IMS
バージョン 14

アプリケーション・プログラミング API

SC43-3857-03

IMS

バージョン 14

アプリケーション・プログラミング API

SC43-3857-03

お願い

本書および本書で紹介する製品をご使用になる前に、 921 ページの『特記事項』に記載されている情報をお読みください。

本書は、IMS 14 (プログラム番号 5635-A05)、IMS Database Value Unit Edition V14.01.00 (プログラム番号 5655-DSE)、IMS Transaction Manager Value Unit Edition V14.01.00 (プログラム番号 5655-TM3)、および新しい版で明記されていない限り、以降のすべてのリソースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC19-4209-03
IMS
Version 14
Application Programming APIs
(November 1, 2017 edition)

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

本書について

これらのトピックは、IMS™ アプリケーション・プログラミング・インターフェース (API) に関する参照情報を提供します。 また、IMS の SQL プログラミング、IMS Adapter for REXX、DL/I テスト・プログラム (DFSDDLT0)、および IMS メッセージ形式サービス (MFS) の参照情報も提供します。IMS アプリケーション・プログラムを作成するためのガイダンス情報は、「*IMS V14* アプリケーション・プログラミング」に記載されています。

この情報は、IBM® Knowledge Center で参照できます。

前提知識

本書は、以下の環境での IMS アプリケーション・プログラミングに関する API (アプリケーション・プログラミング・インターフェース) 解説書です。

- IMS Database Control (DBCTL) を含む IMS Database Manager (IMS DB)
- IMS Transaction Manager (IMS TM)
- CICS® EXEC DLI
- WebSphere® Application Server for z/OS®
- 分散プラットフォーム用 WebSphere Application Server
- Java™ 従属領域 (JMP および JBP)
- スタンドアロン Java アプリケーション開発用のすべての環境

本書では、DL/I、EXEC DLI、IMS Universal ドライバー、および IMS 用の Java クラス・ライブラリーを含む、IMS アプリケーション・プログラミング・インターフェース (API) に関する参照情報を提供します。また、IMS Adapter for REXX、DL/I テスト・プログラム (DFSDDLT0)、および IMS メッセージ形式サービス (MFS) の参照情報も提供します。IMS アプリケーション・プログラムを作成するためのガイダンス情報は、「*IMS V14* アプリケーション・プログラミング」に記載されています。

本書を使用する前に、「*IMS V14* アプリケーション・プログラミング」で説明されているアプリケーション設計の概念を理解しておく必要があります。その資料では、読者が z/OS および IMS の基本概念と IMS 環境について理解しているものと想定しています。読者は、アセンブラー言語、 C 言語、 COBOL、 Pascal、または PL/I の使用方法についても理解しておく必要があります。CICS プログラムは、アセンブラー言語、 C 言語、 COBOL、 PL/I、および C++ で作成することができます。

Java アプリケーションを作成するには、Java 言語および JDBC を完全に理解する必要があります。本書は、読者が Java および JDBC を十分理解していることを前提としています。 Java または JDBC の概念に関する説明は含まれません。

Java データベース・メタデータ・クラスを作成するには (これは IMS Universal ドライバー または Java クラス・ライブラリーを使用した IMS 用 Java アプリケ

ーションの作成に必要なステップです)、IMS データベースを理解する必要があります。IMS データベース概念は、「*IMS V14* データベース管理」で説明されています。

XML を保管または検索するアプリケーションを作成するには、XML スキーマなど、XML とその関連テクノロジーを理解しておく必要があります。

z/OS の詳細については、IBM Knowledge Centerの『z/OS basic skills』トピックを参照してください。

IMS の基本概念を理解するには、「*An Introduction to IMS*」(IBM Press 出版) をお読みになると役立ちます。

IBM では、IMS の学習に役立つような講習会や自習講座を数多く提供しています。利用可能な講習の詳しいリストについては、IBM Skills Gateway にアクセスして、IMS を検索してください。

新規および変更された情報の識別方法

IMS ライブラリーの PDF 資料のほとんどの新規および変更された情報は、左マージン内の文字 (改訂マーカー) によって示されています。「リリース計画」、ならびに「*Program Directory*」および「*Licensed Program Specifications*」の第 1 版 (-00) には、改訂マーカーは含まれていません。

改訂マーカーは、以下の一般的な規則に従っています。

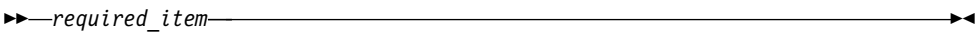
- 技術的な変更のみにマークが付けられています。形式上の変更や文法的な変更には、マークは付けられていません。
- 段落、構文図、リスト項目、操作手順、または図などの要素の一部が変更された場合、その要素の一部だけの変更であっても、要素全体に改訂マーカーが付けられています。
- トピックの変更が 50% を超えた場合には、そのトピック全体に改訂マーカーが付けられています (そのため、新規トピックではなくても、新規トピックのように見えることがあります)。

改訂マーカーは情報に加えられたすべての変更を示しているとは限りません。削除されたテキストとグラフィックスには、改訂マーカーでマークを付けることはできないためです。

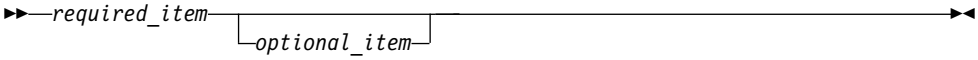
構文図の読み方

本書で使用されている構文図には、以下の規則が適用されています。

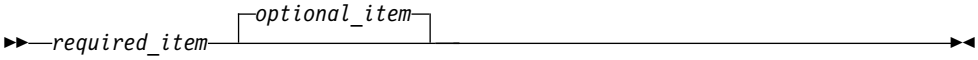
- 構文図は、経路を示す線に沿って、左から右、上から下に読み取ります。以下の規則が使用されます。
  - >>--- 記号は、構文図の始まりを示します。
  - ---> 記号は、構文図が次の行に続くことを示します。
  - >--- 記号は、この構文図が直前の行から続いていることを示します。
  - --->< 記号は、構文図の終わりを示します。
- 必須項目は、水平線 (メインパス) 上に表示されます。



- オプション項目は、メインパスより下に示されます。

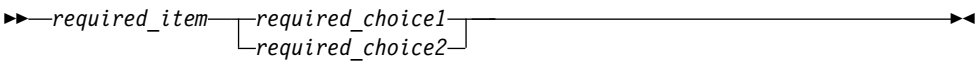


メインパスより上にオプショナル項目が示されている場合は、その項目が構文エレメントの実行に影響することはなく、読みやすくするためのみの表記です。

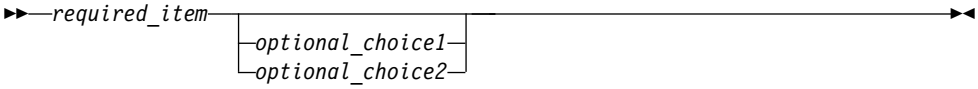


- 複数の項目から選択できる場合は、縦方向に並べて (スタック) 示されます。

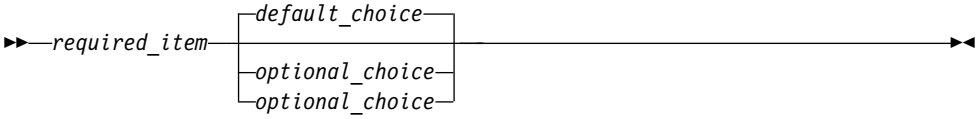
それらの項目の中から 1 つを選択する必要がある 場合は、スタックの中の 1 つの項目がメインパス上に表示されます。



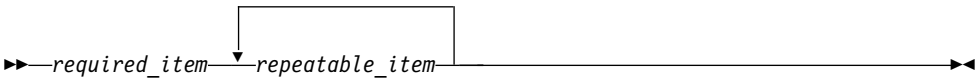
それらの項目から 1 つを選択することがオプションである場合は、スタック全体がメインパスの下に表示されます。



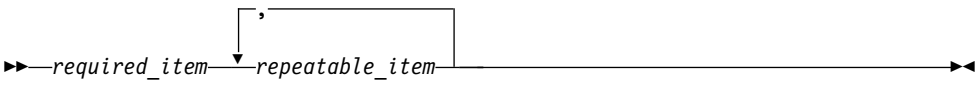
デフォルト項目が含まれている場合、その項目はメインパスより上に示され、他の選択項目はメインパスより下に示されます。



- メインパスの上方にある左に戻る矢印線は、項目が反復可能であることを示します。

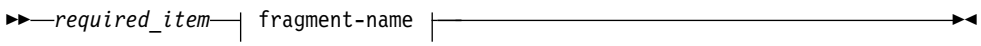


反復矢印線にコンマが含まれている場合は、反復項目をコンマで区切る必要があります。

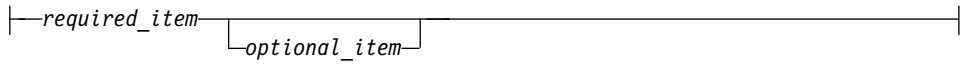


スタック上方の反復矢印線は、スタック内の項目を反復できることを示しています。

- 1 つの構文図を複数のフラグメントに分割しなければならない場合もあります。構文フラグメントはメインの構文図とは別に示されますが、フラグメントの内容は、図のメインパス上にあるものとして読む必要があります。



fragment-name:



- IMS では、b 記号は、該当位置にブランクが 1 つあることを示します。
- キーワード、および該当する場合はキーワードの最小の省略語は、大文字で表されます。これらは、示されているとおりに入力する必要があります。変数は、すべて小文字のイタリック文字で示されます (例えば、*column-name*)。これらは、ユーザーが指定する名前または値を表します。
- キーワードとパラメーターは、構文図で間に句読点が示されていない場合は、少なくとも 1 つのスペースで分離します。
- 句読記号、括弧、算術演算子、およびその他の記号は、構文図で示されたとおりに入力します。
- 脚注は、例えば (1) のように、数字を括弧で囲んで示してあります。

IMS 14 のアクセシビリティー機能

アクセシビリティー機能は、運動障害または視覚障害など身体に障害を持つユーザーが情報技術製品を快適に使用できるようにサポートします。

アクセシビリティー機能

以下のリストは、 IMS 14 を含む z/OS 製品の主なアクセシビリティー機能を示しています。これらの機能は、以下をサポートしています。

- キーボードのみの操作。
- スクリーン・リーダー (読み上げソフトウェア) およびスクリーン拡大鏡によって通常使用されるインターフェース。
- 色、コントラスト、フォント・サイズなど表示属性のカスタマイズ。

キーボード・ナビゲーション

IMS 14 ISPF パネル機能には、キーボードまたはキーボード・ショートカット・キーを使用してアクセスできます。

TSO/E または ISPF を使用して IMS 14 ISPF パネルをナビゲートする詳細については、「*z/OS TSO/E* 入門」、「*z/OS TSO/E* ユーザーズ・ガイド」、および「*z/OS* 対話式システム生産性向上機能 *(ISPF)* ユーザーズ・ガイド 第 *1* 巻」を参照してください。上記の資料には、キーボード・ショートカットまたはファンクション・キー (PF キー) の使用方法を含む、各インターフェースのナビゲート方法が

記載されています。それぞれの資料では、PF キーのデフォルトの設定値とそれらの機能の変更方法についても説明しています。

関連のアクセシビリティー情報

IMS 14 のオンライン資料は、IBM Knowledge Center で参照できます。

IBM におけるアクセシビリティー

IBM のアクセシビリティーに対する取り組みについて詳しくは、*IBM Human Ability and Accessibility Center* (www.ibm.com/able) を参照してください。

ご意見の送付方法

お客様のご意見を送り返していただくことは、弊社が正確な情報を提供し、品質の高い情報を提供するうえで重要なことです。本書またはその他の IMS 関連資料についてコメントのある場合、次のいずれかの方法でお送りください。

- IBM Knowledge Center のトピックの下部にある「**Contact Us**」タブをクリックします。
- imspubs@us.ibm.com に E メールを送信します。必ず、資料タイトルと資料番号を記載してください。

弊社が迅速かつ正確に対応するために、ご意見をお送りいただく資料の内容、その掲載個所、改善のためのご提案について可能な限り多くの情報を記載してください。

第 1 章 DL/I 呼び出し参照

これらのトピックには、IMS DL/I 呼び出しに関する参照情報が記載されています。

データベース管理

以下の DL/I 呼び出しを使用して、IMS データベースにアクセスしてそれを管理します。

データベース管理のための DL/I 呼び出し

IMS DB で以下の DL/I 呼び出しを使用して、アプリケーション・プログラムでのデータベース管理機能を実行します。

各呼び出しの説明には以下の内容が含まれます。

- 構文図
- その呼び出しで使用可能なパラメーターの定義
- アプリケーション・プログラムでの呼び出しの詳しい使用方法
- 呼び出しの使用に関する制約事項 (該当する場合)

各パラメーターは入力パラメーターまたは出力パラメーターとして説明されています。「入力」とは、アプリケーション・プログラムから IMS への入力のことを指します。「出力」とは、IMS からアプリケーション・プログラムへの出力のことを指します。

データベース管理呼び出しでは、*db pcb* または *aib* パラメーターを使用する必要があります。これらの呼び出しの構文図は、機能 パラメーターで始まります。この構文図には、呼び出し、呼び出しインターフェース (xxxTDLI)、および *parmcount* (必要な場合) は含まれていません。

関連資料: アセンブラー言語、C 言語、COBOL、Pascal、および PL/I でのプログラムのコーディングについての詳細は、「*IMS V14* アプリケーション・プログラミング」のトピック『アプリケーション・プログラム・エレメントの定義』を参照してください。

関連資料:

143 ページの『IMS TM システム・サービスのための DL/I 呼び出し』

94 ページの『トランザクション管理のための DL/I 呼び出し』

188 ページの『EXEC DLI コマンド』

データベース管理呼び出しの要約

以下の表は、各データベース管理呼び出しに有効なパラメーターを示します。

オプション・パラメーターは、大括弧 ([ ]) で囲まれています。

制約事項: 言語依存パラメーターは、ここでは示されていません。PLITDLI 呼び出しには、*parmcount* 変数が必要です。アセンブラー言語呼び出しには、*parmcount* か **VL** のどちらかが必要です。COBOL、C、および Pascal プログラムでは、parmcount はオプションです。

関連資料: 言語依存アプリケーション・エレメントの詳細については、「*IMS V14* アプリケーション・プログラミング」の『アプリケーション・プログラム・エレメントの定義』を参照してください。

表 *1. DB* 呼び出しの要約

機能コード	意味および用途	オプション	パラメーター	有効な環境
CIMS		z/OS アプリケーション領域で ODBA インターフェースを初期化し、終了させる。	aib	DB/DC、DBCTL、ODBA
CLSE	クローズ	GSAM データベースを明示的にクローズする。	function、gsam pcb または aib	DB/DC、DBCTL、DB バッチ、ODBA
DEQb	Dequeue	Q コマンド・コードによって予約されたセグメントを解放する。	function、i/o pcb (全機能のみ)、または aib、i/o area (全機能のみ)	DB バッチ、BMP、MPP、IFP、DBCTL、ODBA
DLET	Delete	セグメントおよびその従属セグメントをデータベースから除去する。	function、db pcb または aib、i/o area、[ssa]	DB/DC、DBCTL、DB バッチ、ODBA
FLDb	フィールド	セグメント内のフィールドにアクセスする。	function、db pcb または aib、i/o area、rootssa	DB/DC、ODBA
GHNb	Get Hold Next	後続のメッセージ・セグメントを検索する。	function、db pcb または aib、i/o area、[ssa]	DB/DC、DBCTL、DB バッチ、ODBA
GHNP	Get Hold Next in Parent	従属セグメントを順次に検索する。	function、db pcb または aib、i/o area、[ssa]	DB/DC、DBCTL、DB バッチ、ODBA
GHUb	Get Hold Unique	セグメントを検索し、データベース内で開始位置を設定する。	function、db pcb または aib、i/o area、[ssa]	DB/DC、DBCTL、DB バッチ、ODBA
GNbb	後続取り出し	後続のメッセージ・セグメントを検索する。	function、db pcb または aib、i/o area、[ssa または rsa]	DB/DC、DBCTL、DB バッチ、ODBA
GNPb	親における後続セグメントの取り出し	従属セグメントを順次に検索する。	function、db pcb または aib、i/o area、[ssa]	DB/DC、DBCTL、DB バッチ、ODBA
GUbb	初回取り出し	セグメントを検索し、データベース内で開始位置を設定する。	function、db pcb または aib、i/o area、[ssa または rsa]	DB/DC、DBCTL、DB バッチ、ODBA

表 *1. DB* 呼び出しの要約 *(*続き*)*

機能コード	意味および用途	オプション	パラメーター	有効な環境
GUR	初回レコード取り出し	IMS カタログから XML 形式で完全なレコードを取り出す。	function、aib、i/o area、[ssa]	DB/DC、DBCTL、DB バッチ、ODBA
ISRT	挿入	1 つ以上のセグメントをロードし、データベースに追加する。	function、db pcb または aib、i/o area、[ssa または rsa]	DB/DC、DCCTL、DB バッチ、ODBA
OPEN	オープン	GSAM データベースを明示的にオープンする。	function、gsam pcb または aib、[i/o area]	DB/DC、DBCTL、DB バッチ、ODBA
POSb	位置	特定の従属セグメントまたは最後に挿入された順次従属セグメントの位置を検索する。	function、db pcb または aib、i/o area、[ssa]	DB/DC、DBCTL、DB バッチ、ODBA
REPL	置換	セグメント内の 1 つ以上のフィールドの値を変更する。	function、db pcb または aib、i/o area、[ssa]	DB/DC、DBCTL、DB バッチ、ODBA
RLSE	ロック解除	無変更データについて、保留されているすべてのロックを解除する	function、db pcb	DB/DC、DBCTL、DB バッチ、ODBA

CIMS 呼び出し

CIMS 呼び出しは、z/OS アプリケーション領域で ODBA インターフェースを初期化し、終了させるために用いられます。

フォーマット

呼び出し名	**DB/DC**	**IMS DB**	**DCCTL**	**DB** バッチ	**TM** バッチ
CIMS	X	X			

パラメーター

aib

この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
文字値。

AIBSFUNC
副次機能コード。このフィールドには、以下のような 8 バイトの副次機能コードのいずれかを入れる必要があります。

INIT
AIBRSNM2。ODBA 始動テーブルの 4 文字 ID。

CONNECT
AIBRSA1。CONNECT パラメーター・リストのアドレス。

以下の表は、CIMS CONNECT パラメーター・リストの形式を示しています。

表 2. *CIMS CONNECT* パラメーター・リストの形式

オフセット	長さ	フィールド	使用上の説明
X'00'	X'04'	入力	接続要求テーブルの項目数。
X'04'	X'04'	入力	接続要求テーブルのアドレス。

TERM
AIBRSNM2。終了させたい IMS 接続を表す ODBA 始動テーブルの 4 文字 ID。

TALL
すべての IMS 接続の終了。

使用法

CIMS 呼び出しは、アプリケーション・アドレス・スペースで動作するアプリケーション・プログラムにより、ODBA 環境を確立もしくは終了させる目的に使用されます。

INIT*bbbb*
z/OS アプリケーション・アドレス・スペースに ODBA 環境を確立しようとするアプリケーションは、CIMS 副次機能の INIT を使用しなければなりません。

AIBRSNM2 には、ODBA 始動テーブル・メンバーの 4 文字 ID を指定できます (指定は任意です)。このメンバーは DFSxxxx0 という名前で、その xxxx が 4 文字 ID に相当します。AIBRSNM2 が指定されると、まず z/OS アプリケーション・アドレス・スペースで ODBA 環境が初期化されます。次に、メンバー DFSxxxx0 で指定された IMS と、ODBA が接続を確立しようとします。

CONNECT*b*
CIMS CONNECT 呼び出しを使用して、CSL Open Database Manager (ODBM) から IMS システムへの ODBA 接続を複数確立します。

CIMS CONNECT 呼び出しは、CIMS INIT 呼び出しの代わりに、あるいはその呼び出しに追加して出すことができます。ODBA がまだ初期設定されていない場合は、CIMS CONNECT 呼び出しによって ODBA が初期設定されます。初期設定のみを実行するには、AIBRSA1 を -1 (X'FFFFFFFF') に設定して CIMS CONNECT 呼び出しを出します。

接続要求テーブルでは、1 つ以上の接続要求項目が連続したストレージに収められます。どの項目にも、以下のフィールドが入ります。

- 1 文字から 4 文字の別名。左揃えされ、右側はブランクで埋められます。この別名は、始動プロパティー・テーブル DFScccc0 から取得された値 (*cccc*) です。このパラメーターは必須です。
- 接続プロパティー・テーブル (DFSPRP) の 4 バイトのアドレスまたは 0。

  値 0 は、IMS 接続プロパティーを取得するために ODBA が DFScccc0 をロードする必要があることを示します。このメンバーを作成するには、DFScccc0 で DFSPRP マクロを指定し、メンバーのアセンブルとリンクを行います。このメンバーは、ODBA アプリケーション・ジョブの STEPLIB または JOBLIB 内にあることが必要です。

  値がゼロでない場合は、呼び出し元から接続プロパティー・パラメーター・テーブルのアドレスが渡されます。接続プロパティー・パラメーターは、DFSPRP マクロによってマップされます。
- 接続要求の戻りコードが入る 4 バイトのフィールド。戻りコードは AIBRETRN コードのいずれかです。
- 接続要求の理由コードが入る 4 バイトのフィールド。理由コードは AIBREASN コードのいずれかです。
- 接続要求のエラー拡張情報コードが入る 4 バイトのフィールド。エラー拡張は、戻りコードおよび理由コードに固有の付加的な診断情報を含んでいます。

以下の表は、CIMS CONNECT テーブル項目の形式を要約したものです。

表 *3. CIMS CONNECT* テーブル項目の形式

オフセット	長さ	フィールド	使用上の説明
X'00'	X'04'	入力	始動プロパティー・テーブル DFScccc0 から取得した 1 文字から 4 文字の IMS 別名 (*cccc*)。この *cccc* は別名。

表 3. *CIMS CONNECT* テーブル項目の形式 (続き)

オフセット	長さ	フィールド	使用上の説明
X'04'	X'04	入力	0 または ODBA 始動プロパティー・テーブルのアドレス。 値 0 は、ODBA が DFScccc0 (cccc は提供されている別名) という始動プロパティー・テーブルをロードする必要があることを示す。 アドレスは、呼び出し元から始動プロパティー・テーブルが提供されることを示す。このテーブルは DFSPRP マクロによってマップされる。
X'08'	X'04'	出力	この項目の接続要求戻りコード。
X'0C'	X'04'	出力	この項目の接続要求理由コード。
X'10'	X'04'	出力	この項目の接続要求エラー拡張コード。

TERM*bbbb*

CIMS 副次機能の TERM は、IMS 接続を 1 つ終了させます。AIBRSNM2 には、終了させたい IMS 接続を表す始動テーブル・メンバーの 4 文字 ID を指定します。TERM 副次機能が完了しても、z/OS アプリケーション・アドレス・スペースでは ODBA 環境がそのまま存続しています。

注: CIMS INIT を出したアプリケーションが、CIMS TERM の完了後にオペレーティング・システムに戻ることを選択すると、アドレス・スペースはシステム異常終了 A03 で終了します。これを避けるには、オペレーティング・システムに戻る前に CIMS TALL を出してください。

TALL*bbbb*

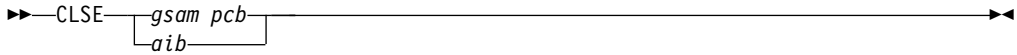
すべての IMS 接続を終了させ、アプリケーション・アドレス・スペースの ODBA 環境を終了させるためには、CIMS 副次機能の TALL を出さなければなりません。

CLSE 呼び出し

クローズ (CLSE) 呼び出しは、GSAM データベースを明示的にクローズするために使用されます。

GSAM の詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『GSAM データベースの処理』を参照してください。

フォーマット



	呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
GSAM の場合:	CLSE	X	X	X	X	X

パラメーター

gsam pcb
呼び出しで使用する GSAM PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB 長。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、GSAM PCB の名前を指定しなければなりません。

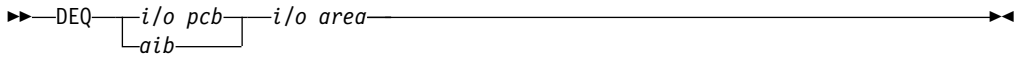
使用法

CLSE の使用法については、「*IMS V14* アプリケーション・プログラミング」のトピック『GSAM に対する明示的なオープンおよびクローズ呼び出し』を参照してください。

DEQ 呼び出し

デキュー (DEQ) 呼び出しは、Q コマンド・コードを使用して検索されるセグメントを解放するために使用します。

形式 (全機能)



形式 (高速機能 DEDB)

	呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
全機能および DEDB 用:	DEQ	X	X		X	

パラメーター

DEDB pcb **(高速機能のみ)**
呼び出しで使用する DEDB PCB を指定します。

i/o pcb **(全機能のみ)**
DEQ 呼び出しで使用する入出力 PCB を指定します。 これは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。これは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbbを指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area **(全機能のみ)**
(A から J までの) 1 文字を納める 1 バイト域を指定します。この文字は解放されるロックのロック・クラスを表します。これは必須の入力パラメーターです。

使用法

DEQ 呼び出しは、Q コマンド・コードを使用して検索されるすべてのセグメントを解放します。ただし、次の場合は除きます。

- ユーザーのプログラムによって修正されたセグメントは、そのプログラムがコミット・ポイントに達するまで解放されません。
- 階層内でユーザーの位置を保持するために必要なセグメントは、ユーザーのプログラムが別のデータベース・レコードに移動するまで解放されません。
- 別のロック・クラスを使用してロックされていたセグメントのクラス

ユーザーのプログラムがセグメントの読み取りだけを行う場合には、プログラムで DEQ 呼び出しを出してセグメントを解放できます。ユーザーのプログラムが DEQ 呼び出しを出さない場合にも、IMS は、そのプログラムがコミット・ポイントに達したときに予約済みのセグメントを解放します。ユーザーのプログラムがコミット・ポイントに達する前に DEQ 呼び出しを使用してセグメントを解放すると、他のプログラムが、より早くそのセグメントを使用できるようになります。

DEQ 呼び出しと Q コマンド・コードの関連についての詳細は、「*IMS V14* アプリケーション・プログラミング」のトピック『ユーザー・プログラムの排他使用に対するセグメントの予約』を参照してください。

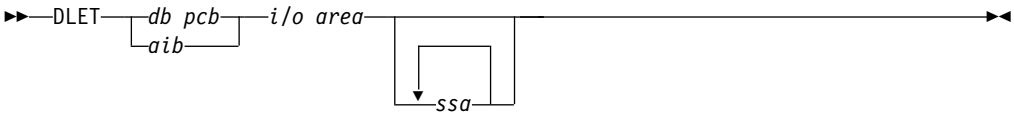
制約事項

リモート環境が CICS (DBCTL) でもある場合、CICS DL/I 環境では、1 つの CICS (DBCTL) システムから出された呼び出しはそのリモート CICS DL/I 環境でサポートされます。

DLET 呼び出し

削除 (DLET) 呼び出しは、あるセグメントとその従属セグメントをデータベースから除去するために使用します。

フォーマット



	呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
全機能の場合:	DLET	X	X		X	
DEDB の場合:	DLET	X	X			
MSDB の場合:	DLET	X				

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
ユーザーのプログラム内で IMS と通信する入出力域を指定します。このパラメーターは入力パラメーターです。セグメントを削除する前に、入出力域にそのセグメントを入れるために Get Hold 呼び出しを出さなければなりません。その後で、DLET 呼び出しを出すことにより、そのセグメントおよび従属セグメントをデータベースから削除できます。

ssa
呼び出しで使用したい SSA があれば、それを指定します。 このパラメーターは入力パラメーターです。呼び出しで提供する SSA は、その呼び出しのための SSA が定義されている、ユーザー・プログラム内のデータ域を指します。このパラメーターでは、1 つの SSA だけを使用できます。 DLET 呼び出しの場合には、このパラメーターはオプションです。

使用法

DLET 呼び出しの前に、3 つの Get Hold 呼び出しのうちのいずれかを出さなければなりません。 DLET 呼び出しを出すと、保留セグメント、およびそれに物理的に従属するすべてのセグメントは、それらのすべてのセグメントがユーザーのプログラムで感知できるかどうかにかかわらず IMS によってデータベースから削除されます。その PCB に関して Get Hold、REPL、または DLET 呼び出しが先に出されていない場合には、IMS は DLET 呼び出しを拒否します。この DLET 呼び出しが成功すると、前に検索されたセグメントとその従属セグメントはすべてデータベースから削除され、再び検索することはできなくなります。

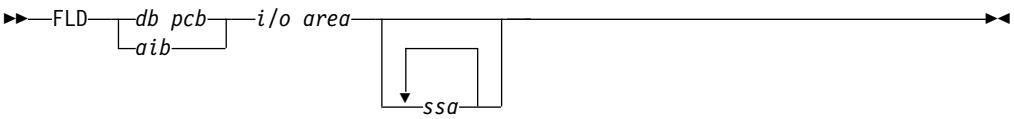
DLET 呼び出しの前に出された Get Hold 呼び出しがパス呼び出しであり、検索されたセグメントのすべてを削除するのではない場合は、検索されたセグメント (従属セグメントがあればそれも含めて) のうちでどれを削除したいかを、IMS に非修飾 SSA を指示する必要があります。この方法でセグメントを削除すると、そのセグメントのすべての従属セグメントが自動的に削除されます。DLET 呼び出しでは 1 つの SSA だけを指定することができ、また、DLET 呼び出しで SSA が適用されるのはこの場合に限られます。

DLET 呼び出しに適用されるコマンド・コードはありません。DLET 呼び出しでコマンド・コードを使用した場合、IMS はそのコマンド・コードを無視します。

FLD 呼び出し

フィールド (FLD) 呼び出しは、MSDB または DEDB 用のセグメント内のフィールドにアクセスするために使用されます。

フォーマット



	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
MSDB の場合:	FLD	X				
DEDB の場合:	FLD	X	X			

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
ユーザーのプログラム内のこの呼び出しに関する FSA (フィールド検索引数) が含まれている入出力域を指定します。このパラメーターは入力パラメーターです。

ssa
呼び出しで使用したい SSA があれば、それを指定します。この入力パラメーターでは、 SSA を最大 15 個使用できます。ユーザーの提供する SSA は、その呼び出しのために定義したデータ域を指します。FLD 呼び出しの場合には、このパラメーターはオプションです。

使用法

セグメント内のフィールドの内容をアクセスおよび変更するには、FLD 呼び出しを使用してください。

FLD 呼び出しはユーザーに代わって 2 つのことを実行します。すなわち、フィールドの値とユーザーが指定した値を比較し (FLD/VERIFY)、指定した方法でそのフィールドの値を変更します (FLD/CHANGE)。

FLD 呼び出しを使用すると、DEDB ですべての DL/I コマンド・コードを利用できるようになります。DEDB 用の FLD 呼び出しの形式は他の DL/I 呼び出し用の形式と同じです。したがって、ユーザーの MSDB が DEDB に変換されている場合、FLD 呼び出しを使用するアプリケーション・プログラムを変更する必要はありません。FLD 呼び出しの詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『セグメントの更新: REPL、DLET、ISRT、および FLD』を参照してください。

DEDB 用のアプリケーション・プログラムでは、GHU、REPL、および DL/I 呼び出しを組み合わせて使用する代わりに FLD 呼び出しを使用することもできます。

FSA

フィールド検索引数 (FSA) は、他の DL/I データベース呼び出しで使用される入出力域と同等です。FLD 呼び出しの場合、データは入出力域に移動はしません。逆に、 FSA が入出力域に移動します。

1 つの FLD 呼び出しで、複数の FSA が認められています。これは、 FSA のコネクター・フィールドで指定されます。各 FSA は、ターゲット・セグメント内の同一フィールドででもあるいは異なるフィールドででも機能します。

FLD 呼び出しでユーザーが参照する FSA には、5 つのフィールドがあります。これらのフィールドをコーディングするための規則は次のとおりです。

フィールド名
: このフィールドは 8 バイト長でなければなりません。使用するフィールド名が 8 バイト未満の場合、左寄せして右側をブランクで埋めなければなりません。

FSA 状況コード
: このフィールドは、1 バイト・フィールドです。FLD 呼び出しのあとで、IMS は次のいずれかの状況コードを戻します。

  b 成功

  A 無効操作

  B オペランドの長さが無効

  C 無効な呼び出し - プログラムがキー・フィールドを変更しようとした

  D 検証チェックが不成功

  E パック 10 進数または 16 進数フィールドが無効

  F プログラムが、所有していないセグメントを変更しようとした

  G 算術オーバーフロー

  H セグメント内にフィールドが見つからないない

命令コード
: この 1 バイト・フィールドには、変更操作用に以下の演算子のいずれかが含まれます。

  + フィールド値にオペランドを加えます。

  - フィールド値からオペランドを減算します。

  = フィールド値をオペランドの値にセットします。

  検査操作の場合、このフィールドには次のいずれかが含まれていなければなりません。

  E フィールド値とオペランドが等しいことを検査します。

  G フィールド値がオペランドよりも大きいことを検査します。

  H フィールド値がオペランドも大きいかまたは等しいことを検査します。

  L フィールド値がオペランドよりも小さいことを検査します。

M フィールド値がオペランドよりも小さいかまたは等しいことを検査します。

N フィールド値がオペランドに等しくないことを検査します。

オペランド

この可変長フィールドには、フィールド値のテストのテスト基準にする値が入ります。このフィールド内のデータは、セグメント・フィールドのデータと同じタイプでなければなりません。(これは、DBD で定義されます。)データが 16 進数である場合、オペランドの値の長さはデータベース内のフィールドの長さの 2 倍になります。データがパック 10 進数の場合、オペランドには先行ゼロが含まれないため、オペランドの長さは実際のフィールドよりも短いことがあります。その他のタイプのデータの場合、長さは同じでなければなりません。

結合子

この 1 バイト・フィールドには、これが最後または唯一の FSA である場合にはブランクが入り、このあとに別の FSA が続く場合にはアスタリスク (*) が入らなければなりません。

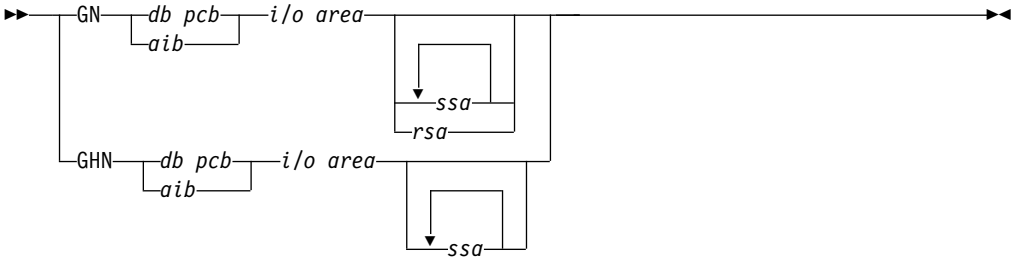
FLD 呼び出しで使用する SSA の形式は、DL/I 呼び出しにおける SSA の形式と同じです。SSA が指定されていない場合には、 MSDB 内または DEDB 内の最初のセグメントが検索されます。

FLD 呼び出しの詳細および例については、「*IMS V14* アプリケーション・プログラミング」のトピック『MSDB と DEDB の処理』を参照してください。

GN/GHN 呼び出し

Get Next (GN) 呼び出しは、データベースからセグメントを順に検索するために使用されます。Get Hold Next (GHN) は、GN 呼び出しの保留の形式です。

フォーマット



	呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
全機能の場合:	GN/GHN	X	X		X	
GSAM の場合:	GN	X	X	X	X	X
DEDB の場合:	GN	X	X	X		
MSDB の場合:	GN	X				

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
入出力域を指定します。このパラメーターは出力パラメーターです。いずれかの Get 呼び出しが成功すると、IMS は要求されたセグメントをこの区域に戻します。ユーザーのプログラムがなんらかのパス呼び出しを出す場合には、入出力域は、パス呼び出しの後の連結セグメントの最長の経路を収めるのに十分な長さでなければなりません。この区域は、常に、左寄せのセグメント・データが入ります。入出力域は、この区域の最初のバイトを指します。

GSAM で GN 呼び出しを使用すると、*i/o area* パラメーターで指定された区域に検索対象のレコードが入ります。

ssa
呼び出しで使用したい SSA があれば、それを指定します。 このパラメーターは入力パラメーターです。呼び出しで提供する SSA は、その呼び出しのための SSA が定義されている、ユーザーのプログラム内のデータ域を指します。このパラメーターでは、SSA を最大 15 個使用できます。 GN 呼び出しの場合には、このパラメーターはオプションです。

rsa
ユーザーのプログラムにおける、レコードの RSA が戻される区域を指定します。この出力パラメーターは GSAM の場合にだけ使用され、オプションです。 RSA の詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『GSAM データ域』を参照してください。

使用法: Get Next (GN)

Get Next (GN) 呼び出しはセグメントに関する要求であり、ユーザーが提供する SSA で記述されているように、GN 呼び出しの前に発行された呼び出しにリンクされています。 IMS は、現在位置で検索を開始します。

GN 呼び出しを使用する際には、以下の点に留意してください。

- (呼び出しに F コマンド・コードが含まれていない限り) 処理は現在位置から前方に向かって進められます。
- IMS は、(前の呼び出しでセットされた) 現在位置を検索の開始点として使用します。
- 検索されるセグメントは、階層における次の順序位置と呼び出しに含まれている SSA の組み合わせによって決められます。
- GN は、注意して使用してください。SSA を使用した場合に、IMS がセグメントの検索を実行せずにデータベースの終わりまで検索だけを実行する可能性があるためです。特に、「等しくない」または「より大きい」関係演算子の場合にはこの可能性が高くなります。

GN 呼び出しは、ユーザーが指定した SSA を満足させる、階層内の次のセグメントを検索します。GN 呼び出しで検索されるセグメントは、階層内の現在位置によって異なるため、GN は多くの場合に、GU 呼び出しの後で出されます。階層内で位置が設定されていない場合には、GN はデータベース内の最初のセグメントを検索します。 GN 呼び出しは、データベース内の現在位置から前方に移動しながらセグメントまたはセグメントの経路を検索します。処理が進むにつれ、IMS は呼び出しを満たす各レベルのセグメントを検索します。

例えば、階層内の順次検索は、常に上から下の順で、左から右に向かって行われます。例えば、以下の図の階層に対して非修飾の GN 呼び出しを繰り返し出すと、IMS はデータベース・レコード内のセグメント・オカレンスを次の順序で戻します。

1. A1 (ルート・セグメント)
2. B1 およびその従属セグメント (C1、D1、F1、D2、D3、E1、E2、および G1)
3. H1 およびその従属セグメント (I1、I2、J1、および K1)

IMS が K1 を戻したあとで非修飾 GN を再び出すと、IMS はデータベース内でセグメント A1 に続くキーが設定されているルート・セグメント・オカレンスを戻します。

セグメント・タイプで修飾された GN 呼び出しでは、データベース内の特定セグメント・タイプのすべてのオカレンスを検索できます。

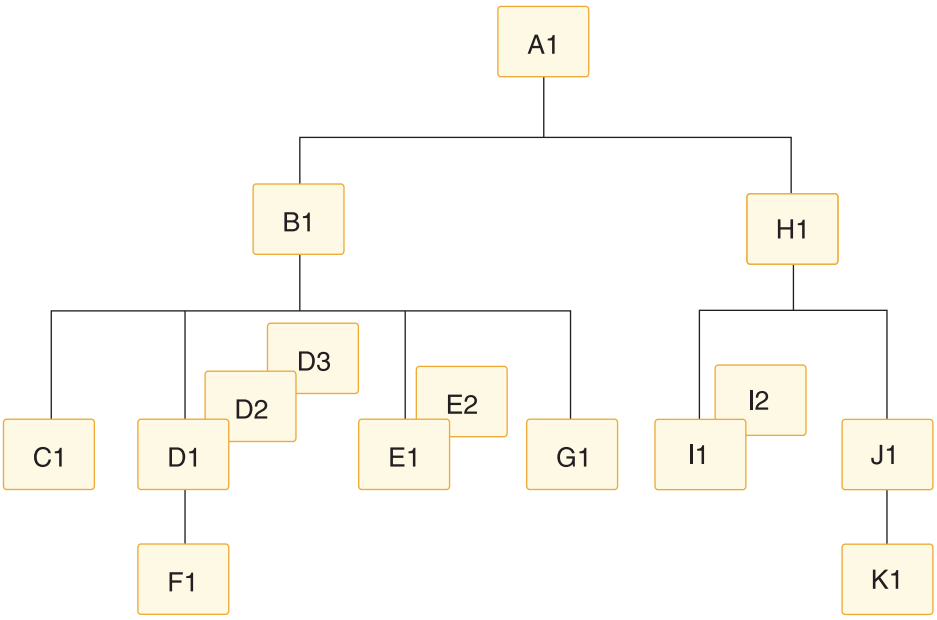


図 *1.* 階層シーケンス

例えば、セグメント A1 および B1 に対しては修飾 SSA を指定し、セグメント・タイプ D に対しては非修飾 SSA を指定して GN 呼び出しを出すと、IMS は、ユーザーが 1 回目にその呼び出しを出した時にはセグメント D1 を戻し、2 回目にその呼び出しを出した時にはセグメント D2 を戻し、3 回目にその呼び出しを出した時にはセグメント D3 を戻します。4 回目にその呼び出しを出すと、IMS は状況コード GE を戻します。これは IMS が要求されたセグメントを見つけられなかったことを意味します。

現在位置から初めて、階層内のセグメントのすべてのオカレンスを階層シーケンスで検索したい場合には、非修飾 GN 呼び出しを使用できます。それぞれの非修飾 GN 呼び出しは、現在位置から前方に進んで次の順序のセグメントを検索します。例えば、次の処理要求を満たしたいものとします。

医療データベース全体を印刷する。

この場合には、IMS が GB 状況コードを戻して、データベースの最後に達したが呼び出しを満たすことはできなかったことが示されるまで、非修飾 GN 呼び出しを繰り返し出してください。GB 状況コードが戻された後で GN を再び出すと、IMS はデータベース内の最初のセグメント・オカレンスを戻します。

GU と同様に、GN 呼び出しには階層内のレベル数と同数の SSA を使用できます。完全修飾 SSA を GN 呼び出しで使用すると、ユーザーが要求する階層経路およびセグメントを明確に識別でき、呼び出しの文書化の際に役立ちます。

非修飾 SSA を使用する GN 呼び出しは、現在位置から前方に進みながら、そのセグメント・タイプの次のオカレンスを検索します。

修飾された SSA を使用する GN は、指定されたセグメント・タイプのオカレンスのうちでその SSAs を満たす次のオカレンスを検索します。

GN で複数の SSA を指定した場合、非修飾 SSA でコマンド・コードを使用しない限り、その呼び出しに非修飾 SSA があるかどうかは、操作に影響を与えません。IMS は、修飾された SSA と最後の SSA だけを使用して経路を判別し、セグメントを検索します。階層内のより高いレベルのセグメントに関する SSA が未指定であったり、修飾されていなかったりする場合、より低いレベルの正しく指定されたセグメントあるいは修飾されたセグメントの親である高レベルのセグメントがその要求を満たします。

ルートのキーで SSA が修飾されている GN 呼び出しを出すと、データベース内の位置およびデータベース内のキーの順序によっては、同じ SSA を使用する GU と異なる結果が得られることがあります。データベース内の現在位置が SSA を満足させるセグメントをすでに越えている場合、そのセグメントはこの GN では検索されません。これらの両方の条件が満たされる場合、GN は GE 状況コードを戻します。

- SSA 内のキーの値に上限値が設定されている (例えば、その値よりも「小さいかまたは等しい」演算子がセットされている)。
- 順次検索で、指定されたセグメントが見つからないうちに、SSA 内のその値よりも大きなキーのセグメントが検出された。

指定のセグメントが存在していて、GU 呼び出しを使用したときにそのセグメントが検索されるような場合であっても、GN は GE 状況コードを戻します。

使用法: Get Hold Next (GHN)

ユーザーのプログラムは、セグメントの削除または置き換えを実行する前に、そのセグメントを検索し、なんらかの方法でセグメントを変更することを IMS に通知しなければなりません。プログラムはこれを実行するために、セグメントの削除または置き換えに先立って、『hold』 を指定する Get 呼び出しを出します。 Get Hold 呼び出しによるセグメントの検索が成功すると、プログラムはセグメントを削除したり、そのセグメント内の (キー・フィールド以外の) 1 つ以上のフィールドを変更したりできます。

Get Hold 呼び出しとそれ以外の Get 呼び出しの違いは、Get Hold 呼び出しのあとでは REPL または DLET を出せるという 1 点です。

検索されたセグメントの保留状況は取り消されるため、 DLET または REPL 呼び出しを再び出す前に保留状況を再設定する必要があります。Get Hold 呼び出しを出したあとで、同一の PCB に対して他の呼び出しを介在して出さない場合には、そのセグメントに対して複数の REPL または DLET 呼び出しを出すことができます。

Get Hold 呼び出しを出してから、そのセグメントを更新する必要がないことが分かった場合には、そのセグメントを解放しないで、引き続き他の処理を行うことができます。このセグメントは、現在位置が変更されると (その Get Hold 呼び出しに使用された PCB に対して別の呼び出しを出したときに)、ただちに解放されます。すなわち、REPL や DLET 呼び出しの前には必ず Get Hold 呼び出しを出す必要があります。ただし、Get Hold 呼び出しを出しても、そのセグメントの置き換えまたは削除を実行しなければならないわけではありません。

使用法: HDAM、PHDAM、または DEDB データベースでの GN

HDAM、PHDAM、DEDB 以外のデータベース編成では、GN 呼び出しを使用してデータベースを順次処理すると、ルート・セグメントがキーの昇順で戻されます。ただし、HDAM、PHDAM、DEDB データベースのルート・セグメントの順序は、そのデータベースに指定されるランダム化ルーチンによって決まります。 順次ランダム化ルーチンが指定されている場合を除き、データベース内のルート・セグメントの順序が昇順キー配列になることはありません。

階層直接アクセス方式 (HDAM、PHDAM) または DEDB データベースの場合、一連の非修飾 GN 呼び出しまたはルート・セグメントのみで修飾される GN 呼び出しを出すと、次のような結果になります。

1. 1 つのアンカー・ポイントからすべてのルートを戻します。
2. 次のアンカー・ポイントに移動します。
3. そのアンカー・ポイントからルートを戻します。

順次ランダム化ルーチンが指定されていない場合、連続するアンカー・ポイントにあるルートは、昇順キー配列にはなっていません。 HDAM、PHDAM、DEDB 編成で注意しなければならないのは、 GN 呼び出しが「等しい」演算子または「より大か等しい」演算子によってルート・セグメントのキー・フィールドが修飾されているときです。 IMS は、データベース内に既存の位置がある場合には、要求されたキーが現在のルートのキーに等しいかあるいはそれよりも大きいことを確認します。そのとおりになっていない場合には、GE 状況コードが戻されます。キーが現行キーに等しいか、あるいはそれよりも大きい場合に、現在位置ではそのキーを満たすことができないときには、IMS はランダム化ルーチンを呼び出してそのキーのアンカー・ポイントを判別します。IMS は、選択されたアンカーの最初のルートから始めて、その呼び出しを満たすものを探します。

制約事項

GN を使用して GSAM データベースの次のレコードを検索できますが、GSAM では GHN は有効ではありません。

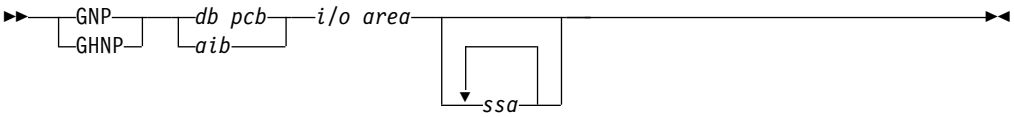
関連資料:

『GNP/GHNP 呼び出し』

GNP/GHNP 呼び出し

Get Next in Parent (GNP) 呼び出しは、従属セグメントを順に検索するために使用します。Get Hold Next in Parent (GHNP) 呼び出しは、 GNP 呼び出しに対応する保留形式です。

フォーマット



	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
全機能の場合:	GNP/GHNP	X	X		X	

	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
DEDB の場合:	GNP/GHNP	X	X	X		
MSDB の場合:	GNP/GHNP	X				

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
入出力域を指定します。このパラメーターは出力パラメーターです。ユーザーが出した Get 呼び出しが成功すると、IMS は要求されたセグメントをこの区域に戻します。ユーザーのプログラムがなんらかのパス呼び出しを出す場合には、入出力域は、パス呼び出しの後の連結セグメントの最長の経路を収めるのに十分な長さでなければなりません。この区域に入るセグメント・データは、常に左寄せされます。入出力域は、この区域の最初のバイトを指します。

ssa
呼び出しで使用したい SSA があれば、それを指定します。 このパラメーターは入力パラメーターです。ユーザーが提供する SSA は、その呼び出しのための SSA が定義されている、ユーザーのプログラム内のデータ域を指します。このパラメーターでは、SSA を最大 15 個使用できます。 GNP 呼び出しの場合、このパラメーターはオプションです。

使用法: Get Next in Parent (GNP)

GNP 呼び出しはセグメントを順次に検索します。GN と GNP の相違点は、GNP の場合には、呼び出しを満足させることのできるセグメントが、設定された親の従属セグメントに限られる点です。

非修飾 GNP は、現在の親に従属する最初の従属セグメント・オカレンスを検索します。現在位置がすでに現在の親に従属するセグメント上にある場合、非修飾 GNP は、次のセグメント・オカレンスを検索します。

データベース内のすべてのセグメントを検索しない場合であっても、データベース内を順方向に移動する時には、GNP を使用することにより、特定セグメントの子セグメントだけが戻されるように制限できます。

先行の ***DL/I*** 呼び出しとのリンク

GNP 呼び出しは、次の 2 通りの方法で、ユーザーのプログラムによって出された直前の DL/I 呼び出しとリンクされます。

- 現在位置: 要求されたセグメントの検索は、先行する GU、GN、または GNP 呼び出しで設定された現在位置から開始されます。
- 親子関係 : 要求されたセグメントの検索は、GU または GN 呼び出しによって最後にアクセスされた最低位セグメントの従属セグメントだけを対象として行われます。親子関係は、検索の終わりを決定するものであり、成功した GU または GN 呼び出しのあとでのみ有効です。

親子関係を使用する処理

親子関係は、次の 2 通りの方法で設定できます。

- 正常な GU または GN 呼び出しを発行する。ユーザーが出した GU または GN 呼び出しが成功すると、IMS は、呼び出しによって戻された最低レベルのセグメントで親子関係を設定します。他の PCB に対して別の GU または GN 呼び出しを出しても、前の呼び出しで最初の PCB を使用して設定した親子関係は影響を受けません。GU または GN 呼び出しが成功しなかった場合には、親子関係は取り消されます。
- GU、GN、または GNP 呼び出しで P コマンド・コードを使用すると、どのレベルでも親子関係を設定できます。

DL/I 呼び出しが親子関係に与える影響

GNP 呼び出しは、P コマンド・コードを含んでいない限り、親子関係に影響を与えません。

2 次索引を使用していない限り、REPL は親子関係に影響を与えることはありません。2 次索引を使用している場合に索引付きセグメントを置き換えると、親子関係は失われます。

DLET 呼び出しの場合には、設定された親を削除するのでない限り、親子関係は影響を受けません。設定された親を削除した場合には、GNP 呼び出しを出す前に親子関係を再設定しなければなりません。

ISRT は、設定された親の従属セグメント以外のセグメントをユーザーが挿入した場合にのみ、親子関係に影響を与えます。この場合、ISRT によって親子関係が取り消されます。ユーザーが挿入するセグメントが、設定された親のなんらかのレベルにおける従属セグメントである場合には、親子関係は影響を受けません。例えば、「*IMS V14* アプリケーション・プログラミング」のトピック『ISRT の後の位置』

では、セグメント B11 が設定された親であるものとしています。次の 2 つの ISRT 呼び出しのどちらも、親子関係に影響を与えません。

```
ISRT   Abbbbbbb(AKEYbbbb=A1)
       Bbbbbbbb(BKEYbbbb=bB11)
       Cbbbbbbbb

ISRT   Abbbbbbbb(AKEYbbbb=bA1)
       Bbbbbbbbb(BKEYbbbb=bB11)
       Cbbbbbbb(CKEYbbbb=bC111)
       Dbbbbbbbb
```

次の ISRT 呼び出しを出すと、F セグメントは設定された親である B に直接従属するセグメントではないため、親子関係は取り消されます。

```
ISRT   Abbbbbbbb(AKEYbbb=bA1)
       Fbbbbbbbb
```

1 つの GNP 呼び出しには、1 つ以上の SSA を含めることができます。この SSA は修飾されていても、非修飾であってもかまいません。SSA を使用しない GNP 呼び出しでは、設定された親の従属セグメントのうちの、次の順序のものが検索されます。GNP で SSA を使用すると、設定された親の特定の従属セグメントまたは従属セグメント・タイプを、IMS に指示できます。

非修飾 SSA を使用した GNP は、設定された親のもとにある従属セグメント・オカレンスの中から、ユーザーが指定したセグメント・タイプのものを順次に検索します。

修飾された SSA を使用した GNP は、ユーザーが検索したいセグメント、またはユーザーが検索したいセグメントへの階層経路の一部となるセグメントを、IMS に対して指示します。修飾された GNP は、データ・フィールドまたは非固有キー・フィールドではなく固有キー・フィールドで修飾されている場合に限り、固有セグメントを指示します。

複数の SSA を使用する GNP は、ユーザーが希望するセグメントへの階層経路を定義します。設定された親レベルよりも上のレベルにあるセグメントに関して SSA を指定する場合、それらの SSA は、そのレベルの現在位置で満たされなければなりません。それらの SSA が現在位置で満たされない場合、GE 状況コードが戻され、既存の位置は変更されません。最後の SSA は、設定された親レベルよりも低いセグメントに関するものでなければなりません。そのようになっていない場合には、GP 状況コードが戻されます。複数の非修飾 SSA は、指定したセグメント・タイプの最初のオカレンスを、ユーザーが希望する経路の一部として設定します。GNP 呼び出しで要求されたセグメントと親との間に欠落している SSA がある場合、それらの SSA は非修飾 SSA として内部的に生成されます。すなわち、IMS が、欠落 SSA からのセグメントの最初のオカレンスを、ユーザーが要求したセグメントへの階層経路の一部として含むことになります。

使用法: Get Hold Next in Parent (GHNP)

GHNP 呼び出しの検索は、GHN 呼び出しの場合と同じです。

関連概念:

2 次索引がユーザー・プログラムに与える影響 (アプリケーション・プログラミング)

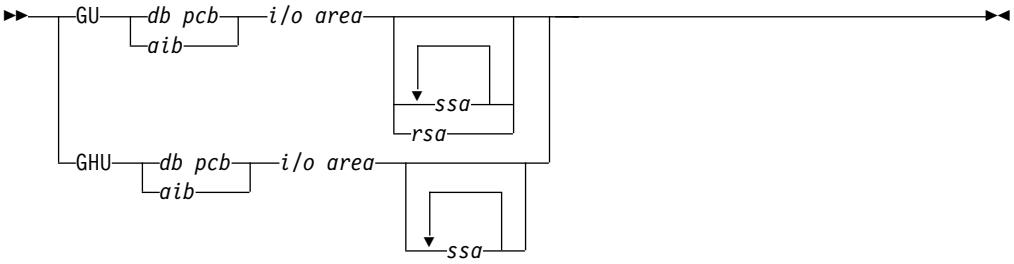
関連資料:

13 ページの『GN/GHN 呼び出し』

GU/GHU 呼び出し

Get Unique (GU) 呼び出しは、セグメントを直接検索して、データベース内で順次処理のための開始点を設定するために使用します。Get Hold Unique (GHU) は、GU 呼び出しに対応する保留形式です。

フォーマット



	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
全機能の場合:	GU/GHU	X	X		X	
GSAM の場合:	GU	X	X	X	X	X
DEDB の場合:	GU	X	X	X		
MSDB の場合:	GU	X				

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
入出力域を指定します。このパラメーターは出力パラメーターです。いずれかの Get 呼び出しが成功すると、IMS は要求されたセグメントをこの区域に戻します。ユーザーのプログラムがなんらかのパス呼び出しを出す場合には、入出力域は、パス呼び出しの後の連結セグメントの最長の経路を収めるのに十分な長さでなければなりません。この区域に入るセグメント・データは、常に左寄せされます。入出力域は、この区域の最初のバイトを指します。

GSAM で GU 呼び出しを使用すると、*i/o area* パラメーターで指定された区域に検索対象のレコードが入ります。

ssa
呼び出しで使用したい SSA があれば、それを指定します。 このパラメーターは入力パラメーターです。呼び出しで提供する SSA は、その呼び出しのための SSA が定義されている、ユーザーのプログラム内のデータ域を指します。このパラメーターでは、SSA を最大 15 個使用できます。 GU 呼び出しの場合には、このパラメーターはオプションです。

rsa
ユーザーのプログラム内の、レコード検索引数が入っている区域を指定します。この必須入力パラメーターは、GSAM のみで使用されます。RSA の詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『GSAM データ域』を参照してください。

使用法: Get Unique (GU)

GU は、ユーザーが指定した SSA で記述されているセグメントに関する要求です。この呼び出しは、特定のセグメントが必要なとき使用してください。また、データベース内のユーザー位置を設定するために使用することもできます。

GU 呼び出しは、データベース内で逆方向に位置を設定するための唯一の方法です。(F コマンド・コードとともに GN および GNP 呼び出しを使用すると、データベース内で後退することができますが、制約があります。GN や GNP とは異なり、GU 呼び出しはデータベース内で自動的に前方へ移動することはありません。

同じ GU 呼び出しを繰り返して出した場合、IMS はそのたびに同じセグメントを検索します。特定のセグメントだけを検索したい場合には、GN の代わりに、それらのセグメントに関する完全な修飾を行った GU を使用してください。特定のセグメント・オカレンスを検索したり、データベース内の特定の位置を取得したりするためには、GU を使用してください。

特定のセグメントを検索したり、データベース内のユーザー位置を特定の場所に設定したりしたい場合には、一般には、修飾した GU 呼び出しを使用します。GU 呼び出しでは、DB PCB により定義された階層内の既存レベルと同数の SSA を使用できます。必要なセグメントが階層の 4 番目のレベルにある場合には、4 つの SSA を使用してこのセグメントを検索できます。 (階層のレベル数を超える数の SSA が必要になることは考えられません。ユーザーの階層が 3 つのレベルだけからなる場合、3 番目のレベルよりも低位のセグメントを探す必要はありません。) 以下は、SSA を伴う GU 呼び出しについての補足情報です。

- ルート・レベルで非修飾 SSA を使用した GU 呼び出しは、データベースの最初から始めて、その呼び出しを満たそうとします。ルート・レベルの SSA が唯一の SSA である場合には、IMS はデータベース内の最初のセグメントを検索します。
- 修飾された SSA を使用する GU 呼び出しは、オブジェクト・セグメントの現在位置に対するオフセットとは無関係に、SSA で記述されたセグメントを検索できます。
- 修飾された SSA と非修飾 SSA が各レベルで混在している GU 呼び出しを出すと、IMS は、その呼び出しを満たすセグメント・タイプの最初のオカレンスを検索します。
- 複数の SSA を使用する GU 呼び出しのレベルの 1 つに SSA を指定しない場合には、IMS はそのレベルに関する SSA があるものと想定します。IMS が想定する SSA は、現在位置によって異なります。
  - IMS の位置が欠落レベルで設定されている場合には、IMS が使用する SSA は、DB PCB に反映されるように、その位置から取り込まれます。
  - IMS の位置が欠落レベルで設定されていない場合には、IMS はそのレベルに関して非修飾 SSA を想定します。
  - IMS が、より高いレベルで設定された位置から前進した場合、IMS はそのレベルに関して非修飾 SSA を想定します。
  - ルート・レベルの SSA が抜けている場合に IMS の位置がルート・レベルで設定されているときには、IMS は、呼び出しを満たそうとする際にそのルートから移動しません。

使用法: Get Hold Unique (GHU)

ユーザーのプログラムは、セグメントの削除または置き換えを実行する前に、そのセグメントを検索し、なんらかの方法でセグメントを変更することを IMS に通知しなければなりません。プログラムはこれを実行するために、セグメントの削除または置き換えに先立って、『hold』 を指定する Get 呼び出しを出します。 Get Hold 呼び出しによるセグメントの検索が成功すると、プログラムはセグメントを削除したり、そのセグメント内の (キー・フィールド以外の) 1 つ以上のフィールドを変更したりできます。

Get Hold 呼び出しとそれ以外の Get 呼び出しの違いは、Get Hold 呼び出しのあとでは REPL または DLET 呼び出しを出せるという 1 点です。

検索されたセグメントの保留状況は取り消されるため、 DLET または REPL 呼び出しを再び出す前に保留状況を再設定する必要があります。Get Hold 呼び出しを出したあとで、同一の PCB に対して他の呼び出しを介在して出さない場合には、そのセグメントに対して複数の REPL または DLET 呼び出しを出すことができます。

Get Hold 呼び出しを出してから、そのセグメントを更新する必要がないことが分かった場合には、そのセグメントを解放しないで、引き続き他の処理を行うことができます。このセグメントは、現在位置が変更されると (その Get Hold 呼び出しに使用された PCB に対して別の呼び出しを出したときに)、ただちに解放されます。すなわち、REPL や DLET 呼び出しの前には必ず Get Hold 呼び出しを出す必要があります。ただし、Get Hold 呼び出しを出しても、そのセグメントの置き換えまたは削除を実行しなければならないわけではありません。

制約事項

GU を使用して GSAM データベースで提供した RSA を持つレコードを検索できます。ただし、GHU は GSAM に対して有効ではありません。

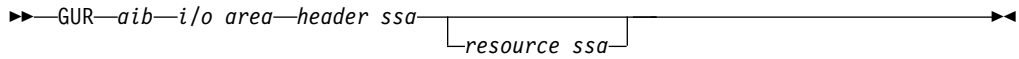
関連概念:

250 ページの『F コマンド・コード』

GUR 呼び出し

初回レコード取り出し (GUR) 呼び出しは、IMS カタログ・データベースからレコード全体を取得するために使用されます。レコードは、XML インスタンス文書として返されます。

フォーマット



パラメーター

aib

呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID

目印。この 8 バイトのフィールドには、DFSAIB*bb* を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。この 8 バイトの左寄せフィールドには、DB PCB の名前を指定する必要があります。

AIBOALEN

入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

AIBRTKN

AIB 戻りトークン。この 8 バイトのフィールドには、GUR 呼び出しが入出力域に収まるより多くのデータを戻したときにトークン値が入ります。後続の GUR 呼び出しを出すときにこのフィールドを戻り値に設定すれば、残りのデータを取得できます。IMS は次のデータを戻し、ユーザーはすべてのデータが取得されるまで AIBRTKN フィールドの設定を続行して、順次呼び出しを出し続けることができます。

AIBRTKN フィールドがゼロでない場合、その呼び出しについてすべての SSA が無視されます。

無効なトークン値または認識されないトークン値を指定した場合、呼び出しは失敗します。

AIBOAUSE
GUR 呼び出しで戻される XML インスタンス文書の合計長さを指定します。この値は、GUR 呼び出しが正常に実行された後で IMS によって設定されます。この値はバイト単位です。

AIBOAUSE フィールドの値が AIBOALEN フィールドの値より小さい場合、アプリケーション・プログラムは XML 文書全体を入出力域から取得することができます。

AIBOAUSE フィールドの値が AIBOALEN フィールドの値より大きい場合は、アプリケーション・プログラムは XML インスタンス文書全体を取得するために、最初の呼び出しの戻りトークン値に AIBRTKN 値を設定して追加の GUR 呼び出しを行う必要があります。

リンク・シリーズの最後の GUR 呼び出しのサイズは、残りのデータのサイズに一致しない場合があります。例えば、AIBOALEN=4096 の要求に対して 9000 バイトのデータを戻す GUR 呼び出しの場合、すべてのデータを取得するためには 3 回のリンク GUR 呼び出しを必要とします。3 回目の呼び出しでは、808 バイトのデータのみが入出力域に戻されます。

AIBOAUSE 値はリンク・シリーズのすべての GUR 呼び出しに対して戻され、常に XML インスタンス文書の合計サイズを反映します。

AIBRETRN
戻りコード

AIBREASN
理由コード。

i/o area
呼び出しで戻された XML インスタンス文書を IMS が置く入出力域を指定します。このパラメーターは出力パラメーターです。呼び出しが成功すると、IMS は要求されたレコードをこの領域に戻します。この領域に入る XML インスタンス文書は常に左寄せされます。入出力域パラメーターはこの領域の最初のバイトを指します。

header ssa

検索対象の HEADER セグメントの名前を指定します。このパラメーターは必須です。

resource ssa

検索対象の DBD または PSB セグメントの名前を指定します。このパラメーターはオプションであり、HEADER SSA を指定した場合にのみ有効です。

IMS は、対応するリソースをカタログ内で検索するために、ACBLIB 内でアクティブ・リソース (DBD または PSB のいずれか) にタイム・スタンプを使用します。

使用法

初回レコード取り出し (GUR) 呼び出しは、IMS カタログからの完全なレコードの要求です。

カタログ・レコードは XML インスタンス文書として戻され、使用可能な入出力域より大きい場合があります。IMS は、成功した GUR 呼び出しの完全な XML イン

スタンス文書を内部検索キャッシュに保管し、それを使用可能な入出力域のサイズの各断片に分けてアプリケーション・プログラムに戻すことができます。XML インスタンス文書の別の断片を取り出す後続の GUR 呼び出しは、それぞれ、元の呼び出し後に IMS によって AIBRTKN フィールドに設定されたトークン値を使用する必要があります。

この呼び出しへの応答として戻された文書の XML スキーマは、次のように IMS.ADFSSMPL データ・セットに組み込まれます。

- DFS3XDBD.xsd (DBD レコードの場合)
- DFS3XPSB.xsd (PSB レコードの場合)

z/OS XML システム・サービスを使用して、応答文書の構文解析を行うことができます。z/OS XML パーサーは、呼び出し可能サービスとして開始されます。サービス・スタブは CSSLIB に配送されます。

GUR 呼び出し SSA は HEADER セグメントで開始する必要があります。

非修飾 SSA で実行された GUR 呼び出しは、ターゲット・データベースの始まりに開始されることによって、要求を満たそうとします。ルート・レベルの SSA が唯一の SSA である場合には、IMS はデータベース内の最初のセグメントを検索します。修飾された SSA を持つ GUR 呼び出しは、現在のカーソル一に相対的なセグメントの位置とは無関係に、SSA で記述されたセグメントを検索することができます。GUR 呼び出しで使用できる SSA 修飾の 2 つのレベルは、カタログに保管されている DBD または PSB のレベルに対応します。

IMS カタログは、各レコードのルートとしてヘッダー・セグメントを使用する構造を持っています。各ヘッダー・セグメント・インスタンスには、PSB または DBD のいずれかの子セグメント・インスタンスがあります。非修飾 GUR 呼び出し (次の例のようなもの) は予期した結果を戻さない可能性があるため、この構造を理解することが重要です。

```
GUR  HEADER
  PSB
```

この呼び出しは最初のレコードを見つけます。英数字順で DBD が PSB より先にあるため、この最初のレコードは常に DBD レコードです。最初のレコードには PSB セグメント・インスタンスが含まれていないため、呼び出しは予期した PSB レコードを戻しません。次のように、セグメント・ヘッダー・レベルで所要のレコード・タイプを修飾する必要があります。

```
GUR HEADER (TYPE = PSB     )
    PSB
```

PSB レベルまたは DBD レベルでの修飾なしに出された GUR 呼び出しは、現在 ACB ライブラリー内でアクティブなメンバーのレコードを検索します。アクティブ・メンバーに対応するカタログ・レコードが見つからない場合、呼び出しは戻りコード X'108'および理由コード X'338'を出して失敗します。このエラーは、ACB ライブラリーの非アクティブ・メンバーに 1 つ以上のカタログ・レコードが存在する場合、または現在 ACB ライブラリー内に存在しないメンバーのレコードが存在する場合に発生します。それらのレコードを取得するには、所要のカタログ・レコードに対応するメンバーの ACB 世代タイム・スタンプを判別して、そのタイム・スタンプを PSB レベルまたは DBD レベルの修飾として組み込む必要があります。

例えば、BMP255 にアクティブな ACB ライブラリー・メンバーがない場合、次の GUR 呼び出しは失敗します。

```
GUR   HEADER  (RHDRSEQ ==PSB     BMP255  )
```

非アクティブまたは削除済みの ACB ライブラリー・メンバーのレコードを取得するには、正しい ACB 世代タイム・スタンプ用の SSA 修飾を追加します。

```
GUR   HEADER  (RHDRSEQ ==PSB     BMP255  )
      PSB     (TSVERS  ==xxxxxxxxxxxxx)
```

注**:** タイム・スタンプで修飾されていない GUR 呼び出しは、アクティブ ACB ライブラリーがない環境 (バッチ領域など) では常に失敗します。

特殊な AIB 戻りコードおよび理由コード

AIB 戻りコードと理由コードの以下の組み合わせは、GUR 呼び出しで特殊な意味を持ちます。

AIBRETRN = X'000' (CALLCOMP)
AIBREASN = X'000' (CALLOK)
GUR 呼び出しが正常に完了しました。

AIBRETRN = X'100' (CALLOKWE)
AIBREASN = X'00C' (PARTDATA)
XML 応答文書が入出力域に収まりません。GUR 継続トークンが AIBRTKN フィールドに設定されています。

AIBRETRN = X'004' (CALLOKWI)
AIBREASN = X'004' (LASTSEG)
GUR 呼び出しには以前に発行された継続 GUR 呼び出しの応答データの最後の部分が含まれています。呼び出しの GUR 継続トークンは無効になりました。

AIBRETRN = X'104' (APPLERR)
AIBREASN = X'224' (INVAOITK)
呼び出しで渡された GUR 継続トークンは無効です。

AIBRETRN = X'104' (APPLERR)
AIBREASN = X'248' (INVPCBN)
IMS™ カタログにアクセスするために指定された正しい PCB 名が見つかりませんでした。

AIBRETRN = X'104' (APPLERR)
AIBREASN = X'404' (INVFUNC)
DL/I 呼び出しに指定された機能コードが、不明または無効でした。

AIBRETRN = X'108' (SYSERROR)
AIBREASN = X'338' (NOCATMBR)
要求されたカタログ・メンバーがカタログにありません。IMS はアクティブ ACBLIB メンバーのタイム・スタンプを持つメンバーを検索しましたが、一致するタイム・スタンプを持つメンバーは見つかりませんでした。

AIBRETRN = X'108' (SYSERROR)
AIBREASN = X'340' (NOGURDLI)
GUR 呼び出しが、指定された IMS カタログ・リソースをバッチ領域内で検出しませんでした。

AIBRETRN = X'108' (SYSERROR)
AIBREASN = X'342' (NOGURXML)
GUR 呼び出しは有効な XML 応答文書を作成できませんでした。

AIBRETRN = X'108' (SYSERROR)
AIBREASN = X'344' (NOGURNFD)
要求されたカタログ・メンバーがカタログにありません。

例

次のサンプル GUR では、DX41SK01 という名前の DBD のカタログ・レコードを検索しています。

```
GUR HEADER (RHDRSEQ==DBD DX41SK01)
```

制約事項

GUR 呼び出しは、IMS カタログ・データベースからレコードを検索する場合にのみ有効です。IMS カタログ・データベースが使用可能でなければなりません。

GUR 呼び出しは AIB インターフェースでのみサポートされます。

SSA コマンド・コードは許可されません。

関連概念:

IMS カタログを使用したアプリケーション・プログラミング (アプリケーション・プログラミング)

IMS カタログの概要 (データベース管理)

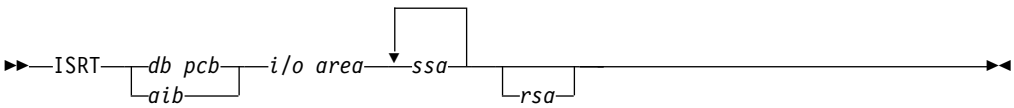
関連資料:

AIB 理由コードと戻りコード (メッセージおよびコード)

ISRT 呼び出し

挿入 (ISRT) 呼び出しは、データベースをロードしたり、データベースに 1 つ以上のセグメントを追加したりするために使用します。ISRT を使用すると、レコードを GSAM データベースの終わりに追加したり、IAFP 処理用に用意された代替 PCB として追加できます。

フォーマット



	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
全機能の場合:	ISRT	X	X		X	

	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
GSAM の場合:	ISRT	X	X	X	X	X
DEDB の場合:	ISRT	X	X	X		
MSDB の場合:	ISRT	X				

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
入出力域を指定します。 このパラメーターは入力パラメーターです。データベースに新しいセグメントを追加したい場合には、ISRT 呼び出しを出す前にこの区域に新しいセグメントを入れてください。この区域は、IMS がこの区域に戻す最長のセグメントを収めるのに十分な長さにしなければなりません。例えば、ユーザーのプログラムで検索または更新されるセグメントの中に 48 バイトを超える長さのものがない場合には、ユーザーの入出力域は 48 バイトにできます。

ユーザーのプログラムがなんらかのパス呼び出しを出す場合には、入出力域は、パス呼び出しの後の最長の連結セグメントを収めるのに十分な長さでなければなりません。この区域に入るセグメント・データは、常に左寄せされます。入出力域は、この区域の最初のバイトを指します。この区域に入るセグメント・データは、常に左寄せされます。入出力域は、この区域の最初のバイトを指します。

GSAM で ISRT 呼び出しを使用する場合、*i/o area* パラメーターで指定された区域には、ユーザーが追加したいレコードが入ります。この区域は、これらのレコードを収めるのに十分な長さでなければなりません。

ssa
呼び出しで使用したい SSA があれば、それを指定します。 このパラメーターは入力パラメーターです。呼び出しで提供する SSA は、その呼び出しのための

SSA が定義されている、ユーザーのプログラム内のデータ域を指します。この呼び出しでは、SSA を最大 15 個使用できます。このパラメーターは必須です。

rsa
ユーザーのプログラムにおける、RSA が DL/I によって戻される区域を指定します。この出力パラメーターは GSAM に対してのみ使用され、オプションです。RSA の詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『GSAM データ域』を参照してください。

使用法

ユーザーのプログラムでは、データベースを初期ロードしたり、既存のデータベースに情報を追加したりする際に ISRT 呼び出しを使用します。この呼び出しは、いずれの場合にも同じような形になります。ただし、その使用方法は、PCB の処理オプションによって決定します。

ISRT を使用すると、既存のセグメント・タイプの新しいオカレンスを HIDAM、PHIDAM、HISAM、HDAM、PHDAM、DEDB、MSDB の各データベースに追加できます。

制約事項**:** データベース全体を再処理するか、あるいはデータベースの終わりに新しいセグメントを追加するのでなければ、新しいセグメントを HSAM データベースに追加することはできません。

ISRT 呼び出しを出す前に、入出力域で新しいセグメントを作成してください。新しいセグメントのフィールドの順序と長さは、そのセグメントに関して定義された順序および長さと同じでなければなりません。(フィールド・センシティビティーを使用する場合には、セグメントのアプリケーション・プログラムのビューに定義された順序どおりになっていなければなりません。)DBD は、セグメントに含まれるフィールド、およびセグメント内でのそれらのフィールドの順序を定義します。

ルート・セグメント・オカレンス

ルート・セグメント・オカレンスを追加する場合、IMS は、ユーザーが入出力域に提供したキーを使用して、オカレンスを正しい順序でデータベースに配置します。挿入するセグメントがルート・セグメントではない場合に、その親セグメントを挿入したばかりのときには、非修飾 SSA を使用する ISRT 呼び出しによって子セグメントを挿入できます。この ISRT 呼び出しを出す前に、新しいセグメントを入出力域に作成しておく必要があります。また、ルート・セグメントを挿入する場合にも、非修飾 SSA を使用します。既存のデータベースに新しいセグメント・オカレンスを追加する場合は、DBD の中にセグメント・タイプ が定義されていなければなりません。 プログラムの入出力域に新しいセグメント・オカレンスを作成したあとは、それらを直接もしくは順次に追加することができます。 ISRT 呼び出しでは、少なくとも 1 つは SSA を使用する必要があります。挿入されるセグメントを、最後の (または唯一の) SSA で指定します。セグメントの経路を挿入するために、経路内の最高レベルのセグメントに関する D コマンド・コードをセットできます。

挿入規則

挿入対象のセグメント・タイプが固有キー・フィールドを持っている場合、IMS が新しいセグメント・オカレンスを追加する位置は、キー・フィールドの値によって決定します。セグメントがキー・フィールドをもっていない場合、またはキーが固有でない場合は、FIRST、LAST、または HERE 挿入規則のいずれかを指定することによって、新しいセグメント・オカレンスをどこに追加するかを制御することができます。 このデータベースに関する DBDGEN の SEGM ステートメントの RULES パラメーターで規則を指定してください。

RULES パラメーターで指定されるこれらの規則は、以下のとおりです。

FIRST
FIRST を指定すると、IMS は、このセグメント・タイプの最初の既存オカレンスの前に新しいセグメント・オカレンスを挿入します。このセグメントが非ユニーク・キーを持っている場合には、IMS は、そのセグメントの既存オカレンスのうちの、同じキー・フィールドを持つすべてのセグメントの前に、新しいオカレンスを挿入します。

LAST LAST を指定すると、IMS は、そのセグメント・タイプの最後の既存オカレンスのあとに新しいオカレンスを挿入します。セグメント・オカレンスが非ユニーク・キーをもっている場合、IMS は、そのセグメント・タイプの既存オカレンスのうちの、同じキーをもつすべてのオカレンスのあとに、新しいオカレンスを挿入します。

HERE HERE を指定すると、ユーザーの位置が前の IMS 呼び出しで得られたセグメント・タイプ上にあるものと IMS は見なします。IMS は、最後の呼び出しによって検索または削除されたセグメント・オカレンスの前 (すなわち、現在位置の直前) に新しいオカレンスを配置します。現在位置が、挿入されるセグメントのオカレンス内にない場合、IMS は、そのセグメント・タイプのすべての既存オカレンスの前に新しいオカレンスを追加します。セグメントが非ユニーク・キーを持ち、そのセグメント・タイプの等しいキー値のオカレンス内に現在位置がない場合、IMS は、等しいキー・フィールドを持つすべての既存オカレンスの前に、新しいオカレンスを追加します。

FIRST 挿入規則は、L コマンド・コードを使用して指定変更できます。 HERE 挿入規則は、F または L コマンド・コードを使用して指定変更できます。これが該当するのは HDAM および PHDAM のルート・セグメント、および非ユニーク・キーを持つかまたはキーをまったく持たないずべてのデータベース・タイプの従属セグメントです。

ISRT 呼び出しでは、データベースに追加されるセグメントごとに少なくとも 1 つずつの非修飾 SSA を使用しなければなりません。ISRT がパス呼び出しでない限り、最低レベルの SSA で、挿入されるセグメントを指定します。この SSA は、非修飾でなければなりません。D コマンド・コードを使用する場合には、その D コマンド・コードを含んでいる SSA とそれよりも低いレベルのすべての SSA は、非修飾でなければなりません。

挿入されるセグメントの位置を設定するために、より高いレベルに関して修飾された SSA を用意してください。セグメントに対する経路を指定するためには、修飾された SSA と非修飾 SSA を使用することができますが、最後の SSA は非修飾でなければなりません。この最後の SSA では、挿入されるセグメント・タイプを指定します。

新しいセグメント・オカレンスに関する非修飾 SSA を 1 つだけ提供する場合には、現在位置が、そのセグメントを挿入する正しいデータベースの位置になっていることを確認する必要があります。

修飾 ***SSA*** と非修飾 ***SSA*** の混在

修飾された SSA と非修飾 SSA は混在できますが、最後の SSA は非修飾でなければなりません。 SSA が非修飾の場合、IMS は、経路が正しければ、セグメント・タイプの最初のオカレンスでそれぞれの非修飾 SSA を満足させます。複数の SSA を使用する ISRT で 1 つのレベルの SSA を省略すると、IMS はそのレベルに関する SSA があるものと想定します。IMS が想定する SSA は、現在位置によって異なります。

- IMS の位置が欠落レベルで設定されている場合には、IMS が使用する SSA は、DB PCB に反映されるように、その位置から取り込まれます。
- IMS の位置が欠落レベルで設定されていない場合には、IMS はそのレベルに関して非修飾 SSA を想定します。
- IMS が、より高いレベルで設定された位置から前進した場合、IMS はそのレベルに関して非修飾 SSA を想定します。
- ルート・レベルの SSA が抜けている場合に IMS の位置がルート・レベルで設定されているときには、IMS は、呼び出しを満たそうとする際にそのルートから移動しません。

ISRT 呼び出しでの ***SSA*** の使用

ISRT で SSA を使用すると、挿入したいセグメントの親セグメントがあるかどうかをチェックできます。対応する親セグメントがデータベース内になければ、セグメントを追加することはできません。親セグメントに関する Get 呼び出しを出す代わりに、すべての親に関して完全に修飾された一連の SSA を定義して、新しいセグメントのための ISRT 呼び出しを出すことができます。IMS が GE 状況コードを戻したときには、少なくとも 1 つの親が存在していないことになります。その場合、DB PCB 内のセグメント・レベル番号をチェックして、どの親が抜けているのかを調べることができます。DB PCB 内のレベル番号が 00 になっている場合、IMS は、ユーザーが指定したどのセグメントも検出できていません。01 は、IMS がルート・セグメントだけを検出したことを意味し、02 は、IMS が検出した最低レベルのセグメントが 2 番目のレベルであることを意味します (以下同様)。

OPEN 呼び出し

OPEN 呼び出しは、GSAM データベースを明示的にオープンするために使用します。

フォーマット

	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
GSAM の場合:	OPEN	X	X	X	X	X

パラメーター

gsam pcb
呼び出しで使用する GSAM PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、GSAM PCB の PCB 名を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
オープンするデータ・セットの種類を指定します。このパラメーターは入力パラメーターです。

使用法

詳しくは、*IMS V14* アプリケーション・プログラミング のトピック「GSAM に対する明示的なオープンおよびクローズ呼び出し」を参照してください。

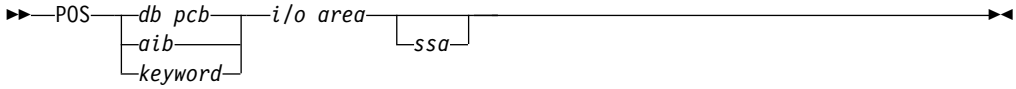
POS 呼び出し

修飾位置 (POS) 呼び出しは、特定の順次従属セグメントの位置を検索するために使用します。コミット済みセグメントに関する SSA を使用する修飾 POS 呼び出しは、位置のほかに、順次従属セグメント (SDEP) タイム・スタンプと、それを挿入した IMS 所有者の ID も戻します。

修飾 POS 呼び出しの詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『高速機能データベースの処理』を参照してください。

非修飾の POS は、順次従属セグメント (SDEP) データの論理終了を指摘します。デフォルトでは、非修飾の POS 呼び出しは、DMACNXTS 値を戻します。この値は、次に割り振られる SDEP CI です。この CI はまだ割り振られていないので、EXCLUDE キーワードを使用せずにそれを指定すると、SDEP ユーティリティーから DFS2664A メッセージが出されることがあります。

フォーマット



	呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
DEDB の場合:	POS	X	X			

パラメーター

db pcb
この呼び出しで使用する DEDB の DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
この呼び出しで使用する DEDB の AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

keyword
この呼び出しで使用する DEDB のキーワードを指定します。 フィールド・コードを含む 6 個のワードを入出力域に戻します。以下の表は、5 個のキーワードおよび対応する出力を示しています。

i/o area
ユーザーのプログラム内で、成功した POS 呼び出しにより戻された位置付け情報を入れる入出力域を指定します。このパラメーターは入力と出力の両用パラメーターです。入出力域は、戻されたすべての項目を収めるのに十分な長さでなければなりません。IMS は、DEDB 内の各区域について項目を 1 つずつ戻します。

POS 呼び出しの I/O 戻り領域には、戻される各出力ごとに 9 個の潜在的なデータ・フィールドを持つ 6 個のワードが含まれていました。各フィールドは 4 バイトか 8 バイトです。成功した POS が非修飾呼び出しの場合、入出力域にはデータ域の長さ (LL) を含む 2 バイト・フィールドが含まれ、そのあとに 24 バイトの位置付け情報が続きます。 I/O データ域には、DEDB 内のすべてのエ

リアに 24 バイトの位置付け情報があります。入力の入出力域の位置 0 にある 5 個のキーワードのいずれかを選択して、戻り入出力域のデータの種類を指定します。以下の表には、5 個のキーワードと、入力の入出力域内の位置 0 で選択されたキーワードに基づいて非修飾の POS 呼び出しが戻すデータがリストされています。

表 4. 非修飾の *POS* 呼び出し: キーワードと入出力域に戻される出力のマップ

キーワード	バイト **2**	ワード **0**	ワード **1**	ワード **2**	ワード **3**	ワード **4**	ワード **5**
<null>	LL	フィールド 1		フィールド 2		フィールド 4	フィールド 5
V5SEGRBA	LL	フィールド 1		フィールド 3		<null>	
PCSEGRTS	LL	フィールド 1		フィールド 3		フィールド 6	
PCSEGHWM	LL	フィールド 1		フィールド 3		フィールド 7	
PCHSEGTS	LL	フィールド 1		フィールド 8		フィールド 6	
PCLBSGTS	LL	フィールド 1		フィールド 9		フィールド 6	

フィールド **1**

エリア名
: この 8 バイト・フィールドには、AREA ステートメントから得られた dd 名が入ります。

フィールド **2**

次に割り振る順次従属 **CI**
: このフィールドは、入出力域の位置 0 にキーワードが指定されていない場合のデフォルトです。戻されるデータは、8 バイトの循環カウントと、グローバル DMACNXTS フィールドから獲得した RBA (CC+RBA) です。このデータは、次の事前割り振り CI を CI 境界として表します。

フィールド **3**

次のローカル順次従属セグメント
: 戻されるデータは、最も現在に近い時期にコミットされた SDEP セグメントの、8 バイトの CC+RBA セグメント境界です。このデータは、POS 呼び出しを実行する IMS に特有のものです。その有効範囲はローカル IMS 用に限られています。

フィールド **4**

順次従属部分の未使用 **CI**
: この 4 バイト・フィールドには、順次従属部分の未使用制御インターバルの数が入ります。

フィールド **5**

独立オーバーフロー部分の未使用 **CI**
: この 4 バイト・フィールドには、独立オーバーフロー部分の未使用制御インターバルの数が入ります。

フィールド **6**

順次従属セグメントのタイム・スタンプ
戻されるデータは、IMS パートナー全体で最も現在に近い時期にコミットされた SDEP セグメントの 8 バイトのタイム・スタンプであるか、またはローカル SDEP の場合は、ローカル IMS で最初に事前割り振りされた SDEP ダミー・セグメントのタイム・スタンプです。区域 (ローカル、共用いずれの場合でも) が開かれていないか、または後続の SDEP セグメントが挿入されずに /DBR が実行された場合は、現在時刻が戻されます。

フィールド **7**

順次従属最高水準点 **(HWM)**
この 8 バイト・フィールドには、循環カウントのほかに、最高水準点 (HWM) SDEP CI である最後の事前割り振り CI の RBA (CC+RBA) が含まれます。

フィールド **8**

最高位コミット済み **SDEP** セグメント
戻されるデータは、8 バイトの循環カウントのほかに、IMS パートナー全体で最も現在に近い時期にコミットされた SDEP セグメントの RBA (CC+RBA) であるか、または、ローカル SDEP の場合は、最も現在に近い時期にコミットされた、ローカル IMS の SDEP セグメントの CC+RBA です。区域 (ローカル、共用、いずれの場合でも) が開かれていないか、または後続の SDEP セグメントが挿入されずに /DBR が実行された場合は、 HWM CI が戻されます。

フィールド **9**

論理開始タイム・スタンプ
この 8 バイト・フィールドには、DMACSDEP_LOGICALBEGIN_TS フィールドから得られた論理開始タイム・スタンプが含まれます。

ssa
この呼び出しで使用したい SSA を指定します。 このパラメーターは入力パラメーターです。POS 呼び出しにおける SSA の形式は、DL/I 呼び出しにおける SSA の形式と同じです。 このパラメーターでは、1 つの SSA のみを使用できます。 POS 呼び出しの場合には、このパラメーターはオプションです。

使用法

POS 呼び出しは以下のことを行います。

- 特定の順次従属セグメントの位置を検索する。
- 最後に挿入された順次従属セグメントの位置、そのタイム・スタンプ、および IMS ID を検索する。
- 順次従属セグメントまたは論理開始のタイム・スタンプを検索する。
- 各 DEDB エリア内の未使用スペースの量をユーザーに知らせる。例えば、 POS 呼び出しに対応して IMS が戻した情報を使用して、特定の期間に関する順次従属セグメントをスキャンまたは削除できます。

POS 呼び出しで指定された区域が使用不能な場合には、入出力域は変更されず、状況コード FH が戻されます。

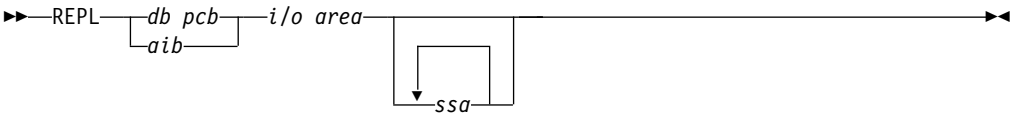
制約事項

POS 呼び出しは、DEDB でのみ使用できます。

REPL 呼び出し

置き換え (REPL) 呼び出しは、セグメント内の 1 つ以上のフィールドの値を変更するために使用します。

フォーマット



	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
全機能の場合:	REPL	X	X		X	
DEDB の場合:	REPL	X	X			
MSDB の場合:	REPL	X				

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
ユーザーのプログラム内で IMS と通信する区域を指定します。このパラメーターは入力パラメーターです。データベース内の既存のセグメントを新しいセグメントに置き換えたい場合には、最初に Get Hold 呼び出しを出して、その新しいセグメントを入出力域に収めます。入出力域のデータを変更してから、REPL 呼び出しを出してデータベース内のセグメントを置き換えることができます。

ssa

呼び出しで使用したい SSA があれば、それを指定します。 このパラメーターは入力パラメーターです。ユーザーが提供する SSA は、その呼び出しのための SSA が定義されている、ユーザーのプログラム内のデータ域を指します。このパラメーターでは、SSA を最大 15 個使用できます。REPL 呼び出しの場合には、このパラメーターはオプションです。

使用法

REPL 呼び出しの前に、3 つの Get Hold 呼び出しのうちのいずれかを出す必要があります。セグメントを検索した後で、入出力域でそれを修正し、さらに REPL 呼び出しを出してデータベース内でそれを置き換えます。IMS は、ユーザーが入出力域内で修正したセグメントによってデータベース内のセグメントを置き換えます。REPL 呼び出しを出すまでは、入出力域のセグメントのフィールド長を変更することはできません。

例えば、Get Hold 呼び出しで戻された 1 つ以上のセグメントを変更しない場合、または入出力域のセグメントを変更してもその変更をデータベースに反映させたくない場合には、IMS に対して、セグメントを置き換えないように通知できます。N コマンド・コードを使用する非修飾 SSA は、IMS に対して、そのセグメントを置き換えないように指示します。

N コマンドを使用すると、D コマンド・コードで戻された複数のセグメントのうち 1 つ以上を置き換えないように IMS に指示できます。ただし、先行の Get Hold 呼び出しで D コマンド・コードが使用されていない場合にも、N コマンド・コードを指定できます。

REPL 呼び出しには、修飾された SSA を含めないようにしてください。これを含めると、AJ 状況コードが戻されます。

ユーザーのプログラムがセグメントを正常に置き換えるためには、そのセグメントが、PCB 内の SENSEG ステートメントで PROCOPT=A または PROCOPT=R によって置き換え感知可能として定義されていなければなりません。

セグメント内のフィールドが REPL 呼び出しで変更されていない場合は、アプリケーションが別のデータベース・レコードに移動する際にロックが解除されます。セグメントをプログラムで排他的に使用できるように保存する必要がある場合は、Q コマンド・コードを使用します。

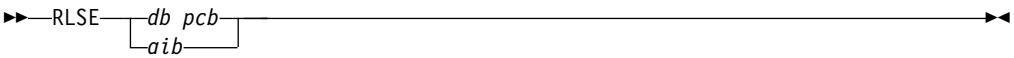
関連資料: PROCOPT オプションについての詳細は、「*IMS V14* システム・ユーティリティー」を参照してください。

ユーザーのプログラムが、置き換えセンシティビティーを持たないセグメントのパスを置き換えようとしたときに、N コマンド・コードが指定されていない場合、REPL 呼び出し用の入出力域内のセグメントに関するデータは、先行の Get Hold 呼び出しで戻されたセグメントと同じでなければなりません。この状況でデータが変更されると、ユーザーのプログラムは AM 状況コードを受け取り、REPL 呼び出しが行われてもデータは変更されません。

RLSE 呼び出し

ロック解除 (RLSE) 呼び出しは、無変更データで保持されているすべてのロックを解除するために使用されます。

フォーマット



	呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
全機能の場合:	RLSE	X	X		X	
DEDB の場合:	RLSE	X	X		X	

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、 DB PCB の名前を指定しなければなりません。

使用法

高速機能データベースの場合は、RLSE 呼び出しを使用して、アプリケーションが所有する、無変更データで保持されているすべてのロックを解除します。全機能データベースの場合は、RLSE 呼び出しを使用して、呼び出しで参照されている DB PCB によって保持されているロックを解除します。更新されたリソースがロックによって保護されている場合、そのロックは解除されません。RLSE 呼び出しを実行すると、データベース位置情報がすべて失われます。

制約事項

RLSE 呼び出しは、DB PCB を使用して発行する必要があります。PCB に入出力 PCB または MSDB PCB を指定することはできません。

IMS DB システム・サービスのための DL/I 呼び出し

以下の DL/I 呼び出しを使用して、IMS DB システム・サービスを取得します。

各呼び出しの説明には以下の内容が含まれます。

- 構文図
- その呼び出しで使用可能なパラメーターの定義
- アプリケーション・プログラムでの呼び出しの詳しい使用方法
- 呼び出しの使用に関する制約事項 (該当する場合)

各パラメーターは入力パラメーターまたは出力パラメーターとして説明されています。「入力」とは、アプリケーション・プログラムから IMS への入力のことを指します。「出力」とは、IMS からアプリケーション・プログラムへの出力のことを指します。

これらの呼び出しの構文図は、*function* パラメーターで始まります。呼び出しインターフェース (xxxTDLI) および *parmcount* (必要な場合) は、この構文図には含まれていません。

関連資料**:** アセンブラー言語、C 言語、COBOL、Pascal、および PL/I でのプログラムのコーディングについての詳細は、「*IMS V14* アプリケーション・プログラミング」のトピック『アプリケーション・プログラム・エレメントの定義』で完全な構造に関する記述を参照してください。

関連資料:

143 ページの『IMS TM システム・サービスのための DL/I 呼び出し』

94 ページの『トランザクション管理のための DL/I 呼び出し』

188 ページの『EXEC DLI コマンド』

344 ページの『IMSCALL コマンド (X'C803')』

システム・サービス呼び出しの要約

以下の表は、各タイプの IMS DB アプリケーション・プログラムで使用できるシステム・サービス呼び出しと、各呼び出しのパラメーターを要約したものです。オプション・パラメーターは、大括弧 ([ ]) で囲まれています。

例外**:** 言語依存パラメーターは、ここでは示されていません。

言語依存アプリケーション・エレメントについての詳細は、「*IMS V14* アプリケーション・プログラミング」の『アプリケーション・プログラム・エレメントの定義』を参照してください。

表 *5.* システム・サービス呼び出しの要約 :

機能コード	意味	用途/オプション	パラメーター	有効な環境
APSB	PSB 割り振り	ODBA アプリケーションに PSB を割り振る。	aib	DB/DC、IMS DB
DPSB	割り振り解除	ODBA アプリケーションの PSB を割り振り解除する。	aib	DB/DC、IMS DB

表 5. システム・サービス呼び出しの要約 (続き) :

機能コード	意味	用途/オプション	パラメーター	有効な環境
CHKP (基本)	基本チェックポイント	リカバリーに備える	function、 i/o pcb または aib、 i/o area	DB バッチ、 TM バッチ、 BMP、 MPP、 IFP
CHKP (シンボリック)	シンボリック・チェックポイント	リカバリーに備える。保管するプログラム域を 7 つまで指定する。	function、 i/o pcb または aib、 i/o area len、 i/o area [、 area len、area]	DB バッチ、 TM バッチ、 BMP
GMSG	メッセージ取り出し	メッセージを AO 出口ルーチンから検索する。何も利用できない場合は、AOI メッセージを待つ。	function、 aib、 i/o area	DB/DC と DCCTL (BMP、 MPP、 IFP)、 DB/DC と DBCTL (DRA スレッド)、 DBCTL (BMP 非メッセージ・ドリブン)、 ODBA
GSCD[1]	システム目録ディレクトリー取り出し	システム目録ディレクトリーのアドレスを獲得する。	function、 db pcb、 i/o pcb または aib、 i/o area	DB バッチ、 TM バッチ
ICMD	コマンド発行	IMS コマンドを出し、最初のコマンド応答セグメントを検索する。	function、 aib、 i/o area	DB/DC と DCCTL (BMP、 MPP、 IFP)、 DB/DC と DBCTL (DRA スレッド)、 DBCTL (BMP 非メッセージ・ドリブン)、 ODBA
INIT	アプリケーションの初期設定	データ可用性およびデッドロック発生に関する状況コードを受け取り、各 PCB データベースについてデータ可用性をチェックする。	function、 i/o pcb または aib、 i/o area	DB バッチ、 TM バッチ、 BMP、 MPP、 IFP、 DBCTL、 ODBA
INQY	照会	入出力または代替 PCB の宛先タイプ、位置、およびセッション状況に関する情報と状況コードを戻す。	function、 aib、 i/o area、 AIBFUNC=FIND|DBQUERY|ENVIRON	DB バッチ、 TM バッチ、 BMP、 MPP、 IFP、 ODBA
LOGb	ログ	システム・ログにメッセージを書き込む。	function、 i/o pcb または aib、 i/o area	DB バッチ、 TM バッチ、 BMP、 MPP、 IFP、 DBCTL、 ODBA
PCBb	プログラム連絡ブロック	もう 1 つの PSB を指定し、スケジュールする。	function、 psb name、 uibptr [、 sysserve]	CICS (DBCTL または DB/DC)

表 5. システム・サービス呼び出しの要約 (続き) :

機能コード	意味	用途/オプション	パラメーター	有効な環境
RCMD	コマンド検索	ICMD 呼び出しに起因する 2 番目以降のコマンド応答セグメントを検索する。	function、 aib、 i/o area	DB/DC と DCCTL (BMP、 MPP、 IFP)、 DB/DC と DBCTL (DRA スレッド)、 DBCTL (BMP 非メッセージ・ドリブン)、 ODBA
ROLB	ロールバック	データベースの更新を取り消し、最後のメッセージを入出力域に戻す。	function、 i/o pcb または aib、 i/o area	DB バッチ、 TM バッチ、 BMP、 MPP、 IFP
ROLL	ロール	データベースの更新を取り消す。	function	DB バッチ、 TM バッチ、 BMP、 MPP、 IFP
ROLS	SETS へのロールバック	DB PCB または入出力 PCB の名前を使用して呼び出しを発行し、データベース変更を SETS ポイントにバックアウトする。	function、 db pcb、 i/o pcb または aib、 i/o area、 token	DB バッチ、 TM バッチ、 BMP、 MPP、 IFP、 DBCTL、 ODBA
SETS/SETU	バックアウト・ポイントを設定する。	既存のすべてのバックアウト・ポイントを取り消し、9 つの中間バックアウト・ポイントを設定する。	function、 i/o pcb または aib、 i/o area、 token	DB バッチ、 TM バッチ、 BMP、 MPP、 IFP、 DBCTL、 ODBA
SNAP[2]		診断情報の収集。 SNAP オプションを選択する。	function、 db pcb または aib、 i/o area	DB バッチ、 BMP、 MPP、 IFP、CICS (DBCTL または DB/DC)、 ODBA
STAT[3]	統計	IMS システム統計の検索。タイプおよびフォーマットを選択する。	function、 db pcb または aib、i/o area、 stat function	DB バッチ、 BMP、 MPP、 IFP、 DBCTL、 ODBA
SYNC	同期	ロックされたリソースを解放し、コミット・ポイント処理を要求する。	function、 i/o pcb または aib	BMP
TERM	終了	PSB を解放し、別の PSB をスケジュールしてデータベース変更をコミットできるようにする。	function	CICS (DBCTL または DB/DC)

表 5. システム・サービス呼び出しの要約 (続き) :

機能コード	意味	用途/オプション	パラメーター	有効な環境
XRST	拡張再始動	保管する区域を 7 つまで指定する。シンボリック・チェックポイントとともに使用され、アプリケーション・プログラムを再始動する。	function、 i/o pcb または aib、 i/o area len、 i/o area [、 area len、 area]	DB バッチ、 TM バッチ、 BMP

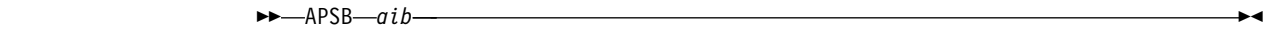
注:

1. GSCD は、プロダクト・センシティブ・プログラミング・インターフェースです。
2. SNAP は、プロダクト・センシティブ・プログラミング・インターフェースです。
3. STAT は、プロダクト・センシティブ・プログラミング・インターフェースです。

APSB 呼び出し

PSB 割り振り (APSB) 呼び出しは、ODBA アプリケーションに PSB を割り振る目的に使用されます。

フォーマット



呼び出し名	DB/DC	IMS DB	DCCTL	DB バッチ	TM バッチ
APSB	X	X			

パラメーター

aib

この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。8 バイトの左寄せされるこのフィールドには、PSB 名を入れます。

AIBRSNM2

APSB のターゲット IMS を表す ODBA 始動テーブルの 4 文字 ID です。

使用法

ODBA アプリケーションは、 ODBA アプリケーション・インターフェース AERTDLI をロードするか、それとのリンク・エディットを必要とします。

APSB 呼び出しは、どの DLI 呼び出しにも先行しなければなりません。

APSB 呼び出しは AIB を用いて、ODBA アプリケーションに PSB を割り振ります。

APSB 呼び出しのためには、その時点で z/OS リソース・リカバリー・サービス (RRS) がアクティブでなければなりません。 RRS がアクティブでないと、APSB 呼び出しは失敗し、アプリケーションに次のものが返されます。

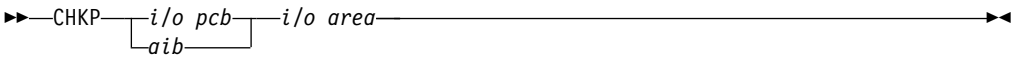
```
AIBRETRN = X'00000108'
AIBREASN = X'00000548'
```

CHKP (基本) 呼び出し

基本チェックポイント (CHKP) 呼び出しは、リカバリーを目的として使用します。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
CHKP	X	X	X	X	X

パラメーター

i/o pcb
呼び出しで使用する入出力 PCB を指定します。基本 CHKP 呼び出しは、入出力 PCB を参照しなければなりません。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
8 バイトのチェックポイント ID が入るプログラム入出力域を指定します。このパラメーターは入力パラメーターです。プログラムが MPP またはメッセージ・ドリブン BMP である場合には、 CHKP 呼び出しは次の入力メッセージを入出力域に暗黙的に戻します。そのため、この区域は、戻されたメッセージで最長のものを納めるのに十分な大きさである必要があります。

使用法

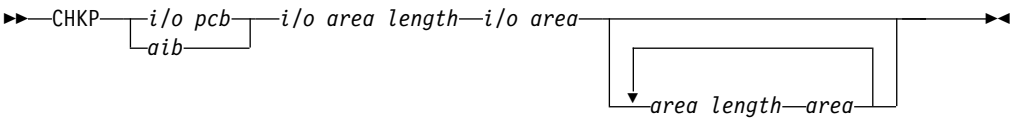
基本 CHKP は、ユーザーのプログラムがデータベースに対して行った変更をコミットし、プログラムが異常終了した場合に再始動することのできる、プログラム内の位置を設定します。

CHKP (シンボリック) 呼び出し

記号チェックポイント (CHKP) 呼び出しは、リカバリーを目的として使用します。プログラムで記号チェックポイント呼び出しを使用する場合は、XRST 呼び出しも使用する必要があります。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
CHKP	X	X	X	X	X

パラメーター

i/o pcb
呼び出しで使用する入出力 PCB を指定します。このパラメーターは入出力パラメーターです。シンボリック CHKP 呼び出しは、入出力 PCB を参照しなければなりません。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area length
このパラメーターは、IMS では使用されなくなりました。互換性を維持するために、このパラメーターを呼び出しに含めて、そこに有効なアドレスを入れなければなりません。プログラム内の任意の区域の名前を指定することにより、有効なアドレスを獲得することができます。

i/o area
このチェックポイントの 8 バイト ID が入るユーザーのプログラムの入出力域を指定します。このパラメーターは入力パラメーターです。プログラムがメッセージ・ドリブン BMP である場合、CHKP 呼び出しは、暗黙のうちに次の入力メッセージをこの入出力域に返します。そのため、この区域は、戻されたメッセージで最長のものを納めるのに十分な大きさである必要があります。

area length
チェックポイントを取る区域の長さ (2 進数) が入るユーザーのプログラムの 4 バイト・フィールドを指定します。このパラメーターは入力パラメーターです。最大 7 つの区域長を指定することができます。それぞれの区域長について、*area* パラメーターも指定する必要があります。7 つの *area* パラメーター (および対応する長さパラメーター) は、すべてオプションです。プログラムを再始動すると、IMS は、CHKP 呼び出しに指定した区域だけを復元します。

area
プログラム内で、IMS がチェックポイントをとる区域を指定します。このパラメーターは入力パラメーターです。区域は 7 つまで指定できます。 指定される各区域の前に、*area length* パラメーターを指定する必要があります。

使用法

シンボリック CHKP 呼び出しは、ユーザーのプログラムがデータベースに対して行った変更をコミットし、プログラムが異常終了した場合に再始動できる位置をプログラム内で設定します。さらに、この CHKP 呼び出しは次のことも行うことができます。

- ユーザーのプログラムが異常終了した場合、拡張再始動 (XRST) 呼び出しを使用して再始動する。
- プログラムの最大 7 つのデータ域を保管する。これは、プログラムの再始動時に復元されます。

XRST 呼び出しを CHKP 呼び出しの前に出す必要があるのは、シンボリック・チェックポイントが取ろうとしていることを IMS に示すためです。

制約事項

シンボリック CHKP 呼び出しを出すことができるのは、バッチおよび BMP のアプリケーションに限られます。

DPSB 呼び出し

DPSB 呼び出しは、IMS DB リソースの割り振り解除に用いられます。

フォーマット

呼び出し名	DB/DC	IMS DB	DCCTL	DB バッチ	TM バッチ
DPSB	X	X			

パラメーター

aib

この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。8 バイトの左寄せされるこのフィールドには、PSB 名を入れます。

AIBSFUNC

副次機能コード。このフィールドには、以下のような 8 バイトの副次機能コードのいずれかを入れる必要があります。

```
bbbbbbbb (Null)
PREPbbbb
```

使用法

DPSB 呼び出しは、z/OS アプリケーション領域で動作するアプリケーションにより、PSB の割り振りを解除する目的に使用されます。PREP 副次機能を使用しないときは、アプリケーションで DPSB を出す前に同期点処理をアクティブにしておかなければなりません。同期点処理をアクティブにするには、z/OS リソース・リカバリー・サービス (RRS) の SRRCMIT/ATRCMIT 呼び出しを使用します。これらの呼び出しについて詳しくは、「*z/OS MVS Programming: Resource Recovery*」を参照してください。

変更がまだコミットされず、あるいはロックがまだ解放されないうちに DPSB を出すと、アプリケーションには次のものが返されます。

```
AIBRETRN = X'00000104'
AIBREASN = X'00000490'
```

スレッドは終了しません。この場合、アプリケーションでは SRRCMIT 呼び出しか SRRBACK 呼び出しを出し、DPSB をやり直すとよいでしょう。

PREP 副次機能を使用すれば、前もって同期点処理をアクティブにしておかなくてもアプリケーションで DPSB を出すことができます。同期点の活動化はもっと後で行うこともできますが、必ずどこかでは行わなければなりません。

GMSG 呼び出し

メッセージ取得 (GMSG) 呼び出しは、AO 出口ルーチン (DFSAOE00 または別の AOIE タイプの出口ルーチン) からメッセージを検索するための自動化操作プログラム (AO) アプリケーション・プログラムで使用されます。

フォーマット

パラメーター

aib

この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーションが実際に取得する AIB の長さが入っている必要があります。

AIBSFUNC

副次機能コード。このフィールドには、以下の 8 バイトの副次機能コードのいずれかを入れます。

8 個のブランク (ヌル)

AIBRSNM1 フィールドで AOI トークンにこれを指定しておくと、アプリケーション・プログラムあての AOI メッセージがないとき、IMS はそれを待たずに戻ります。

WAITAOI

AIBRSNM1 フィールドで AOI トークンに WAITAOI を指定しておくと、アプリケーション・プログラムあての AOI メッセージがないとき、IMS は AOI メッセージが出るのを待ちます。この副次機能の値は、AIBRSNM1 フィールドに AOI トークンがコード化されていない場合は無効です。この場合、AIB にエラー戻りコードと理由コードが返されます。

値 WAITAOI は左揃えし、右側を 1 個のブランク文字で埋めなければなりません。

AIBRSNM1

リソース名。このフィールドには、AOI トークンまたはブランクを入れま

す。 AOI トークンは、AO アプリケーションが取り出すメッセージを識別します。このトークンは、メッセージの最初のセグメントに与えられます。メッセージが複数セグメントのメッセージの場合は、このフィールドをブランクに設定して、2 番目のセグメントから最後のセグメントまでを取り出します。 AIBRSNM1 は、8 バイトの英数字左寄せフィールドです。右側はブランクで埋められます。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。このフィールドは、IMS により変更されません。

AIBOAUSE
入出力域に返されるデータの長さ。このパラメーターは出力パラメーターです。

入出力域に十分な大きさがないため、データの一部が返された場合、AIBOAUSE にすべてのデータを取り出すために必要な長さが入っており、AIBOALEN にデータの実際の長さが入っています。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。この入出力域は、IMS から AO アプリケーション・プログラムに渡される最大セグメントを収めるのに十分な長さがなければなりません。入出力域の大きさが、データをすべて収めるのに十分でない場合、IMS はデータの一部だけを返します。

使用法

GMSG は、AOI トークンに関連するメッセージを検索するために、AO アプリケーション・プログラム内で使用されます。AO アプリケーション・プログラムは、メッセージの最初のセグメントを検索するために、 8 バイトの AOI トークンを IMS に渡さなければなりません。IMS は、AOI トークンを使用して、タイプ AOIE の AO 出口ルーチンからのメッセージを、AO アプリケーション・プログラムから出される GMSG 呼び出しに関連付けます。IMS は、AOI トークンに関連するメッセージのみをアプリケーション・プログラムに戻します。さまざまな AOI トークンを使用することによって、AOIE タイプの出口ルーチンはさまざまな AO アプリケーション・プログラムにメッセージを送ることができます。インストール・システムでは、AOI トークンを定義するように注意してください。

複数セグメント・メッセージの 2 番目から最後までのセグメントを取り出すには、トークンを指定せずに (トークンをブランクに設定) GMSG 呼び出しを発行します。メッセージのすべてのセグメントを取り出す場合、すべてのセグメントが取り出されるまでに GMSG 呼び出しを発行する必要があります。AOI トークンを指定する新規 GMSG 呼び出しが出されると、IMS は複数セグメント・メッセージのうち、検索されなかったすべてのセグメントを廃棄します。

ユーザーの AO アプリケーション・プログラムは、GMSG 呼び出し上での待機を指定できます。関連する AOI トークン用のメッセージが現行では入手できない場合、メッセージが入手されるまで、ユーザーの AO アプリケーション・プログラムは待機状態になります。待機の決定は AO アプリケーション・プログラムが指定します。これは、待機がトランザクション定義で指定される WFI トランザクション

とは異なっています。待機は呼び出しベースで実行されます。すなわち、単一アプリケーション・プログラム内では、GMSG 呼び出しが待機を指定できます。その他の呼び出しは待機を指定できません。以下の表は、IMS 環境において GMSG を出すことのできる AO アプリケーション・プログラムの型を示しています。GMSG はまた、CPI-C ドリブン・プログラムからもサポートされます。

表 *6.* アプリケーション領域タイプ別の *GMSG* サポート

	IMS 環境		
アプリケーション領域のタイプ	**DBCTL**	**DB/DC**	**DCCTL**
DRA スレッド	あり	あり	N/A
BMP (非メッセージ・ドリブン)	あり	あり	あり
BMP (メッセージ・ドリブン)	N/A	あり	あり
MPP	N/A	あり	あり
IFP	N/A	あり	あり

制約事項

CPI-C ドリブン・プログラムは、GMSG を出す前に割り振り PSB (APSB) 呼び出しを出さなければなりません。

GSCD 呼び出し

システム目録ディレクトリー獲得 (GSCD) 呼び出しは、IMS システム目録ディレクトリーのアドレスを検索するためにバッチ・プログラムの中で使用します。

このトピックにはプロダクト・センシティブ・プログラミング・インターフェース情報が含まれています。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット

呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
GSCD				X	X

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

i/o pcb
呼び出しで使用する入出力 PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、PCB 名、IOPCBbbb (I/O PCB 使用時)、または DB PCB (DB PCB 使用時) を指定する必要があります。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
入出力域を指定します。この入出力域は 8 バイト長でなければなりません。IMS は、最初の 4 バイトにシステム目録ディレクトリー (SCD) のアドレスを入れ、2 番目の 4 バイトにプログラム仕様テーブル (PST) のアドレスを入れます。このパラメーターは出力パラメーターです。

使用法

プログラムが出した GSCD 呼び出しが成功した後では、IMS はプログラムに状況コードを返しません。同じ PCB を使用した、直前の呼び出しからの状況コードは、PCB に未変更のまま残っています。

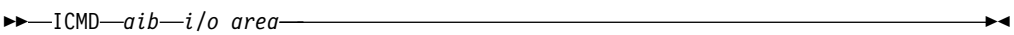
制約事項

GSCD 呼び出しは、バッチ・アプリケーション・プログラムの中からのみ出すことができます。

ICMD 呼び出し

コマンド発行 (ICMD) 呼び出しを使用すると、自動化操作プログラム (AO) アプリケーション・プログラムは IMS コマンドを出し、最初のコマンド応答セグメントを検索できます。

フォーマット



パラメーター

aib
この呼び出しで使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。このフィールドは、IMS により変更されません。

AIBOAUSE
入出力域に戻されるデータの長さ。このパラメーターは出力パラメーターです。

ユーザー・プログラムでは、このフィールドで ICMD 呼び出しによって入出力域にデータが返されたかどうかを検査する必要があります。このコマンドに対する唯一の応答が DFS058 メッセージであり、このコマンドが進行中または終了済みであると示す場合は、応答は戻されません。

入出力域に十分な大きさがないため、データの一部が返された場合、AIBOAUSE にすべてのデータを取り出すために必要な長さが入っており、AIBOALEN にデータの実際の長さが入っています。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは入出力パラメーターです。入出力域は、AO アプリケーション・プログラムから IMS に渡される最大のコマンド、あるいは IMS から AO アプリケーション・プログラムに渡される最大のコマンド応答セグメントが入るのに十分な大きさがなければなりません。入出力域の大きさが、データをすべて収めるのに十分でない場合、IMS はデータの一部だけを返します。

使用法

ICMD を使用すると、AO アプリケーションで IMS コマンドを発行することができ、コマンドの応答の最初のセグメントを取り出すことができます。

ICMD を使用する場合は、出すべき IMS コマンドをアプリケーション・プログラムの入出力域に入れてください。IMS はそのコマンドの処理を終了した後、応答メッセージの最初のセグメントを AO アプリケーション・プログラムの入出力域に戻します。後続のセグメントを検索するには (一度に 1 つのセグメント)、 RCMD 呼び出しを使用します。

IMS コマンドが正常に終了すると、その完了を示す DFS058 メッセージが出されます。IMS コマンドが非同期的に処理されると、コマンドが処理中であることを示す DFS058 メッセージが出されます。ICMD 呼び出しで入力されるコマンドの場合、DFS058 というメッセージは、AO アプリケーション・プログラムに戻されません。その場合、AIBOAUSE フィールドはゼロにセットされ、どのセグメントも戻されなかったことが示されます。その結果、ユーザーの AO アプリケーション・プログラムは、戻りコードと理由コードとともに AIBOAUSE フィールドをチェックし、応答が戻されたどうかを判別します。

関連資料: AOI 出口についての詳細は、「*IMS V14* 出口ルーチン」を参照してください。

以下の表は、IMS 環境において ICMD を出すことのできる AO アプリケーション・プログラムの型を示しています。ICMD はまた、CPI-C ドリブン・プログラムからもサポートされます。

表 *7.* アプリケーション領域タイプ別の *ICMD* サポート

	IMS 環境		
アプリケーション領域のタイプ	**DBCTL**	**DB/DC**	**DCCTL**
DRA スレッド	あり	あり	N/A
BMP (非メッセージ・ドリブン)	あり	あり	あり
BMP (メッセージ・ドリブン)	N/A	あり	あり
MPP	N/A	あり	あり
IFP	N/A	あり	あり

ICMD 呼び出しで出せるコマンドのリストについては、「*IMS V14* オペレーションおよびオートメーション」を参照してください。

制約事項

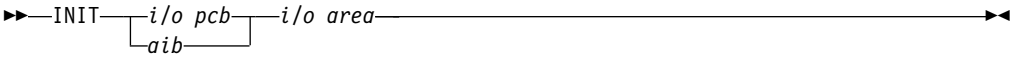
ICMD を出す前に、CPI-C ドリブン・プログラムは割り振り PSB (APSB) 呼び出しを出さなければなりません。

INIT 呼び出し

初期設定 (INIT) 呼び出しを使用すると、(各 DB PCB を検査することによって) アプリケーションはデッドロックの発生とデータ可用性に関する状況コードを受け取ることができます。

GSAM データベースの場合、初期設定 (INIT) 呼び出しを使用して、ラージ・フォーマット・データ・セットのレコードを検索するときにプログラムが 12 バイトのレコード検索引数 (RSA) を受け入れることができると IMS に伝えることができます。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
INIT	X	X	X	X	X

パラメーター

i/o pcb
呼び出しで使用する入出力 PCB を指定します。INIT は、入出力 PCB を参照しなければなりません。このパラメーターは入出力パラメーターです。

aib

呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN

入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area

どの INIT 機能が要求されたのかを示す 1 つまたは複数の文字ストリングが入っている、ユーザーのプログラム内の入出力域を指定します。このパラメーターは入力パラメーターです。

指定できる機能には、以下のものがあります。

- DBQUERY
- RSA12
- STATUS GROUPA
- STATUS GROUPB
- VERSION

使用法

この呼び出しは、共用環境の IMS バッチを含む任意のアプリケーション・プログラムで使用できます。

アプリケーション・プログラム内の機能を、入出力域に文字ストリングで指定してください。

例えば、LLZZ 文字ストリングの形式を使用してください。ここで、LL は文字ストリングの長さで、LLZZ 部分の長さを含みます。ZZ は 2 進数のゼロでなければなりません。PL/I の場合、LL フィールドはフルワードとして定義する必要があります。この値は、LLZZ 部分の長さを含む文字ストリングの長さから 2 を引いた値です。この入出力域が無効の場合には、AJ 状況コードが戻されます。以下の表には、アセンブラー言語、COBOL、C 言語、Pascal、および PL/I で使用される場合の INIT の入出力域の例を示しています。

データベース可用性の判別*: **INIT DBQUERY***

INIT 呼び出しを、入出力域に DBQUERY 文字ストリングを指定して発行すると、アプリケーション・プログラムで各 PCB のデータの使用可能性に関する情報を入手することができます。

言語非依存 AIB インターフェース、またはアセンブラー、COBOL、C、または Pascal のプログラミング言語の言語固有インターフェースを使用するアプリケーション・プログラムは、2 バイトの LL フィールドを使用して、入出力域の長さを指定します。以下の表は、LLZZ の長さフィールドおよび DBQUERY が指定された INIT 呼び出し入出力域の例を示しています。

表 *8. AIB*、*ASMTDLI*、*CBLTDLI*、*CTDLI*、および *PASTDLI* の各インターフェースの *INIT DBQUERY* 例

L	L	Z	Z	文字ストリング
00	0B	00	00	DBQUERY

注: X'0B' の LL 値は、10 進数の 11 を 16 進数で表した値です。ZZ フィールドは 2 進数です。

以下の表は、PL/I で DBQUERY を指定した INIT 呼び出しの入出力域の例を示します。PLITDLI インターフェースでは、入出力域の長さに 4 バイトの LLLL フィールドを使用します。

表 *9. INIT DBQUERY: PLITDLI* の場合の入出力域の例

L	L	L	L	Z	Z	文字ストリング
00	00	00	0B	00	00	DBQUERY

注: X'0B' の LL 値は、10 進数の 11 を 16 進数で表した値です。ZZ フィールドは 2 進数です。

LL または **LLLL**
文字ストリングの長さに、LL のための 2 バイトを加えた長さが入る 2 バイト・フィールドです。PLITDLI インターフェースの場合には、4 バイトのフィールド LLLL を使用します。AIB インターフェース (AIBTDLI) を使用する場合の PL/I プログラムでは、2 バイト・フィールドだけが必要となります。

ZZ 2 進数ゼロが入る 2 バイト・フィールドです。

各データベースの PCB について、以下のいずれかの状況コードが戻されます。

NA この PCB を使用してアクセスできるデータベースのうちの少なくとも 1 つが使用できない。この PCB を使用して呼び出しを行うと、おそらく、INIT STATUS GROUPA 呼び出しが発行されている場合は BA または BB の状況コードが返され、発行されていない場合は DFS3303I メッセージが出されて 3303 疑似異常終了になります。動的割り振りが失敗したためにデータベースが使用できない場合は例外です。この場合、呼び出しを発行すると、状況コード AI (オープン不能) が返されることになります。

DCCTL 環境では、状況コードは常に NA です。

NU この PCB を使用して更新できるデータベースのうちの少なくとも 1 つが更新に使用することができない。この PCB を使用して ISRT、DLET、または REPL 呼び出しを発行すると、INIT STATUS GROUPA 呼び出しが出されていれば状況コード BA が、出されていなければ DFS3303I メッセージが送られ、3303 の疑似異常終了が発生します。状況コード NU の原因となった

データベースは、削除処理の場合のみ必要になる場合があります。その場合、DLET 呼び出しは失敗しますが、ISRT 呼び出しおよび REPL 呼び出しは成功します。

bb この PCB でアクセスできるデータは、PCB が許容するすべての機能に使用可能です。DEDB および MSDB の状況コードは常に bb です。

データ可用性状況に加えて、ルート・セグメントのデータベース編成の名前が、PCB のセグメント名フィールドに返されます。セグメント名フィールドにはデータベース編成が入ります。DEDB、MSDB、GSAM、HDAM、PHDAM、HIDAM、PHIDAM、HISAM、HSAM、INDEX、SHSAM、SHISAM のいずれかです。

DCCTL 環境の場合、データベース編成は UNKNOWN です。

重要**:** High Availability Large Database (HALDB) を扱っている場合、データ可用性に関する PSB スケジュール時のフィードバックは、HALDB マスターが使用可能かどうかを表すだけで、HALDB 区画が使用可能かどうかを示すものではないことに注意する必要があります。ただし、HALDB 区画のデータが使用できないことを伝えるエラー設定値は、非 HALDB データベースのそれと同じであり、状況コード 'BA' か、疑似異常終了 U3303 となります。

自動 ***INIT DBQUERY***

プログラムが最初にスケジューリングされたときには、データベース PCB 内の状況コードは INIT DBQUERY 呼び出しが出されたときと同じように初期設定されます。したがって、アプリケーション・プログラムは、INIT 呼び出しを出さなくてもデータベースの可用性を判別できます。

INIT 呼び出しに関するパフォーマンスの考慮事項 ***(IMS*** オンラインのみ***)***

DCCTL 環境の場合、状況コードは NA です。

パフォーマンスの低下を防ぐために、入出力 PCB に対する最初の GU 呼び出しを出す前には INIT 呼び出しを出さないでください。INIT 呼び出しを最初に出すと、GU 呼び出しが効率よく処理されなくなります。

異常終了を発生させずにデータ可用性状況を確認する

異常終了 U3303 を回避するには、最初に INIT STATUS GROUPx (x=A or B) を使用します。 IMS から、使用不能なデータベース (または HALDB 区画) の状況コードが通知されます。その後に、INIT DBQUERY を使用して各 DB PCB に状況コードを設定します。DB 呼び出しを試行する前に、すべての PCB で非ブランクの状況を検査できます。

データ可用性状況コードを使用可能にする***: INIT STATUS GROUPA***

以下の表は、アセンブラー言語、COBOL、 C 言語、Pascal での INIT 呼び出しの入出力域の例を示します。

表 *10. ASMTDLI*、*CBLTDLI*、*CTDLI*、および *PASTDLI* の場合の *INIT* 入出力域の例

L	L	Z	Z	文字ストリング
00	11	00	00	STATUS GROUPA

表 *10. ASMTDLI*、*CBLTDLI*、*CTDLI*、および *PASTDLI* の場合の *INIT* 入出力域の例 *(*続き*)*

L	L	Z	Z	文字ストリング

注: X'11' の LL 値は、10 進数の 17 を 16 進数で表した値です。ZZ フィールドは 2 進数です。

以下の表は、PL/I での INIT 呼び出しの入出力域の例を示します。

表 *11. PLITDLI* の場合の *INIT* 入出力域の例

L	L	L	L	Z	Z	文字ストリング
00	00	00	11	00	00	STATUS GROUPA

注: X'11' の LL 値は、10 進数の 17 を 16 進数で表した値です。ZZ フィールドは 2 進数です。

LL または **LLLL**
: LL はハーフワード長フィールドです。非 PLITDLI 呼び出しの場合、LLLL は PLITDLI 用のフルワード長フィールドです。

ZZ
: 2 進数ゼロが入る 2 バイト・フィールドです。

LLZZ データまたは LLLLZZ データの値は、常に (LLZZ または LLLLZZ のための) 4 バイトにデータ長を加えた値です。

推奨事項: 本書の読者は、データ可用性について理解している必要があります。

入出力域に文字ストリング STATUS GROUPA を指定して INIT 呼び出しを出すと、アプリケーション・プログラムは IMS に対してデータの非可用性に関する状況コードを受け取るための用意ができていることを通知します。これにより、IMS は、引き続いて出される呼び出しが使用不能データへのアクセスを必要とする場合には、結果として生じる疑似異常終了ではなく状況コードを戻します。IMS が使用不能データを検出したときに戻す状況コードは BA と BB です。状況コード BA および BB はともに、アクセスする必要のあるデータが使用不能であったために呼び出しを完了できなかったことを表します。 DEDB は BA または BB 状況コードを受け取ることができます。

状況コード BA が戻されると、システムは、使用不能データを検出する前に現行呼び出しで行われた更新だけをバックアウトします。直前の呼び出しにより変更が行われている場合には、アプリケーション・プログラムは、この変更内容をコミットするかしないかを決定しなければなりません。データベースは、失敗した呼び出しが出される前の状態になります。呼び出されたのが REPL または DLET 呼び出しである場合には、PCB 位置は変化しません。呼び出しが Get タイプまたは ISRT 呼び出しである場合には、PCB 位置は予測できません。

状況コード BB が戻されると、システムは、最後のコミット・ポイント以降にプログラムが行ったすべてのデータベース更新をバックアウトし、最後のコミット・ポイント以降に送られたすべての非緊急メッセージを取り消します。どの PCB でも、データベースの開始点が PCB 位置となります。

デッドロック発生状況コードを使用可能にする*: INIT STATUS GROUPB*

以下の表は、アセンブラー言語、COBOL、 C 言語、Pascal での INIT 呼び出しの入出力域の例を示します。

表 *12. ASMTDLI*、*CBLTDLI*、*CTDLI*、および *PASTDLI* の場合の *INIT* 入出力域の例

L	**L**	**Z**	**Z**	文字ストリング
00	11	00	00	STATUS GROUPB

注: X'11' の LL 値は、10 進数の 17 を 16 進数で表した値です。ZZ フィールドは 2 進数です。

以下の表は、PL/I での INIT 呼び出しの入出力域の例を示します。

表 *13. PLITDLI* の場合の *INIT* 入出力域の例

L	**L**	**L**	**L**	**Z**	**Z**	文字ストリング
00	00	00	11	00	00	STATUS GROUPB

注: X'11' の LL 値は、10 進数の 17 を 16 進数で表した値です。ZZ フィールドは 2 進数です。

LL または **LLLL**
: LL はハーフワード長フィールドです。非 PLITDLI 呼び出しの場合、LLLL は PLITDLI 用のフルワード長フィールドです。

ZZ
: 2 進数ゼロが入る 2 バイト・フィールドです。

LLZZ データまたは LLLLZZ データの値は、常に (LLZZ または LLLLZZ のための) 4 バイトにデータ長を加えた値です。

入出力域に文字ストリング STATUS GROUPB を指定して INIT 呼び出しを出すと、アプリケーション・プログラムは IMS に対してデータの非可用性およびデッドロック発生に関する状況コードを受け取るための用意ができていることを通知します。データ非可用性に関する状況コードは、『データ可用性状況コードを使用可能にする: INIT STATUS GROUPA』 で述べたとおり BA と BB です。

INITSTATUS GROUPB 呼び出しが出されていたときにバッチでデッドロックが発生すると、次のことが起こります。

- データベースが変更されていない場合、BC 状況コードが戻されます。
- データベースが変更されている場合、データ・ログが存在していて BKO=YES が指定されていると、BC 状況コードが戻されます。
- データベースが変更されている場合、ディスク・ログが存在しないか、BKO=YES が指定されていないときには、777 疑似異常終了が発生します。

アプリケーション・プログラムがデッドロックの発生を検出すると、IMS は次のことを行います。

- (GSAM および DB2® を除く) すべてのデータベース・リソースを最後のコミット・ポイントまでバックアウトします。GSAM PCB は純バッチまたは BMP 環境用に定義することができますが、GSAM の変更はバックアウトされません。DB2 の場合にデータベース・リソースがバックアウトされるのは、IMS が同期点コーディネーターであるときだけです。

純バッチ環境で INIT STATUS GROUPB を使用するときには、 DISKLOG および BACKOUT オプションを指定しなければなりません。

- すべての出力メッセージを最後のコミット・ポイントにバックアウトします。
- すべての入力メッセージを次のようにキューに入れなおします。

  環境　処置

  MPP および **BMP**
  すべての入力メッセージがメッセージ・キューに戻されます。アプリケーション・プログラムはその入力メッセージを制御しなくなります。

  IFP　すべての入力メッセージが IMS 高速機能 (IFP) 平衡グループ・キュー (BALGRP) に戻され、 BALGRP 上のその他の IFP 領域で使用可能になります。デッドロック状態になった IFP は、BALGRP で使用可能な次のトランザクションまたはメッセージを受け取ります。

  DBCTL
  処置の対象は、DBCTL (例えば、データベース更新) によって管理されるリソースに限られます。
- データベース PCB に BC 状況コードを入れてプログラムに戻します。

ラージ・フォーマット・データ・セット用の *GSAM* データベースの判別*: INIT RSA12*

入出力域で文字ストリング 『RSA12』 を設定して INIT 呼び出しを実行すると、GSAM アプリケーション・プログラムは、ラージ・フォーマット・データ・セットのレコードを検索するときにプログラムが 12 バイトの RSA を受け入れることができると IMS に伝えます。次の表は、アセンブラー言語、COBOL、C 言語、および Pascal で RSA12 を指定した INIT 呼び出しの入出力域のサンプルを示しています。

表 *14. INIT RAS12: ASMTDLI*、*CBLTDLI*、*CTDLI*、および *PASTDLI* の場合の例

L	**L**	**Z**	**Z**	文字ストリング
00	09	00	00	RSA12

注: X'09' の LL 値は 10 進数 9 の 16 進表記です。ZZ フィールドは 2 進数です。

以下の表は、PL/I で RSA12 を指定した INIT 呼び出しの入出力域の例を示します。

表 *15. INIT RSA12: PLITDLI* の例

L	**L**	**L**	**L**	**Z**	**Z**	文字ストリング
00	00	00	09	00	00	RSA12

注: X'09' の LL 値は 10 進数 9 の 16 進表記です。ZZ フィールドは 2 進数です。

LL または **LLLL**
入出力域の合計長さが入る 2 バイトまたは 4 バイトのフィールド。PL/I では、LLLL フィールドの長さは、実際には 4 バイト・フィールドであっても 2 バイトと見なされます。AIBTDLI インターフェースを使用する場合、レコードの長さは「LL + ZZ + 文字ストリング」の合計の長さに等し

くなります。PLITDLI インターフェースの場合は、レコードの長さは「LLLL + ZZ + 文字ストリング」の合計長さに等しくなります。ここで LLLL は 2 バイトと見なされます。

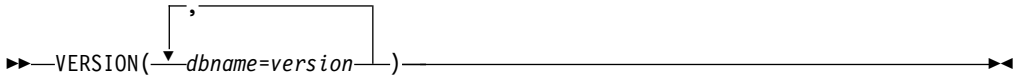
ZZ 2 進数ゼロが入る 2 バイト・フィールドです。

データベースのバージョン番号を ***INIT VERSION(dbname=version)*** で指定します。

データベースのバージョン管理が可能な場合、アプリケーション・プログラムは「VERSION」関数を使用して、データベースのバージョン (PCB 上でそのアプリケーション・プログラム用に指定されているバージョン番号または IMS によって返されるデフォルトのバージョンとは異なるもの) を要求できます。 INIT VERSION 呼び出しで指定されるバージョン番号は、他のすべてのバージョン指定およびデフォルトに優先します。

データベースにアクセスするための DL/I より前に INIT VERSION 呼び出しが発行されない場合、アプリケーション・プログラムに返されるデータベースのバージョンは、PCB ステートメントの DBVER キーワードによって決定されます。DBVER キーワードが指定されない場合、IMS は、ACB ライブラリーでアクティブなデータベースのバージョン、またはデータベースのバージョン 0 を返します。これは、PSBGEN ステートメント内、または DFSDFxxx PROCLIBメンバーのデータベース・セクション内の DBLEVEL キーワードによって決定されます。

入出力域で、VERSION 関数は次の形式を使用して指定されます。



各データベース名は英字を使用して指定され、指定可能な回数は 1 回のみです。指定するのは物理データベースの名前のみです。論理データベースの名前はサポートされていません。

各バージョンは 0 から 2147483647 までの数値として指定されます。指定される番号は、指名されているデータベースの DBD で定義され、IMS カタログに保管されているバージョン番号と一致している必要があります。

入出力域に必要なサイズは、入力の入出力域に指定されているデータベースの数に 20 を乗算して計算します。

例えば、次の表は、アセンブラー言語、COBOL、C 言語、および Pascal での INIT VERSION 呼び出しの入出力域の例を示しています。次の表で、LL 値 X'3C' は、10 進数 60 の 16 進表記です。これは、3 つのデータベース名が入力に指定されている場合に、その出力を入出力域に保持するために必要な長さ (バイト単位) です。 ZZ フィールドは 2 進数です。

表 16. INIT VERSION: ASMTDLI、CBLTDLI、CTDLI、および PASTDLI での形式の例

L	L	Z	Z	文字ストリング
00	3C	00	00	VERSION (DBa=1,DBb=2,DBc=3)

次の表は、PL/I の場合の、VERSION を指定した INIT 呼び出しの入出力域の例を示しています。この表で、LL 値 X'3C' は 10 進数 60 の 16 進表記です。 ZZ フィールドは 2 進数です。

表 17. INIT VERSION: PLITDLI での形式の例

L	L	L	L	Z	Z	文字ストリング
00	00	00	3C	00	00	`VERSION (DBa=1,DBb=2,DBc=3)`

LL または **LLLL**
入出力域の合計長さが入る 2 バイトまたは 4 バイトのフィールド。PL/I では、LLLL フィールドの長さは、実際には 4 バイト・フィールドであっても 2 バイトと見なされます。AIBTDLI インターフェースを使用する場合、レコードの長さは「LL + ZZ + 文字ストリング」の合計の長さに等しくなります。PLITDLI インターフェースの場合、レコードの長さは「LLLL + ZZ + 出力に必要な長さ」の合計長さに等しくなります。ここで LLLL は 2 バイトと見なされます。

ZZ 2 進数ゼロが入る 2 バイト・フィールドです。

文字ストリング
入力での関数の指定。LL または LLLL に指定される長さは、出力に必要な長さ (すなわち、入力文字ストリングに指定されているデータベースごとに 20 バイトずつ) です。

制約事項

CICS 環境の機能シップの場合、ローカルおよびリモート CICS はともに DBCTL でなければなりません。

「*IMS V14* システム管理」で説明されているデッドロック発生についてよく理解しておく必要があります。

関連概念:

GSAM レコードのリトリーブと挿入 (アプリケーション・プログラミング)

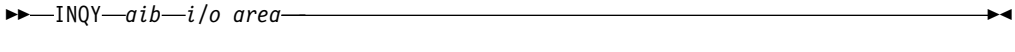
HDAM および HIDAM データベースから HALDB への変換 (データベース管理)

データの可用性に関する考慮事項 (アプリケーション・プログラミング)

INQY 呼び出し

照会 (INQY) 呼び出しは、実行環境、宛先のタイプと状況、ならびにセッションの状況に関する情報の要求に使用します。INQY は、AIB 構造を使用するアプリケーション・インターフェースのみに有効です。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
INQY	X	X	X	X	X

パラメーター

aib

呼び出しに使用するアプリケーション・インターフェース・ブロック (DFSAIB) のアドレスを指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBSFUNC
副次機能コード。このフィールドには、以下のような 8 バイトの副次機能コードのいずれかを入れる必要があります。

- bbbbbbbb (Null)
- DBQUERYb
- ENVIRONb
- FINDbbbb
- LERUNOPT
- MSGINFOb
- PROGRAMb (ODBA インターフェースではサポートされません)

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、PSB 内の名前付き PCB の名前を入れる必要があります。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。このフィールドは、IMS により変更されません。

i/o area

この呼び出しで使用するデータ出力域を指定します。このパラメーターは出力パラメーターです。INQY 副次機能の ENVIRONb、MSGINFOb、および PROGRAMb には、入出力域が必要です。副次機能 DBQUERYb、FINDbbbb、および LERUNOPT では必要ではありません。

制約事項

INQY 呼び出しは、AIB を使用している場合にのみ有効です。PCB インターフェースを使用して INQY 呼び出しを出すと、AD 状況コードによって拒否されます。

使用法

INQY 呼び出しは、バッチ環境でもオンライン IMS 環境でも使用できます。IMS アプリケーション・プログラムは INQY 呼び出しを使用して、出力の宛先、セッションの状況、現在の実行環境、データベースの可用性、および (PCB 名にもとづく)

PCB アドレスに関する情報を要求できます。INQY 呼び出しを出すときには、AIB を使用しなければなりません。INQY 呼び出しを出す前に、AIB のフィールドを初期設定してください。

INQY 呼び出しを使用する場合には、AIB で渡される 8 バイトの副次機能コードを指定します。アプリケーション・プログラムが受け取る情報は、 INQY 副次機能によって決まります。

INQY 呼び出しは、呼び出し側の入出力域に情報を返します。INQY 呼び出しから戻されるデータの長さは AIB フィールドの 1 つである AIBOAUSE の中のアプリケーションに渡されます。

入出力域のサイズは AIB フィールド AIBOALEN に指定します。INQY 呼び出しは、1 回の呼び出しでその区域に入るだけのデータを返します。区域の長さが、すべての情報を入れるのに十分ではない場合、AG 状況コードが戻され、データの一部が入出力域に戻されます。この場合、AIB フィールドの AIBOALEN には、入出力域に返されたデータの実際の長さが入っており、AIBOAUSE フィールドには、すべてのデータを受け取るために必要な出力域の長さが入っています。

データ可用性の照会: INQY DBQUERY

DBQUERY 副次機能を指定して INQY 呼び出しを発行すると、アプリケーション・プログラムは、各 PCB のデータに関する情報を入手します。AIBRSNM1 に渡せる有効な PCB 名は IOPCBbbb だけです。INQY DBQUERY 呼び出しは、INIT DBQUERY 呼び出しに似ています。INQY DBQUERY 呼び出しは、入出力域には情報を戻しませんが、INIT DBQUERY 呼び出しと同じように、データベース PCB 内の状況コードを更新します。

INQY FIND 呼び出しが実行されるまで、アプリケーション・プログラムには各 PCB の状況が知らされません。データベースの状況を検索するには、INQY FIND 呼び出しでそのデータベースの DB PCB を渡す必要があります。

INIT DBQUERY 状況コードのほかに、 INQY DBQUERY 呼び出しは、次の状況コードを入出力 PCB に戻します。

bb その呼び出しは成功し、すべてのデータベースが使用可能。

BJ PSB 内のどのデータベースも使用可能でないか、あるいは PSB に PCB が存在しない。すべてのデータベース PCB (GSAM を除く) には、INQY DBQUERY 呼び出しの処理の結果として状況コード NA が入っています。

BK PSB の少なくとも 1 つのデータベースが使用できない状態であるか、または使用できる範囲が制限されている。少なくとも 1 つのデータベース PCB には、INQY DBQUERY 呼び出しの処理の結果として NA または NU の状況コードが入っています。

INQY 呼び出しは、各 DB PCB に以下の状況コードを返します。

NA この PCB を使用してアクセスできるデータベースのうちの少なくとも 1 つが使用できない。この PCB に基づく呼び出しの結果は、おそらく、INIT STATUS GROUPA 呼び出しが出されていれば BA または BB 状況コード、出されていなければ DFS3303I メッセージと 3303 疑似異常終了となり

ます。動的割り振りが失敗したためにデータベースが使用できない場合は例外です。この場合、呼び出しを発行すると、状況コード AI (オープン不能) が返されることになります。

DCCTL 環境では、状況コードは常に NA です。

NU この PCB を使用して更新できるデータベースのうちの少なくとも 1 つが更新に使用することができない。この PCB に基づく ISRT、DLET、 REPL 各呼び出しの結果は、おそらく、INIT STATUS GROUPA 呼び出しが出されていれば BA 状況コード、出されていなければ DFS3303I メッセージと 3303 疑似異常終了になります。状況コード NU の原因となったデータベースは、削除処理の場合のみ必要になる場合があります。その場合、DLET 呼び出しは失敗しますが、ISRT 呼び出しおよび REPL 呼び出しは成功します。

bb この PCB でアクセスできるデータは、PCB が許可するすべての機能に使用することができます。DEDB および MSDB は、常に bb になります。

環境の照会: INQY ENVIRON

ENVIRON 副次機能を指定して INQY 呼び出しを発行すると、アプリケーション・プログラムは、現在の実行環境に関する情報を入手します。 AIBRSNM1 に渡せる有効な PCB 名は IOPCBbbb だけです。この中には、IMS ID、リリース、領域、および領域タイプがあります。

INQY ENVIRON 呼び出しは、文字ストリング・データを戻します。出力データは左寄せされ、右側にはブランクが埋め込まれます。

推奨事項**:** 応答データの長さの拡張に対応するには、入出力域の長さを 512 バイトに指定します。

リカバリー・トークンまたはアプリケーション・パラメーター・ストリングが入っているフィールドを参照するには、INQY ENVIRON 呼び出しのデータ出力で返されたフィールドのアドレスを使用して、フィールドの位置を指定するようにアプリケーション・プログラムをコーディングします。この方法が、リカバリー・トークン・フィールドおよびアプリケーション・パラメーター・ストリング・フィールドを参照する唯一の有効なプログラミング手法です。これらのフィールドを参照するのに、他のプログラミング手法を使用することはできません。

リカバリー・トークンまたはアプリケーション・パラメーター・ストリングはオプションであるため、常に返されるわけではありません。これらのフィールドが返されない場合、アドレス・フィールドの値はゼロになります。

リカバリー・トークン・フィールドおよびアプリケーション・パラメーター・フィールドについて詳しくは、以下の表の後に示された注 2 を参照してください。

以下の表は、INQY ENVIRON 呼び出しから戻される出力のリストです。戻される情報には、出力のバイト長、実際の値、および説明が含まれます。

表 *18. INQY ENVIRON* データ出力

返される情報	バイト長 実際の値	説明
IMS ID	8	実行パラメーターから得られた ID を示す。

表 18. INQY ENVIRON データ出力 (続き)

返される情報	バイト長	実際の値	説明
IMS リリース・レベル	4		IMS のリリース・レベルを提供する。例えば、X'00000410'。
IMS 制御領域のタイプ	8	BATCH	IMS バッチ領域がアクティブであることを示す。
		DB	IMS Database Manager だけが活動状態であることを示す。(DBCTL システム)
		TM	IMS Transaction Manager だけが活動状態であることを示す。(DCCTL システム)
		DB/DC	IMS Database Manager と IMS Transaction Manager の両方が活動状態であることを示す。(DB/DC システム)
IMS アプリケーション領域のタイプ	8	BATCH	IMS バッチ領域がアクティブであることを示す。
		BMP	バッチ・メッセージ処理領域がアクティブであることを示す。
		DRA	データベース・リソース・アダプターのスレッド領域がアクティブであることを示す。
		IFP	IMS 高速機能領域がアクティブであることを示す。
		JBP	Java バッチ処理領域がアクティブであることを示す。
		JMP	Java メッセージ処理領域がアクティブであることを示す。
		MPP	メッセージ処理領域がアクティブであることを示す。
領域 ID	4		領域 ID を提供する。例えば、X'00000001'。
アプリケーション・プログラム名	8		実行中のアプリケーション・プログラムの名前を提供する。
PSB 名 (現在割り振られているもの)	8		現在割り振られている PSB の名前を提供する。
トランザクション名	8		トランザクションの名前を提供する。
		b	関連するトランザクションがないことを示す。
ユーザー ID[1]	8		ユーザー ID を提供する。
		b	ユーザー ID が使用できないことを示す。
グループ名	8		グループ名を提供する。
		b	グループ名が使用できないことを示す。
状況グループ標識	4	A	INIT STATUS GROUPA 呼び出しが出されることを示す。
		B	INIT STATUS GROUPB 呼び出しが出されることを示す。
		b	状況グループが初期設定されないことを示す。
リカバリー・トークンのアドレス[2]	4		あとにリカバリー・トークンが続く LL フィールドのアドレスを提供する。
		0	リカバリー・トークンが使用不可であることを示す。
アプリケーション・パラメーター・ストリングのアドレス [2]	4		アプリケーション・プログラム・パラメーター・ストリングが後に続く、LL フィールドのアドレスを示す。
		0	従属領域 JCL の実行パラメーターで APARM= パラメーターがコーディングされていないことを示す。
共用キュー標識	4		IMS が共用キューを使用していないことを示す。
		SHRQ	IMS が共用キューを使用していることを示す。
アドレス・スペースのユーザー ID	8		従属アドレス・スペースのユーザー ID

表 *18. INQY ENVIRON* データ出力 (続き)

返される情報	バイト長	実際の値	説明
ユーザー ID 標識	1		ユーザー ID フィールドの内容を示す以下の可能な値のいずれかを含む。 **U** サインオン時にソース端末から得られたユーザーの ID を示す。 **L** サインオンが有効になっていない場合のソース端末の LTERM 名を示す。 **P** ソース BMP またはトランザクションの PSBNAME を示す。 **O** その他の特定の名前を示す。
z/OS リソース・リカバリー・サービス (RRS) 標識	3	b	IMS が RRS を持つ UR に利害関係があることを示さなかったことを表す。したがって、IMS がコミット有効範囲には関係しないので、アプリケーションは、RRS が UR の同期点マネージャーになるような作業を実行してはなりません。例えば、アプリケーションは、アウトバウンド保護会話を出してはなりません。
		RRS	IMS が RRS をもつ UR に利害関係があることを示したことを表す。したがって、RRS が UR の同期点マネージャーになっている場合、IMS はコミット有効範囲で使用されます。
IMS カタログ使用可能化標識	7	b	IMS カタログが DFSDFxxx PROCLIB メンバーで使用可能でないことを表す。 IMS カタログのセットアップおよび使用可能化については、IMS カタログの定義と調整 (システム定義)を参照してください。 DFSDFxxx PROCLIB メンバーでの IMS カタログの使用可能化については、IMS PROCLIB データ・セットの DFSDFxxx メンバー (システム定義)を参照してください。
		CATALOG	IMS カタログが使用可能であることを表す。データベースとアプリケーション・メタデータが IMS で使用可能です。

注:

1. ユーザー ID は、INQY ENVIRON 呼び出しを発行する領域を表す PST の PSTUSID フィールドから取り出されます。PSTUSID フィールドは、以下のいずれかになります。
   - IMS メッセージ・キューに対する GU 呼び出しを正常に完了させなかったメッセージ・ドリブン BMP 領域の場合、および非メッセージ・ドリブン BMP 領域の場合、PSTUSID フィールドは、現在 BMP 領域にスケジューリングされている PSB の名前から得られます。
   - GU 呼び出しを正常に完了させたメッセージ・ドリブン BMP 領域および MPP 領域の場合、通常は入力端末の RACF® ID の入っている PSTUSID フィールドの値が得られます。端末が RACF にサインオンされていない場合には、ID は入力端末の LTERM になります。
2. このポインターは長さフィールド (LL) を識別するアドレスであり、このフィールドには、リカバリー・トークンまたはアプリケーション・プログラム・パラメーター・ストリングの長さ (LL に必要な 2 バイトを含む) が 2 進数形式で入ります。このポインターを使用して、バッチ・プログラム内でリリース間の AIB のアドレス可能度をセットアップします。

入力メッセージ情報の照会: INQY MSGINFO

現行の入力メッセージに関する情報を取得するには、MSGINFO 副次機能で INQY 呼び出しを使用します。AIBRSNM1 フィールドに渡せる有効な PCB 名は

IOPCBbbb だけです。出力は、メッセージ情報のバージョン番号および出力フィールドを返します。 INQY MSGINFO 呼び出しは、入出力域の応答を返します。

以下の表は、INQY MSGINFO 呼び出しから戻される出力のリストです。戻される情報に含まれるのは、バイト長、実際の値、および出力の説明です。

表 *19. INQY MSGINFO* データ出力

返される情報	バイト長	実際の値	説明
バージョン番号	4	1	出力応答バージョン 1。
発信元 IMSID	8		入力メッセージの発信元の IMS ID。
IMS に予約済み	68		このフィールドは将来の出力拡張のために予約されています。

PCB の照会: INQY FIND

FIND 副次機能を指定して INQY 呼び出しを発行すると、アプリケーション・プログラムには、要求された PCB 名の PCB アドレスが返されます。AIBRSNM1 で渡すことのできる有効な PCB 名は、IOPCBbbb または PSB で定義された代替 PCB または DB PCB の名前です。PCB アドレスは AIB マスクの AIBRSA1 フィールドに返されます。INQY 呼び出しが完了すると、AIBRSA1 フィールドには呼び出し固有の情報が入ります。

データベースの状況を検索するには、INQY FIND 呼び出しでそのデータベースの DB PCB を渡す必要があります。必要な PCB ごとに 1 回の呼び出しを行う必要があります。

FIND 副次機能では、要求された PCB が修正されず、入出力域には情報が返されません。

FIND 副次機能を使用すると、INQY DBQUERY 呼び出しに続く PCB アドレスが得られます。このプロセスにより、アプリケーション・プログラムで PCB 状況コードを分析して、PCB に NA または NU の状況コードが設定されているかどうか判別することができます。

次の PL/I コード・サンプルは、データベース状況値を検索する方法を示しています。

```
II00_INITSTAT: PROC;
     DCL DUMMY_LENGTH CHAR(4) INIT(' '); /* TO PLEASE IMS */
     AIB.PCBNAME     = 'IOPCB';
     CALL AIBTDLI($3,INIT,AIB,STATUS_CALL2);
     IF AIB.RETURN = 0 THEN
        PUT SKIP LIST('INIT ISSUED');
     ELSE
       DO;
        PUT SKIP LIST ('AIB RETURN CODE   ',AIB.RETURN);
        PUT SKIP LIST ('AIB REASON CODE   ',AIB.REASON);
        PUT SKIP LIST ('IOPCB STATUS CODE ',IO_PCB.STATUS_CODE);
        PUT SKIP LIST ('INIT  UNSUCCESSFULL');
       END;
     SELECT  (IO_PCB.STATUS_CODE);
```

```
      WHEN ('  ')
        GROUPA_STATUS = '  ';
      WHEN ('NA')
        GROUPA_STATUS = 'NA';
      WHEN ('NU')
        GROUPA_STATUS = 'NU';
      OTHERWISE
        DO;
          PUT SKIP LIST
          ('INIT STATUS GROUPA FAILED ',IO_PCB.STATUS_CODE);
        END;
    END;
    PUT SKIP LIST
      ('INIT STATUS GROUPA =  ',IO_PCB.STATUS_CODE);
END II00_INITSTAT;
JJ00_INQY: PROC;
    DCL DUMMY_LENGTH CHAR(4) INIT(' '); /* TO PLEASE IMS */
    AIB.PCBNAME     = 'IOPCB';
    AIB.SUB_FUNC    = 'DBQUERY   ';
    AIB.OUT_LEN_TOT = 2000;
    CALL AIBTDLI($3,INQY,AIB,IO_AREA);
    PUT SKIP LIST('INQY ISSUED ON IOPCB BEFORE CHECK OF AIB RETURN');
    IF AIB.RETURN = 0 THEN
       PUT SKIP LIST('INQY ISSUED - 0 RC ON AIB.RETURN');
    ELSE
      DO;
       PUT SKIP LIST ('AIB RETURN CODE   ',AIB.RETURN);
       PUT SKIP LIST ('AIB REASON CODE   ',AIB.REASON);
       PUT SKIP LIST ('IOPCB STATUS CODE ',IO_PCB.STATUS_CODE);
       PUT SKIP LIST ('INQY IOPCB DBQUERY  UNSUCCESSFULL');
      END;
    SELECT  (IO_PCB.STATUS_CODE);
      WHEN ('  ')
        DO;
          PUT SKIP DATA (IO_AREA);
          PUT SKIP DATA (IO_PCB.STATUS_CODE);
        END;
      WHEN ('NA')
        PUT SKIP LIST ('NA STATUS ON IO_PCB.STATUS_CODE');
      WHEN ('NU')
        PUT SKIP LIST ('NU STATUS ON IO_PCB.STATUS_CODE');
      OTHERWISE
        DO;
          PUT SKIP LIST
          ('INQY  FAILED ',IO_PCB.STATUS_CODE);
        END;
    END;
    PUT SKIP LIST ('START B1CSTP FIND CALL');
    AIB.PCBNAME     = 'B1CSTP';
    AIB.SUB_FUNC    = 'FIND     ';
    AIB.OUT_LEN_TOT = 2000;
    CALL AIBTDLI($3,INQY,AIB,IO_AREA);
    PUT SKIP LIST('INQY B1CSTP FIND READY TO BE CALLED');
    IF AIB.RETURN = 0 THEN
       PUT SKIP LIST('INQY B1CSTP FIND CALLED  - 0 RC');
    ELSE
      DO;
       PUT SKIP LIST ('AIB RETURN CODE   ',AIB.RETURN);
       PUT SKIP LIST ('AIB REASON CODE   ',AIB.REASON);
       PUT SKIP LIST ('CSTP_PCB STATUS CODE ',CSTP_PCB.STATUS_CODE);
       PUT SKIP LIST ('INQY B1CSTP FIND  UNSUCCESSFULL');
      END;
      PUT SKIP LIST ('CSTP  STATUS  ' ,CSTP_PCB.STATUS_CODE);
      PUT SKIP LIST ('IO PCB        ', IO_PCB.STATUS_CODE);
    SELECT  (CSTP_PCB.STATUS_CODE);
      WHEN ('  ')
        DO;
```

```
      PUT SKIP DATA (CSTP_PCB.STATUS_CODE);
      PUT SKIP DATA (IO_AREA);
    END;
  WHEN ('NA')
    PUT SKIP LIST ('NA STATUS ON  B1CSTP CSTPPCB.STATUS_CODE');
  WHEN ('NU')
    PUT SKIP LIST ('NU STATUS ON  B1CSTP CSTPPCB.STATUS_CODE');
  OTHERWISE
    DO;
      PUT SKIP LIST
      ('INQY  FAILED ',IO_PCB.STATUS_CODE);
    END;
END;
PUT SKIP LIST ('START D1CSTP FIND CALL');
AIB.PCBNAME     = 'D1CSTP';
AIB.SUB_FUNC    = 'FIND     ';
AIB.OUT_LEN_TOT = 2000;
CALL AIBTDLI($3,INQY,AIB,IO_AREA);
PUT SKIP LIST('INQY D1CSTP FIND READY TO BE CALLED');
IF AIB.RETURN = 0 THEN
   PUT SKIP LIST('INQY D1CSTP FIND CALLED  - 0 RC');
ELSE
  DO;
   PUT SKIP LIST ('AIB RETURN CODE   ',AIB.RETURN);
   PUT SKIP LIST ('AIB REASON CODE   ',AIB.REASON);
   PUT SKIP LIST ('CSTP_PCB STATUS CODE ',CSTP_PCB.STATUS_CODE);
   PUT SKIP LIST ('INQY D1CSTP FIND  UNSUCCESSFULL');
  END;
  PUT SKIP LIST ('CSTP  STATUS  ' ,CSTP_PCB.STATUS_CODE);
  PUT SKIP LIST ('IO PCB        ', IO_PCB.STATUS_CODE);
SELECT  (CSTP_PCB.STATUS_CODE);
  WHEN ('  ')
    DO;
      PUT SKIP DATA (CSTP_PCB.STATUS_CODE);
      PUT SKIP DATA (IO_AREA);
    END;
  WHEN ('NA')
    PUT SKIP LIST ('NA STATUS ON  D1CSTP CSTPPCB.STATUS_CODE');
  WHEN ('NU')
    PUT SKIP LIST ('NU STATUS ON  D1CSTP CSTPPCB.STATUS_CODE');
  OTHERWISE
    DO;
      PUT SKIP LIST
      ('INQY  FAILED ',IO_PCB.STATUS_CODE);
    END;
END;
PUT SKIP LIST ('START S1CSTP FIND CALL');
AIB.PCBNAME     = 'XXCSTP';
AIB.SUB_FUNC    = 'FIND     ';
AIB.OUT_LEN_TOT = 2000;
CALL AIBTDLI($3,INQY,AIB,IO_AREA);
PUT SKIP LIST('INQY S1CSTP FIND READY TO BE CALLED');
IF AIB.RETURN = 0 THEN
   PUT SKIP LIST('INQY S1CSTP FIND CALLED  - 0 RC');
ELSE
  DO;
   PUT SKIP LIST ('AIB RETURN CODE   ',AIB.RETURN);
   PUT SKIP LIST ('AIB REASON CODE   ',AIB.REASON);
   PUT SKIP LIST ('CSTP_PCB STATUS CODE ',CSTP_PCB.STATUS_CODE);
   PUT SKIP LIST ('INQY S1CSTP FIND  UNSUCCESSFULL');
  END;
  PUT SKIP LIST ('CSTP  STATUS  ' ,CSTP_PCB.STATUS_CODE);
  PUT SKIP LIST ('IO PCB        ', IO_PCB.STATUS_CODE);
SELECT  (CSTP_PCB.STATUS_CODE);
  WHEN ('  ')
    DO;
      PUT SKIP DATA (CSTP_PCB.STATUS_CODE);
```

```
      PUT SKIP DATA (IO_AREA);
    END;
  WHEN ('NA')
    PUT SKIP LIST ('NA STATUS ON  S1CSTP CSTPPCB.STATUS_CODE');
  WHEN ('NU')
    PUT SKIP LIST ('NU STATUS ON  S1CSTP CSTPPCB.STATUS_CODE');
  OTHERWISE
    DO;
      PUT SKIP LIST
      ('INQY  FAILED ',IO_PCB.STATUS_CODE);
    END;
END;
```

LE オーバーライドの照会: INQY LERUNOPT

LERUNOPT 副次機能を指定して LERUNOPT 呼び出しを発行すると、IMS は LE オーバーライドが LEOPT システム・パラメーターに基づいて許可されているかどうか判別します。 LE オーバーライド・パラメーターは、 UPDATE LE コマンドで IMS に定義されます。IMS は、発呼者の環境におけるトランザクション名、lterm 名、ユーザー ID、またはプログラム名の特定の組み合わせに基づいて、発呼者に該当するオーバーライドがあるかどうかをチェックします。IMS は、オーバーライド・パラメーターが検索された場合、発呼者にストリングのアドレスを戻します。IMS 提供の CEEBXITA 出口、DFSBXITA は、LE ランタイム・パラメーター用の動的オーバーライドを許可するため、LE オーバーライドを使用します。

呼び出しストリングは、機能コードおよび AIB アドレスを含まなければなりません。入出力域は必要パラメーターではなく、指定した場合は無視されます。AIBRSNM1 に渡せる有効な PCB 名は IOPCB だけです。 AIBOALEN および AIBOAUSE フィールドは使用されません。

DL/I INQY LERUNOPT 呼び出しで戻されることになる項目、マッチングするルールは、以下の通りです。

- MPP または JMP 領域は、トランザクション名、LTERM、ユーザー ID、およびプログラムを使用して、各項目とマッチングします。
- IFB、JBP、または非メッセージ・ドリブン BMP は、それぞれのエントリーとマッチングするためにプログラム名を使用します。項目に、トランザクション名、LTERM、またはユーザー ID に対する定義されたフィルターがある場合は、その項目は一致しません。 メッセージ・ドリブン BMP もまたトランザクション名を使用します。
- 複数のエントリーがスキャンされ、フィルターに掛けた際のマッチングが最も多かったエントリーが検出されます。フィルターに掛けた際のマッチングが最も正確なリスト内の最初のエントリーが選択されます。すべてのフィルターでマッチングする 1 つのエントリーを検出するとスキャンは停止します。

  注: 表項目の検索は、項目の作成方法および検索方法により、ユーザーが混乱することがあります。例えば、 DL/I INQY 呼び出しで指定された複数のフィルターでマッチングするエントリーがテーブルに 2 つあると想定します。1 番目のトランザクションがトランザクション名と LTERM 名でマッチングします。2 番目のエントリーがトランザクション名とプログラム名でマッチングします。IMS は 1 番目のエントリーを選択します。なぜなら、それがフィルターに掛けた際のマッチングが最も多かった最初のエントリーだからです。2 番目のエントリーがより長いパラメーター・ストリングと共に更新されると、新規のエントリーが

構築され、キューのヘッドに追加されます。次の検索では、トランザクション名とプログラム名がある項目が選択されることになります。その結果、ユーザーが予想しなかったランタイム・オプションのセットが選択されることになります。

プログラム名の照会: INQY PROGRAM

PROGRAM 副次機能を指定して INQY 呼び出しを発行すると、アプリケーション・プログラム名が、入出力域の先頭の 8 バイトに返されます。 AIBRSNM1 に渡せる有効な PCB 名は IOPCBbbb だけです。

INQY 戻りコードおよび理由コード

INQY 呼び出しを発行すると、戻りコードと理由コードが AIB に返されます。状況コードが PCB に返される場合もあります。INQY に適用されるもの以外の戻りコードと理由コードが戻された場合は、ユーザーのアプリケーション・プログラムは PCB を検査して、どのような状況コードが戻されているのかを調べる必要があります。

PCB タイプへの INQY 副次機能のマップ

表 *20. INQY* 呼び出しの場合の副次機能、*PCB*、および入出力域の組み合わせ

副次機能	**I/O PCB**	代替 **PCB**	**DB PCB**	必要な入出力域
FIND	OK	OK	OK	NO
ENVIRON	OK	NO	NO	YES
DBQUERY	OK	NO	NO	NO
LERUNOPT	OK	NO	NO	NO
PROGRAM	OK	NO	NO	YES
MSGINFO	OK	NO	NO	YES

LOG 呼び出し

ログ (LOG) 呼び出しは、情報を送信して IMS システム・ログに書き込むときに使用します。

フォーマット

呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
LOG	X	X	X	X	X

パラメーター

i/o pcb
呼び出しで使用する入出力 PCB を指定します。このパラメーターは入出力パラメーターです。

aib

呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN

入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area

プログラム内の区域を指定します。ここには、システム・ログに書き込むレコードが入ります。これは入力パラメーターです。このレコードは、以下の表に示す形式に従っていなければなりません。

表 *21. COBOL*、*C*、アセンブラー、*Pascal*、および *AIBTDLI*、*ASMTDLI*、*CBLTDLI*、*CEETDLI*、 *CTDLI*、および *PASTDLI* インターフェースの *PL/I* プログラムのログ・レコード形式

LL	ZZ	C	テキスト
2	2	1	可変

表 *22. COBOL*、*C*、アセンブラー、*Pascal*、および *PLITDLI* インターフェースの *PL/I* プログラムのログ・レコード形式

LLLL	ZZ	C	テキスト
4	2	1	可変

フィールドは次のとおりでなくてはなりません。

LL または **LLLL**

レコードの長さが入る 2 バイト長フィールド (または、PL/I の場合には、 4 バイト長フィールド) を指定します。レコードの長さはレコードの LL + ZZ + C + テキストです。ログ・レコードの長さを計算する際は、これらのフィールドのすべてを考慮する必要があります。指定する長さの合計には、以下のものが含まれます。

- LL または LLLL の 2 バイト (PL/I の場合、LLLL は 4 バイトですが、長さは 2 として計算されます。)
- ZZ フィールドの 2 バイト
- C フィールドの 1 バイト
- レコード自体の長さを表す *n* バイト

PLITDLI インターフェースを使用している場合には、プログラムで、長さフィールドを 2 進数のフルワードで定義しなければなりません。

ZZ 2 バイトの 2 進数ゼロのフィールドを指定します。

C ログ・コードが入る 1 バイトのフィールドを指定します。これは、X'A0' 以上でなければなりません。

テキスト
ログに書き込むデータを指定します。

使用法

アプリケーション・プログラムは、LOG 呼び出しを発行して、システム・ログへレコードを書き込むことができます。 LOG 呼び出しを出すときには、システム・ログに書き込みたいレコードが入る入出力域を指定してください。ログには任意の情報を書き込むことができ、異なるタイプの情報を区別するために異なるログ・コードを使用できます。

LOG 呼び出しは次のように出すことができます。

- バッチ・プログラム内で出すと、レコードは IMS ログに書き込まれます。
- DBCTL 環境のオンライン・プログラム内で出すと、レコードは DBCTL ログに書き込まれます。
- IMS DB/DC 環境では、レコードは IMS ログに書き込まれます。

制約事項

(すべてのフィールドを含む) 入出力域の長さは、システム・ログ・データ・セットの論理レコード長 (LRECL) から 4 バイトを引いた値を超えてはなりません。あるいは、PSB の PSBGEN ステートメントの IOASIZE キーワードで指定された入出力域を超えてはなりません。

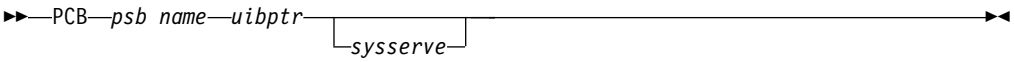
CICS 環境の機能シップの場合、ローカルおよびリモート CICS はともに DBCTL でなければなりません。

PCB 呼び出し (CICS オンライン・プログラムのみ)

PCB 呼び出しは、PSB 呼び出しをスケジューリングするために使用します。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
PCB	X	X			

パラメーター

AIB は PCB 呼び出しには無効です。

psb name
PSB を指定します。このパラメーターにアスタリスクを使用すると、デフォルト値を表すことができます。 このパラメーターは入力パラメーターです。

uibptr
ポインターを指定します。このポインターは、呼び出しの後の UIB のアドレスを指します。このパラメーターは出力パラメーターです。

sysserve
IOPCB または NOIOPCB が入るオプションの 8 バイト・フィールドを指定します。 このパラメーターは入力パラメーターです。

使用法

CICS オンライン・プログラムは、DL/I 呼び出しを出す前に、特定の PSB を使用する予定であることを DL/I に通知しなければなりません。PCB 呼び出しは、ユーザーのプログラムで使用する PSB を示すほかに、PSB 内の PCB のアドレスを入手します。PCB 呼び出しを出すときには、以下のものを指定してください。

- 呼び出し機能: PCBb
- 使用したい PSB またはアスタリスク (アスタリスクは、デフォルト名を使用することを意味します)。プログラムは他のプログラムによって呼び出されている可能性があるので、デフォルトの PSB 名は、必ずしも PCB 呼び出しを出すプログラムの名前にはなりません。
- ポインター。これは、呼び出しが出された後の UIB のアドレスを指します。

  UIB に対するアドレス可能度の定義方法および設定方法についての詳細は、「*IMS V14* アプリケーション・プログラミング」のトピック『UIB の指定 (CICS オンライン・プログラムのみ)』を参照してください。
- IOPCB または NOIOPCB が入っているオプションの 8 バイト・フィールドの名前を指定する、システム・サービス呼び出しパラメーター。

制約事項

CICS 環境の機能シップの場合、ローカルおよびリモート CICS はともに DBCTL でなければなりません。

RCMD 呼び出し

コマンド検索 (RCMD) 呼び出しを使用すると、自動化操作プログラム (AO) アプリケーション・プログラムは ICMD 呼び出しの後の 2 番目以降のコマンド応答セグメントを検索できます。

フォーマット

パラメーター

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。このフィールドは、IMS により変更されません。

AIBOAUSE
入出力域に戻されるデータの長さ。このパラメーターは出力パラメーターです。

入出力域に十分な大きさがないため、データの一部が返された場合、AIBOAUSE にすべてのデータを取り出すために必要な長さが入っており、AIBOALEN にデータの実際の長さが入っています。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。この入出力域は、IMS から AO アプリケーション・プログラムに渡される最大のコマンド応答セグメントを収めるのに十分な長さがなければなりません。その入出力域にすべての情報が入るだけの大きさがない場合には、入出力域には一部のデータが返されます。

使用法

RCMD を使用すると、AO アプリケーション・プログラムは ICMD 呼び出しが出された後の 2 番目以降のコマンド応答セグメントを検索できます。

関連資料 AOI 出口についての詳細は、「*IMS V14* 出口ルーチン」を参照してください。

以下の表は、IMS 環境において RCMD を出すことのできる AO アプリケーション・プログラムの型を示しています。RCMD はまた、CPI-C ドリブン・プログラムからもサポートされます。

表 *23.* アプリケーション領域タイプ別の *RCMD* サポート

	IMS 環境		
アプリケーション領域のタイプ	**DBCTL**	**DB/DC**	**DCCTL**
DRA スレッド	あり	あり	N/A
BMP (非メッセージ・ドリブン)	あり	あり	あり
BMP (メッセージ・ドリブン)	N/A	あり	あり
MPP	N/A	あり	あり
IFP	N/A	あり	あり

RCMD は、一度に 1 つの応答セグメントしか取り出しません。さらに応答セグメントが必要な場合は、IMS によって出される応答セグメントごとに一度ずつ RCMD を出さなければなりません。

制約事項

ICMD 呼び出しは、RCMD 呼び出しを発行する前に発行する必要があります。

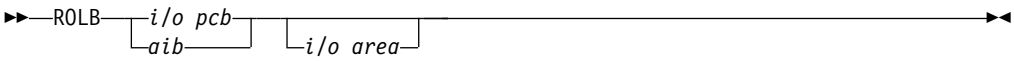
ROLB 呼び出し

ロールバック (ROLB) 呼び出しは、データベースの変更を動的にバックアウトし、制御をユーザーのプログラムに戻すために使用します。

ROLB 呼び出しの詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『データベース保全性の維持』を参照してください。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
ROLB	X	X	X	X	X

パラメーター

i/o pcb
呼び出しで使用する入出力 PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
ユーザーのプログラム内の、IMS によって最初のメッセージ・セグメントが戻される区域を指定します。このパラメーターは出力パラメーターです。

制約事項

この呼び出しに対して、AIB は入出力 PCB を指定しなければなりません。

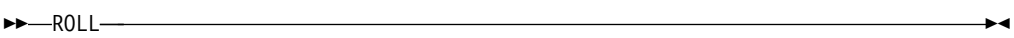
ROLL 呼び出し

ロール (ROLL) 呼び出しは、ユーザーのプログラムを異常終了させ、データベースの変更を動的にバックアウトするために使用します。

ROLL 呼び出しの詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『データベース保全性の維持』を参照してください。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
ROLL	X	X	X	X	X

パラメーター

ROLL 呼び出しに必要なパラメーターは、呼び出し機能だけです。

使用法

ROLL 呼び出しを出すと、IMS は U0778 異常終了でアプリケーション・プログラムを終了させます。

制約事項

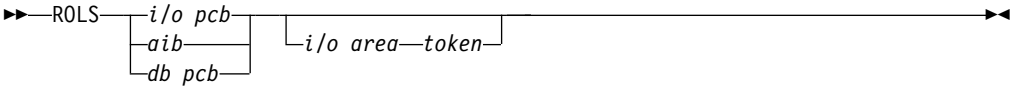
ROLB 呼び出しとは異なり、ROLL 呼び出しは、制御をプログラムに戻しません。

ROLS 呼び出し

SETS へのロールバック (ROLS) 呼び出しは、先行する SETS または SETU 呼び出しによって設定された処理ポイントにバックアウトするために使用します。

ROLS 呼び出しの詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『データベース保全性の維持』を参照してください。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
ROLS	X	X	X	X	X

パラメーター

db pcb
呼び出しで使用する DB PCB を指定します。このパラメーターは入出力パラメーターです。

i/o pcb
呼び出しで使用する入出力 PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、PCB 名、IOPCBbbb、または DB PCB 名を指定します。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
SETS 呼び出しで指定した入出力域と同じ形式の入出力域を指定します。このパラメーターは出力パラメーターです。

token
ユーザーのプログラム内の、4 バイト ID が入っている区域を指定します。 このパラメーターは入力パラメーターです。

使用法

先行する SETS または SETU 呼び出しによって設定された処理ポイントにバックアウトするために Roll Back to SETS (ROLS) 呼び出しを使用すると、ROLS によりユーザーは処理を継続したり、あるいは前のコミット・ポイントにバックアウトし、それ以降の処理の中断キュー上に入力メッセージを書くことができます。

DB PCB で ROLS 呼び出しを出すと、結果としてユーザー異常終了コード 3303 が戻されます。

制約事項

CICS 環境の機能シップの場合、ローカルおよびリモート CICS はともに DBCTL でなければなりません。

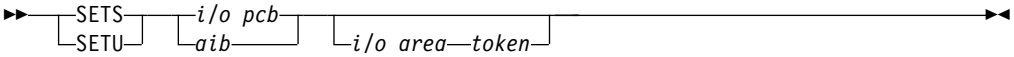
ROLS 呼び出しは、PSB に DEDB または MSDB PCB が入っているとき、または DB2 データベースに対して呼び出しが発行されるときは無効です。

SETS/SETU 呼び出し

バックアウト点の設定 (SETS) 呼び出しは、中間バックアウト・ポイントを設定したり、既存のすべてのバックアウト点を取り消したりするために使用します。

無条件 SET (SETU) 呼び出しは、 SETS 呼び出しと似た働きをしますが、サポートされていない PCB が含まれていたり、外部サブシステムが使用されている場合でも、SETU 呼び出しは受け入れられます。SETS および SETU 呼び出しの詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『データベース保全性の維持』を参照してください。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
SETS/SETU	X	X	X	X	X

パラメーター

i/o pcb
呼び出しで使用する入出力 PCB を指定します。SETS および SETU は、入出力 PCB を参照しなければなりません。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
ユーザーのプログラム内の、対応する ROLS 呼び出しで戻されるデータが入る区域を指定します。このパラメーターは入力パラメーターです。

token
ユーザーのプログラム内の、4 バイト ID が入っている区域を指定します。 このパラメーターは入力パラメーターです。

使用法

SETS と SETU では、呼び出し機能が SETS と SETU であるという 1 点を除けば、形式もパラメーターも同じです。

SETS および SETU 呼び出しは、IMS が ROLS 呼び出しで使用するバックアウト・ポイントを提供します。 ROLS 呼び出しは、SETS および SETU 呼び出しのバックアウト・ポイントを使用して機能します。

SETS および SETU における SC 状況コードの意味は次のとおりです。

SETS SETS 呼び出しは拒否されます。入出力 PCB の状況コード SC は、PSB にサポートされないオプションが入っているか、またはアプリケーション・プログラムが外部サブシステムに呼び出しを発行したことを示します。

SETU SETU 呼び出しは拒否されません。SC 状況コードは、PSB 内にサポートされない PCB が存在すること、またはアプリケーション・プログラムが外部サブシステムに対して呼び出しを出したことのどどちらかを示します。

制約事項

CICS 環境の機能シップの場合、ローカルおよびリモート CICS はともに DBCTL でなければなりません。

SETS 呼び出しは、PSB に DEDB または MSDB PCB が入っているとき、または DB2 データベースに対して呼び出しが発行されるときは無効です。 PSB にサポートされない PCB が含まれている場合、あるいはプログラムが外部サブシステムを使用した場合、SETU 呼び出しは有効ですが機能しません。

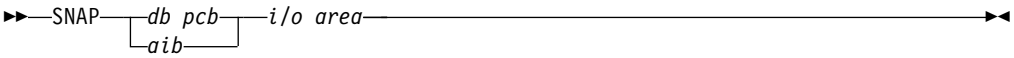
ROLS 呼び出しの前に、同一のトークンを使用して最大 255 の SETS 呼び出しを指定でき、しかも、正しいメッセージ・レベルにバックアウトすることができます。 255 の SETS 呼び出しの後で、メッセージのバックアウトが続行されますが、255 番目の SETS 呼び出しと同じメッセージ・レベルにのみバックアウトされます。 SETS のトークン・カウントは、同期点処理時にゼロにリセットされます。

SNAP 呼び出し

SNAP 呼び出しは、診断情報を収集するために使用します。

このトピックにはプロダクト・センシティブ・プログラミング・インターフェース情報が含まれています。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
SNAP	X	X		X	

パラメーター

db pcb
呼び出し元プログラムの PSB で定義された全機能 PCB を参照するアドレスを指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、全機能 DB PCB の名前を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area
ユーザーのプログラム内の、SNAP 操作パラメーターが入る区域を指定します。このパラメーターは入力パラメーターです。以下の図は、ユーザーが指定する、以下の SNAP 操作パラメーターを示します。

- 長さ (1 - 2 バイト)
- 宛先 (3 - 10 バイト)
- 識別番号 (11 - 18 バイト)
- SNAP オプション (19 - 22 バイト)

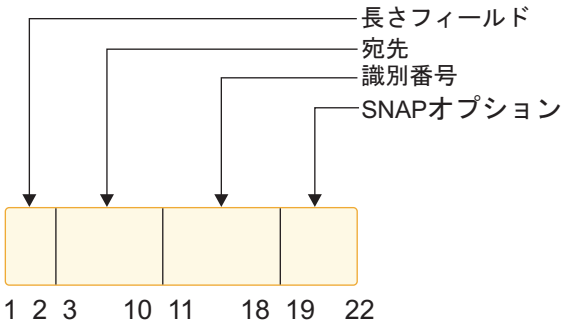


図 2. *SNAP* 操作パラメーターの入出力域

以下の表は、ユーザーが指定できる値を説明しています。

表 24. *SNAP* 操作パラメーター

バイト	値	意味
1-2	*xx*	この 2 バイトの 2 進数フィールドには、SNAP 操作パラメーターの長さを指定する。長さには、この 2 バイト長フィールドも含めなければならない。 操作パラメーターを指定しなかった場合、IMS はデフォルト値を使用する。この表は、パラメーターの指定値によって決まる長さを示している。

ユーザーが指定した値	**IMS** が提供するデフォルト値	長さ **(16** 進数**)**
宛先、識別番号、SNAP オプション		16
宛先、識別番号	SNAP オプション	12
宛先	識別番号、SNAP オプション	10
	宛先、識別番号、SNAP オプション	2
別の長さを指定した場合には、IMS は宛先、識別番号、および SNAP 操作パラメーターにデフォルト値を使用する。		

表 *24. SNAP* 操作パラメーター *(*続き*)*

バイト	値	意味
3 から 10		この 8 バイト・フィールドは、SNAP 出力の送信先を IMS に知らせる。出力を IMS ログあてに送るには、LOG または bbbbb を指定する。
		出力を IMS ログに送る。これはデフォルト宛先である。
	dcbaddr	この DCB アドレスで定義されたデータ・セットに出力を送る。 アプリケーション・プログラムは、SNAP 呼び出しがデータ・セットを参照する前にそのデータ・セットをオープンする必要がある。このオプションは、バッチ環境でのみ有効です。出力データ・セットは、z/OS SNAP データ・セットに関する規則に従っている必要がある。
	ddname	対応する DD ステートメントで定義されたデータ・セットに出力を送る。この DD ステートメントは、z/OS SNAP データ・セットに関する規則に従っている必要がある。*ddname* で指定されたデータ・セットは、この SNAP 要求のためにオープンされクローズされる。 DB/DC 環境では、DD ステートメントを制御領域用の JCL の中に含めなければならない。 宛先が無効な場合、IMS は出力を IMS ログに送る。
11 から 18	*cccccccc*	この 8 文字名を使用して、 SNAP を識別することができる。名前を指定しない場合、IMS はデフォルト値である NOTGIVEN を使用する。
19 から 22	*cccc*	この 4 文字フィールドは、SNAP 出力に含めたいデータ・エレメントを特定する。デフォルトは YYYN である。
19		バッファー・プール**:**
	Y	SNAP 呼び出しで、すべてのバッファー・プールと順次バッファー制御ブロックをダンプする。
	N	SNAP 呼び出しで、すべてのバッファー・プールまたは順次バッファー制御ブロックをダンプしない。
20		制御ブロック**:**
	Y	SNAP 呼び出しで、現行 DB PCB に関連した制御ブロックをダンプする。
	N	SNAP 呼び出しで、現行 DB PCB に関連した制御ブロックをダンプしない。
21	Y	SNAP 呼び出しで、この PSB に関連するすべての制御ブロックをダンプする。バイト 21 に Y を指定すると、現行値に関係なく、バイト 20 の現行 DB PCB 要求が Y にセットされ、それに対してスナップ・ダンプが生成されます。
	N	SNAP 呼び出しで、この PSB に関連するすべての制御ブロックをダンプしない。この場合、20 桁目の現行 DB PCB SNAP 要求が指定されたとおりに使用される。
19 から 21	ALL	これは、19-21 桁目に YYY を指定する場合と同じになる。

表 *24. SNAP* 操作パラメーター *(*続き*)*

バイト	値	意味
22		領域**:**
	Y	SNAP 呼び出しで、バイト 3-10 で指定した DCB アドレスまたはデータ・セット DD 名に全領域をダンプする。IMS は、バイト 19-21 で指定された SNAP 要求を処理する前に、この要求を処理する。宛先が IMS ログになっている場合、IMS は全領域をダンプしない。その代わり、ALL が指定されているものとして要求が処理される。
	N	SNAP 呼び出しで、全領域をダンプしない。
	S	SNAP 呼び出しで、サブプール 0-127 をダンプする。

SNAP 呼び出しの後、IMS は AB、AD、bb (ブランク) のいずれかを状況コードとして返します。これらのコードおよび必要な応答については、「*IMS V14* メッセージおよびコード 第 *4* 巻*: IMS* コンポーネント・コード」を参照してください。

使用法

どのアプリケーション・プログラムも、この呼び出しを出すことができます。

制約事項

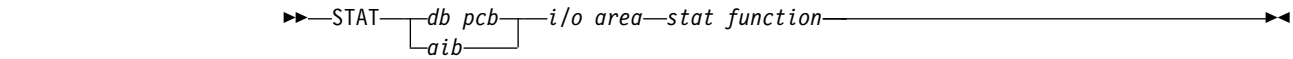
CICS 環境の機能シップの場合、ローカルおよびリモート CICS はともに DBCTL でなければなりません。

STAT 呼び出し

統計 (STAT) 呼び出しは、パフォーマンス・モニターに役立つデータベース統計を入手するために、CICS、IMS オンライン、またはバッチ・プログラムで使用します。

このトピックにはプロダクト・センシティブ・プログラミング・インターフェース情報が含まれています。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
STAT	X	X		X	

パラメーター

db pcb

アプリケーション・プログラムに状況情報を渡すために使用される DB PCB を指定します。この PCB に関連したデータ・セットによって使用される VSAM 統計は、STAT 呼び出しによって戻される統計のタイプとは関連付けられていません。この PCB は、全機能データベースを参照しなければなりません。このパラメーターは入出力パラメーターです。

aib

呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。この 8 バイトの左寄せフィールドには、全機能 DB PCB の名前を指定しなければなりません。

AIBOALEN

入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。

i/o area

必要な統計を収めるのに十分な大きさの、アプリケーション・プログラム内の区域を指定します。このパラメーターは出力パラメーターです。PL/I の場合、このパラメーターは大構造、配列、または文字ストリングを指すポインターとして指定してください。

stat function

要求された統計のタイプと形式を記述する内容の 9 バイト域を指定します。最初の 4 バイトでは、希望する統計タイプを定義し、5 番目のバイトでは、提供される形式を定義します。残りの 4 バイトには、EBCDIC ブランクが入ります。指定した *stat function* が定義された機能に該当しない場合には、AC 状況コードが戻されます。このパラメーターは入力パラメーターです。9 バイト・フィールドには、以下の情報が入ります。

- 希望する統計のタイプを定義する 4 バイト

  DBAS
  OSAM データベース・バッファー・プールの統計

  DBES OSAM データベース・バッファー・プールの統計。機能強化または拡張されたもの。

  VBAS
  VSAM データベース・サブプールの統計

  VBES VSAM データベース・サブプールの統計。機能強化または拡張されたもの。

- 統計の形式を指定する 1 バイト

  F 統計全体を形式設定します。F を指定した場合の入出力域は、DBAS/VBAS の場合には少なくとも 360 バイト、DBES/VBES の場合には少なくとも 600 バイトでなければなりません。

  O 形式設定された OSAM データベース・サブプールの全統計。O を指定した場合、入出力域は少なくとも 360 バイトでなければなりません。

S 統計の要約を形式設定します。S を指定した場合の入出力域は、DBAS/VBAS の場合には少なくとも 120 バイト、DBES/VBES の場合には少なくとも 360 バイトでなければなりません。

U 統計全体を形式設定しません。U を指定した場合、入出力域は少なくとも 72 バイトでなければなりません。

- 通常または強化 STAT 呼び出しのための 4 バイトの EBCDIC ブランク、または bE1b

  制約事項**:** 拡張形式パラメーターがサポートされているのは、DBESO、DBESU、および DBESF 機能だけです。

  強化呼び出し機能の後にパラメーター bE1bを指定すると、拡張 OSAM のバッファー・プール統計を取り出すことができます。拡張 STAT 呼び出しでは、強化呼び出しによって返されるすべての統計値に加え、カップリング・ファシリティー・バッファーの無効化、OSAM キャッシング、順次バッファリングでの IMMED と SYNC 読み取りカウントが返されてきます。

使用法

STAT 呼び出しは IMS データベース統計を検索するため、デバッグに利用できます。また、パフォーマンスのモニターおよびチューニングに利用することもできます。 STAT 呼び出しは、OSAM データベース・バッファー・プール統計および VSAM データベース・バッファー・サポートを検索します。

VSAM 統計を要求すると、STAT 呼び出しを出すたびにサブプールの統計が検索されます。すべての VSAM ローカル共用リソース・プールに関する統計が、定義された順に検索されます。それぞれのローカル共用リソース・プールについて、バッファー・サイズの昇順に統計が検索されます。共用リソース・プールに索引サブプールがある場合、データ・サブプールに関する統計の後に、常に索引サブプールに関する統計が続きます。索引サブプールも、バッファー・サイズの昇順で検索されます。

STAT 呼び出しの詳細については、「*IMS V14* アプリケーション・プログラミング」を参照してください。

制約事項

CICS 環境の機能シップの場合、ローカルおよびリモート CICS はともに DBCTL でなければなりません。

関連概念:

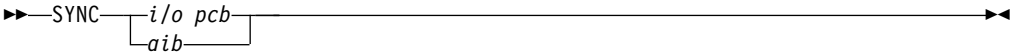
データベース統計のリトリーブ: STAT 呼び出し (アプリケーション・プログラミング)

SYNC 呼び出し

同期点 (SYNC) 呼び出しは、IMS がアプリケーション・プログラム用にロックしたリソースをリリースするために使用します。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
SYNC	X	X	X		

パラメーター

i/o pcb
呼び出しで使用する PCB を指定します。このパラメーターは入出力パラメーターです。

aib
呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

使用法

SYNC は、ユーザーのプログラムがデータベースに対して行った変更をコミットし、ユーザーのプログラムが異常終了した場合にそのプログラムを再始動するための位置を設定します。

制約事項

SYNC 呼び出しは、非メッセージ・ドリブン BMP でのみ有効です。ユーザーは CPI-C ドリブン・アプリケーション・プログラムから SYNC 呼び出しを出すことはできません。

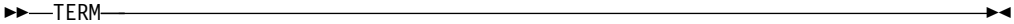
SYNC 呼び出しの使用に関する重要な考慮事項については、「*IMS V14* データベース管理」を参照してください。

TERM 呼び出し (CICS オンライン・プログラムのみ)

終了 (TERM) 呼び出しは、CICS オンライン・プログラムの PSB を終了させるために使用します。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
TERM	X	X			

使用法

ユーザーのプログラムで複数の PSB を使用する必要がある場合、使用する最初の PSB を解放するために TERM 呼び出しを出してから、もう一度 PCB 呼び出しを出して 2 番目の PSB をスケジュールしなければなりません。 TERM 呼び出しは、データベース変更のコミットも行います。

TERM 呼び出しの唯一のパラメーターは、呼び出し機能の TERM または bbb です。ユーザーのプログラムがこの呼び出しを発行すると、CICS は、スケジュール済みの PSB を終了し、CICS 同期点をとり、変更をコミットし、リソースを解放して他のタスクで使用できるようにします。

制約事項

CICS 環境の機能シップの場合、ローカルおよびリモート CICS はともに DBCTL でなければなりません。

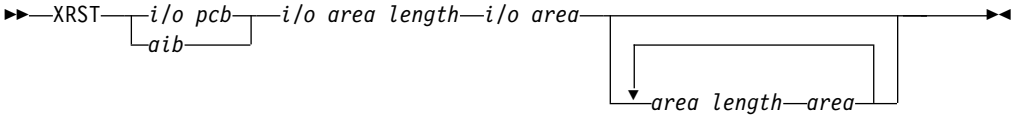
XRST 呼び出し

拡張再始動 (XRST) 呼び出しは、プログラムを再始動するときに使用します。

ユーザーのプログラムでシンボリック・チェックポイント呼び出しを使用する場合には、それに先立って XRST 呼び出しを出して、ブランクのチェックポイント・データを指定する必要があります。

ODBA インターフェースはこの呼び出しをサポートしません。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
XRST	X	X	X	X	X

パラメーター

i/o pcb
: 呼び出しで使用する入出力 PCB を指定します。XRST は、入出力 PCB を参照しなければなりません。このパラメーターは入出力パラメーターです。

aib

呼び出しで使用する AIB を指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN

入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。このパラメーターは、XRST 呼び出しの間は使用されません。互換性の理由により、このパラメーターは引き続きコーディングする必要があります。

i/o area length

このパラメーターは、IMS では使用されなくなりました。互換性の理由により、引き続きこのパラメーターはこの呼び出しに組み込む必要があり、また有効なアドレスを入れなければなりません。プログラム内の任意の区域の名前を指定することにより、有効なアドレスを獲得することができます。

i/o area

プログラムに 14 バイトの区域を指定します。この区域は、プログラムを通常に開始する場合はブランクに設定され、拡張再始動を実行する場合はチェックポイント ID を持ちます。

area length

リストアする区域の長さ (2 進数) が入るユーザーのプログラムの 4 バイト・フィールドを指定します。このパラメーターは入力パラメーターです。最大 7 つの区域長を指定することができます。それぞれの区域長について、 *area* パラメーターを指定する必要があります。7 つの *area* パラメーター (および対応する *area length* パラメーター) は、すべてオプションです。プログラムが再始動されると、IMS は、CHKP 呼び出しで指定された区域だけをリストアします。

XRST 呼び出しで指定する区域の数は、CHKP 呼び出しで指定した区域の数より小さいか等しくなければなりません。

area

IMS にリストアさせたいユーザーのプログラム内の区域を指定します。区域は 7 つまで指定できます。 指定される各区域の前に、*area length* を指定する必要があります。これは入力パラメーターです。

使用法

記号チェックポイント呼び出し (CHKP) を発行したいプログラムは、拡張再始動呼び出し (XRST) も発行する必要があります。XRST 呼び出しは、一度だけ、プログラム実行の早い時期に発行される必要があります。この呼び出しは、プログラムの最初

の呼び出しでなくてもかまいません。ただし、CHKP 呼び出しよりは先に出さなければなりません。XRST 呼び出しより前に出されるすべてのデータベース呼び出しは、再始動の有効範囲内には入りません。

正常に始動するかあるいは再始動するかを決めるために、IMS は、XRST 呼び出しにより提供される入出力域、またはユーザーのプログラムの JCL 内の EXEC ステートメント上の PARM フィールドにある CKPTID= の値により提供される入出力域を評価します。

プログラムの正常始動

プログラムを正常に始動するときは、 XRST 呼び出しで指し示す入出力域にブランクが入り、かつ、PARM フィールドの CKPTID= 値がヌルである必要があります。これが、以降の CHKP 呼び出しは、基本チェックポイントではなくシンボリック・チェックポイントであることを、IMS に指示します。 XRST 呼び出しを発行した後、プログラムで入出力域を検査する必要があります。プログラムを正常に始動するとき、IMS はこの区域を変更しません。ただし、変更済みの入出力域は、ユーザー・プログラムを再始動させようとしていることを示します。したがって、ユーザーのプログラムは、直前に保管されていて現時点でリストアされている指定済みのデータ域を処理する必要があります。

ユーザー・プログラムの再始動

プログラムの直前の実行時にとられたシンボリック・チェックポイントから、プログラムを再始動することができます。再始動の実行に使用されるチェックポイントは、以下の方法で識別されます。すなわち、XRST 呼び出し (左寄せで、残りの区域にはブランクが入る) により示される入出力域に、チェックポイント ID を入力するか、プログラムの JCL の EXEC ステートメントの PARM= パラメーターの CKPTID= フィールドに ID を指定します。 (両方指定された場合は、IMS は、EXEC ステートメントのパラメーター・フィールドの CKPTID= 値を使用します。)

ID 指定は、次のいずれかが可能です。

- 1 文字から 8 文字の拡張チェックポイント ID。
- メッセージ DFS0540I からの 14 文字のタイム・スタンプ ID。ここで、
  - IIII は、領域 ID。
  - DDD は、年間通算日。
  - HHMMSST は、時、分、秒、秒の十分の一の値による時刻。
- 4 文字の定数 "LAST"。(BMP のみ : これは、BMP で出した、最後のチェックポイントを、プログラムの再始動に使用するように IMS に指示します。)

システム・メッセージ DFS0540I では、チェックポイント ID とタイム・スタンプが提供されます。

システム・メッセージ DFS682I は、最後に完了したチェックポイントのチェックポイント ID を提供します。これを使用して、異常終了したバッチ・プログラムまたはバッチ・メッセージ処理プログラム (BMP) を再始動できます。

XRST 呼び出しの完了時には、入出力域には、再始動に使用される 8 文字のチェックポイント ID が必ず入ります。チェックポイント ID が 8 個のブランク文字のときは例外です。入出力域には 14 文字のタイム・スタンプ (IIIIDDDHHMMSST) が入ります。

再始動されるプログラムが DL/I バッチ領域にある場合、ログ・データ・セットを定義する IMSLOGR DD ステートメントを JCL に提供する必要があります。IMS は、これらのデータ・セットを読み取り、指定された ID を持つチェックポイント・レコードを検索します。

ただし、再始動されるプログラムが BMP 領域にあり、以下の条件がすべて満たされる場合は、IMSLOGR DD ステートメントは必要ありません。

- BMP プログラムが CKPTID=LAST を指定して再始動される。
- BMP プログラムが同じ IMS システム上で、同じジョブ名、同じ PSB、および異常終了時に使用されていたのと同じプログラム名で再始動される。
- BMP プログラムの異常終了以後に IMS のコールド・スタートが行われていない。
- プログラムの再始動に必要なチェックポイント・レコードがあるデータ・セットは、異常終了時以後にアーカイブも再使用もされていない OLDS データ・セットである。あるいは、SLDSREAD ロガー機能が IMS でアクティブになっている。

上記の条件のいずれかが存在しない場合、必要なチェックポイント・レコードが入っているデータ・セットを指す DD ステートメントを提供する必要があります。

IMSLOGR DD ステートメントを提供する場合、必要なチェックポイント・ログ・レコードを入れる必要があります。IMSLOGR DD ステートメントがある場合、IMS は BMP のチェックポイント・レコードを自動的に探して取り出すことはしません。 IMSLOGR DD ステートメントのみが検索され、レコードが見つからない場合は BMP プログラムは異常終了 U0102 で終了します。

注**:** IMSLOGR DD ステートメント用の DD DUMMY ステートメントを提供することもできます。DD DUMMY ステートメントは IMSLOGR DD ステートメントが存在しないかのように処理されます。

XRST 呼び出しの完了時には、入出力域には、再始動に使用される 8 文字のチェックポイント ID が必ず入ります。チェックポイント ID が 8 個のブランク文字のときは例外です。入出力域には 14 文字のタイム・スタンプ (IIIIDDDHHMMSST) が入ります。

入出力 PCB の状況コードも検査してください。XRST 呼び出しの唯一の成功状況コードは、ブランクです。

XRST を出した後のデータベース内の位置

XRST 呼び出しは、すべてのデータベースの位置を、最後にチェックポイントが取られたときに保管された位置に変更しようとします。これは、各 PCB および PCB キー・フィードバック域をチェックポイント・レコードに入れることによって行われます。XRST を出すと、キー・フィードバック域がチェックポイント・レコードの PCB から、再始動する PSB 内の対応する PCB に移動します。その後で IMS

は、チェックポイントが取られたときに位置を保管した各 PCB に関して、 (C コマンド・コードを使用して) 連結キーで修飾した GU 呼び出しを出します。

XRST 呼び出しが出された後の PCB は、チェックポイントが取られたときに存在していた値ではなく、GU 再位置付け呼び出しの結果を反映しています。チェックポイントが取られたときに PCB がルート・セグメントまたはそれよりも低いレベルのセグメントに位置を保管しなかった場合には、GU 呼び出しは出されません。PCB に GE 状況コードが戻された場合には、連結キーに関する GU が完全には満たされていません。PCB 内のセグメント名、セグメント・レベル、およびキー・フィードバック長は、その GU 呼び出しで最後に満たされたレベルを反映しています。次のいずれかの理由によって IMS が Get 呼び出しに関連するセグメント検索引数を満たすセグメントを検出できないために、GE 状況コードが戻されることがあります。

- チェックポイント呼び出しの前に出された呼び出しが、同じ PCB に対する DLET 呼び出しであった。この場合、位置は適正です。Get 呼び出しでのターゲットの未検出後の位置が、DLET 呼び出しの後に存在するはずの位置と同じであるためです。

  制約事項: DLET 呼び出しの直後にチェックポイントを取るのは避けてください。
- ユーザーのプログラムが異常終了してからそのプログラムが再始動されるまでの間に、別のアプリケーション・プログラムによってそのセグメントが削除された。再始動のあとに出された GN 呼び出しは、削除された 1 つまたは複数のセグメントの後の最初のセグメントを戻します。

この説明は、チェックポイント時の位置がユニーク・キーをもつセグメントにあることを前提としています。そのセグメントまたはそのセグメントの親のうちのいずれかが非ユニーク・キーをもっている場合には、XRST でそのセグメントに再位置付けすることはできません。

DEDB の場合、位置がセグメント上ではなく作業単位 (UOW) 境界にあるときには、GC 状況コードが戻されます。XRST 呼び出しは、シンボリック・チェックポイントがとられたときに PCB が位置付けられたセグメントに位置を再設定しようとするため、PCB に GC 状況コードが入っているときにシンボリック・チェックポイントがとられた場合には、XRST 呼び出しは PCB 上に位置を再設定しません。

ユーザーのプログラムが GSAM データベースにアクセスする場合、 XRST 呼び出しはこれらのデータベースも位置変更します。

GSAM の XRST 処理中には、位置変更する GSAM 出力データ・セットが空であるかどうか、および異常終了したジョブがデータ・セットにレコードを前に挿入したかどうかについて検査が行われます。

制約事項

プログラムを正常に始動する場合には、入出力域の先頭の 5 バイトはブランクに設定しなければなりません。

プログラムを再始動させる場合で、EXEC ステートメント上の PARM フィールドの CKPTID= の値が使用されていない場合には、入出力域で使用されるチェックポイント ID の残りの右端のバイトはブランクになっている必要があります。

XRST 呼び出しは、バッチ・アプリケーション・プログラムおよび BMP アプリケーション・プログラムのみから出すことができます。

トランザクション管理

以下の参照情報を使用して、トランザクション管理のための DL/I 呼び出しを行います。

トランザクション管理のための DL/I 呼び出し

IMS TM で以下の DL/I 呼び出しを使用して、アプリケーション・プログラムでのトランザクション管理機能を実行します。

トランザクション管理呼び出しには、*i/o pcb* または *aib* パラメーターのどちらかを使用しなければなりません。

各呼び出しの説明には以下の内容が含まれます。

- 構文図
- 呼び出しで使用できる各パラメーターの定義
- アプリケーション・プログラムでの呼び出しの詳しい使用方法
- 呼び出しの使用についての制約事項

各パラメーターは、入力パラメーターまたは出力パラメーターとして記述されています。『入力』とは、アプリケーション・プログラムから IMS への入力のことをいいます。「出力」とは、IMS からアプリケーション・プログラムへの出力のことを指します。

トランザクション管理呼び出しのための構文図には、完全な呼び出し構造を含んでいるわけではありません。完全な構文ではなく、各呼び出しは *function* パラメーターで始まっています。この構文図には、呼び出し、呼び出しインターフェース (xxxTDLI)、および *parmcount* (必要な場合) は含まれていません。完全な構造については、「*IMS V14* アプリケーション・プログラミング」のトピック『アプリケーション・プログラム・エレメントの定義』に記載されている言語固有の情報 (COBOL、C 言語、Pascal、PL/I、アセンブラー言語別) を参照してください。

トランザクション管理呼び出しの要約

以下の表は、各トランザクション管理メッセージ呼び出しに有効なパラメーターを要約したものです。以下の表には、機能コード、その意味、用途、パラメーター、および機能コードの有効な領域がリストしてあります。オプション・パラメーターは、大括弧 ([ ]) で囲まれています。

例外: 言語依存パラメーターは、ここでは示されていません。PLITDLI 呼び出しには、*parmcount* 変数が必要です。アセンブラー言語呼び出しには、*parmcount* か **VL** のどちらかが必要です。COBOL、C、および Pascal プログラムでは、parmcount はオプションです。言語固有の情報については、「*IMS V14* アプリケーション・プログラミング」のトピック『言語インターフェースのための DL/I 呼び出しのフォーマット設定』を参照してください。

関連資料: プログラミング言語のインターフェースを使用する呼び出しの作成については、「*IMS V14* アプリケーション・プログラミング」の『アプリケーション・プログラム・エレメントの定義』を参照してください。

表 *25. TM* メッセージ呼び出しの要約

機能コード	意味	用途	パラメーター	有効な環境
AUTH	許可	ユーザーのセキュリティー許可を検証する	function、i/o pcb または aib、i/o area	DB/DC、DCCTL
CHNG	変更	変更可能代替 PCB に宛先を設定する。	function, alt pcb または aib, destination name[, options list, feedback area]	DB/DC、DCCTL
CMD	コマンド	プログラムが IMS コマンドを出すために使用する。	function、i/o pcb または aib、i/o area	DB/DC、DCCTL
GCMD	コマンド結果の取り出し	コマンドに対する 2 回目以降の応答を検索する。	function、i/o pcb または aib、i/o area	DB/DC、DCCTL
GN	後続取り出し	2 回目以降のメッセージ・セグメントを検索する。	function、i/o pcb または aib、i/o area	DB/DC、DCCTL
GU	初回取り出し	メッセージの初回のセグメントを検索する。	function、i/o pcb または aib、i/o area	DB/DC、DCCTL
ICAL	IMS 呼び出し	データやサービスに対する同期要求を、分散環境で稼働する非 IMS アプリケーション・プログラムまたはサービスに送信する。	aib、request area、response area	DB/DC、DCCTL
ISRT	挿入	プログラムの入出力域に出力メッセージを作成する。	function、i/o または alt pcb または aib、i/o area [、mod name]	DB/DC、DCCTL
PURG	パージ	メッセージを PCB から宛先のキューに入れる。	function、i/o または alt pcb または aib[、i/o area、mod name]	DB/DC、DCCTL
SETO	拡張印刷機能および APPC/IMS メッセージ処理のための処理オプションの設定	オプション・リストのエラーに関する情報がフィードバック域に返される。	function、i/o pcb または alternate pcb または aib、i/o area、options list[、feedback area]	BMP、MPP、IFP DB/DC、DCCTL

関連資料: DCCTL ユーザーは、GSAM データベース PCB を使用して呼び出すことができます。「*IMS V14* アプリケーション・プログラミング」を参照してください。

関連資料:

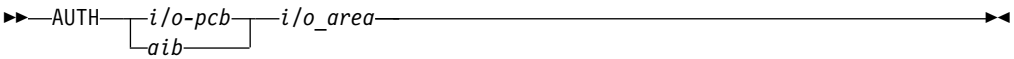
41 ページの『IMS DB システム・サービスのための DL/I 呼び出し』

1 ページの『データベース管理のための DL/I 呼び出し』
188 ページの『EXEC DLI コマンド』

AUTH 呼び出し

許可 (AUTH) 呼び出しは、各ユーザーのセキュリティー許可を検査します。ユーザーが、AUTH 呼び出しに指定されているリソースに、アクセスする許可を与えられているかどうかを判別します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
AUTH	X		X		

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbbを指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは入出力パラメーターです。

入出力域

以下の表に、AUTH 呼び出しを発行する前の入出力域のパラメーター・リストの形式を示します。

AUTH 呼び出し前の入出力域

表 *26. AIBTDLI*、*ASMTDLI*、 *CBLTDLI*、*CEETDLI*、*CTDLI* および *PASTDLI* インターフェースの場合、*AUTH* 呼び出し発行前の入出力域は以下のとおりです。

フィールド名	フィールド長
LL	2
ZZ	2
CLASSNAME	8
RESOURCE	8
USERDATA	8

表 *27. PLITDLI* インターフェースの場合、*AUTH* 呼び出し発行前の入出力域は以下のとおりです。

フィールド名	フィールド長
LLLL	4
ZZ	2
CLASSNAME	8
RESOURCE	8
USERDATA	8

LL または LLLL
2 バイトのフィールドを指定します。ここには、LL の 2 バイトを含むパラメーター・リストの長さが入ります。 PLITDLI インターフェースの場合には、4 バイトのフィールド LLLL を使用します。ただし、AIBTDLI インターフェースを使用する場合には、PL/I プログラムに必要なフィールドは 2 バイトだけです。

ZZ 2 バイトのフィールドを指定します。ここには、2 進数のゼロが入ります。

CLASSNAME
8 バイトのフィールドを指定します。ここには、以下のいずれかの値が入ります。

TRANbbbb

DATABASE

SEGMENTbb

FIELDbbb

OTHERbbb

すべてのパラメーターは長さ 8 バイトで左寄せされ、右側には必ずブランクが埋め込まれます。

呼び出しパラメーター・リストで総称クラス名を使用すれば、アプリケーションが、使用される実際のリソース・アクセス管理機能 (RACF) クラス名に依存する必要がなくなります。トランザクション許可はアクティブでなければならないので、定義する必要があるのは、そのトランザクション・クラスの総称クラス名 ID に関連付けられた RACF クラスだけです。呼び出しパラメーター・リストの総称クラス名により、許可機能が適切な RACF クラスを選択し、そのクラスを検査するアクセスを要求します。

RESOURCE
8 バイトのフィールドを指定します。ここには、検査するリソースの名前が入ります。総称クラス TRAN を除いては、IMS TM にとって名前は意味を持たないため、アプリケーションはどのようなリソース名を指定してもかまいません。

IMS TM は、リソース名については妥当性検査は行いません。

USERDATA
8 バイトのキーワードを指定します。唯一サポートされる値は、定数 USERDATA です。パラメーター・リストにこのキーワードがある場合、アプリケーションが RACF アクセス機能環境エレメント (ACEE) に存在する RACF インストール・データを必要としていることを意味します。

以下の表は、AUTH 呼び出し後の入出力域を示しています。

AUTH 呼び出し後の入出力域

表 *28. AIBTDLI*、*ASMTDLI*、 *CBLTDLI*、*CEETDLI*、*CTDLI* および *PASTDLI* インターフェースの場合、*AUTH* 呼び出し発行後の入出力域は以下のとおりです。

フィールド名	フィールド長
LL	2
ZZ	2
FEEDBACK	2
EXITRC	2
STATUS	2
RESERVED	16
UL	2
USERDATA	可変

表 *29. PLITDLI* インターフェースの場合、*AUTH* 呼び出し発行後の入出力域は以下のとおりです。

フィールド名	フィールド長
LLLL	4
ZZ	2
FEEDBACK	2
EXITRC	2
STATUS	2
RESERVED	16
UL	2
USERDATA	可変

LL または LLLL
この 2 バイトのフィールドには、文字ストリングの長さに LL の 2 バイトを加算した値が入ります。 PLITDLI インターフェースの場合には、4 バイトのフィールド LLLL を使用します。ただし、AIBTDLI インターフェースを使用する場合には、PL/I プログラムに必要なフィールドは 2 バイトだけです。

ZZ 2 バイトのフィールドを指定します。ここには、2 進数のゼロが入ります。

FEEDBACK

2 バイトのフィールドを指定します。ここには、以下のいずれかの RACF 戻りコードが入ります。

0000 ユーザーは許可されている。

0004 リソースまたはクラスが定義されていない。

0008 ユーザーは許可されていない。

000C RACF がアクティブではない。

0010 インストール・システム出口戻りコードが無効。

EXITRC

2 バイトのフィールドを指定します。ここには、ユーザー出口 (使用された場合) からの戻りコードが入ります。EXITRC フィールドには、最後に入ったユーザー出口からの戻りコードが入ります。ユーザー出口がないかまたは呼び出されない場合には、このフィールドには 2 進数のゼロが入ります。インストール・システム・データが出口から返される場合には、EXITRC フィールドはゼロに設定され、出口からの戻りコードが許可されたことを示します。

STATUS

2 バイトのフィールドを指定します。ここには、インストール・システム・データ状況を示す以下の 16 進状況コードが入ります。

0000 RACF インストール・システム・データが入出力域内にある。

0004 セキュリティー出口インストール・システム・データが入出力域内にある。

0008 ユーザーが現在サインオンされていない。

000C ユーザーが許可されていないためにインストール・システム・データが使用可能にならないか、あるいはユーザーは許可されているがインストール・システム・データが定義されていない。

0010 ユーザーは許可されたが、インストール・システム・データが要求されなかった。

0014 USERDATA が PSBWORK 域の長さを超えている。

0018 RACF がアクティブではなく、TRN=N が定義されている。

RESERVED

2 進数のゼロ (予約済み)

UL 2 バイトのフィールドを指定します。ここには、UL パラメーターの長さを含む、インストール・システム・データの長さを指定します。

USERDATA

可変長フィールドを指定します。ここには、ACEE またはユーザー・セキュリティー出口からのインストール・システム・データが入ります。インストール・システム・データの長さは、長さ (UL) フィールドを含めて 1026 バイトまでに制限されています。セキュリティー出口が 1026 よりも大きい値を返した場合には、IMS はそのインストール・システム・データを切り捨て、実際にアプリケーション・プログラムに返されたインストール・システム・データの量を表すために、長さフィールドを調整します。セキュリティー出口インストール・シス

テム・データが返されると、たとえパラメーター・リストに USERDATA パラメーターが入っていなくても、 IMS はデータをアプリケーション・プログラムに渡します。

RACF からの戻りコードが、呼び出しパラメーター・リストに指定されたリソースに対してユーザーが許可されていることを示している場合には、使用可能なインストール・システム・データが返されます。トランザクションを発信したユーザーが、トランザクションに関与する端末にサインオンされなくなった場合には、インストール・システム・データは返されません。セキュリティー出口がセキュリティーの決定に関与していると、インストール・システム・データはその出口によって提供されない場合もあります。ただし、いずれかの出口がインストール・システム・データを返す場合には、IMS はそれをアプリケーション・プログラムに返します。

提供される場合には、呼び出しパラメーター・リストが USERDATA パラメーターを指定しないときでも、インストール・システム・データがセキュリティー出口からアプリケーション・プログラムに返されます。その場合、入出力域の STATUS フィールドには、インストール・システム・データがあることを示すコード X'0004' が入ります。

使用法

AUTH 呼び出しは、ユーザーが AUTH 呼び出しで指定したリソースに対するアクセス許可を与えられているかどうかを判別します。AUTH は入出力 PCB を伴って出され、その機能は、アプリケーション・プログラムによって異なります。許可検査は、従属領域のタイプと GU 呼び出しが発行されているかどうかによって決まります。呼び出し機能は以下のとおりです。

- BMP では、AUTH は IMS 制御領域のユーザー ID またはインストール・システム固有のユーザー出口を使用して、呼び出しの状況を判別する。
- 入出力 PCB に GU 呼び出しを発行して成功した BMP の場合、AUTH は MPP の場合と同様に機能する。
- MPP では、AUTH は、アプリケーション・プログラムで使用するリソースに指定されたリソース・クラスについて、RACF を使用してユーザーの許可を検査する。

この呼び出しはインストール・システム・データとして RACF ユーザー・データを入出力域内に返すように要求することができるので、呼び出しの処理では必ず入出力域の STATUS フィールドを変更することになります。この STATUS フィールドは、アプリケーションに入出力域内のインストール・システム・データの状況を、使用可能か使用不能かで通知します。インストール・システム・データが定義されていないか、あるいは発信元のユーザーが IMS システムにサインオンされなくなったために、使用可能でない場合があります。

トランザクション許可にサポートされるセキュリティー出口 (DFSCTRN0 または DFSCTSE0) のいずれかで、IMS に戻るときにインストール・システム・データを示すことができます。出口がインストール・システム・データを返す場合、パラメーター・リストに USERDATA パラメーターが入っていなくても、そのデータはアプリケーションに返されます。STATUS フィールドは、インストール・システム・

データの発信を示すように設定されます。STATUS フィールドは、RACF インストール・システム・データか、セキュリティー出口インストール・システム・データのいずれかがあることを示します。

また、アプリケーション・プログラムは、実際の RACF 戻りコードの通知を受け取ります。入出力域内に FEEDBACK と表されているこの戻りコードは、アプリケーション・プログラムで使用して、矛盾する操作モードを検出し、代替アクションをとることができます。矛盾する操作モードの例としては、適切な RACF クラスが定義されていないこと、または要求されたリソースが RACF に正しく定義されていないことがあります。

入出力域内の FEEDBACK、EXITRC、および STATUS を検査することにより、アプリケーション・プログラムは、適切な RACF 定義およびリソースが定義されていないなどの問題を認識できるようになります。RACF を使用しており、定義されていないリソースを AUTH 呼び出しで参照する場合、PCB 状況コードはブランクに設定され、入出力域の FEEDBACK フィールドは、そのリソースが保護されていないことを示すように設定されます。

EXITRC の値はユーザー・セキュリティー出口により提供されるので、このフィールドを使用するには、終了操作を理解しておくことと、出口へのすべての変更がアプリケーション・エラーにつながる可能性があることを、知っておくことが必要です。操作エラーのために適切なリソースが保護されていない場合には、アプリケーションは、なんらかの方法でそのエラーに対処することができます。このフィードバックは、操作制御をより単純化し、アプリケーションをより柔軟なものにします。

関連資料: RACF の用語および概念については、他の情報単位に詳しい説明があります。詳細については、「*IMS V14* システム管理」および「*IMS V14* 出口ルーチン」を参照してください。

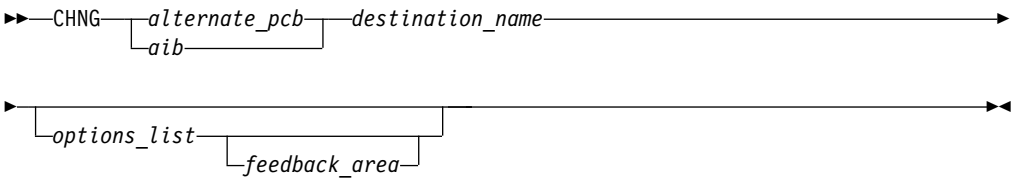
制約事項

入出力 PCB への GU 呼び出しが成功する前に、AUTH 呼び出しを発行してはなりません。

CHNG 呼び出し

変更 (CHNG) 呼び出しは、変更可能代替 PCB の宛先を、ユーザー指定の論理端末、LU 6.2 記述子、またはトランザクション・コードに設定します。スプール・アプリケーション・プログラム・インターフェース (スプール API) で CHNG 呼び出しを使用して、印刷データ・セットの特性を指定することもできます。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
CHNG	X		X		

パラメーター

alternate pcb
この呼び出しに使用する、変更可能代替 PCB を指定します。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。8 バイトの左寄せされるこのフィールドには、変更可能代替 PCB の名前を入れます。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

destination name
8 バイトのフィールドを指定します。ここには、メッセージの宛先名 (論理端末またはトランザクション・コード) が入ります。このパラメーターは入力パラメーターです。宛先名は、最大で 8 バイトです。LU 6.2 オプションを指定すると、IMS TM は代替 PCB 内の宛先名を DFSLU62bに設定します。LU 6.2 オプション・リストを指定する場合、宛先名パラメーターは無視されます。

LU 6.2 の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

宛先名は、OTMA から非 OTMA 宛先へのメッセージ通信を実装する場合にも使用されることがあります。この場合は、宛先名が IMS.PROCLIB の DFSYDTx メンバーにあるルーティング記述子の名前と一致していなければなりません。

制約事項: 宛先名の中には無効なものがあります。リソースの命名規則の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

options list
オプション・キーワードの 1 つを指定します。 このパラメーターは入力パラメーターです。リスト内の各オプションはコンマで区切られ、埋め込まれたブランクを含めてはなりません。オプション・リストの処理は、リスト内の最初のブラ

ンクに達したとき、または指定した長さのオプション・リストの処理が終了したときに終了します。拡張印刷機能または APPC のためにオプションを指定することができます。

APPC の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

options list の形式は、以下に示されたとおりです。

LL または **LLLL** [1, 2, 3]	**ZZ**	**keyword1=***variable1*
LLZZ または LLLLZZ の 4 バイトを含む、オプション・ストリングのハーフワードの長さ	ハーフワードのゼロ	コンマで区切られる CHNG オプション

注:

1. PLITDLI インターフェースを使用するアプリケーション・プログラムの場合、長さフィールドはフルワードです (LLLL)。ただし、LLLLZZ フィールドの長さはそれでも 4 バイトと見なされます。
2. 長さフィールドをゼロに設定すると、オプション・リストは無視されます。IMS TM は、*options list* パラメーターが指定されていないかのように、CHNG 呼び出しを処理します。
3. キーワードは、等号 (=) によって後続の変数と区切られます。変数のないキーワードは、必ずコンマまたはブランクで区切ります。

feedback area

オプション・リストに関するエラー情報をアプリケーション・プログラムに返すために使用するオプション・パラメーターを指定します。このパラメーターは出力パラメーターです。アプリケーション・プログラムが受け取る情報の量は、フィードバック域のサイズによって決まります。フィードバック域を指定しないと、返される状況コードはオプション・リスト・エラーの指示だけです。フィードバック域のサイズを、指定したオプション・リスト (最小で 8 ワード) のサイズの 1½ から 2 倍に指定すると、IMS TM はそのオプション・リストのエラーについて、さらに詳細な情報を返します。

以下の表は、呼び出しリスト内で IMS に渡されるフィードバック域の形式を示しています。

LL または **LLLL** [1, 2]	**ZZ**
LLZZ フィールドの 4 バイトを含む、フィードバック域のハーフワードの長さ	ハーフワードのゼロ

注:

1. PLITDLI インターフェースを使用するアプリケーション・プログラムの場合、長さフィールドはフルワードです (LLLL)。ただし、LLLLZZ フィールドの長さはそれでも 4 バイトと見なされます。
2. 長さフィールドをゼロに設定すると、フィードバック域は無視されます。IMS TM は、*feedback area* パラメーターが指定されていないかのように、CHNG 呼び出しを処理します。

IMS からアプリケーション・プログラムに返されるフィードバック域の出力形式は、以下のとおりです。

LLZZ または LLLLZZ	LL	フィードバック・データ
フィードバック域の入力形式で指定される長さフィールド	LL フィールドの 2 バイトを含む、IMS TM が返すフィードバック・データのハーフワードの長さ	IMS TM が返すデータ。一般に、フィードバック・データには、エラーが検出されたオプション・キーワードと、エラーの理由を示す 4 バイトの EBCDIC コード (括弧付き) が含まれます。複数のエラーは、それぞれコンマで区切られます。

使用法

CHNG 呼び出しは、出力メッセージをユーザー・システムまたは他のシステムの代替宛先に送信するときに使用します。CHNG 呼び出しを発行するときに、メッセージの宛先の名前を指定します。指定された代替 PCB は、以下のいずれかを行うまでは、その宛先に設定されたままになります。

- 別の CHNG 呼び出しを発行して宛先をリセットする。
- メッセージ・キューに GU (初回取り出し) 呼び出しを発行して、新しいメッセージの処理を開始する。この場合、CHNG 呼び出しで指定した PCB の名前は、たとえ無効になっても代替 PCB 内に残ります。
- アプリケーション・プログラムを終了する。アプリケーションを終了すると、IMS TM は宛先をブランクにリセットします。

CHNG 呼び出しを使用して、スプール API 機能を実行することができます。

スプール API 機能では、非急送代替 PCB への CHNG 呼び出しは、別個の JES スプール・データ・セットを作成します。(PURG 呼び出しは、非急送代替 PCB に対して発行された場合は、効果がありません。) PCB の宛先が JES スプールである場合、データ・セットが同期点により解放されるまで、非 JES スプール宛先に CHNG されることはできません。 CHNG 呼び出しで指定できるキーワードについては、以下で説明されています。

OTMA 環境において

IMS アプリケーション・プログラムが代替 PCB に対する CHNG 呼び出しを発行するときに、オプション・リストを指定している場合、出力の宛先を IMS オープン・トランザクション・マネージャーのクライアントに指定することはできません。

代替 PCB に対する CHNG 呼び出しを発行する、IMS アプリケーション・プログラム (オプション・リストを指定) は、宛先を判別するために、IMS に OTMA 事前ルーティングおよび宛先解決出口ルーチンを呼び出させることはありません。代替 PCB に対する CHNG 呼び出しを発行する IMS アプリケーション・プログラム (APPC 記述子を指定) では、IMS に OTMA 出口ルーチンを呼び出して宛先を判別します。これらの出口ルーチンについては、「*IMS V14* 出口ルーチン」を参照してください。

このアプリケーション・プログラムは、入出力 PCB に対する ISRT 呼び出しを発行して OTMA の宛先にデータを送信することもできます。

OTMA アプリケーション・プログラムでは、APPC の宛先に対する CHNG および ISRT 呼び出しを使用することができます。詳しくは、*IMS V14* コミュニケーションおよびコネクションを参照してください。

拡張印刷機能のオプション

IAFP キーワードは、CHNG 呼び出しをスプール API 機能の要求として識別します。 IAFP キーワードのパラメーターを以下に示します。

キーワード
　　説明

IAFP=*abc*

　　a — 紙送り制御オプションを指定

　　b — 保全性オプションを指定

　　c — メッセージ処理オプションを指定

以下の各オプションは、CHNG 呼び出しに拡張印刷機能を指定します。

紙送り制御オプション:1 文字の紙送り制御オプションは、ISRT または PURG 呼び出しが出されるときにメッセージ・データ内にある紙送り制御のタイプを示します。アプリケーション・プログラムで、データ・ストリームに適切な紙送り制御文字を挿入しなければなりません。IAFP キーワードについて、以下のいずれかの値を指定することができます。

A　データ・ストリームには、ASA 紙送り制御文字が入ります。

M　データ・ストリームには、機械紙送り制御文字が入ります。

N　データ・ストリームには、紙送り制御文字は入りません。

保全性オプション:1 文字の保全性オプションは、IAFP メッセージが入る IMS スプール・データ・セットの割り振りに IMS TM が使用する方式を示します。IAFP キーワードには、以下のいずれかのオプションを指定することができます。

0　IMS TM は、データ・セットの保護は試みません。アプリケーション・プログラムは、適切な OUTPUT 記述子オプションを使用して、後処理または保留状況を提供しなければなりません。IMS TM は、部分メッセージの印刷の防止と、すでに同期点に達しているメッセージが入るデータ・セットの割り振り解除を試みます。IMS スプール・データ・セットについてのエラー・メッセージを出すかどうかを制御するには、IAFP キーワードにメッセージ処理オプションを使用します。

1　IMS スプール・データ・セットは、割り振り時に SYSOUT HOLD キューに入れられます。IMS TM がメッセージ DFS00121 または DFS00141 を出す場合、オペレーターは、適切なデータ・セットを見つけるために SYSOUT HOLD キューを照会しなければなりません。IMS TM はデータ・セットを解放し、これを割り振り解除して同期点で印刷します。

　　保全性オプションに 1 を指定するときには、IAFP キーワードのメッセージ処理オプションには *M* を指定しなければなりません。

2 リモート宛先は、CHNG 呼び出しの *destination name* パラメーターで指定されています。IMS スプール・データ・セットは、割り振り時に SYSOUT リモート・ワークステーション IMSTEMP に入れられます。この宛先は、選択不能として定義に組み込まなければなりません。そのため、データ・セットが印刷されるように自動的に選択されることはありません。If IMS TM がメッセージ DFS00121 または DFS00141 を出す場合、オペレーターは、適切なデータ・セットを見つけるために IMSTEMP を照会しなければなりません。同期点で、IMS TM はデータ・セットを解放し、*destination name* パラメーターに指定されたリモート・ワークステーション ID に割り振り解除します。 2 の値は、IAFP OUTPUT オプションに指定されたすべての宛先を上書きします。

メッセージ処理オプション:1 文字のメッセージ処理オプションは、IMS TM が再始動時にメッセージ DFS00141 を出すかどうか、および動的割り振り障害に際してメッセージ DFS00121 を出すかどうかを示します。以下のいずれかのオプションを指定することができます。

0 DFS00121 および DFS00141 は出されません。ユーザーのアプリケーション・プログラムで、IAFP メッセージの保全性を制御します。

M 必要に応じて、DFS00121 および DFS00141 が出されます。 IMS TM で IAFP メッセージの保全性を制御します。

CHNG 呼び出しでは、以下の方法でデータ・セット特性を提供することができます。

- PRTO= オプションを直接使用する。
- TXTU= オプションを使用して、事前に作成したテキスト単位を参照する。
- OUTN= オプションを使用して、従属領域の JCL で OUTPUT JCL ステートメントを参照する。

IAFP キーワードを使用するときには、PRTO、TXTU、または OUTN のいずれかのオプションも指定しなければなりません。(オプション PRTO、TXTU、および OUTN のいずれか 1 つしか指定できません。) これらの追加オプションのいずれも指定しないか、複数指定するか、または無効な値で IAFP を指定した場合、IMS TM は、アプリケーション・プログラムに状況コード AR を返します。

キーワード
: 説明

PRTO=*outdes options*
: TSO OUTDES ステートメントで指定した、データ・セット処理オプションについて記述します。

PRTO= キーワードの形式は以下のとおりです。

LL	*outdes options*
LL の 2 バイトを含む、OUTDES プリンター・オプションのハーフワードの長さ	OUTDES プリンター・オプションの有効な組み合わせ
注: オプションによっては、MVS™ のリリース・レベルによって異なるものがあります。	

TXTU=アドレス
: テキスト単位ポインターのリストのアドレスを指定します。リスト (関係す

るテキスト単位がある) は、前の SETO 呼び出しで、あるいはユーザーのアプリケーション・プログラムで作成することができます。接頭部 LLZZ または LLLLZZ は、リストが入るバッファーに組み込まなければなりません。TXTU により、CHNG 呼び出しを発行する前に、アプリケーション・プログラムで SETO 呼び出しを発行して、OUTDES オプションのテキスト単位を作成することができます。

アプリケーション・プログラムで、CHNG 呼び出しを同じ OUTDES プリンター・オプションで複数回発行する場合、TXTU オプションは、CHNG 呼び出しごとに OUTDES オプションを作成する必要はないことを意味します。

OUTN=名前
最大 8 文字の文字ストリングを指定します。ここには、使用するプリンター処理オプションを識別する OUTPUT JCL ステートメントの名前が入ります。指定した OUTPUT DD ステートメントが、アプリケーション・プログラムが実行される領域の JCL に組み込まれていない場合には、アプリケーションがデータ・セットへのデータの挿入を試みた時点で動的割り振りエラーが発生します。

APPC オプション*:* 以下の APPC オプションは、CHNG 呼び出しに使用することができます。

キーワード
説明

LU=論理装置名
パートナー・アプリケーション・プログラムとの、LU 6.2 会話のパートナーの論理装置 (LU) 名を指定します。このキーワードは、MODE オプションおよび TPN オプションと共に使用され、会話を成立させます。LU 名では、国別文字を含む英数字のストリングを指定することができますが、先頭の文字を数字にすることはできません。LU 名がネットワーク修飾名の場合、名前は最高 17 文字の長さで、発信元システムのネットワーク ID のあとに '.' が付き、そのあとに LU 名が続く形になります。例えば、netwrkid.luname となります。LU 名とネットワーク ID はどちらも 1 文字から 8 文字の長さです。このオプションのデフォルトは DFSLU です。

MODE=モード名
パートナー・アプリケーション・プログラムとの、LU 6.2 会話のパートナーのモードを指定します。このキーワードは、LU オプションおよび TPN オプションと共に使用され、会話を成立させます。モード名には、国別文字を含む最大 8 文字の英数字ストリングを指定することができますが、先頭の文字を数字にすることはできません。MODE と SIDE の両オプションを指定した場合は、SIDE 項目に指定されたモード名は無視されますが、変更はされません。このオプションのデフォルトは DFSMODE です。

TPN=トランザクション・プログラム名
LU 6.2 会話でのパートナー・アプリケーション・プログラムのトランザクション・プログラム (TP) 名を指定します。このオプションは、LU キーワードおよび MODE キーワードと共に使用され、会話を成立させます。

TP 名には、最大 64 文字を指定することができ、00640 文字セットからブランクを除く任意の文字を入れることができます。00640 文字セットには、A から Z の文字、0 から 9 の数字、および 20 個の特殊文字が含まれて

います。このオプションのデフォルトは DFSASYNC です。00640 文字セットの詳細については、「CPI Communications Reference」を参照してください。 TPN オプションの形式は、以下のようになります。

LL	*tpn*
LL の 2 バイトを含む、TP 名のハーフワードの長さ	最大 64 文字の TP 名

IMS コマンド・プロセッサーで処理する TP 名には、IMS に対して有効な文字を入れなければなりません。例えば、英小文字の入った名前は処理できず、IMS コマンドのオペランドに使用されている場合には拒否されます。

SIDE=サイド情報項目名
サイド情報項目名を指定します。これを使用して、パートナー・アプリケーション・プログラムとの LU 6.2 会話を成立させることができます。 SIDE 名には、大文字の英字 (A から Z) および 0 から 9 の数字を含む、最大 8 文字を指定することができます。LU、MODE、または TPN キーワードを指定すると、SIDE キーワードが上書きされますが、サイド情報項目名は変更されません。このオプションにデフォルトはありません。

SYNC=NC
APPC/IMS 会話の同期レベルを変更します。N を選ぶと、同期レベルは NONE に設定されます。C を選ぶと、同期レベルは CONFIRM に設定されます。このオプションのデフォルトは C です。

TYPE=BM
APPC/IMS 会話のタイプを変更します。B を選ぶと、会話タイプは BASIC に設定されます。M を選ぶと、会話タイプは MAPPED に設定されます。このオプションのデフォルトは M です。

関連資料**:** APPC オプションおよびデフォルト・オプションの詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

オプション・リストのフィードバック域:オプション・リスト内にエラーが検出されると、オプション・リストのフィードバック域を使用してエラー情報がアプリケーションに返されます。

IMS はオプション・リスト全体の解析を試み、できるだけ多くのエラーに関する情報を返します。フィードバック域にすべてのエラー情報が入るだけの大きさがない場合には、スペースが許すかぎりの情報が返されます。この区域を指定しないと、状況コードがオプション・リストのエラーを示すだけです。

フィードバック域は、アプリケーション・プログラムが区域の長さを示す長さフィールドで初期設定しなければなりません。フィードバック域は、オプション・リストのおよそ 1.5 から 2 倍の長さ、または少なくとも 8 語あれば十分です。

エラー・コード

この節では、アプリケーションが受け取るエラー・コードについて説明します。

エラー・コード
理由

(0002) オプション・キーワードが認識されない。

このエラーの理由としては、以下が考えられます。

- キーワードのスペルが間違っている。
- キーワードのスペルは正しいが、後続の区切り文字が無効である。
- PRTO を表す長さが指定されたフィールドが、オプションの実際の長さよりも短い。
- 指示された呼び出しに対してキーワードが無効である。

(0004) オプション変数に指定した文字数が少なすぎるか、または多すぎる。呼び出しのオプション・リストのキーワードに続くオプション変数が、オプションの長さの制限を超えています。

(0006) オプション変数の長さフィールド (LL) が長すぎて、オプション・リストに入らない。オプション・リストの長さフィールド (LL) は、指定したオプション変数の終わりに達する前に、オプション・リストが終了することを示します。

(0008) オプション変数に無効な文字が入っているか、または英字で始まっていない。

(000A)

必須指定のオプション・キーワードが指定されなかった。

このエラーの理由としては、以下が考えられます。

- オプション・リストに 1 つ以上のキーワードが指定されたため、1 つ以上のキーワードを追加する必要がある。
- オプション・リストに指定した長さはゼロより大きいが、リストにオプションが入っていない。

(000C)

指定したオプション・キーワードの組み合わせが無効である。このエラーの原因としては、以下が考えられます。

- オプション・リストに指定された他のキーワードのためにそのキーワードが許可されない。
- オプション・キーワードが複数回指定されている。

(000E) 印刷データ・セット記述子の解析中に、IMS が 1 つ以上のオペランドでエラーを検出した。通常、IMS は z/OS サービス (SJF) を使用して印刷記述子 (PRTO= オプション変数) の妥当性検査を行います。 IMS が SJF を呼び出すとき、IMS は TSO OUTDES コマンドと同様の妥当性検査を要求します。したがって、IMS は出力記述子の変更を重要と見なしません。ユーザーのシステムについて有効な記述子は、MVS リリース・レベルの機能です。有効な記述子のリストおよび正しい構文については、TSO HELP OUTDES コマンドを使用してください。

IMS は、まず PRTO オプションの形式が SJF サービスを使用できる形式であることを確認します。それ以外の形式の場合には、IMS は、状況コード **AS**、エラー・コード (000E)、および記述エラー・メッセージを返します。SJF 処理中にエラーが検出された場合、SJF からのエラー・メッセージには、(R.C.=xxxx,REAS.=yyyyyyyy) 形式の情報およびエラーを示すエラー・メッセージが含まれます。

変数によっては、その範囲が初期設定パラメーターによって制御されます。初期設定パラメーターによって影響を受ける変数の例としては、コピーの最大数の値、許容されるリモート宛先、クラス、および用紙名があります。

制約事項

CHNG 呼び出しを使用して代替 PCB の宛先を設定または変更する前に、PURG 呼び出しを発行して、その PCB で作成していたメッセージが終了していることを、IMS に示さなければなりません。

LU 6.2 アーキテクチャーでは、LU 6.2 会話での CHNG 呼び出しでの ALTRESP PCB の使用を禁止しています。LU 6.2 会話は、IOPCB とのみ連動できます。アプリケーションは、既存の LU 6.2 会話 (同期) でメッセージを送信するか、または IOPCB を使用して、IMS に新規の会話 (非同期) を作成させます。LU 6.2 会話に関連した LTERM がないので、IOPCB のみが元の LU 6.2 会話を表します。

スプール API 機能では、非急送代替 PCB への CHNG 呼び出しは、別個の JES スプール・データ・セットを作成します。(PURG 呼び出しは、非急送代替 PCB に対して発行された場合は、効果がありません。) PCB の宛先が JES スプールである場合、データ・セットが同期点により解放されるまで、非 JES スプール宛先に CHNG されることはできません。

関連資料:

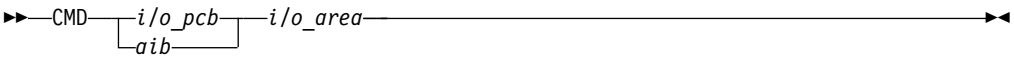
131 ページの『ISRT 呼び出し』

134 ページの『PURG 呼び出し』

CMD 呼び出し

コマンド (CMD) 呼び出しは、アプリケーション・プログラムが IMS コマンドを出せるようにします。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
CMD	X		X		

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは入出力パラメーターです。入出力域には、プログラムと IMS TM との間で受け渡しされる最大のセグメントが入るだけの大きさがなければなりません。

使用法

CMD 呼び出しは、IMS TM にコマンドを送り、その応答を受け取るために、GCMD 呼び出しとともに使用します。 CMD 呼び出しで IMS TM にコマンドを発行した後、CMD 呼び出しで CC 状況コードが返された場合のみ、IMS TM はそのコマンドを処理して、応答メッセージの最初のセグメントをアプリケーション・プログラム入出力域に返します。その後、アプリケーション・プログラムで GCMD 呼び出しを発行して、後続のすべてのメッセージ・セグメントを一度に 1 セグメントずつ取り出す必要があります。一般に、CMD コマンドと GCMD コマンドの呼び出しは、通常、ユーザーが端末で処理する機能を実行するために使用されます。これらのプログラムは、自動化操作プログラム (AO) アプリケーションと呼ばれます。

CMD 呼び出しを発行する前に、実行する IMS コマンドは、呼び出しで参照する入出力域になければなりません。CMD 呼び出しを発行すると、IMS TM は、入出力域から IMS 制御領域にコマンドを渡して処理を行います。IMS TM は、コマンドが処理されるまで、アプリケーション・プログラムを待ち状態にします。アプリケーション・プログラムは、IMS TM が応答を返すまでは待ち状態のままです。 (応答は、IMS TM がコマンドを受け取り、処理したことを意味します。) 非同期コマンドの場合、ユーザーは、コマンドが完了したときではなく、処理中に応答を受け取ります。

また、IMS TM アプリケーション・プログラムから DB2 コマンドを出すこともできます。コマンド呼び出しを発行し、/SSR コマンドと、それに続けて DB2 コマンドを使用してください。 IMS TM はコマンドを DB2 に経路指定します。DB2 はコマンドに対する応答を出し、IMS TM は DB2 応答をマスター端末オペレーター (MTO) に送ります。

制約事項

この呼び出しに対して、AIB は入出力 PCB を指定しなければなりません。

この呼び出しを使用するすべてのアプリケーション・プログラムは、セキュリティー管理者によって許可されていなければなりません。

CPI-C ドリブン・アプリケーション・プログラムから CMD 呼び出しを発行することはできません。

この呼び出しは、IFP または非メッセージ・ドリブン BMP ではサポートされません。

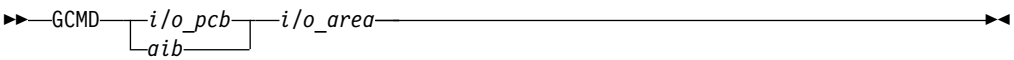
関連資料:

『GCMD 呼び出し』

GCMD 呼び出し

コマンド結果の取り出し (GCMD) 呼び出しは、アプリケーション・プログラムで CMD 呼び出しを使用して IMS コマンドを処理するときに、IMS TM から応答セグメントを取り出します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
GCMD	X		X		

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。入出力域には、プログラムと IMS TM との間で受け渡しされる最大のセグメントが入るだけの大きさがなければなりません。

使用法

CMD 呼び出しを発行すると、IMS TM は、最初のコマンド応答セグメントをアプリケーション・プログラムの入出力域に返します。コマンド応答セグメントを複数個返すコマンドを処理している場合は、GCMD 呼び出しを使用して、2 番目およびそれ以降のコマンド応答セグメントを取り出します。アプリケーション・プログラムで GCMD 呼び出しを発行するたびに、IMS TM は、コマンド応答セグメントを 1 つそのアプリケーション・プログラムの入出力域に返します。入出力域には、アプリケーション・プログラムが要求する、最長のメッセージ・セグメントが入るだけの長さがなければなりません。 IMS で可能な最大のセグメント・サイズは 132 バイト (4 バイトの LLZZ フィールドを含む) です。

一般的に、CMD 呼び出しおよび GCMD 呼び出しは、端末使用者が行う機能を実行するために使用されます。これらのプログラムは、自動化操作プログラム (AO) アプリケーションと呼ばれます。

PCB 状況コードは、GCMD 呼び出しの結果を示します。状況コードは、メッセージの GN 呼び出しの結果として出される状況コードと類似しています。QD の状況コードは、応答にそれ以上セグメントがないことを示します。QE の状況コードは、CMD 呼び出しが応答セグメントを生成しなかった後、GCMD 呼び出しが発行されたことを示します。ブランクの状況コード ('bb') は、セグメントが正常に取り出されたことを示します。

制約事項

この呼び出しに対して、AIB は入出力 PCB を指定しなければなりません。

この呼び出しを使用する AO アプリケーション・プログラムは、セキュリティー管理者によって許可されていなければなりません。

CPI-C ドリブン・アプリケーション・プログラムから、GCMD 呼び出しを発行することはできません。

この呼び出しは、IFP または非メッセージ・ドリブン BMP ではサポートされません。

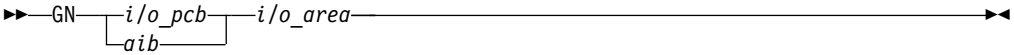
関連資料:

110 ページの『CMD 呼び出し』

GN 呼び出し

入力メッセージに複数のセグメントが入っている場合には、初回取り出し (GU) 呼び出しがメッセージの最初のセグメントを取り出し、後続取り出し (GN) 呼び出しが残りのセグメントを取り出します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
GN	X		X		

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。入出力域には、プログラムと IMS TM との間で受け渡しされる最大のセグメントが入るだけの大きさがなければなりません。

使用法

複数のセグメントが入っているメッセージを処理する場合には、GN 呼び出しを使用して、メッセージの 2 番目以降のセグメントを取り出します。アプリケーション・プログラムで GN 呼び出しを発行するたびに、IMS TM はメッセージ・セグメントを 1 つ、そのアプリケーション・プログラムの入出力域に返します。

BMP プログラムから GN 呼び出しを発行することができます。

制約事項

この呼び出しに対して、AIB は入出力 PCB を指定しなければなりません。

CPI-C ドリブン・アプリケーション・プログラムから GN 呼び出しを発行することはできません。

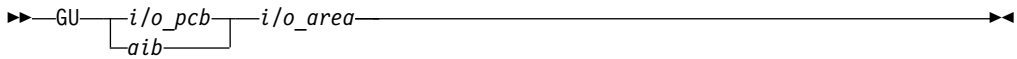
関連資料:

『GU 呼び出し』

GU 呼び出し

初回取り出し (GU) 呼び出しは、メッセージの最初のセグメントを取り出します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
GU	X		X		

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。入出力域には、プログラムと IMS TM との間で受け渡しされる最大のセグメントが入るだけの大きさがなければなりません。

使用法

MPP またはメッセージ・ドリブン BMP は、GN と GU の 2 つの呼び出しを使用してホストから入力メッセージを取り出します。GU 呼び出しでメッセージの最初のセグメントを取り出します。後続取り出し (GN) 呼び出しで 2 番目以降のセグメントを取り出します。

GU または GN を出して成功すると、IMS TM は、呼び出しで指定した入出力域にメッセージ・セグメントを返します。メッセージ・セグメントは、すべてが同じ長さではありません。セグメントの長さはさまざまなので、ユーザーの入出力域には、ユーザー・プログラムが受け取る最長のセグメントが入るだけの長さがなければなりません。セグメントの最初の 2 バイトには、セグメントの長さが入ります。

アプリケーション・プログラムは、他の DL/I 呼び出しを発行する前に、メッセージ・キューに GU 呼び出しを発行しなければなりません。 IMS TM が MPP をスケジュールすると、トランザクション・マネージャーは、最初のメッセージの最初のセグメントをメッセージ処理領域に転送します。 MPP が最初のメッセージに対する GU を出すとき、IMS TM はすでにそのメッセージを待ち状態にしています。アプリケーション・プログラムがプログラムの最初の呼び出しとして GU メッセージ呼び出しを発行しない場合、IMS TM はメッセージを再度転送しなければならず、メッセージを準備することによる効率は失われます。

MPP が複数のトランザクション・コードに応答する場合、MPP は、入力メッセージのテキストを検査して、そのメッセージに必要な処理を判別しなければなりません。

GU 呼び出しが成功すると、IMS TM は以下の情報を入出力 PCB マスクに入れます。

- メッセージを送った論理端末の名前。
- この呼び出しの状況コード。(*IMS V14* アプリケーション・プログラミングのトピック「入出力 PCB マスク」を参照)
- 入力接頭部。最初にキューに入れられたメッセージの日付、時刻、およびシーケンス番号を示します。IMS は、2 桁の年数を含む 8 バイトのローカル日付と、4 桁の年数を含む 12 バイトのタイム・スタンプ (地方時刻または UTC 時刻) の両方を返します。
- MOD 名 (MFS を使用している場合)。
- 端末使用者のユーザー ID。あるいはシステムでユーザー ID を使用しない場合には論理端末名。メッセージが BMP からのものである場合、IMS TM は BMP の PSB 名をこのフィールドに入れます。
- グループ名。SQL 呼び出しのセキュリティーのため DB2 により使用されます。

関連資料: 入出力 PCB マスクの形式の詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『入出力 PCB マスクの指定』を参照してください。

制約事項

この呼び出しに対して、AIB は入出力 PCB を指定しなければなりません。

CPI-C ドリブン・アプリケーション・プログラムから GU 呼び出しを発行することはできません。

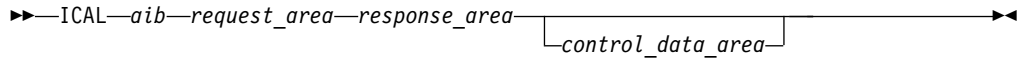
関連資料:

113 ページの『GN 呼び出し』

ICAL 呼び出し

IMS Call (ICAL) 呼び出しにより、IMS TM 環境で稼働するアプリケーション・プログラムは、z/OS または分散環境で稼働する非 IMS アプリケーション・プログラムまたはサービスにデータまたはサービスに関する同期要求を送信でき、また IMS トランザクションへの同期プログラム間通信を開始できます。

SENDRECV 副次機能の形式



RECEIVE 副次機能の形式

呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
ICAL	X		X		

パラメーター

aib

この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBERRXT

この長さ 4 バイトのフィールドには、OTMA、IMS Connect、IMS TM Resource Adapter、IMS Enterprise Suite SOAP Gateway サーバー、またはユーザー作成の IMS Connect クライアント・アプリケーションから戻された追加のエラー情報が入ります。デフォルトは 0 です。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBOALEN

4 バイトのフィールド。ICAL 呼び出しが発行された場合、このフィールドには要求域の長さが入ります。

ICAL 呼び出しに対する応答を受信すると、応答データが大きすぎて応答域に入らない場合は、AIBOALEN フィールドに応答データ全体の合計長が示

されます。応答域が小さすぎてすべての応答データが入らない場合、AIB 戻りコードは X'100'、AIB 理由コードは X'00C' です。それ以外の戻りコードを応答で受信した場合は、このフィールドは変更されません。

部分データが返された場合、このフィールドの値を使用して、応答データ・バッファーに必要なスペースを判別することができます。その後、アプリケーション・プログラムのバッファーを拡張し、RECEIVE 副次機能を指定して ICAL 呼び出しを発行することで、完全な応答メッセージをリトリーブすることができます。

AIBOAUSE
4 バイトのフィールド。ICAL が発行された場合、このフィールドには呼び出しリストで指定された出力応答域の長さが入ります。

ICAL 呼び出しに対する応答を受信すると、IMS はフィールドを更新し、応答域に返された応答メッセージの長さを示します。応答域の大きさが十分でないために部分データのみが返された場合は、AIBOAUSE には応答域に返されたデータの長さが示され、AIBOALEN には応答メッセージの合計長が示されます。

AIBOPLEN
4 バイトのフィールド。ICAL 呼び出しが発行された場合、このフィールドには呼び出しリストで指定された制御データ域の合計の長さが入ります。制御域が ICAL 呼び出しに指定されていない場合、このパラメーターは無視されます。制御域は、1 つから多数の制御データ項目で構成できます。制御データ域の合計の長さが 8,160,000 を超えるものであってはなりません。

AIBREASN
AIB 理由コード。

AIBRETRN
AIB 戻りコード。

AIBRSFLD
同期呼び出し処理の完了を待つ時間。この 4 バイトのフィールドには、100 分の 1 秒単位の時間値が入っています。

有効な範囲は 0 から 999999 までです。システム・デフォルトは 10 秒です。

- 指定した値が最大値より大きい場合は、最大値が使用されます。
- 値を 0 に設定した場合は、OTMA 記述子に指定されているタイムアウト値が使用されます。OTMA 記述子にタイムアウト値が指定されていない場合は、システム・デフォルトがタイムアウトとして使用されます。
- OTMA 宛先記述子に指定されたタイムアウト値と ICAL 呼び出しで指定されたタイムアウト値が異なる場合、OTMA は 2 つの値のうち小さい方を使用します。

タイムアウト値に達すると、同期コールアウト要求を発行した IMS アプリケーションは、戻りコード X'0100' および理由コード X'0104' を受信します。メッセージは廃棄されます。

AIBRSNM1
OTMA 記述子名。この 8 バイトの左寄せ英数字フィールドには、IMS 呼び出しの宛先を定義する OTMA 記述子の名前を入れる必要があります。

AIBRSNM2

この 8 バイト、英数字、左揃えのフィールドには、同期プログラム間通信での ICAL 呼び出しのターゲット・トランザクションに使用する IMS アプリケーション・プログラムの I/O PCB 内で、LTERM 名をオーバーライドするのに使用する論理端末名が入っています。 AIB で指定された名前は、OTMA 宛先記述子で指定された名前の代わりに使用されます。ただし、AIBRSNM2 で名前が指定されていない場合は、OTMA 記述子で指定された名前が使用されます。記述子にも AIB にも名前が検出されない場合は、IMS アプリケーション端末シンボル (PSTSYMBO) がターゲット・トランザクションのデフォルトの論理端末名として使用されます。

AIBSFUNC

副次機能コード。このフィールドには、8 バイトの副次機能コードを入れる必要があります。有効な副次機能コードは以下のとおりです。

SENDRECV

IMS アプリケーション・プログラムは、この副次機能を使用してメッセージを送信し、応答を待ちます。この副次機能は、同期型プログラム間通信に使用されます。

RECEIVE

IMS アプリケーション・プログラムは、この副次機能を使用して、以前に不完全であった ICAL 呼び出しから完全な応答データをリトリーブします。 SENDRECV 副次機能呼び出しが AIB 戻りコード X'0100' および理由コード X'000C' で完了した場合、応答データが応答域に入らなかったことを示します。アプリケーション・プログラムは、応答域を拡張することで、RECEIVE 副次機能呼び出しを使用して完全な応答をリトリーブすることができます。

AIBUTKN

マップ名。この 1 バイトから 8 バイトの英数字が指定された場合、左寄せフィールドには、メッセージのフォーマット設定やサービス識別の目的で使用する 1 文字から 8 文字のマップ名が示されます。このマップ名は、コールアウトの宛先に送信される OTMA 状態データ接頭部に含まれています。

request_area

この呼び出しで使用する要求域を指定します。 このパラメーターは入力パラメーターです。

この要求域には、IMS アプリケーション・プログラムから OTMA 記述子に指定されたアプリケーションに送信される要求メッセージ・データが入ります。AIBOALEN フィールドは、要求メッセージ・データの長さを指定します。ICAL 呼び出しは IMS TM メッセージ・キューをバイパスするため、要求域の形式に LLZZ フィールドは必要ありません。

要求メッセージを別の IMS アプリケーション・プログラムに送付する必要があると OTMA 記述子で指定されている場合 (TYPE=IMSTRAN)、LLZZ に続くデータ域の最初の 8 バイトに、LLZZ フィールドとトランザクション・コードを指定する必要があります。 MULTSEG で指定されたトランザクションでは、要求データに複数セグメント・メッセージの全体を含める必要があります。標準 IMS LLZZ 形式はセグメントごとに必要ですが、トランザクション・コードは最初のセグメントにのみ必要です。

LL　セグメントの長さを指定します。

ZZ　セグメントを 2 進ゼロに設定します。

response_area
この呼び出しで使用する応答域を指定します。このパラメーターは出力パラメーターです。

返されたデータ全体を応答域に収容できない場合、IMS はデータの一部を返します。データの一部が返された場合、AIBOAUSE フィールドには応答域に返されたデータの長さが入り、AIBOALEN には応答メッセージの実際の長さが入ります。

同期コールアウトの ICAL 呼び出しは IMS TM メッセージ・キューをバイパスするため、応答域の形式に LLZZ フィールドは必要ありません。ただし、別の IMS アプリケーションへの同期プログラム間通信の ICAL 呼び出しでは、LLZZ フィールドが必要です。同期プログラム間通信での LLZZ フィールドには、ターゲット IMS アプリケーションからの出力が入ります。同期プログラム通信要求は、メッセージ・キューをバイパスしません。

元の要求メッセージが別の IMS アプリケーション・プログラムに経路指定されていた場合、応答データは応答メッセージのセグメントごとに標準 LLZZ 形式に従います。

control_data_area
この呼び出しで使用する制御域を指定します。このパラメーターは、オプションの入力パラメーターです。この制御域は、IMS アプリケーション・プログラムから、OTMA 記述子に指定されたターゲット・クライアント・アプリケーションに送信されます。AIBOPLEN フィールドは、制御データの長さを指定する必要があります。ICAL 制御データは、1 つから多数の制御データ項目で構成できるので、同じ ICAL 呼び出しに複数のサービスと操作を指定できます。

それぞれの制御データ項目は、4 バイトの長さフィールドから始まり、タグ、データ、および終了タグが続きます。タグの長さは任意です。開始タグは、より小記号 (<)、タグ名、およびより大記号 (>) からなります。終了タグは、より小記号 (<)、スラッシュ (/)、開始タグ名と一致するタグ名、およびより大記号 (>) からなります。より小記号 (<)、およびスラッシュが前に付いたより大記号 (/>) は、EBCDIC で指定する必要があります。タグ名とデータ内容はバイナリーとして扱われ、ターゲット・クライアントに「現状のまま」渡されます。

ICAL 制御データの制御データ項目のフォーマットは、次のとおりです。

```
LLLL | <tag> | data ... | </tag>
```

IBM が初期に設定した制御データ項目が存在する場合があります。これらの項目のタグは、DFS から始まります。DFS 接頭部は、IBM 指定の制御データ項目に制限されています。

SOAP Gateway メッセージの場合は、タグ <DFSCNVTR>*CONVERTER_NAME*</DFSCNVTR> を使用して制御データにコンバーター名を指定できます。コンバーター名とタグは大文字の EBCDIC である必要があります。コンバーター名を指定すると、IMS Connect がメッセージの処理に使用する予定だったコンバーターの名前がオーバーライドされます。

次の表に、IBM が初期に設定する制御データ・タグ名とその説明を示します。

表 30. IBM が初期に設定する制御データ・タグ

開始タグ	データ	終了タグ	説明
<DFSCNVTR>	コンバーター名	</DFSCNVTR>	IMS Connect がメッセージの処理に使用するコンバーターの名前を指定します。

使用法: SENDRECV 副次機能

ICAL 呼び出しは、IMS アプリケーション・プログラムで、IMS メッセージ・キューを使用しない同期コールアウトに使用されます。IMS メッセージ・キューが使用されないため、同期コールアウト・メッセージは 32 KB のメッセージ・セグメントの制限を受けません。

ただし、IMS アプリケーション・プログラムで IMS トランザクションへの同期プログラム間通信処理に使用される ICAL 呼び出しは、IMS メッセージ・キューを使用します。同期プログラム間通信要求には、32 KB のメッセージ制限が適用されます。

この呼び出しを出す IMS アプリケーションを実行する前に、以下の作業が必要となります。

- アウトバウンド宛先ルーティング情報に対する OTMA 記述子を定義しておく必要があります。
- ICAL 要求が同期コールアウト目的である場合、IMS アプリケーションがコールアウトする外部アプリケーションまたはサーバーは、IMS OTMA RESUME TPIPE 機能を使用してコールアウト・メッセージを listen するように構成する必要があります。 ICAL 呼び出しがタイムアウトになる前に RESUME TPIPE がセットアップされないと、IMS アプリケーションにタイムアウト・エラーが返されます。
- ICAL 要求が同期プログラム間通信目的である場合、そのターゲットは、TRANSACT マクロまたは同等のタイプ 2 コマンド CREATE TRAN および UPDATE TRAN で定義された IMS トランザクションです。トランザクションを開始する必要があります。
- 共用キュー環境での同期プログラム間通信要求の場合、同じ共用キュー・グループ内のすべての IMS システムが 13.1 の MINVERS 値を持っている必要があります。

OTMA 宛先記述子と DL/I ICAL 呼び出しの両方に同期コールアウト・タイムアウト値がおいされている場合、IMS は 2 つのうち小さい方の値を使用します。

同期プログラム間通信の場合、ターゲット・トランザクションは、同じ IMS システム内、共用キューを介してアクセスできる IMS 内、または MSC でアクセスできるリモート IMS 内のいずれにあっても構いません。同期プログラム間通信要求は OTMA トランザクションとしてキューに入れられますが、OTMA 自体は必要ありません。

同期プログラム間通信のターゲット・アプリケーションは、元のアプリケーション・プログラムに返す前に、追加の同期プログラム間通信要求を発行することができます。任意の数の同期プログラム間通信要求をチェーニングすることができま

す。ただし、ネストされた同期プログラム間通信要求を作成する場合は、各 ICAL 呼び出しのタイムアウト値を検討してください。また、スケジュールする各ターゲット・トランザクションで使用可能な IMS 従属領域がなければなりません。最後に、通信シーケンス全体が解決するまでは、複数のプログラム間通信の流れによってデータベースがロック状態のまま保持される可能性があることを考慮してください。同じ同期プログラム間通信チェーン内にある複数のアプリケーションで、データベース・ロック競合が相互に発生する可能性があります。

同期プログラム間通信要求の ICAL 呼び出しがタイムアウトになった場合、あるいは最初の応答の後に複数の応答が返された場合は、IMS は追加の応答を遅延メッセージとして処理します。遅延メッセージに対するデフォルト応答は、そのメッセージをデキューすることです。遅延メッセージを保持したい場合、遅延メッセージの保持を要求するための tpipe を OTMA 宛先記述子内に構成するか、DFSMSCE0 出口ルーチンをコーディングしてそれらのメッセージを転送することができます。

高速機能領域から作成された同期プログラム間通信要求は、遅延応答メッセージをサポートしません。遅延応答メッセージは、後続の重複応答も含めてすべて破棄されます。

同期プログラム間通信要求の遅延応答メッセージは、OTMA クライアントに送信されますが、元の要求は OTMA クライアントから開始されたものではないため、DFSYIOE0 出口ルーチンを使用して、応答メッセージ用にデフォルトの 1 KB の OTMA メッセージ・ユーザー・データ接頭部を再ビルドする必要があります。

同期プログラム間通信要求の宛先記述子が、遅延メッセージを T パイプのキューに入れるように構成されている、または遅延メッセージを DFSMSCE0 ユーザー出口ルーチンを使用して転送するように構成されている場合、そのメッセージに対するアプリケーション GU 時刻における OTMA トランザクションの期限切れ検査は無効にされます。

トランザクション・セキュリティー指定 (TRN=Y) に応じて、ICAL 要求を発行するアプリケーションを実行している IMS 領域は、RACF および DFSCTRN0 ユーザー出口を呼び出して、ユーザーが ICAL 呼び出しを発行する権限があるかどうかを判別します。 APPC または OTMA トランザクションの場合、追加のセキュリティー指定が検査されます。 APPC または OTMA のセキュリティー・レベルが NONE に設定されている場合、TRN=Y が指定されている場合でも RACF および DFSCTRN0 ユーザー出口は呼び出されません。

同期プログラム間通信要求の場合、IMS はトランザクションを OTMA トランザクションとしてスケジュールします。 OTMA がアクティブではない場合でも、OTMA セキュリティー構成 (NONE、CHECK、FULL、または PROFILE) が使用されます。デフォルトのセキュリティー設定は FULL です。これは、IMS システムで OTMA が有効にされていない場合も使用されます。

同期プログラム間通信のセキュリティー構成は、次のコマンドを発行して変更することができます。

```
/SECURE OTMA TMEMBER DFSYICAL value
```

DFSYICAL は、TMEMBER を処理する同期プログラム間通信専用です。他のタイプの要求では使用されません。必要に応じて、*value* を NONE、CHECK、FULL、または PROFILE に置き換えます。

DFSYICAL の OTMA セキュリティーが FULL に設定されている場合、IMS は、ACEE をスケジュールする場合は必ずその ACEE を従属領域内に作成します。セキュリティー検査が必要な場合、IMS はこの ACEE を使用します。

DFSYICAL の OTMA セキュリティーが CHECK に設定されている場合、IMS はスケジュール時に ACEE を作成しません。セキュリティー検査が必要な場合、IMS は ACEE を制御領域内に作成します。

DFSYICAL の OTMA セキュリティーが NONE に設定されている場合、セキュリティー検査は実行されません。

使用法: RECEIVE 副次機能

SENDRECV 副次機能呼び出しが返したデータが多すぎて割り振り済みの応答バッファーに入らない場合 (AIB 戻りコード X'0100' および理由コード X'000C')、AIBOLEN フィールドの値が更新されて、完全な応答メッセージの長さが示されます。応答域のサイズを拡張してから RECEIVE 副次機能コードを指定して ICAL 呼び出しを発行し、完全な応答メッセージをリトリーブします。

元の ICAL 呼び出しの完全な応答データは、以下のいずれかのイベントが発生するまでは、IMS 制御領域に保持されています。

- アプリケーションが SENDRECV 副次機能コードを指定した新規の ICAL 呼び出しを発行する
- IMS アプリケーションが同期点に達するか異常終了する
- IMS アプリケーションが ROLB または CHECKPOINT 呼び出しを発行する

制約事項

外部コールアウト用の ICAL 呼び出しには、以下の制約事項があります。

- IMS アプリケーション・プログラムと外部アプリケーション・プログラムとの間の整合 2 フェーズ・コミットはサポートされていません。
- IMS Connect に接続されていない共用キュー環境では、IMS から ICAL 呼び出しを出すことができません。

同期プログラム間通信要求には、以下の制約事項があります。

- 同期プログラム間通信要求または応答メッセージでは、OTMA 入出力編集出口ルーチン (DFSYIOE0) は呼び出されません。
- 同期プログラム間通信要求では、「TM および MSC メッセージ経路指定および制御」出口ルーチン (DFSMSCE0) は呼び出されません。
- アプリケーション・プログラムの開始では、ターゲット・トランザクションは RRS コミットの有効範囲の一部ではありません。
- BMP および JBP アプリケーションは、DBCTL 環境では同期プログラム間通信要求を作成することができません。

- ターゲット・トランザクションには、高速機能 MSDB への読み取り専用アクセス権があります。
- ターゲット・トランザクションは、IMS 会話型トランザクションであってはなりません。
- 共用キュー環境に参加しているすべての IMS システムの DBRC MINVERS 値が 13.1 以上でなければなりません。

戻りコードおよび理由コード

次の表は、ICAL 呼び出しの戻りコードおよび理由コードの一覧です。

表 *31. ICAL* 呼び出しの戻りコードおよび理由コード

戻りコード	理由コード	拡張理由コード	説明
X'0000'	X'0000'	X'0000'	呼び出しは正常に完了しました。先へ進む。
X'0100'	X'000C'	X'0000'	出力応答データの一部が返された。 完全な応答メッセージをリトリーブするには、応答データ領域を拡張し、RECEIVE 副次機能を指定して新規の ICAL 呼び出しを発行する。
X'0100'	X'000C'	X'000D'	同期プログラム間通信要求への応答として IMS 通知メッセージまたはエラー・メッセージが返された。
X'0100'	X'0100'	デフォルト値は 0。値がゼロでない場合は外部アプリケーションにより設定されている。	出力応答データにエラー・メッセージが返された。
X'0100'	X'0100'	X'000D'	同期プログラム間通信要求が IMS メッセージと一緒に返された。
X'0100'	X'0100'	X'0004'	同期プログラム間通信要求への応答として IMS 通知メッセージまたはエラー・メッセージが返された。
X'0100'	X'0104'	X'0004'	要求がタイムアウトした。ICAL は外部アプリケーションに送信されなかった。
X'0100'	X'0104'	X'0008'	要求がタイムアウトした。ICAL は送信されたが、ACK を受信しなかった。
X'0100'	X'0104'	X'000C'	要求がタイムアウトした。ICAL は送信されたが、応答を受信しなかった。
X'0100'	X0104'	X'0010'	要求がタイムアウトした。ICAL は送信されたが、IMS が応答の処理に失敗した。
X'0100'	X'0104'	X'0020'	要求がタイムアウトした。同期プログラム間通信の ICAL 要求が送信されたが、応答を受信しなかった。

表 *31. ICAL* 呼び出しの戻りコードおよび理由コード *(*続き*)*

戻りコード	理由コード	拡張理由コード	説明
X'0100'	X'0108'	デフォルト値は 0。値がゼロでない場合は外部アプリケーションにより設定されている。	外部アプリケーションが要求メッセージを拒否した。
X'0100'	X'010C'	X'0000'	同期呼び出しがコマンド (/STOP や /PSTOP コマンドなど) によってクリアされた。
X'0100'	X'0110'	X'0000'	指定されたトランザクションがサポートされないため、要求メッセージは拒否された。trancode が見つからないか、指定されたトランザクションが IMS 会話型トランザクションか、CPIC トランザクションか、IMS コマンド・トランザクションであった。
X'0100'	X'0110'	X'0004'	ユーザーに同期プログラム間通信要求を出す権限がないため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0005'	同期プログラム間通信要求の処理に IMS が使用する tmember (DFSYICAL) が停止しているため、要求メッセージが拒否された。問題を解決するには、コマンド `/START TMEMBER DFSYICAL` を発行する。
X'0100'	X'0110'	X'0006'	同期プログラム間通信要求の処理に IMS が使用する tpipe (OTMA tmember の DFSTPIPE) が停止しているため、要求メッセージが拒否された。問題を解決するには、コマンド `/START TMEMBER DFSYICAL TPIPE DFSTPIPE` を発行する。
X'0100'	X'0110'	X'000D'	メッセージを処理するための内部ストレージ YTIB を IMS が取得できなかったため、要求メッセージは拒否された。
X'0100'	X'0110'	X'000E'	メッセージを処理するための DFSYTIB0 を IMS が活動化できなかったため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0010'	無効文字が含まれているため、前の応答メッセージ・ルーティングの TMEMBER 名または TPIPE 名は無効。宛先記述子を検査する。

表 31. ICAL 呼び出しの戻りコードおよび理由コード (続き)

戻りコード	理由コード	拡張理由コード	説明
X'0100'	X'0110'	X'0011'	宛先記述子から前の応答メッセージ・ルーティングの TMEMBER 名または TPIPE 名が欠落している。いずれかの値を指定する場合、両方を含める必要がある。
X'0100'	X'0110'	X'0012'	前の応答メッセージ・ルーティングの TMEMBER 名または TPIPE 名が正しくない。宛先記述子を検査する。
X'0100'	X'0110'	X'0013'	SMEM パラメーターと SYNCTP パラメーターを同時に指定することはできない。
X'0100'	X'0110'	X'0014'	宛先記述子内で、遅延メッセージ処理用の TPIPE 名が欠落しているか無効である。
X'0100'	X'0110'	X'0015'	要求が異なる IMS MINVERS 値を持つ共用キュー環境で作成されたため、要求メッセージは拒否された。共用キュー・グループ内の IMS システムの MINVERS 値は 13.1 でなければならない。
X'0100'	X'0110'	X'0016'	OTMA グローバル・メッセージがあふれ状態のため要求が拒否された。システム内で割り振られた OTMA メッセージ・ブロック (TIB) が多すぎる。
X'0100'	X'0110'	X'0020'	入力データの長さが正しくないため、要求メッセージは拒否された。セグメントの長さは要求で指定された LLZZ 値に一致しなければならない。要求のセグメントの合計長さは AIB の AIBOALEN 値に一致しなければならない。
X'0100'	X'0110'	X'0030'	トランザクションが現在使用不可なため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0031'	トランザクションが停止したため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0033'	プログラム間通信の宛先名が RCNT であるため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0034'	プログラム間通信の宛先名が CNT であるため、要求メッセージは拒否された。

表 *31. ICAL* 呼び出しの戻りコードおよび理由コード *(*続き*)*

戻りコード	理由コード	拡張理由コード	説明
X'0100'	X'0110'	X'0035'	宛先トランザクションは単一の入力セグメントのみを受け入れることができるため、要求メッセージは拒否された。複数の入力セグメントが要求に指定されている。
X'0100'	X'0110'	X'0036'	IMS キュー・マネージャーが挿入エラーを検出したため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0037'	IMS キュー・マネージャーが内部エラーを検出したため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0038'	キュー・オーバーフローが検出されたため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0039'	IMS が高速機能トランザクションを処理できなかったため、要求メッセージは拒否された。
X'0100'	X'0110'	X'003A'	IMS キュー・マネージャーがメッセージ接頭語を更新できなかったため、要求メッセージは拒否された。
X'0100'	X'0110'	X'003B'	IMS がトランザクションをエンキューできなかったため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0060'	応答を受信する前に同期プログラム間通信が取り消されたため、要求メッセージは拒否された。
X'0100'	X'0110'	X'0061'	ターゲット・トランザクションが IOPCB に応答せず、プログラム間通信を実行しないため、要求メッセージは拒否された。タイムアウトを回避するために ICAL は拒否された。この拒否は、REPLYCHK 記述子が宛先トランザクションについて YES に設定されているときに発生する。ICAL に関する非同期応答がある場合、REPLYCHK を NO に設定すれば ICAL は有効として処理される。
X'0100'	X'0110'	X'0070'	IMS は同期プログラム間通信 ICAL 呼び出しへの応答メッセージの処理に失敗した。出力メッセージ・セグメントの長さが 32K 制限を超えていた。

表 31. ICAL 呼び出しの戻りコードおよび理由コード (続き)

戻りコード	理由コード	拡張理由コード	説明
X'0100'	X'0110'	X'0071'	IMS は同期プログラム間通信 ICAL 呼び出しへの応答メッセージの処理に失敗した。IMS は応答メッセージを処理するための LUMP ストレージ・スペースが不足。
X'0100'	X'0110'	X'0072'	IMS は同期プログラム間通信 ICAL 呼び出しへの応答メッセージの処理に失敗した。IMS は応答メッセージの処理に必要なサブプール 231 からストレージを割り振ることができなかった。
X'0100'	X'0110'	X'0073'	IMS は IMS メッセージ・キューから応答メッセージの取得に失敗した。
X'0104'	X'0210'	X'0000'	入力域の長さ (AIBOALEN) がゼロに設定されている。
X'0104'	X'0214'	X'0000'	出力域の長さ (AIBOAUSE) がゼロに設定されている。
X'0104'	X'0218'	X'0000'	副次機能コードが不明または無効である。
X'0104'	X'0610'	X'0000'	要求入力域のアドレス・パラメーターが欠落している。
X'0104'	X'0614'	X'0000'	応答出力域のアドレス・パラメーターが欠落している。
X'0104'	X'1020'	X'0000'	記述子名が無効である。
X'0104'	X'1024'	X'0000'	タイムアウト値が無効である。
X'0104'	X'1028'	X'0000'	使用可能な追加応答データがないため、ICAL RECEIVE 呼び出しが拒否された。以前の ICAL SENDRECV 呼び出しからの追加応答データが既にリトリーブされたか、後続の ICAL SENDRECV 呼び出しが応答バッファーを消去した。
X'0104'	X'102C'	X'0000'	制御データを指定した ICAL 呼び出しが正しくない。AIBOPLEN 値がゼロである。
X'0104'	X'102C'	X'0004'	制御データを指定した ICAL 呼び出しが正しくない。制御データ域の後に追加のデータ域が検出された。
X'0104'	X'102C'	X'0008'	制御データを指定した ICAL 呼び出しが正しくない。OTMA 宛先記述子が TYPE=IMSCON ではない。

表 *31. ICAL* 呼び出しの戻りコードおよび理由コード *(*続き*)*

戻りコード	理由コード	拡張理由コード	説明
X'0104'	X'102C'	X'000C'	制御データを指定した ICAL 呼び出しが正しくない。RESUME TPIPE は制御データを受信できない (TMAMCRHQ_MODE に TMAMCRHQ_CTLDATA が指定されていない)。
X'0104'	X'102C'	X'0010'	制御データを指定した ICAL 呼び出しが正しくない。制御データの長さが制御データ項目と一致しない。
X'0104'	X'102C'	X'0014'	制御データを指定した ICAL 呼び出しが正しくない。制御データのタグ・エラー。
X'0104'	X'102C'	X'0018'	制御データを指定した ICAL 呼び出しが正しくない。AIBOPLEN 値が、許容最大長の 8,160,000 を超えている。
X'0108'	X'0008'	X'0000'	IMS が ICAL 呼び出し用の PSTICALO (内部ストレージ) の解放に失敗した。
X'0108'	X'0010'	X'0000'	専用ストレージを取得できない。入力要求データのサイズが大きすぎる可能性がある。
X'0108'	X'0570'	X'0000'	PSTICALO にある内部バッファー・ストレージが無効であるため、ICAL RECEIVE 呼び出しが拒否された。
X'0108'	X'0580'	X'0004'	要求メッセージを外部アプリケーションに送信できない。 IMS がシャットダウンしているとき。
X'0108'	X'0580'	X'0008'	要求メッセージを外部アプリケーションに送信できない。 IMS コールアウト機能が使用不可。
X'0108'	X'0580'	X'000C'	要求メッセージを外部アプリケーションに送信できない。 OTMA メンバーが見つからないか非アクティブ。
X'0108'	X'0580'	X'0010'	要求メッセージを外部アプリケーションに送信できない。 OTMA TPIPE が見つからないか停止している。
X'0108'	X'0580'	X'0014'	要求メッセージを外部アプリケーションに送信できない。 IMS は要求をキューに入れるためのストレージを獲得できなかった。

表 31. ICAL 呼び出しの戻りコードおよび理由コード (続き)

戻りコード	理由コード	拡張理由コード	説明
X'0108'	X'0580'	X'0018'	要求メッセージを外部アプリケーションに送信できない。 IMS はメッセージを処理するための LUMP ストレージを獲得できなかった。
X'0108'	X'0580'	X'001C'	要求メッセージを外部アプリケーションに送信できない。 IMS は ICAL 呼び出しを処理するために OTMA に接触できなかった。
X'0108'	X'0580'	X'0100'	IMS は同期プログラム間通信要求の処理に必要な LUMP ストレージ・スペースの獲得に失敗した。
X'0108'	X'0580'	X'0104'	OTMA は同期プログラム間通信の処理に失敗した。関連 X'67D0' ログ・レコードを参照。
X'0108'	X'0584'	X'0004'	外部アプリケーションからの応答出力メッセージを処理できない。応答メッセージにデータがない。
X'0108'	X'0584'	X'0008'	外部アプリケーションからの応答出力メッセージを処理できない。応答メッセージの XCF バッファー長が正しくない。
X'0108'	X'0584'	X'000C'	外部アプリケーションからの応答メッセージを処理できない。 IMS は応答メッセージ処理のためのストレージ割り振りに失敗した。
X'0108'	X'0584'	X'0010'	外部アプリケーションからの応答メッセージを処理できない。複数セグメント応答メッセージにヌル・セグメントが見つかった。
X'0108'	X'0588'	デフォルト値は 0。値がゼロでない場合は IMS Connect により設定されている。	IMS Connect が応答の処理に失敗した。応答データが返されなかった。
X'0108'	X'058C'	デフォルト値は 0。値がゼロでない場合は IMS Connect により設定されている。	IMS Connect が応答の処理に失敗した。外部クライアント・アプリケーションから完全または部分的な生データが返された。

関連概念:

OTMA 記述子 (コミュニケーションおよびコネクション)

関連資料:

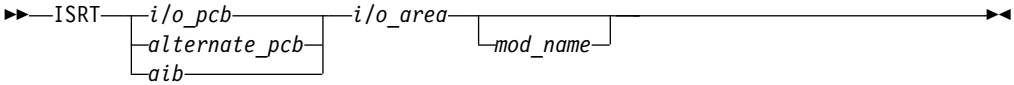
IMS によって設定される AIB 戻りコードおよび理由コード (メッセージおよびコード)

ISRT 呼び出し

挿入 (ISRT) 呼び出しは、呼び出しに指定する宛先にメッセージ・セグメントを 1 つ送ります。宛先は、呼び出しパラメーターに指定した入出力 PCB、代替 PCB、または AIB で表されます。

スプール API 機能の場合、ISRT 呼び出しは JES スプールへのデータの書き込みにも使用されます。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
ISRT	X		X		

パラメーター

i/0 pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

alternate pcb
この呼び出しに使用する PCB を指定します。これらのパラメーターは、入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、PCB 名 IOPCBbbb (TP PCB を使用する場合)、または 代替 PCB (代替 PCB を使用する場合) を指定する必要があります。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。 このパラメーターは入力パラ

メーターです。入出力域には、アプリケーション・プログラムと IMS TM との間で受け渡しされる最大のセグメントが入るだけの大きさがなければなりません。

mod name
この出力メッセージに使用する MOD を指定します。 このパラメーターは入力パラメーターです。8 バイトの MOD 名は、左寄せして、必要なだけのブランクを埋め込みます。出力を受け取る端末が MFS を使用しない場合には、このパラメーターは無視されます。有効な MOD 名を指定すると、IMS TM はその MOD を使用して、ユーザーが送る出力メッセージのために画面の形式を設定します。

使用法

ISRT 呼び出しを発行して成功させるには、アプリケーション・プログラムで、まずアプリケーション・プログラムの入出力域に送りたいメッセージを作成しなければなりません。 ISRT は、入出力 PCB または代替 PCB 内の宛先名、および呼び出しに指定した入出力域を使用して、送るべきメッセージを位置指定します。さらに、ISRT 呼び出しは、出力メッセージをアプリケーション・プログラムから別の端末に送ります。ISRT は、出すたびにメッセージ・セグメントを 1 つ送るので、アプリケーション・プログラムは、入出力域のメッセージの各セグメントごとに ISRT 呼び出しを 1 つ出さなければなりません。

また、画面の形式を変更したい場合にも MOD 名を指定することができます。例えば、アプリケーション・プログラムがエラーを検出し、それを端末使用者に通知しなければならない場合、MOD 名を指定して、画面の形式をエラー・メッセージを受け取れるように設定することができます。出力メッセージの最初のセグメントに MOD 名を指定することができる DL/I 呼び出しは、ISRT と PURG だけです。

アプリケーション・プログラムで 1 つ以上の ISRT 呼び出しを発行すると、IMS TM は、送られるメッセージ・セグメントをグループ化してメッセージ・キューに入れます。IMS TM は、アプリケーション・プログラムが以下のいずれかを行うと、宛先にメッセージ・セグメントを送ります。

- GU 呼び出しを発行して、次のメッセージの最初のセグメントを取り出す。
- コミット・ポイントに達する。
- 急送代替 PCB で PURG 呼び出しを発行する。

アプリケーションは、ISRT 呼び出しを使用して会話型プログラムの他の端末に応答を出し、複数のプログラム間で会話の受け渡しを行わなければなりません。

共用キュー環境において

MSGQ 構造が完全である場合は、共用キュー環境内では、 ISRT 呼び出し上で STATUSQF を受信することができます。 MSGQ 構造が完全である場合は、以下のうちの 1 つが起こります。

- ISRT が複数セグメントメッセージの場合は、STATUSQF が受信されます。
- 複数セグメント用の ISRT が正常に(十分なスペースで)終了したが 、 PURG または CHKP 時間で使用可能な十分なスペースが検出されなかった場合、アプリケーションは、ABENDU0370 で異常終了します。

- ISRT が複数セグメントメッセージの場合は、STATUSQF が受信できます。すべての使用可能な装置相対レコード番号 (DRRN) が消耗する原因となるメッセージをプログラムが挿入し続ける場合に、IMS は、ABENDU0758 で失敗します。すべて使用可能な DRRN を消耗する前に、プログラムがチェックポイントを発行する場合は、キュー・バッファーは解放され、メッセージは「unresolved UOWEs.」としてログに書き込まれます。オリジナルの type01 と type03 のログ・レコードを含む複数のログは、スペースが使用可能になり、再利用が不可能である場合に、後からメッセージをその構造に挿入する必要があります。 IMS はユーザーにチェックポイントの時間を毎回指摘する DFS1994I メッセージを発行します。

スプール *API* 機能

JES スプールへのデータの書き込みに、ISRT 呼び出しを使用することができます。これらの書き込みは BSAM を使用して行われ、可能であれば、各 BSAM 『書き込み』 はアプリケーション・プログラムのバッファー域から直接行われます。

制約事項**:** BSAM は、16 MB 境界より上の SYSOUT データ・セットの入出力域をサポートしません。 IMS は、16 MB 境界より上に入出力域を検出した場合、BSAM 書き込みを実行する前に、アプリケーション・データをその境界より下の作業域に移します。入出力域がすでにその 16 MB 境界より下にある場合は、書き込みは入出力域から直接行われます。パフォーマンス上必要にならないかぎり、通常と違う手順で 16 MB 境界より下へ入出力域を移動しないでください。

IAFP 処理用に設定された代替 PCB に対する ISRT 呼び出しを発行する場合、入出力域の接頭部として可変長レコードの BSAM ブロック記述子ワードを付けます。

LL または **LLLL**[1, 2]	**ZZ**[2]	**II**[3]	**zz**[3]
LLZZ フィールドの 4 バイトを含む、入出力域またはブロックのハーフワードの長さ	ハーフワードのゼロ	llzz フィールドの 4 バイトを含む、論理レコードまたはセグメントのハーフワードの長さ	ハーフワードのゼロ

注:

1. PLITDLI インターフェースを使用するアプリケーション・プログラムの場合、長さフィールドはフルワードです (LLLL)。ただし、LLLLZZ フィールドの長さはそれでも 4 バイトと見なされます。
2. LLZZ は、BSAM ブロック記述子ワード (BDW) と同等です。
3. llzz は、BSAM レコード記述子ワード (RDW) と同等です。

制約事項

CPI-C ドリブン・アプリケーション・プログラムは、代替 PCB に対する ISRT 呼び出しだけを出すことができます。

次のメッセージを取り出す前、あるいはコミット・ポイントを出す前にメッセージ・セグメントを送りたい場合には、PURG 呼び出しを使用しなければなりません。

MOD 名は、メッセージごとに 1 回だけ、メッセージを始める最初の ISRT 呼び出しまたは PURG 呼び出しで指定することができます。

BSAM は、16 MB 境界より上の SYSOUT データの入出力域をサポートしません。

関連資料:

101 ページの『CHNG 呼び出し』

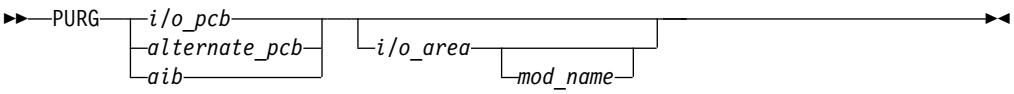
『PURG 呼び出し』

PURG 呼び出し

パージ (PURG) 呼び出しを使用すると、アプリケーション・プログラムは、次の入力メッセージを取り出すか、またはコミット・ポイントを出す前に、指定した宛先に 1 つ以上の出力メッセージ・セグメント (ISRT 呼び出しで設定) を送ることができます。

スプール API 機能では、PURG 呼び出しは、即時印刷のために印刷データ・セットを解放するためにも使用できます。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
PURG	X		X		

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

alternate pcb
呼び出しに使用する PCB を指定します。これらのパラメーターは、入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、PCB 名 IOPCBbbb (TP PCB を使用する場合)、または 代替 PCB (代替 PCB を使用する場合) を指定する必要があります。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。 このパラメーターは入力パラメーターです。入出力域には、プログラムと IMS TM との間で受け渡しされる最大のセグメントが入るだけの大きさがなければなりません。

mod name
この出力メッセージに使用する MOD を指定します。 このパラメーターは入力パラメーターです。8 バイトの MOD 名は、左寄せして、必要なだけのブランクを埋め込みます。PURG は、出力メッセージの最初のメッセージ・セグメントに MOD 名を指定することができます。出力を受け取る端末が MFS を使用しない場合には、このパラメーターは無視されます。有効な MOD 名を指定すると、IMS TM はその MOD を使用して、ユーザーが送る出力メッセージのために画面の形式を設定します。

使用法

PURG 呼び出しは、複数の異なる端末に出力メッセージを送信する場合に使用します。 PURG 呼び出しは、指定した入出力 PCB または代替 PCB (ISRT 呼び出しで) に対して作成したメッセージが完了したことを、IMS TM に通知します。 IMS TM は、1 つのメッセージとして 1 つの PCB に挿入されたメッセージ・セグメントを収集し、PURG 呼び出しにリストされた代替 PCB の宛先名にそのメッセージを送ります。

PURG 呼び出しのパラメーターに入出力域を指定する場合、PURG は、次のメッセージの最初のセグメントを挿入する ISRT 呼び出しとして実行されます。入出力域を識別する際、MOD 名を指定して画面の形式を変更することもできます。

OTMA 環境において

IMS アプリケーション・プログラムが PURG 呼び出しを発行すると、IMS が Open Transaction Manager Access (OTMA) の事前経路指定および宛先解決出口ルーチンを呼び出して宛先を判別します。これらの出口ルーチンについては、「*IMS V14* 出口ルーチン」を参照してください。

共用キュー環境において

MSGQ 構造が完全である場合は、共用キュー環境内では、 PURG 呼び出し上で STATUSQF を受信することができます。 MSGQ 構造が完全である場合は、以下のうちの 1 つが起こります。

- PURG が複数セグメントメッセージの場合は、STATUSQF が受信されます。
- 複数セグメント用の PURG が正常に終了したが (十分なスペース)、 PURG または CHKP 時間で使用可能な十分なスペースが検出されなかった場合、アプリケーションは、ABENDU0370 で異常終了します。

急送代替 PCB と共に PURG 呼び出しを使用することにより、即時印刷のために印刷データ・セットを解放することができます。入出力域による PURG 呼び出しを発行すると、IMS は呼び出しを、パージ要求および入出力域が提供するデータの挿入、といった 2 つの機能として扱います。

PURG 呼び出しを発行すると、以下のことが行われます。

- 急送代替 PCB に対する呼び出しが出されると、データ・セットがクローズされ、割り振り解除され、印刷のために解放される。宛先はリセットされます。
- 入出力域を伴って非急送代替 PCB に対する呼び出しが出されると、パージ機能は無視され、呼び出しの挿入部分のデータが印刷データ・セットに入れられる。つまり、この呼び出しは ISRT 呼び出しと同様の動作を行います。
- 入出力域を伴わずに急送代替 PCB に対する呼び出しが出されると、データ・セットがクローズされ、割り振り解除され、印刷のために解放される。IMS はブランクの状況コードを返します。
- 入出力域を伴わずに非急送代替 PCB に対する呼び出しが出されると、いかなるアクションも行われない。

制約事項

CPI-C ドリブン・アプリケーション・プログラムは、代替 PCB に対する PURG 呼び出しだけを出すことができます。

MOD 名は、メッセージごとに 1 回だけ、メッセージを始める最初の ISRT 呼び出しまたは PURG 呼び出しで指定することができます。会話型トランザクションで最初の ISRT が SPA の場合、MOD 名は SPA ISRT かメッセージ・セグメントの最初の ISRT のいずれかで指定することができます。

この呼び出しは、IFP ではサポートされません。

同期 APPC/OTMA 会話、または OTMA 接頭部に TMAMIPRG 標識が設定された OTMA コミット後送信 (CM0) トランザクションでは、TP PCB での PURG 呼び出しが無視されます。次の ISRT 呼び出しは、現在のメッセージの次のセグメントについて処理されます。

関連資料:

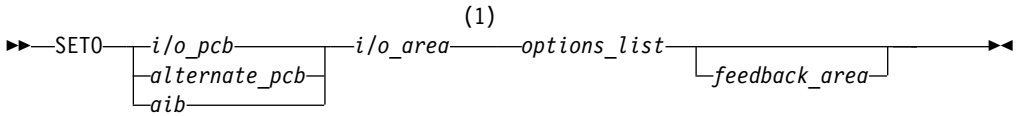
101 ページの『CHNG 呼び出し』

131 ページの『ISRT 呼び出し』

SETO 呼び出し

オプション設定 (SETO) 呼び出しを使用すると、IMS アプリケーション・プログラムで処理オプションを設定することができます。SETO 呼び出しは、スプール API 機能に処理オプションを設定するためにも使用できます。

フォーマット



注:

1 APPC オプションを指定する呼び出しには、入出力域パラメーターは使用されません。

呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
SETO	X		X		

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

alternate pcb
この呼び出しに使用する TP PCB または代替 PCB を指定します。これらのパラメーターは、入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには、PCB 名 IOPCBbbb (TP PCB を使用する場合)、または 代替 PCB (代替 PCB を使用する場合) を指定する必要があります。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。拡張印刷機能が入るオプション・リストを指定する場合には、入出力域を指定しなければなりません。APPC オプションを使用する場合には、入出力域パラメーターは任意指定です。

拡張印刷機能オプションを使用する場合、入出力域は最小でも 4 KB 必要です。入出力域の長さが接頭部 LLZZ または LLLLZZ を含めても 4096 バイトより小さい場合、AJ 状況コードが返されます。いったん入出力域内にテキスト単位区域ができたら、その区域を新しい区域にコピーしてはなりません。SETO 呼び出しで渡す入出力域には、接頭部 LLZZ (PL/I の場合は LLLLZZ) を入れるようにします。

LLLL は DL/I 呼び出しインターフェースにのみ適用されます。

options list

複数のオプション・キーワードを指定します。この入力パラメーターは必須です。リスト内の各オプションはコンマで区切られ、埋め込まれたブランクを含めてはなりません。オプション・リストの処理は、リスト内の最初のブランクに達したとき、または指定した長さのオプション・リストの処理が終了したときに終了します。拡張印刷機能または APPC のためにオプションを指定することができます。指定できるオプションについては、『拡張印刷機能のオプション』と『APPC オプション』で説明されています。

options list の形式は、以下のようになります。

LL または **LLLL**[1]、[2]	**ZZ**	**keyword=***variable1*
LLZZ または LLLLZZ の 4 バイトを含む、オプション・ストリングのハーフワードの長さ	ハーフワードのゼロ	コンマで区切られる SETO オプション

注:

1. PLITDLI インターフェースを使用するアプリケーション・プログラムの場合、長さフィールドはフルワードです (LLLL)。ただし、LLLLZZ フィールドの長さはそれでも 4 バイトと見なされます。
2. 長さフィールドをゼロに設定すると、オプション・リストは無視されます。IMS TM は、*options list* パラメーターが指定されていないかのように、SETO 呼び出しを処理します。

feedback area

オプション・リストに関するエラー情報をアプリケーション・プログラムに返すために使用するオプション・パラメーターを指定します。このパラメーターは出力パラメーターです。アプリケーション・プログラムが受け取る情報の量は、フィードバック域のサイズによって決まります。フィードバック域を指定しないと、返される状況コードはオプション・リスト区域の表示だけです。フィードバック域のサイズを、指定したオプション・リスト (最小で 8 ワード) のサイズの 1½ から 2 倍に指定すると、IMS TM はそのオプション・リストのエラーについて、さらに詳細な情報を返します。

呼び出しリスト内で IMS TM に渡される *feedback area* の形式は、以下のようになります。

LL または **LLLL**[1, 2]	**ZZ**
LLZZ フィールドの 4 バイトを含む、フィードバック域のハーフワードの長さ	ハーフワードのゼロ

LL または **LLLL**[1、 2]	**ZZ**

注:

1. PLITDLI インターフェースを使用するアプリケーション・プログラムの場合、長さフィールドはフルワードです (LLLL)。ただし、LLLLZZ フィールドの長さはそれでも 4 バイトと見なされます。
2. 長さフィールドをゼロに設定すると、フィードバック域は無視されます。IMS TM は、*feedback area* パラメーターが指定されていないかのように、SETO 呼び出しを処理します。

IMS TM からアプリケーション・プログラムに返されるフィードバック域の出力形式は、以下のとおりです。

LLZZ または **LLLLZZ**	**LL**	*feedback area*
フィードバック域の入力形式で指定される長さフィールド	**LL** フィールドの 2 バイトを含む、IMS TM が返すフィードバック・データのハーフワードの長さ	IMS TM が返すデータ。一般に、フィードバック・データには、エラーが検出されたオプション・キーワードと、エラーの理由を示す 4 バイトの EBCDIC コード (括弧付き) が含まれます。複数のエラーは、それぞれコンマで区切られます。

使用法

SETO 呼び出しを使用すると、処理オプションを設定することができます。

SETO 呼び出しを使用すれば、解析の実行、およびデータ・セットの OUTPUT 記述子のテキスト構成に要するオーバーヘッドを減らすことができます。アプリケーション・プログラムでインストール時に一連の記述子を複数回使用できる場合には、アプリケーションで SETO 呼び出しを使用して、スプール API に印刷データ・セットの特性を提供することができます。SETO 呼び出しを処理する際、OUTPUT オプションが解析され、アプリケーションが提供する作業域に動的 OUTPUT テキスト単位が構成されます。事前に作成されたテキスト単位をアプリケーションが受け取った後は、CHNG 呼び出しおよび TXTU= オプションを使用して、解析およびテキスト単位作成のオーバーヘッドなしで、データ・セットに印刷特性を提供することができます。

別のプログラミング手法でテキスト単位が事前に作成されている場合には、SETO 呼び出しを使用してテキスト単位を作成する必要はありません。

関連資料**:** スプール API の詳細については、「*IMS V14* アプリケーション・プログラミング」を参照してください。

OTMA 環境において

SETO 呼び出しを発行する IMS アプリケーション・プログラムでは、宛先の判別のために IMS が Open Transaction Manager Access (OTMA) の事前経路指定および宛先解決出口ルーチンを呼び出すことはありません。これらの出口ルーチンについては、「*IMS V14* 出口ルーチン」を参照してください。

SETO 呼び出しを発行する既存の IMS アプリケーション・プログラムは、要求どおりに実行されないことがあります。これは、プログラムが OTMA によって発信されたトランザクションを処理する場合、プログラムに戻りコードが返されるためです。また、SETO 呼び出しを発行する APPC/IMS アプリケーション・プログラムは、暗黙の OTMA サポートが必要な場合でも、修正する必要がない場合もあります。

この問題を解決するには、INQY 呼び出しを使用してから、SETO 呼び出しを発行します。アプリケーション・プログラムは、INQY 呼び出しの出力を使用してトランザクションが OTMA によって発信されたものであるかどうかを判別し、SETO 呼び出しをバイパスします。

拡張印刷機能のオプション

PRTO= キーワードが、SETO 呼び出しをスプール API 要求と識別します。

キーワード
　　説明

PRTO=*outdes options*
　　TSO OUTDES ステートメントで指定した、データ・セット処理オプションについて記述します。 **PRTO** キーワードの形式は以下のとおりです。

LL	*outdes options*
LL の 2 バイトを含む、OUTDES プリンター・オプションのハーフワードの長さ	複数の OUTDES プリンター・オプションの有効な任意の組み合わせ (コンマで区切る)
注: TSO OUTDES オプションについては、「*z/OS MVS Programming: Authorized Assembler Services Reference*」を参照してください。オプションによっては、MVS のリリース・レベルによって異なるものがあります。	

z/OS が OUTDES プリンター・オプションにエラーを検出した場合、状況コード AS がアプリケーション・プログラムに返されます。

APPC オプション

以下のオプションは、SETO 呼び出しに使用することができます。

SEND_ERROR
　　メッセージが送られたときに、IMS LU マネージャーは、入出力 PCB または代替 PCB に関係する会話で SEND_ERROR を出すようになります。急送 PCB へのメッセージは、PURG 呼び出しまたは同期点処理 (どちらが先かにかかわらず) の際に送られます。非急送 PCB へのメッセージは、同期点処理のときに送られます。

　　このオプションは LU 6.2 でのみ使用され、LU 6.2 以外の装置に指定された場合には無視されます。

　　このオプションは、DEALLOCATE_ABEND オプションと一緒に使用することはできません。オプション・リストに両方のオプションをコーディングすると、アプリケーションに状況コード AR が返されます。

DEALLOCATE_ABEND
　　メッセージを送る際に SEND_ERROR に続いて DEALLOCATE_ABEND を出

すことにより、会話を割り振り解除します。DEALLOCATE_ABEND オプションを指定した SETO 呼び出しを発行すると、PCB に対する後続の ISRT 呼び出しが状況コード QH で拒否されます。

このオプションは、LU 6.2 装置についてのみ適用されます。LU 6.2 以外の装置に指定すると、PCB に対する後続の ISRT 呼び出しが状況コード QH で拒否されます。

IFP 領域内の TP PCB で SETO 呼び出しを発行するときには、DEALLOCATE_ABEND オプションは無効です。このような条件下でこのオプションの使用を試みると、アプリケーションに状況コード AD が返されます。

このオプションは、SEND_ERROR オプションと一緒に使用することはできません。オプション・リストに両方のオプションをコーディングすると、アプリケーションに状況コード AR が返されます。

関連資料**:** APPC および LU 6.2 の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

オプション・リストのフィードバック域

オプション・リスト内にエラーが検出されると、オプション・リストのフィードバック域を使用してエラー情報がアプリケーションに返されます。

IMS はオプション・リスト全体の解析を試み、できるだけ多くのエラーに関する情報を返します。フィードバック域にすべてのエラー情報が入るだけの大きさがない場合には、スペースが許すかぎりの情報が返されます。この区域を指定しないと、状況コードがオプション・リストのエラーを示すだけです。

フィードバック域は、アプリケーション・プログラムが区域の長さを示す長さフィールドで初期設定しなければなりません。フィードバック域は、オプション・リストのおよそ 1½ から 2 倍の長さ、または少なくとも 8 語あれば十分です。

エラー・コード

この節では、アプリケーションが受け取るエラー・コードについて説明します。

エラー・コード
　理由

(0002) オプション・キーワードが認識されない。

このエラーの理由としては、以下が考えられます。

- キーワードのスペルが間違っている。
- キーワードのスペルは正しいが、後続の区切り文字が無効である。
- PRTO を表す長さが指定されたフィールドが、オプションの実際の長さよりも短い。
- 指示された呼び出しに対してキーワードが無効である。

(0004) オプション変数に指定した文字数が少なすぎるか、または多すぎる。呼び出しのオプション・リストのキーワードに続くオプション変数が、オプションの長さの制限を超えています。

(0006) オプション変数の長さフィールド (LL) が長すぎて、オプション・リストに

入らない。オプション・リストの長さフィールド (LL) は、指定したオプション変数の終わりに達する前に、オプション・リストが終了することを示します。

(0008) オプション変数に無効な文字が入っているか、または英字で始まっていない。

(000A)

必須指定のオプション・キーワードが指定されなかった。

このエラーの理由としては、以下が考えられます。

- オプション・リストに 1 つ以上のキーワードが指定されたため、1 つ以上のキーワードを追加する必要がある。
- オプション・リストに指定した長さはゼロより大きいが、リストにオプションが入っていない。

(000C)

指定したオプション・キーワードの組み合わせが無効である。このエラーの原因としては、以下が考えられます。

- オプション・リストに指定された他のキーワードのためにそのキーワードが許可されない。
- オプション・キーワードが複数回指定されている。

(000E) 印刷データ・セット記述子の解析中に、IMS が 1 つ以上のオペランドでエラーを検出した。通常、IMS は z/OS サービス (SJF) を使用して印刷記述子 (PRTO= オプション変数) の妥当性検査を行います。 IMS が SJF を呼び出すとき、IMS は TSO OUTDES コマンドと同様の妥当性検査を要求します。したがって、IMS は出力記述子の変更を重要と見なしません。ユーザーのシステムについて有効な記述子は、MVS リリース・レベルの機能です。有効な記述子のリストおよび正しい構文については、TSO HELP OUTDES コマンドを使用してください。

IMS は、まず PRTO オプションの形式が SJF サービスを使用できる形式であることを確認します。それ以外の形式の場合には、IMS は、状況コード **AS**、エラー・コード (000E)、および記述エラー・メッセージを返します。SJF 処理中にエラーが検出された場合、SJF からのエラー・メッセージには、(R.C.=xxxx,REAS.=yyyyyyyy) 形式の情報およびエラーを示すエラー・メッセージが含まれます。

変数によっては、その範囲が初期設定パラメーターによって制御されます。初期設定パラメーターによって影響を受ける変数の例としては、コピーの最大数の値、許容されるリモート宛先、クラス、および用紙名があります。

制約事項

CPI-C ドリブン・アプリケーション・プログラムは、代替 PCB に対する SETO 呼び出しだけを出すことができます。

関連資料:

415 ページの『REXXTDLI 呼び出し』

IMS TM システム・サービスのための DL/I 呼び出し

以下の DL/I 呼び出しを、IMS Transaction Manager システム・サービスとともに使用します。

呼び出しは、英字順に説明します。各呼び出しの説明には以下の内容が含まれます。

- 構文図
- 呼び出しで使用できる各パラメーターの定義
- アプリケーション・プログラムでの呼び出しの詳しい使用方法
- 呼び出しの使用についての制約事項

各パラメーターは、入力パラメーターまたは出力パラメーターとして記述されています。『入力』とは、アプリケーション・プログラムから IMS への入力のことをいいます。「出力」とは、IMS からアプリケーション・プログラムへの出力のことを指します。

システム・サービス呼び出しは、TP PCB だけを参照しなければなりません。システム・サービス呼び出しは、IMS TM 機能に関係しているものだけが記述されています。

これらの呼び出しの構文図は、*function* パラメーターで始まります。呼び出し、呼び出しインターフェース (xxxTDLI) および *parmcount* (必要な場合) は、以下の構文図には含まれていません。完全な構造については、「*IMS V14* アプリケーション・プログラミング」のトピック『アプリケーション・プログラム・エレメントの定義』に記載されているアセンブラー言語、COBOL、Pascal、および PL/I の固有の情報を参照してください。

システム・サービス呼び出しの要約

以下の表は、各タイプの IMS DB アプリケーション・プログラムで使用できるシステム・サービス呼び出しと、各呼び出しのパラメーターの要約です。以下の表には、機能コード、その意味、用途、パラメーター、および機能コードの有効な領域がリストしてあります。オプション・パラメーターは、大括弧 ([ ]) で囲まれています。

DCCTL 環境で出されるシステム・サービス呼び出しは、入出力 PCB または GSAM データベース PCB のみを参照します。DCCTL 環境で使用されない呼び出しには注が記載されています。

言語依存パラメーターは、ここでは示されていません。言語固有の情報については、「*IMS V14* アプリケーション・プログラミング」のトピック『言語インターフェースのための DL/I 呼び出しのフォーマット設定』を参照してください。

プログラミング言語のインターフェースを使用する呼び出しの作成については、「*IMS V14* アプリケーション・プログラミング」のトピック『アプリケーション・プログラム・エレメントの定義』を参照してください。

表 32. システム・サービス呼び出しの要約

機能コード	意味および用途	オプション	パラメーター	有効な環境
APSB	PSB 割り振り。CPI-C ドリブン・アプリケーション・プログラムで使用する PSB を割り振る。	ありません。	function、aib	MPP
CHKP (基本)	基本チェックポイント。リカバリーのため。	ありません。	function、i/o pcb または aib、i/o area	バッチ、BMP、MPP
CHKP (シンボリック)	シンボリック・チェックポイント。リカバリーのため。	保管するプログラム域を 7 つ指定できる。	function、i/o pcb または aib、i/o area length、i/o area[, area length, area]	バッチ、BMP
DPSB	PSB 割り振り解除。CPI-C ドリブン・アプリケーション・プログラムで使用されている PSB を解放する。	ありません。	function、aib	MPP
GMSG	AO 出口ルーチンからメッセージを取り出す。	使用可能なメッセージがない場合は AOI メッセージを待つ。	function, aib, i/o area	DB/DC および DCCTL (BMP、MPP、IFP)、DB/DC および DBCTL (DRA スレッド)、DBCTL (BMP 非メッセージ・ドリブン)
GSCD[1]	システム目録ディレクトリーのアドレスを取り出す。	ありません。	function、i/o pcb または aib、i/o area	バッチ
ICMD	IMS コマンドを出し、コマンド応答の最初のセグメントを取り出す。	ありません。	function, aib, i/o area	DB/DC および DCCTL (BMP、MPP、IFP)、DB/DC および DBCTL (DRA スレッド)、DBCTL (BMP 非メッセージ・ドリブン)
INIT	アプリケーションがデータ可用性の状況コードを受け取る。	各 PCB のデータ可用性を検査する。	function、i/o pcb または aib、i/o area	バッチ、 BMP、MPP、 IFP
INQY	照会。出力の宛先、セッション状況、実行環境、および PCB アドレスに関する情報を検索する。	ありません。	function, aib, i/o area	バッチ、 BMP、MPP、 IFP
LOGb	ログ。システム・ログにメッセージを書き出す。	ありません。	function、i/o pcb または aib、i/o area	バッチ、 BMP、MPP、 IFP

表 32. システム・サービス呼び出しの要約 (続き)

機能コード	意味および用途	オプション	パラメーター	有効な環境
RCMD	ICMD 呼び出しの結果として生成されたコマンド応答の 2 番目以降のセグメントを取り出す。	ありません。	function, aib, i/o area	DB/DC および DCCTL (BMP、MPP、IFP)、DB/DC および DBCTL (DRA スレッド)、DBCTL (BMP 非メッセージ・ドリブン)
ROLB	ロールバック。アプリケーション・プログラムが送信するメッセージをバックアウトする。	この呼び出しで、最後のメッセージが入出力域に返される。	function、i/o pcb または aib[、i/o area]	バッチ、 BMP、MPP、 IFP
ROLL	ロール。出力メッセージをバックアウトし、会話を終了する。	ありません。	function	バッチ、BMP、MPP
ROLS	SETS または SETU 呼び出しが設定する同期点にメッセージ・キューの位置を戻す。	入出力 PCB か AIB で呼び出しを発行する。	function、i/o pcb または aib i/o area、token	バッチ、 BMP、MPP、 IFP
SETS	中間同期 (バックアウト) ポイントを設定する。	既存のバックアウト・ポイントをすべて取り消す。9 つまでのバックアウト・ポイントを設定できる。	function、i/o pcb または aib、i/o area、token	バッチ、 BMP、MPP、 IFP
SETU	中間同期 (バックアウト) ポイントを設定する。	既存のバックアウト・ポイントをすべて取り消す。9 つまでのバックアウト・ポイントを設定できる。	function、i/o pcb または aib、i/o area、token	バッチ、 BMP、MPP、 IFP
SYNC	同期	コミット・ポイント処理を要求する。	function、i/o pcb または aib	BMP
XRST	再始動。CHKP (記号) と併せて使用し、障害発生後のアプリケーション・プログラムを再始動する。	7 つまでの区域の保管を指定できる。	function、i/o pcb または aib、i/o area length、i/o area[, area length, area]	バッチ、BMP

注:

1. GSCD は、プロダクト・センシティブ・プログラミング・インターフェースです。

関連資料: DCCTL ユーザーは、GSAM データベース PCB を使用して呼び出しを発行することができます。GSAM データベースについては、「*IMS V14* アプリケーション・プログラミング」に説明があります。

関連資料:

41 ページの『IMS DB システム・サービスのための DL/I 呼び出し』

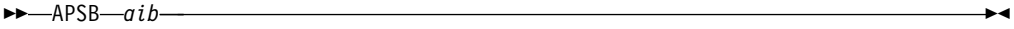
1 ページの『データベース管理のための DL/I 呼び出し』

188 ページの『EXEC DLI コマンド』

APSB 呼び出し

PSB 割り振り (APSB) 呼び出しは、CPI 通信ドリブン・アプリケーション・プログラムに PSB を割り振るときに使用します。このようなアプリケーション・プログラムは、LU 6.2 装置を含む会話に使用されます。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
APSB	X		X		

パラメーター

aib

この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。8 バイトの左寄せされるこのフィールドには、PSB 名を入れます。

使用法

CPI-C ドリブン・アプリケーション・プログラムは、IMS 言語インターフェース・モジュールとリンク・エディットし、アプリケーション・プログラムで DL/I 呼び出しを発行する前に、使用する PSB を指示する必要があります。APSB 呼び出しは、AIB を使用して、このようなアプリケーション・プログラムに PSB を割り振ります。

APSB 呼び出しを発行すると、IMS TM は、指定した PSB に入っている PCB アドレスのリストをアプリケーション・プログラムに返します。AIB の AIBRSA1 フィールドには、PCB リストが返されます。

IMS TM では、PSB が指示するデータベースが利用不能であっても、APSB 呼び出しを完了させることができます。また、INIT 呼び出しを発行して、データ可用性に関する追加状況コードをアプリケーション・プログラムが受け取ることができることを、IMS TM に知らせることができます。

関連資料: CPI 通信ドリブン・アプリケーション・プログラムの詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

制約事項

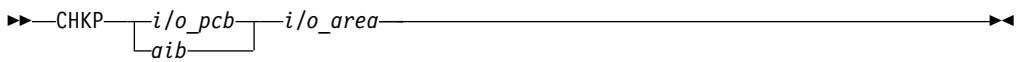
APSB を使用するアプリケーション・プログラムが一度に割り振ることができる PSB は 1 つだけです。アプリケーションに複数の PSB が必要な場合には、まず PSB 割り振り解除 (DPSB) 呼び出しを発行して使用中の PSB を解放しなければなりません。

CPI 通信ドリブン・アプリケーション・プログラムは、他の DL/I 呼び出しを発行する前に APSB 呼び出しを発行しなければなりません。 APSB 呼び出しで PSB が割り振られる前にアプリケーション・プログラムで DL/I 呼び出しを発行しようとすると、アプリケーション・プログラムは、AIB 内にエラー戻りコードおよび理由コードを受け取ります。

CHKP (基本) 呼び出し

基本チェックポイント (CHKP) 呼び出しは、リカバリーを目的として使用します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
CHKP	X	X	X	X	X

パラメーター

i/o pcb
入出力 PCB を指定します。これは、この呼び出しで使用するためにプログラムに渡されるリスト内の最初の PCB アドレスです。これは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbbを指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
呼び出しで使用する入出力域を指定します。このパラメーターは入出力パラメーターです。CHKP 呼び出しの場合、入出力域には 8 文字のチェックポイント ID を入れます。プログラムが MPP、またはメッセージ・ドリブン BMP である場合、CHKP 呼び出しは、暗黙のうちに次の入力メッセージをこの入出力域に返します。したがって、この区域は、返される最長のメッセージが入るだけの長さがなければなりません。

使用法

トランザクション管理アプリケーション・プログラムでは、基本 CHKP 呼び出しは、会話型 SPA の取り出し、またはアプリケーションがスケジュールされる前にキューイングされた初期メッセージ・セグメントの取り出しに使用することができます。CHKP 呼び出しは、そのプログラムが加えたすべての変更をコミットし、また、アプリケーション・プログラムが異常終了した場合には、プログラムを再始動できるポイントを設定します。

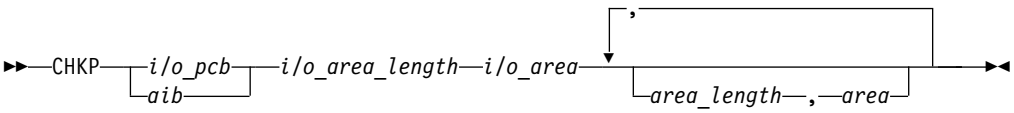
制約事項

CPI 通信ドリブン・アプリケーション・プログラムでは、基本 CHKP 呼び出しを発行することはできません。

CHKP (シンボリック) 呼び出し

記号チェックポイント (CHKP) 呼び出しは、リカバリーを目的として使用します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
CHKP	X	X	X	X	X

パラメーター

i/o pcb
入出力 PCB を指定します。これは、この呼び出しで使用するためにプログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area length
このパラメーターは、IMS では使用されなくなりました。互換性の理由により、引き続きこのパラメーターはこの呼び出しに組み込む必要があり、また有効なアドレスを入れなければなりません。プログラム内の任意の区域の名前を指定することにより、有効なアドレスを獲得することができます。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは入出力パラメーターです。CHKP 呼び出しの場合、この入出力域には 8 文字のチェックポイント ID を入れます。プログラムがメッセージ・ドリブン BMP である場合、CHKP 呼び出しは、暗黙のうちに次の入力メッセージをこの入出力域に返します。したがって、この区域は、返される最長のメッセージが入るだけの長さがなければなりません。

area length
プログラム内に 4 バイトのフィールドを指定します。ここには、チェックポイントをとる最初の区域長が 2 進数で入ります。このパラメーターは入力パラメーターです。最大 7 つの区域長を指定することができます。各区域長に、区域パラメーターも指定しなければなりません。

area
プログラム内で、IMS がチェックポイントをとる区域を指定します。このパラメーターは入力パラメーターです。プログラムで、IMS がチェックポイントをとる区域を、最大 7 つ指定することができます。まず区域長パラメーターを、次に区域パラメーターを指定するようにします。 XRST 呼び出しで指定する区域の数は、プログラムが出す CHKP 呼び出しに指定する区域の数以下でなければなりません。プログラムを再始動すると、IMS は、CHKP 呼び出しに指定した区域だけを復元します。

使用法

トランザクション管理アプリケーション・プログラムでは、記号 CHKP 呼び出しは、会話型 SPA の取り出し、またはアプリケーションがスケジュールされる前にキューイングされた初期メッセージ・セグメントの取り出しに使用することができます。CHKP 呼び出しは、そのプログラムが加えたすべての変更をコミットし、ま

た、アプリケーション・プログラムが異常終了した場合には、プログラムを再始動できるポイントを設定します。さらに、記号 CHKP 呼び出しでは、以下のことを行うことができます。

- 拡張再始動 (XRST) 呼び出しを使用して、プログラムを再始動する (プログラムが異常終了した場合)。
- プログラムの最大 7 つのデータ域を保管する。これは、プログラムの再始動時に復元されます。

制約事項

CPI 通信ドリブン・アプリケーション・プログラムは、記号 CHKP 呼び出しを発行することはできません。記号 CHKP 呼び出しは、バッチおよび BMP アプリケーションからのみ出すことができます。

記号 CHKP 呼び出しを発行する前に、XRST 呼び出しを発行しなければなりません。

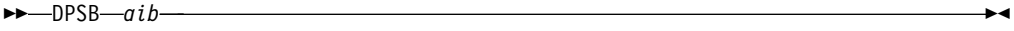
関連資料:

184 ページの『XRST 呼び出し』

DPSB 呼び出し

PSB 割り振り解除 (DPSB) 呼び出しは、APSB 呼び出しで割り振られた PSB を解放します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
DPSB	X		X		

パラメーター

aib

この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1

リソース名。8 バイトの左寄せされるこのフィールドには、PSB 名を入れます。

使用法

DPSB 呼び出しは、CPI 通信ドリブン・アプリケーション・プログラムで、コミット・ポイントが発生した後で別の PSB を割り振る前に、PSB を解放するために使用します。CPI 通信ドリブン・アプリケーション・プログラムでは、コミット・ポイントは、COMMIT verb で実行されます。CPI 通信ドリブン・アプリケーション・プログラムの詳細については、「*IMS V14* コミュニケーションおよびコネクション」のトピック『CPI-C ドリブン・アプリケーション・プログラム』を参照してください。

制約事項

DPSB 呼び出しは、コミット・ポイントの発生後に限って発行することができ、APSB 呼び出しが成功した後に有効になります。

GMSG 呼び出し

メッセージ取得 (GMSG) 呼び出しは、AO 出口ルーチン (DFSAOE00 または別の AOIE タイプの出口ルーチン) からメッセージを検索するための自動化操作プログラム (AO) アプリケーション・プログラムで使用されます。

フォーマット

パラメーター

aib

この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID

目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN

AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBSFUNC

副次機能コード。このフィールドには、リストされている 8 バイトの副次機能コードのいずれかを入れる必要があります。

8 個のブランク (ヌル)

AIBRSNM1 フィールドに AOI トークンでコード化すると、アプリケーションで使用できる AOI メッセージがないときに IMS が制御を戻すことを示します。

WAITAOI

AIBRSNM1 フィールドに AOI トークンでコード化すると、その時点でアプリケーションで使用できるものがないときに IMS が AOI メッセージを待つことを示します。この副次機能の値は、AIBRSNM1 フィ

ールドに AOI トークンがコード化されていない場合は無効です。この場合、AIB にエラー戻りコードと理由コードが返されます。

WAITAOI の値は、左寄せで指定し、ブランク文字を埋め込まなければなりません。

AIBRSNM1
リソース名。このフィールドには、AOI トークンまたはブランクを入れます。 AOI トークンは、AO アプリケーションが取り出すメッセージを識別します。このトークンは、メッセージの最初のセグメントに与えられます。メッセージが複数セグメントのメッセージの場合は、このフィールドをブランクに設定して、2 番目のセグメントから最後のセグメントまでを取り出します。AIBRSNM1 は、8 バイトの英数字フィールドで、左寄せで指定し、ブランクで埋め込まれます。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。このフィールドは、IMS により変更されません。

AIBOAUSE
入出力域に返されるデータの長さ。このパラメーターは出力パラメーターです。

入出力域に十分な大きさがないため、データの一部が返された場合、AIBOAUSE にすべてのデータを取り出すために必要な長さが入っており、AIBOALEN にデータの実際の長さが入っています。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。入出力域には、IMS から AO アプリケーションに渡される最大のセグメントが入るだけの大きさがなければなりません。入出力域にすべてのデータが入るだけの大きさがない場合には、IMS はデータの一部を返します。

使用法

GMSG は、AO アプリケーションで AOI トークンに関連したメッセージを取り出すときに使用します。AO アプリケーションは、メッセージの最初のセグメントを取り出すため、IMS に 8 バイトの AOI トークンを渡す必要があります。IMS は、AOI トークンを使用して、タイプ AOIE の AO 出口ルーチンからのメッセージを、AO アプリケーションから出される GMSG 呼び出しに関連付けます。IMS は、アプリケーションに AOI トークンに関連したメッセージだけを返します。さまざまな AOI トークンを使用することによって、AOIE タイプの出口ルーチンはさまざまな AO アプリケーションにメッセージを送ることができます。インストール・システムでは、AOI トークンを定義するように注意してください。

複数セグメント・メッセージの 2 番目から最後までのセグメントを取り出すには、トークンを指定せずに (トークンをブランクに設定) GMSG 呼び出しを発行します。メッセージのすべてのセグメントを取り出す場合、すべてのセグメントが取り出されるまでに GMSG 呼び出しを発行する必要があります。AOI トークンを指定した新しい GMSG 呼び出しが発行されると、IMS は、複数セグメント・メッセージの取り出されなかったセグメントすべてを廃棄します。

AO アプリケーションでは、GMSG 呼び出しの待機を指定することができます。現時点で対応する AOI トークンで使用できるメッセージがない場合は、AO アプリケーションはメッセージが使用可能になるまで待機します。待機の決定は、トランザクション定義の中で待機が指定されている WFI トランザクションとは異なり、AO アプリケーションによって指定されます。待機は呼び出し単位で行われます。つまり、1 つの AO アプリケーションの中で待機を指定する GMSG 呼び出しもあれば、指定しない呼び出しもあります。

以下の表は、IMS 環境において GMSG を出すことのできるアプリケーション・プログラムの型を示しています。GMSG は、CPI-C ドリブン・アプリケーション・プログラムからもサポートされます。

表 *33.* アプリケーション領域タイプ別の *GMSG* サポート

	IMS 環境		
アプリケーション領域のタイプ	**DBCTL**	**DB/DC**	**DCCTL**
DRA スレッド	あり	あり	N/A
BMP (非メッセージ・ドリブン)	あり	あり	あり
BMP (メッセージ・ドリブン)	N/A	あり	あり
MPP	N/A	あり	あり
IFP	N/A	あり	可

制約事項

CPI-C ドリブン・プログラムでは、 GMSG を発行する前に APSB (PSB 割り振り) 呼び出しを発行しなければなりません。

GSCD 呼び出し

システム目録ディレクトリー取り出し (GSCD) 呼び出しは、バッチ・プログラムの IMS システム目録ディレクトリー (SCD) のアドレスを取り出します。

このトピックにはプロダクト・センシティブ・プログラミング・インターフェース情報が含まれています。

フォーマット

呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
GSCD				X	X

パラメーター

i/o pcb
PCB を指定します。これは、この呼び出しで使用するためにプログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) のアドレスを指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbbを指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。GCSD 呼び出しの場合、入出力域の長さは 8 バイトでなければなりません。IMS TM は、最初の 4 バイトに SCD のアドレスを、次の 4 バイトにプログラム仕様テーブル (PST) のアドレスを入れます。

使用法

プログラムが出した GSCD 呼び出しが成功した後では、IMS はプログラムに状況コードを返しません。同じ PCB を使用した、直前の呼び出しからの状況コードは、PCB に未変更のまま残っています。

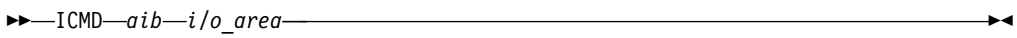
制約事項

GSCD 呼び出しは、DLI または DBB バッチ・アプリケーション・プログラムからのみ出すことができます。

ICMD 呼び出し

コマンド発行 (ICMD) 呼び出しを使用すると、自動化操作プログラム (AO) アプリケーション・プログラムで IMS コマンドを発行することができ、コマンドの応答の最初のセグメントを取り出すことができます。

フォーマット



パラメーター

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。このフィールドは、IMS により変更されません。

AIBOAUSE
入出力域に戻されるデータの長さ。このパラメーターは出力パラメーターです。

ユーザー・プログラムでは、このフィールドで ICMD 呼び出しによって入出力域にデータが返されたかどうかを検査する必要があります。コマンドに対する唯一の応答が「COMMAND IN PROGRESS (コマンドの処理中)」または「COMMAND COMPLETE (コマンドの完了)」を示す DFS058 メッセージである場合、この応答は返されません。

入出力域に十分な大きさがないため、データの一部が返された場合、AIBOAUSE にすべてのデータを取り出すために必要な長さが入っており、AIBOALEN にデータの実際の長さが入っています。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは入出力パラメーターです。入出力域には、AO アプリケーション・プログラムから IMS へ受け渡しされる最大のコマンド、または IMS から AO アプリケーション・プログラムへ渡されるコマンド応答のセグメントが入るだけの十分な大きさが必要です。入出力域にすべてのデータが入るだけの大きさがない場合には、IMS はデータの一部を返します。

ICMD 呼び出しでのユーザーの入出力作業域の一般形式は次のようになります。

```
LLZZ/VERB KEYWORD1 P1 KEYWORD2 P2, P3.
```

LL コマンド・テキストの長さが入った 2 バイトのフィールド (LLZZ を含む)

ZZ IMS 用に予約されている 2 バイトのフィールド

/ または **CRC**
IMS コマンドが続くことを示します。DBCTL 環境では、スラッシュ (/) ではなく CRC (コマンド認識文字) が使用されます。

VERB 発行中の IMS コマンド

KEYWORDX
発行中のコマンドに適用されるキーワード

PX 指定中のキーワードのパラメーター

. (ピリオド**)**
コマンドの終了

コマンドの長さは入出力域のサイズによって制限されます。このサイズは、PCB 生成中に PSBGEN マクロの IOASIZE パラメーターで指定されます。 LL はコマンド・テキストの長さです。入出力域のサイズは、実際のコマンド・テキストの長さに LLZZ 用の 4 バイトを加えた長さです。入出力作業域の最小サイズは 132 バイトです。

5 番目のバイトは、/ (DBCTL の場合は CRC) でなければなりません。また、その直後には verb が続いていなければなりません。 /BROADCAST および /LOOPTEST コマンドは、コマンド・セグメントとテキスト・セグメントの間にピリオドが必要で、テキストのサイズが入った LLZZ フィールドがその前になければなりません。 最後のパラメーターのあとにピリオド (.) を入れると、コメントを追加することができます。

制約事項**:** /SSR コマンドを出すときには、コマンド終了標識 (ピリオド) をコーディングしてはなりません (「*IMS V14* オペレーションおよびオートメーション」を参照)。ピリオドが使用されている場合、これはテキストの一部と見なされます。

使用法

ICMD を使用すると、AO アプリケーションで IMS コマンドを発行することができ、コマンドの応答の最初のセグメントを取り出すことができます。

ICMD を使用するときは、発行する IMS コマンドをアプリケーションの入出力域に入れます。IMS は、コマンドを処理した後、AO アプリケーション・プログラムの入出力域に応答メッセージの最初のセグメントを返し、RCMD 呼び出しを使用して後続のセグメントを (一度に 1 つずつ) 取り出します。

正常に実行された IMS コマンドには、「DFS058 COMMAND COMPLETE」(DFS058 コマンドの完了) というメッセージを返すものもあります。非同期に処理される IMS コマンドには、「DFS058 COMMAND IN PROGRESS」(DFS058 コマンドの処理中) というメッセージを返すものもあります。ICMD 呼び出しで入力されたコマンドでは、AO アプリケーション・プログラムにどちらの DFS058 メッセージも返されません。セグメントが返されなかったことを示すため、AIBOAUSE フィールドがゼロに設定されます。したがって、AO アプリケーション・プログラムで AIBOAUSE フィールドと戻りコードおよび理由コードを検査して、応答が返されたどうか判別する必要があります。

関連資料**:** AOI 出口についての詳細は、「*IMS V14* 出口ルーチン」を参照してください。

以下の表は、IMS 環境において ICMD を出すことのできるアプリケーション・プログラムの型を示しています。ICMD は、CPI-C ドリブン・アプリケーションからもサポートされます。

表 34. アプリケーション領域タイプ別の *ICMD* サポート

	IMS 環境		
アプリケーション領域のタイプ	**DBCTL**	**DB/DC**	**DCCTL**
DRA スレッド	あり	あり	N/A
BMP (非メッセージ・ドリブン)	あり	あり	あり
BMP (メッセージ・ドリブン)	N/A	あり	あり
MPP	N/A	あり	あり
IFP	N/A	あり	可

ICMD 呼び出しで出せるコマンドのリストについては、「*IMS V14* オペレーションおよびオートメーション」を参照してください。

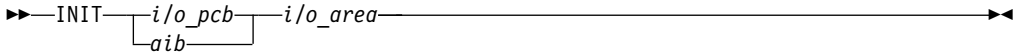
制約事項

CPI-C ドリブン・プログラムでは、ICMD 呼び出しを発行する前に APSB (PSB 割り振り) 呼び出しを発行しなければなりません。

INIT 呼び出し

初期設定 (INIT) 呼び出しを使用すると、アプリケーション・プログラムは、各 DB PCB のデータ可用性を検査することにより、データ可用性状況コードを受け取ることができます。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
INIT	X	X	X	X	X

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) のアドレスを指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
この呼び出しで使用する入出力域を指定します。 このパラメーターは入力パラメーターです。INIT 呼び出しの入出力域には、文字ストリング「DBQUERY」または「VERSION(*dbname1=version,dbname2=version*)」が入ります。

使用法

INIT 呼び出しは、すべての IMS TM アプリケーション・プログラムについて有効です。

INIT 呼び出しに関するパフォーマンスの考慮事項 ***(IMS*** オンラインのみ***)***

パフォーマンス上の理由により、入出力 PCB への最初の GU 呼び出しの前に、オンライン・アプリケーション・プログラムで INIT 呼び出しを発行してはなりません。最初に INIT 呼び出しを発行した場合、入出力 PCB への GU 呼び出しは効率よく処理されません。

アプリケーション・プログラムにデータベース照会副次機能を指定するには、入出力域内に文字ストリング 『DBQUERY』 を指定します。

データベース可用性の判別***: INIT DBQUERY***

INIT 呼び出しを、入出力域に DBQUERY 文字ストリングを指定して発行すると、アプリケーション・プログラムで各 PCB のデータの使用可能性に関する情報を入手することができます。以下の表は、DBQUERY を指定した INIT 呼び出しの入出力域の例を示します。

表 *35. PLITDLI* を除くすべての *xxxTDLI* インターフェースの *INIT* 入出力域の例

L	**L**	**Z**	**Z**	文字ストリング
00	0B	00	00	DBQUERY

注: LL フィールドと ZZ フィールドは 2 進数です。LL の値 X'0B' は、10 進数 11 の 16 進表記です。

表 *36. PLITDLI* インターフェースの *INIT* 入出力域の例

L	**L**	**L**	**L**	**Z**	**Z**	文字ストリング
00	00	00	0B	00	00	DBQUERY

注: LLLL フィールドと ZZ フィールドは 2 進数です。L の値 X'0B' は、10 進数 11 の 16 進表記です。

LL または **LLLL**
この 2 バイトのフィールドには、文字ストリングの長さに LL の 2 バイトを加算した値が入ります。 PLITDLI インターフェースの場合には、4 バ

イトのフィールド LLLL を使用します。AIBTDLI インターフェースを使用する場合、PL/I プログラムには 2 バイトのフィールドのみが必要です。

ZZ 2 進数ゼロが入る 2 バイト・フィールドです。

各データベースの PCB について、以下のいずれかの状況コードが戻されます。

NA この PCB を使用してアクセスできるデータベースのうちの少なくとも 1 つが使用できない。この PCB を使用して呼び出しを発行すると、INIT STATUS GROUPA 呼び出しが出されていれば BA または BB の状況コードが返され、出されていなければ DFS3303I メッセージが送られ、3303 の疑似異常終了が発生します。動的割り振りが失敗したためにデータベースが使用できない場合は例外です。この場合、呼び出しを発行すると、状況コード AI (オープン不能) が返されることになります。

DCCTL 環境では、状況コードは常に NA です。

NU この PCB を使用して更新できるデータベースのうちの少なくとも 1 つが更新に使用することができない。この PCB を使用して ISRT、DLET、または REPL 呼び出しを発行すると、INIT STATUS GROUPA 呼び出しが出されていれば状況コード BA が、出されていなければ DFS3303I メッセージが送られ、3303 の疑似異常終了が発生します。状況コード NU の原因となったデータベースは、削除処理の場合のみ必要になる場合があります。その場合、DLET 呼び出しは失敗しますが、ISRT 呼び出しおよび REPL 呼び出しは成功します。

bb この PCB でアクセスできるデータは、PCB が許可するすべての機能に使用することができます。DEDB および MSDB は、常に bb になります。

データ可用性状況に加えて、ルート・セグメントのデータベース編成の名前が、PCB のセグメント名フィールドに返されます。DCCTL 環境では、データベース編成の名前は UNKNOWN です。

自動 ***INIT DBQUERY***

アプリケーション・プログラムに初めて入るとき、データベース PCB の状況コードは、INIT DBQUERY 呼び出しが出された場合のように初期設定されます。これにより、アプリケーション・プログラムは、INIT 呼び出しを発行せずにデータベース可用性を判別することができます。

DCCTL 環境では、状況コードは NA です。

データベースのバージョン番号を ***INIT VERSION(dbname=version)*** で指定します。

データベースのバージョン管理が可能な場合、アプリケーション・プログラムは「VERSION」関数を使用して、データベースのバージョン (PCB 上でそのアプリケーション・プログラム用に指定されているバージョン番号または IMS によって返されるデフォルトのバージョンとは異なるもの) を要求できます。 INIT VERSION 呼び出しで指定されるバージョン番号は、他のすべてのバージョン指定およびデフォルトに優先します。

データベースにアクセスするための DL/I より前に INIT VERSION 呼び出しが発行されない場合、アプリケーション・プログラムに返されるデータベースのバージ

ョンは、PCB ステートメントの DBVER キーワードによって決定されます。DBVER キーワードが指定されない場合、IMS は、ACB ライブラリーでアクティブなデータベースのバージョン、またはデータベースのバージョン 0 を返します。これは、PSBGEN ステートメント内、または DFSDFxxx PROCLIBメンバーのデータベース・セクション内の DBLEVEL キーワードによって決定されます。

入出力域で、VERSION 関数は次の形式を使用して指定されます。

各データベース名は英字を使用して指定され、指定可能な回数は 1 回のみです。指定するのは物理データベースの名前のみです。論理データベースの名前はサポートされていません。

各バージョンは 0 から 2147483647 までの数値として指定されます。指定される番号は、指名されているデータベースの DBD で定義され、IMS カタログに保管されているバージョン番号と一致している必要があります。

入出力域に必要なサイズは、入力の入出力域に指定されているデータベースの数に 20 を乗算して計算します。

例えば、次の表は、アセンブラー言語、COBOL、C 言語、および Pascal での INIT VERSION 呼び出しの入出力域の例を示しています。次の表で、LL 値 X'3C' は、10 進数 60 の 16 進表記です。これは、3 つのデータベース名が入力に指定されている場合に、その出力を入出力域に保持するために必要な長さ (バイト単位) です。 ZZ フィールドは 2 進数です。

表 *37. INIT VERSION: ASMTDLI*、*CBLTDLI*、*CTDLI*、および *PASTDLI* での形式の例

L	**L**	**Z**	**Z**	文字ストリング
00	3C	00	00	VERSION (DBa=1,DBb=2,DBc=3)

次の表は、PL/I の場合の、VERSION を指定した INIT 呼び出しの入出力域の例を示しています。この表で、LL 値 X'3C' は 10 進数 60 の 16 進表記です。 ZZ フィールドは 2 進数です。

表 *38. INIT VERSION: PLITDLI* での形式の例

L	**L**	**L**	**L**	**Z**	**Z**	文字ストリング
00	00	00	3C	00	00	VERSION (DBa=1,DBb=2,DBc=3)

LL または **LLLL**

入出力域の合計長さが入る 2 バイトまたは 4 バイトのフィールド。PL/I では、LLLL フィールドの長さは、実際には 4 バイト・フィールドであっても 2 バイトと見なされます。AIBTDLI インターフェースを使用する場合、レコードの長さは「LL + ZZ + 文字ストリング」の合計の長さに等しくなります。PLITDLI インターフェースの場合、レコードの長さは「LLLL + ZZ + 出力に必要な長さ」の合計長さに等しくなります。ここで LLLL は 2 バイトと見なされます。

ZZ　2 進数ゼロが入る 2 バイト・フィールドです。

文字ストリング
　入力での関数の指定。LL または LLLL に指定される長さは、出力に必要な長さ (すなわち、入力文字ストリングに指定されているデータベースごとに 20 バイトずつ) です。

INQY 呼び出し

照会 (INQY) 呼び出しは、実行環境、宛先のタイプと状況、ならびにセッションの状況に関する情報の要求に使用します。INQY は、AIB 構造を使用するアプリケーション・インターフェースのみに有効です。

フォーマット

呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
INQY	X	X	X	X	X

パラメーター

aib
　呼び出しに使用するアプリケーション・インターフェース・ブロック (DFSAIB) のアドレスを指定します。このパラメーターは入出力パラメーターです。AIB 内でこれらのフィールドを初期設定する必要があります。

AIBID
　目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
　AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBSFUNC
　副次機能コード。このフィールドには、以下のような 8 バイトの副次機能コードのいずれかを入れる必要があります。

- bbbbbbbb (Null)
- DBQUERYb
- ENVIRONb
- FINDbbbb
- LERUNOPT
- MSGINFOb
- PROGRAMb (ODBA インターフェースではサポートされません)

AIBRSNM1
　リソース名。この 8 バイトの左寄せフィールドには、PSB 内の名前付き PCB の名前を入れる必要があります。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストで指定された入出力域の長さを指定する必要があります。このフィールドは、IMS により変更されません。

i/o area
この呼び出しで使用するデータ出力域を指定します。このパラメーターは出力パラメーターです。INQY 副次機能の ENVIRONb、MSGINFOb、および PROGRAMb には、入出力域が必要です。副次機能 DBQUERYb、FINDbbbb、および LERUNOPT では必要ではありません。

制約事項

CPI 通信ドリブン・アプリケーション・プログラムは、入出力 PCB に対してヌル副次機能を使用して INQY 呼び出しを発行することはできません。

バッチ・プログラムでは、ヌル副次機能を使用して INQY 呼び出しを発行することはできません。

使用法

INQY 呼び出しは、バッチ環境でもオンライン IMS 環境でも使用できます。IMS アプリケーション・プログラムは INQY 呼び出しを使用して、出力の宛先、セッションの状況、現在の実行環境、データベースの可用性、および (PCB 名にもとづく) PCB アドレスに関する情報を要求できます。INQY 呼び出しを出すときには、AIB を使用しなければなりません。INQY 呼び出しを出す前に、AIB のフィールドを初期設定してください。

INQY 呼び出しを使用する場合には、AIB で渡される 8 バイトの副次機能コードを指定します。アプリケーション・プログラムが受け取る情報は、 INQY 副次機能によって決まります。

INQY 呼び出しは、呼び出し側の入出力域に情報を返します。INQY 呼び出しから戻されるデータの長さは AIB フィールドの 1 つである AIBOAUSE の中のアプリケーションに渡されます。

入出力域のサイズは AIB フィールド AIBOALEN に指定します。INQY 呼び出しは、1 回の呼び出しでその区域に入るだけのデータを返します。区域の長さが、すべての情報を入れるのに十分ではない場合、AG 状況コードが戻され、データの一部が入出力域に戻されます。この場合、AIB フィールドの AIBOALEN には、入出力域に返されたデータの実際の長さが入っており、AIBOAUSE フィールドには、すべてのデータを受け取るために必要な出力域の長さが入っています。

PCB からの情報の照会 : INQY ヌル

ヌル副次機能を指定して INQY 呼び出しを発行するとき、アプリケーション・プログラムは、出力宛先のタイプと場所、およびセッション状況を含む、PCB に関連する情報を入手します。INQY 呼び出しには、入出力 PCB または代替 PCB が使用できます。宛先の場所およびセッション状況に関して受け取る情報は、宛先のタイプに基づいています。宛先タイプには
APPC、OTMA、TERMINAL、TRANSACT、および UNKNOWN があります。

関連資料: APPC および LU 6.2 の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

INQY ヌル副次機能は、入出力域に文字ストリング・データを返します。宛先のタイプ APPC、OTMA、TERMINAL、および TRANSACT に返される出力は、左揃えにして、ブランクが埋め込まれます。宛先のタイプ UNKNOWN は、情報をなにも返しません。以下の表は INQY ヌル呼び出しで戻される出力をリストしています。いくつかの項目についての詳細は、表に付随している注を参照してください。

表 *39.* 端末タイプ宛先の場合の *INQY* ヌルのデータ出力

返される情報	バイト長	実際の値	説明
宛先のタイプ	8	端末	I/O PCB または代替 PCB の宛先が端末である。
端末の位置	8	ローカル	端末はローカルと定義されている。
		リモート	端末はリモートと定義されている。
キューの状況	8	開始済み	キューが開始され、作業が受け入れ可能である。
		停止	キューが停止され、作業が受け入れ不能である。
セッション状況	8	b	状況が使用できない。
		ACTIVE	セッションがアクティブである。
		INACTIVE	セッションが非アクティブである。

表 *40.* トランザクション・タイプ宛先の場合の *INQY* ヌルのデータ出力

返される情報	バイト長	実際の値	説明
宛先のタイプ	8	TRANSACT	代替 PCB の宛先がプログラムである。
トランザクションの位置	8	ローカル	トランザクションはローカルと定義されている。
		リモート	トランザクションはリモートと定義されている。
		DYNAMIC	トランザクションは動的と定義されている。[1]
トランザクションの状況	8	STARTED	トランザクションがスケジュール可能である。[2]
		STOPPED	トランザクションがスケジュール不能である。[2]
宛先 PSB 名	8		このフィールドは宛先 PSB の名前を提供する。
		b	プログラム・ルーティング出口ルーチンが、宛先をこのシステム上にはないトランザクションとして定義した、または、トランザクションが動的である。トランザクションの宛先が使用できない。

表 40. トランザクション・タイプ宛先の場合の *INQY* ヌルのデータ出力 (続き)

返される情報	バイト長	実際の値	説明
宛先プログラムまたはセッション状況	8	b	状況が使用できない。
		ACTIVE	MSC リンク・セッションはアクティブである (リモート・トランザクション、または TM およびメッセージ・ルーティングと制御のユーザー出口ルーチン (DFSMSCE0) によってリモート IMS に転送されるトランザクション)。
		INACTIVE	MSC リンク・セッションは非アクティブである (リモート・トランザクション、または TM およびメッセージ・ルーティングと制御のユーザー出口ルーチン (DFSMSCE0) によってリモート IMS に転送されるトランザクション)。
		STARTED	プログラムがスケジュール可能である (ローカル・トランザクション)。
		STOPPED	プログラムがスケジュール不能である (ローカル・トランザクション)。

注:

1. 動的トランザクションは、共用キュー環境でのみ使用可能です。トランザクションは、メッセージを送信している IMS システムには定義されていないで、キューを共用している他の IMS システムに定義されているとき、動的になります。動的トランザクションは、宛先作成出口ルーチン (DFSINSX0) が、IMS にとって未知の宛先を持つトランザクションを指示するときに作成されます。宛先 PSB 名と宛先プログラムに対する出力フィールドは、ブランクに設定されます。
2. TM およびメッセージ・ルーティングと制御のユーザー出口ルーチン (DFSMSCE0) によって、トランザクションがリモート IMS に転送された場合、返される状況は、MSNAME 状況です。

表 41. *APPC* タイプ宛先の場合の *INQY* ヌルのデータ出力

返される情報	バイト長	実際の値	説明
宛先のタイプ	8	APPC	宛先が LU 6.2 装置である。
APPC/MVS サイド情報項目名[1]	8		このフィールドはサイド名を提供する。
		b	サイド名が使用できない。
パートナー論理装置名[2]	8		このフィールドは会話のパートナー LU 名を提供する。
		b	パートナー LU 名が使用できない。
パートナー・モード・テーブル項目名[3]	8		このフィールドは会話のモード名を提供する。
		b	モード名が使用できない。
ユーザー ID	8		このフィールドはユーザー ID を提供する。
		b	ユーザー ID が使用できない。
グループ名	8		このフィールドはグループ名を提供する。
		b	グループ名が使用できない。
同期レベル[4]	1	C	同期レベルは CONFIRM と定義されている。
		N	同期レベルは NONE と定義されている。

表 41. APPC タイプ宛先の場合の INQY ヌルのデータ出力 (続き)

返される情報	バイト長	実際の値	説明
会話タイプ[5]	1	B	会話は BASIC と定義されている。
		M	会話は MAPPED と定義されている。
ユーザー ID 標識	1		ユーザー ID 標識フィールドの値は、ユーザー ID フィールドの内容を示します。ユーザー ID 標識フィールドには、4 つの値のいずれかが入る。
		U	値 U はサインオン時に送信元端末からのユーザーの識別名を示す。
		L	値 L は送信元端末の LTERM 名を示す (サインオンがアクティブでない場合)。
		P	値 P は送信元 BMP またはトランザクションの PSBNAME を示す。
		O	値 O はその他の名前を示す。
TPN のアドレス[6]	4		トランザクション・プログラム名の LL フィールドのアドレス。[7]
		0	トランザクション・プログラム名のアドレスがない。

注:

1. TP PCB に対してこの呼び出しを発行する場合、サイド名は使用できず、b が返されます。変更可能代替 PCB に対してこの呼び出しを発行する場合には、INQY の前に出す CHNG 呼び出しで、サイド名を指定しなければなりません。
2. TP PCB に対してこの呼び出しを発行する場合には、LU 名をコーディングしなければなりません。変更可能代替 PCB に対してこの呼び出しを発行する場合には、INQY の前に出す CHNG 呼び出しで、LU 名を指定しなければなりません。
3. TP PCB に対してこの呼び出しを発行する場合、モード名は使用できず、b が返されます。変更可能代替 PCB に対してこの呼び出しを発行する場合には、INQY の前に出す CHNG 呼び出しで、モード名を指定しなければなりません。
4. 同期レベルが使用できない場合、IMS は CONFIRM のデフォルトを使用します。
5. 会話タイプが使用できない場合は、IMS は MAPPED のデフォルトを使用します。
6. ポインターは、長さフィールド (LL) に必要な 2 バイトを含む、TPN の長さ (2 進数) が入る LL を識別します。
7. TPN の長さは最大で 64 バイトです。

表 42. OTMA タイプ宛先の場合の INQY ヌルのデータ出力

返される情報	長さ (バイト)	実際の値	説明
宛先のタイプ	8	OTMA	宛先は OTMA クライアントである。
T パイプ名	8		このフィールドは OTMA トランザクションのパイプ名を提供する。
		b	T パイプ名が使用できない。

表 42. *OTMA* タイプ宛先の場合の *INQY* ヌルのデータ出力 (続き)

返される情報	長さ (バイト)	実際の値	説明
メンバー名	16		このフィールドは OTMA クライアントの z/OS システム間カップリング・ファシリティー (XCF) メンバー名を提供する。
		b	メンバー名が使用できない。
ユーザー ID	8		このフィールドはユーザー ID を提供する。
		b	ユーザー ID が使用できない。
グループ名	8		このフィールドはグループ名を提供する。
		b	グループ名が使用できない。
同期レベル	1	S	OTMA トランザクション・パイプが同期化されている。
		b	OTMA トランザクション・パイプが同期化されていない。
メッセージ同期レベル[1]	1	C	同期レベルは CONFIRM と定義されている。
		N	同期レベルは NONE と定義されている。
ユーザー ID 標識	1		ユーザー ID 標識フィールドの値は、ユーザー ID フィールドの内容を示します。ユーザー ID 標識フィールドには、4 つの値のいずれかが入る。
		U	値 U はサインオン時に送信元端末からのユーザーの識別名を示す。
		L	値 L は送信元端末の LTERM 名を示す (サインオンがアクティブでない場合)。
		P	値 P は送信元 BMP またはトランザクションの PSBNAME を示す。
		O	値 O はその他の名前を示す。
IMS に予約済み	1		このフィールドは予約済みである。

注:

1. 同期レベルが使用できない場合、IMS は CONFIRM のデフォルトを使用します。

表 43. *UNKNOWN* タイプ宛先の場合の *INQY* ヌルのデータ出力

返される情報	バイト長	実際の値	説明
宛先のタイプ	8	UNKNOWN	宛先が見つからない。

出力フィールドの内容は、INQY 呼び出しで使用する PCB のタイプによって異なります。以下の表は、APPC 宛先の INQY 出力が PCB タイプによってどのように変わるかを示しています。PCB は TP PCB または代替 PCB のいずれかになります。

表 44. *INQY* 出力および *PCB* のタイプ

出力フィールド	**TP PCB**	代替 **PCB** (修正不能)	代替 **PCB** (修正可能)
宛先のタイプ	APPC	APPC	APPC

表 44. *INQY* 出力および *PCB* のタイプ (続き)

出力フィールド	**TP PCB**	代替 **PCB (**修正不能**)**	代替 **PCB (**修正可能**)**
サイド名	ブランク	サイド名 (使用可能な場合) またはブランク	サイド名 (前の CHNG 呼び出しで指定されている場合) またはブランク
LU 名	入力 LU 名	LU 名 (使用可能な場合) またはブランク	LU 名 (前の CHNG 呼び出しで指定されている場合) またはブランク
モード名	ブランク	モード名 (使用可能な場合) またはブランク	モード名 (前の CHNG 呼び出しで指定されている場合) またはブランク
ユーザー ID	USERID (使用可能な場合) またはブランク	USERID (使用可能な場合) またはブランク	USERID (使用可能な場合) またはブランク
グループ名	グループ名 (使用可能な場合) またはブランク	グループ名 (使用可能な場合) またはブランク	グループ名 (使用可能な場合) またはブランク
同期レベル	C または N	C または N	C または N
会話タイプ	B または M	B または M	B または M
ユーザー ID 標識	U または L または P または O	U または L または P または O	U または L または P または O
TPN アドレス	TPN 文字ストリングのアドレス	TPN 文字ストリングのアドレスまたはゼロ	TPN 文字ストリングのアドレスまたはゼロ
TPN 文字ストリング **注:** TPN 名が DFSASYNC の場合、宛先は非同期の会話を表します。	実行中の IMS トランザクションのインバウンド名	パートナー TPN (使用可能な場合)。使用できなければアドレス・フィールドはゼロ。	TP 名 (前の CHNG 呼び出しで指定されている場合)。指定されていなければアドレス・フィールドはゼロ。

関連資料**:** APPC および LU 6.2 の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

データ可用性の照会**: INQY DBQUERY**

DBQUERY 副次機能を指定して INQY 呼び出しを発行すると、アプリケーション・プログラムは、各 PCB のデータに関する情報を入手します。AIBRSNM1 に渡せる有効な PCB 名は IOPCBbbb だけです。INQY DBQUERY 呼び出しは、INIT DBQUERY 呼び出しに似ています。INQY DBQUERY 呼び出しは、入出力域には情報を戻しませんが、INIT DBQUERY 呼び出しと同じように、データベース PCB 内の状況コードを更新します。

INQY FIND 呼び出しが実行されるまで、アプリケーション・プログラムには各 PCB の状況が知らされません。データベースの状況を検索するには、INQY FIND 呼び出しでそのデータベースの DB PCB を渡す必要があります。

INIT DBQUERY 状況コードのほかに、 INQY DBQUERY 呼び出しは、次の状況コードを入出力 PCB に戻します。

bb その呼び出しは成功し、すべてのデータベースが使用可能。

BJ PSB 内のどのデータベースも使用可能でないか、あるいは PSB に PCB が存在しない。すべてのデータベース PCB (GSAM を除く) には、INQY DBQUERY 呼び出しの処理の結果として状況コード NA が入っています。

BK PSB の少なくとも 1 つのデータベースが使用できない状態であるか、または使用できる範囲が制限されている。少なくとも 1 つのデータベース PCB には、INQY DBQUERY 呼び出しの処理の結果として NA または NU の状況コードが入っています。

INQY 呼び出しは、各 DB PCB に以下の状況コードを返します。

NA この PCB を使用してアクセスできるデータベースのうちの少なくとも 1 つが使用できない。この PCB に基づく呼び出しの結果は、おそらく、INIT STATUS GROUPA 呼び出しが出されていれば BA または BB 状況コード、出されていなければ DFS3303I メッセージと 3303 疑似異常終了となります。動的割り振りが失敗したためにデータベースが使用できない場合は例外です。この場合、呼び出しを発行すると、状況コード AI (オープン不能) が返されることになります。

DCCTL 環境では、状況コードは常に NA です。

NU この PCB を使用して更新できるデータベースのうちの少なくとも 1 つが更新に使用することができない。この PCB に基づく ISRT、DLET、 REPL 各呼び出しの結果は、おそらく、INIT STATUS GROUPA 呼び出しが出されていれば BA 状況コード、出されていなければ DFS3303I メッセージと 3303 疑似異常終了になります。状況コード NU の原因となったデータベースは、削除処理の場合のみ必要になる場合があります。その場合、DLET 呼び出しは失敗しますが、ISRT 呼び出しおよび REPL 呼び出しは成功します。

bb この PCB でアクセスできるデータは、PCB が許可するすべての機能に使用することができます。DEDB および MSDB は、常に bb になります。

環境の照会: INQY ENVIRON

ENVIRON 副次機能を指定して INQY 呼び出しを発行すると、アプリケーション・プログラムは、現在の実行環境に関する情報を入手します。 AIBRSNM1 に渡せる有効な PCB 名は IOPCBbbb だけです。この中には、IMS ID、リリース、領域、および領域タイプがあります。

INQY ENVIRON 呼び出しは、文字ストリング・データを戻します。出力データは左寄せされ、右側にはブランクが埋め込まれます。

推奨事項**:** 応答データの長さの拡張に対応するには、入出力域の長さを 512 バイトに指定します。

リカバリー・トークンまたはアプリケーション・パラメーター・ストリングが入っているフィールドを参照するには、INQY ENVIRON 呼び出しのデータ出力で返されたフィールドのアドレスを使用して、フィールドの位置を指定するようにアプリケーション・プログラムをコーディングします。この方法が、リカバリー・トークン・フィールドおよびアプリケーション・パラメーター・ストリング・フィールドを参照する唯一の有効なプログラミング手法です。これらのフィールドを参照するのに、他のプログラミング手法を使用することはできません。

リカバリー・トークンまたはアプリケーション・パラメーター・ストリングはオプションであるため、常に返されるわけではありません。これらのフィールドが返されない場合、アドレス・フィールドの値はゼロになります。

リカバリー・トークン・フィールドおよびアプリケーション・パラメーター・フィールドについて詳しくは、以下の表の後に示された注 2 を参照してください。

以下の表は、INQY ENVIRON 呼び出しから戻される出力のリストです。戻される情報には、出力のバイト長、実際の値、および説明が含まれます。

表 *45. INQY ENVIRON* データ出力

返される情報	バイト長	実際の値	説明
IMS ID	8		実行パラメーターから得られた ID を示す。
IMS リリース・レベル	4		IMS のリリース・レベルを提供する。例えば、X'00000410'。
IMS 制御領域のタイプ	8	BATCH	IMS バッチ領域がアクティブであることを示す。
		DB	IMS Database Manager だけが活動状態であることを示す。(DBCTL システム)
		TM	IMS Transaction Manager だけが活動状態であることを示す。(DCCTL システム)
		DB/DC	IMS Database Manager と IMS Transaction Manager の両方が活動状態であることを示す。(DB/DC システム)
IMS アプリケーション領域のタイプ	8	BATCH	IMS バッチ領域がアクティブであることを示す。
		BMP	バッチ・メッセージ処理領域がアクティブであることを示す。
		DRA	データベース・リソース・アダプターのスレッド領域がアクティブであることを示す。
		IFP	IMS 高速機能領域がアクティブであることを示す。
		JBP	Java バッチ処理領域がアクティブであることを示す。
		JMP	Java メッセージ処理領域がアクティブであることを示す。
		MPP	メッセージ処理領域がアクティブであることを示す。
領域 ID	4		領域 ID を提供する。例えば、X'00000001'。
アプリケーション・プログラム名	8		実行中のアプリケーション・プログラムの名前を提供する。
PSB 名 (現在割り振られているもの)	8		現在割り振られている PSB の名前を提供する。
トランザクション名	8		トランザクションの名前を提供する。
		b	関連するトランザクションがないことを示す。
ユーザー ID[1]	8		ユーザー ID を提供する。
		b	ユーザー ID が使用できないことを示す。
グループ名	8		グループ名を提供する。
		b	グループ名が使用できないことを示す。
状況グループ標識	4	A	INIT STATUS GROUPA 呼び出しが出されることを示す。
		B	INIT STATUS GROUPB 呼び出しが出されることを示す。
		b	状況グループが初期設定されないことを示す。

表 45. INQY ENVIRON データ出力 (続き)

返される情報	バイト長	実際の値	説明
リカバリー・トークンのアドレス[2]	4		あとにリカバリー・トークンが続く LL フィールドのアドレスを提供する。
		0	リカバリー・トークンが使用不可であることを示す。
アプリケーション・パラメーター・ストリングのアドレス [2]	4		アプリケーション・プログラム・パラメーター・ストリングが後に続く、LL フィールドのアドレスを示す。
		0	従属領域 JCL の実行パラメーターで APARM= パラメーターがコーディングされていないことを示す。
共用キュー標識	4		IMS が共用キューを使用していないことを示す。
		SHRQ	IMS が共用キューを使用していることを示す。
アドレス・スペースのユーザー ID	8		従属アドレス・スペースのユーザー ID
ユーザー ID 標識	1		ユーザー ID フィールドの内容を示す以下の可能な値のいずれかを含む。 **U** サインオン時にソース端末から得られたユーザーの ID を示す。 **L** サインオンが有効になっていない場合のソース端末の LTERM 名を示す。 **P** ソース BMP またはトランザクションの PSBNAME を示す。 **O** その他の特定の名前を示す。
z/OS リソース・リカバリー・サービス (RRS) 標識	3	b	IMS が RRS を持つ UR に利害関係があることを示さなかったことを表す。したがって、IMS がコミット有効範囲には関係しないので、アプリケーションは、RRS が UR の同期点マネージャーになるような作業を実行してはなりません。例えば、アプリケーションは、アウトバウンド保護会話を出してはなりません。
		RRS	IMS が RRS をもつ UR に利害関係があることを示したことを表す。したがって、RRS が UR の同期点マネージャーになっている場合、IMS はコミット有効範囲で使用されます。
IMS カタログ使用可能化標識	7	b	IMS カタログが DFSDFxxx PROCLIB メンバーで使用可能でないことを表す。 IMS カタログのセットアップおよび使用可能化については、IMS カタログの定義と調整 (システム定義)を参照してください。 DFSDFxxx PROCLIB メンバーでの IMS カタログの使用可能化については、IMS PROCLIB データ・セットの DFSDFxxx メンバー (システム定義)を参照してください。
		CATALOG	IMS カタログが使用可能であることを表す。データベースとアプリケーション・メタデータが IMS で使用可能です。

表 45. INQY ENVIRON データ出力 (続き)

返される情報	バイト長	実際の値	説明

注:

1. ユーザー ID は、INQY ENVIRON 呼び出しを発行する領域を表す PST の PSTUSID フィールドから取り出されます。PSTUSID フィールドは、以下のいずれかになります。
   - IMS メッセージ・キューに対する GU 呼び出しを正常に完了させなかったメッセージ・ドリブン BMP 領域の場合、および非メッセージ・ドリブン BMP 領域の場合、PSTUSID フィールドは、現在 BMP 領域にスケジューリングされている PSB の名前から得られます。
   - GU 呼び出しを正常に完了させたメッセージ・ドリブン BMP 領域および MPP 領域の場合、通常は入力端末の RACF ID の入っている PSTUSID フィールドの値が得られます。端末が RACF にサインオンされていない場合には、ID は入力端末の LTERM になります。
2. このポインターは長さフィールド (LL) を識別するアドレスであり、このフィールドには、リカバリー・トークンまたはアプリケーション・プログラム・パラメーター・ストリングの長さ (LL に必要な 2 バイトを含む) が 2 進数形式で入ります。このポインターを使用して、バッチ・プログラム内でリリース間の AIB のアドレス可能度をセットアップします。

入力メッセージ情報の照会: INQY MSGINFO

現行の入力メッセージに関する情報を取得するには、MSGINFO 副次機能で INQY 呼び出しを使用します。AIBRSNM1 フィールドに渡せる有効な PCB 名は IOPCBbbb だけです。出力は、メッセージ情報のバージョン番号および出力フィールドを返します。 INQY MSGINFO 呼び出しは、入出力域の応答を返します。

以下の表は、INQY MSGINFO 呼び出しから戻される出力のリストです。戻される情報に含まれるのは、バイト長、実際の値、および出力の説明です。

表 46. INQY MSGINFO データ出力

返される情報	バイト長	実際の値	説明
バージョン番号	4	1	出力応答バージョン 1。
発信元 IMSID	8		入力メッセージの発信元の IMS ID。
IMS に予約済み	68		このフィールドは将来の出力拡張のために予約されています。

PCB アドレスの照会: INQY FIND

FIND 副次機能を指定して INQY 呼び出しを発行すると、アプリケーション・プログラムには、要求された PCB 名の PCB アドレスが返されます。 AIBRSNM1 で渡すことのできる有効な PCB 名は、IOPCBbbb または PSB に定義されている代替 PCB (TP PCB) またはデータベース PCB の名前です。

FIND 副次機能では、要求された PCB が修正されず、入出力域には情報が返されません。

FIND 副次機能を使用すると、INQY DBQUERY 呼び出しに続く PCB アドレスが得られます。このプロセスにより、アプリケーションで PCB 状況コードを分析して、

PCB に NA または NU の状況コードが設定されているかどうか判別することができます。

LE オーバーライドの照会: INQY LERUNOPT

LERUNOPT 副次機能を指定して LERUNOPT 呼び出しを発行すると、IMS は LE オーバーライドが LEOPT システム・パラメーターに基づいて許可されているかどうか判別します。 LE オーバーライド・パラメーターは、 UPDATE LE コマンドで IMS に定義されます。IMS は、発呼者の環境におけるトランザクション名、lterm 名、ユーザー ID、またはプログラム名の特定の組み合わせに基づいて、発呼者に該当するオーバーライドがあるかどうかをチェックします。IMS は、オーバーライド・パラメーターが検索された場合、発呼者にストリングのアドレスを戻します。IMS 提供の CEEBXITA 出口、DFSBXITA は、LE ランタイム・パラメーター用の動的オーバーライドを許可するため、LE オーバーライドを使用します。

呼び出しストリングは、機能コードおよび AIB アドレスを含まなければなりません。入出力域は必要パラメーターではなく、指定した場合は無視されます。AIBRSNM1 に渡せる有効な PCB 名は IOPCB だけです。 AIBOALEN および AIBOAUSE フィールドは使用されません。

DL/I INQY LERUNOPT 呼び出しで戻されることになる項目、マッチングするルールは、以下の通りです。

- MPP または JMP 領域は、トランザクション名、LTERM、ユーザー ID、およびプログラムを使用して、各項目とマッチングします。
- IFB、JBP、または非メッセージ・ドリブン BMP は、それぞれのエントリーとマッチングするためにプログラム名を使用します。項目に、トランザクション名、LTERM、またはユーザー ID に対する定義されたフィルターがある場合は、その項目は一致しません。 メッセージ・ドリブン BMP もまたトランザクション名を使用します。
- 複数のエントリーがスキャンされ、フィルターに掛けた際のマッチングが最も多かったエントリーが検出されます。フィルターに掛けた際のマッチングが最も正確なリスト内の最初のエントリーが選択されます。すべてのフィルターでマッチングする 1 つのエントリーを検出するとスキャンは停止します。

  注: 表項目の検索は、項目の作成方法および検索方法により、ユーザーが混乱することがあります。例えば、 DL/I INQY 呼び出しで指定された複数のフィルターでマッチングするエントリーがテーブルに 2 つあると想定します。1 番目のトランザクションがトランザクション名と LTERM 名でマッチングします。2 番目のエントリーがトランザクション名とプログラム名でマッチングします。IMS は 1 番目のエントリーを選択します。なぜなら、それがフィルターに掛けた際のマッチングが最も多かった最初のエントリーだからです。2 番目のエントリーがより長いパラメーター・ストリングと共に更新されると、新規のエントリーが構築され、キューのヘッドに追加されます。次の検索では、トランザクション名とプログラム名がある項目が選択されることになります。その結果、ユーザーが予想しなかったランタイム・オプションのセットが選択されることになります。

環境:LERUNOPT 副次機能は、 DB/DC、DBCTL、および DCCTL 環境より指定されることが可能です。オーバーライドは、MPP および JMP 領域でのトランザクション名、LTERM 名、ユーザー ID、およびプログラム名の組み合わせに基づいて

います。IFP、BMP、および JBP の領域には、プログラム名に基づいたオーバーライドがあります。メッセージ・ドリブン BMP の領域もまたトランザクション名を使用することができます。

戻りコードおよび理由コード:AIB 戻りコードおよび理由コードをチェックし、呼び出しが正常に完了したかどうかを判断しなければなりません。オーバーライド・パラメーターが呼び出し側用に使用可能な場合、AIBRSA2 を使用してパラメーター・ストリングのアドレスが戻されます。

プログラム名の照会: INQY PROGRAM

PROGRAM 副次機能を指定して INQY 呼び出しを発行すると、アプリケーション・プログラム名が、入出力域の先頭の 8 バイトに返されます。 AIBRSNM1 に渡せる有効な PCB 名は IOPCBbbb だけです。

INQY 戻りコードおよび理由コード

INQY 呼び出しを発行すると、戻りコードと理由コードが AIB に返されます。状況コードが PCB に返される場合もあります。INQY に適用されるもの以外の戻りコードと理由コードが戻された場合は、ユーザーのアプリケーション・プログラムは PCB を検査して、どのような状況コードが戻されているのかを調べる必要があります。

PCB タイプへの INQY 副次機能のマップ

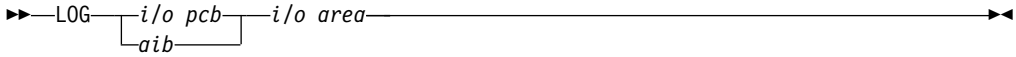
表 *47. INQY* 呼び出しの場合の副次機能、*PCB*、および入出力域の組み合わせ

副次機能	**I/O PCB**	代替 **PCB**	**DB PCB**	必要な入出力域
FIND	OK	OK	OK	NO
ENVIRON	OK	NO	NO	YES
DBQUERY	OK	NO	NO	NO
LERUNOPT	OK	NO	NO	NO
PROGRAM	OK	NO	NO	YES
MSGINFO	OK	NO	NO	YES

LOG 呼び出し

ログ (LOG) 呼び出しは、情報を送信して IMS システム・ログに書き込むときに使用します。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
LOG	X	X	X	X	X

パラメーター

i/o pcb
PCB のアドレスを指定します。これは、この呼び出しで使用するためにプログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbbを指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
プログラム内の区域を指定します。ここには、システム・ログに書き込むレコードが入ります。このパラメーターは入力パラメーターです。このレコードは、必ず以下の表に示す形式になります。

表 *48. AIBTDLI*、*ASMTDLI*、*CBLTDLI*、*CEETDLI*、 *CTDLI*、および *PASTDLI* の各インターフェースの場合の *COBOL*、*PL/I*、*C* 言語、*Pascal*、およびアセンブラーのログ・レコード形式

フィールド名	フィールド長
LL	2
ZZ	2
C	1
テキスト	可変

表 *49. PLITDLI* インターフェースが *COBOL*、*PL/I*、*C* 言語、*Pascal*、およびアセンブラーの場合のログ・レコード形式

フィールド名	フィールド長
LLLL	4
ZZ	2
C	1
テキスト	可変

フィールドは、必ず以下のようになります。

LL または **LLLL**

レコードの長さが入る 2 バイトのフィールドを指定します。 AIBTDLI インターフェースを使用する場合、レコードの長さは、LL + ZZ + C + レコードのテキストになります。 PLITDLI インターフェースの場合は、レコードの長さは、LLLL + ZZ + C + レコードのテキストになります。ログ・レコードの長さを計算する際には、すべてのフィールドを計算に入れなければなりません。指定する長さの合計には、以下のものが含まれます。

- LL または LLLL の 2 バイト (PL/I の場合、LLLL は 4 バイトですが、長さは 2 として計算されます。)
- ZZ フィールドの 2 バイト
- C フィールドの 1 バイト
- レコード自体の長さの n バイト

PLITDLI インターフェースを使用している場合には、プログラムで、長さフィールドを 2 進数のフルワードで定義しなければなりません。

ZZ 2 バイトの 2 進数ゼロのフィールドを指定します。

C ログ・コードが入る 1 バイトのフィールドを指定します。これは、X'A0' 以上でなければなりません。

テキスト

ログに書き込むデータを指定します。

使用法

アプリケーション・プログラムは、LOG 呼び出しを発行して、システム・ログへレコードを書き込むことができます。 LOG 呼び出しを発行するときには、入出力域を指定します。ここには、システム・ログに書き込むレコードが入ります。任意の情報をログに書き込むことができ、また、ログ・コードを使用して情報の各種のタイプを区別することができます。LOG は次の環境で発行できます。

- IMS DB/DC 環境では、レコードは IMS ログに書き込まれます。
- DCCTL 環境では、レコードは DCCTL ログに書き込まれます。

制約事項

入出力域の長さ (すべてのフィールドを含む) は、システム・ログ・データ・セットの論理レコード長 (LRECL) から 4 バイトと論理レコードの接頭部の長さ (長さ x'4A' バイト) を引いた長さ、または PSB の PSBGEN ステートメントの IOASIZE キーワードに指定された入出力域より長くすることはできません。

RCMD 呼び出し

コマンド検索 (RCMD) 呼び出しを使用すると、ICMD 呼び出しの後、自動化操作プログラム (AO) アプリケーション・プログラムでコマンドの応答の 2 番目以降のセグメントを取り出すことができます。

フォーマット

パラメーター

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。このフィールドは、IMS により変更されません。

AIBOAUSE
入出力域に戻されるデータの長さ。このパラメーターは出力パラメーターです。

入出力域に十分な大きさがないため、データの一部が返された場合、AIBOAUSE にすべてのデータを受け取るために必要な長さが入っており、AIBOALEN にデータの実際の長さが入っています。

i/o area
この呼び出しで使用する入出力域を指定します。このパラメーターは出力パラメーターです。入出力域には、IMS から AO アプリケーションに渡されるコマンドの応答の最大のセグメントが入るだけの大きさがなければなりません。その入出力域にすべての情報が入るだけの大きさがない場合には、入出力域には一部のデータが返されます。

使用法

RCMD を使用すると、AO アプリケーションで、ICMD 呼び出しで生成されたコマンドの応答の 2 番目以降のセグメントを取り出すことができます。

関連資料: AOI 出口についての詳細は、「*IMS V14* 出口ルーチン」を参照してください。

RCMD は、CPI-C ドリブン・アプリケーション・プログラムからもサポートされます。

表 *50.* アプリケーション領域タイプ別の *RCMD* サポート

	IMS 環境		
アプリケーション領域のタイプ	**DBCTL**	**DB/DC**	**DCCTL**
DRA スレッド	あり	あり	N/A
BMP (非メッセージ・ドリブン)	あり	あり	あり
BMP (メッセージ・ドリブン)	N/A	あり	あり
MPP	N/A	あり	あり

表 *50.* アプリケーション領域タイプ別の *RCMD* サポート *(*続き*)*

アプリケーション領域のタイプ	IMS 環境		
	DBCTL	**DB/DC**	**DCCTL**
IFP	N/A	あり	可

RCMD は、一度に 1 つの応答セグメントしか取り出しません。応答セグメントがさらに必要な場合は、IMS が発行する応答セグメントごとに RCMD を 1 回発行しなければなりません。

制約事項

ICMD 呼び出しは、RCMD 呼び出しを発行する前に発行する必要があります。

ROLB 呼び出し

ロールバック (ROLB) 呼び出しは、アプリケーション・プログラムによって送られたメッセージをバックアウトします。

フォーマット

呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
ROLB	X	X	X	X	X

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
IMS TM が最初のメッセージ・セグメントを返す、プログラム内の区域を指定する出力パラメーター。会話型トランザクションでは、SPA がアプリケーションに戻される最初の項目です。次の GN 呼び出しが、メッセージの最初のユーザー・セグメントを戻します。

使用法

会話型プログラムで ROLB を出すと、IMS TM は、アプリケーション・プログラムが送ったメッセージをバックアウトします。プログラムが ROLB 呼び出しを発行し、発信元端末に必要な応答を送ることなくコミット・ポイントに達すると、IMS TM は会話を終了し、メッセージ「DFS2171I NO RESPONSE CONVERSATION TERMINATED」を発信元端末に送ります。

IMS TM がロールバックできないリソースをアプリケーション・プログラムが割り振った場合、そのリソースは無視されます。例えば、アプリケーション・プログラムで CPI-C verb を出してリソースを割り振る場合 (変更された DL/I または CPI-C ドリブン・プログラムの場合)、ROLB だけが IMS によって割り振られたそれらのリソースに影響します。アプリケーションで、CPI-C 会話に、ROLB 呼び出しが出されたことを通知しなければなりません。

CPI-C ドリブン・アプリケーション・プログラムでは、急送代替 PCB 以外の代替 PCB に挿入されたすべてのメッセージが廃棄されます。ROLB 呼び出しが出される前に PCB に対して PURG 呼び出しが出されなかった場合には、急送代替 PCB に挿入されたメッセージは廃棄されます。

スプール API 機能を使用し、印刷データ・セットを作成するアプリケーション・プログラムは、ROLB 呼び出しを発行することができます。これにより、JES に解放されていない印刷データ・セットはすべてバックアウトされます。

以下の処理上の考慮事項は、次のような IMS ROLB 呼び出しを発行する変更されたメッセージ・ドリブン IMS アプリケーションに適用されます。すなわち、OTMA または APPC/MVS から保護された入力メッセージを受け取って、他の z/OS リソース・リカバリー・サービス (RRS) リソース・マネージャーに保護されたアウトバウンド・ワークを発行できる ROLB 呼び出しです。

- 保護された入力データを使用する変更メッセージ・ドリブン IMS アプリケーション・プログラムが ROLB 呼び出しを発行した場合、ROLB 呼び出しは IMS アプリケーションに隔離されるため、保護された作業単位全体に影響を及ぼすことはありません。ROLB 呼び出しが発行されると、コミット・ポイントに達するまで、保護された入力メッセージは IMS アプリケーションにとっては処理中のままになります。
- 変更メッセージ・ドリブン IMS アプリケーション・プログラムがアウトバウンド保護会話を発行した場合、そのアウトバウンド保護会話は ROLB 処理に含まれません (つまり、アウトバウンド保護会話は ROLB 呼び出しの一環としてバックアウトされません)。変更メッセージ・ドリブン IMS アプリケーション・プログラムには、バックアウト対象の保護されたすべてのアウトバウンド作業を明示的にクリーンアップする責任があります。

制約事項

この呼び出しに対して、AIB は入出力 PCB を指定しなければなりません。

関連概念:

前のコミット・ポイントへのバックアウト: ROLL、ROLB、および ROLS 呼び出し (アプリケーション・プログラミング)

ROLL 呼び出し

ロール (ROLL) 呼び出しは、会話型アプリケーション・プログラムで送られた出力メッセージをバックアウトし、会話を終了します。

フォーマット

呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
ROLL	X	X	X	X	X

パラメーター

ROLL 呼び出しに必要なパラメーターは、呼び出し機能だけです。

使用法

IMS は、アプリケーションを U0778 異常終了で終了します。

会話中に ROLL 呼び出しを発行すると、IMS TM は更新をバックアウトし、出力メッセージを取り消します。また、IMS TM は U0778 異常終了コードで会話を終了します。

CPI 通信インターフェースを使用するアプリケーションの場合、元のトランザクションは、IMS によって廃棄可能トランザクションとして分類されていれば廃棄されます。

変更された DL/I または CPI-C ドリブン・アプリケーション・プログラムで生成されたリモート LU 6.2 会話トランザクションは、TYPE (ABEND_SVC) により割り振り解除されます。

スプール API 機能を使用し、印刷データ・セットを作成するアプリケーション・プログラムは、ROLL 呼び出しを発行することができます。これにより、JES に解放されていない印刷データ・セットはすべてバックアウトされます。

制約事項

ROLL 呼び出しは、AIBTDLI インターフェースを使用することができません。

関連概念:

前のコミット・ポイントへのバックアウト: ROLL、ROLB、および ROLS 呼び出し (アプリケーション・プログラミング)

APPC/IMS および LU 6.2 装置の管理 (コミュニケーションおよびコネクション)

関連資料:

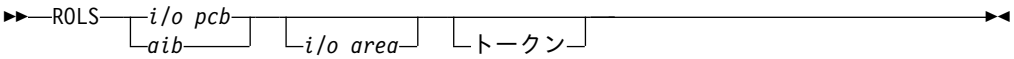
廃棄不能メッセージ・ユーザー出口 (NDMX) (出口ルーチン)

ROLS 呼び出し

SETS/SETU へのロールバック (ROLS) 呼び出しは、メッセージ・キューの位置を、SETS/SETU 呼び出しで設定した同期点に戻します。

ROLS 呼び出しおよび SETS/SETU 呼び出しの詳細については、*IMS V14* アプリケーション・プログラミングのトピック「前のコミット・ポイントへのバックアウト : ROLL、ROLB、および ROLS 呼び出し」を参照してください。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
ROLS	X	X	X	X	X

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
入出力域を指定します。 SETS/SETU 呼び出しの入出力域と同じ形式になります。このパラメーターは出力パラメーターです。

token
プログラムの区域の名前を指定します。ここには、4 バイトの ID が入ります。このパラメーターは入力パラメーターです。

使用法

会話型プログラムで ROLS を出すと、IMS TM は、アプリケーション・プログラムが送ったメッセージをバックアウトします。会話トランザクションの場合、これは、プログラムが ROLS 呼び出しを発行した後、発信元端末に必要な応答を送ることなくコミット・ポイントに達した場合に、IMS TM がその会話を終了させ、メッセージ「DFS2171I NO RESPONSE, CONVERSATION TERMINATED」を発信元端末に送ることを意味します。

トークンを使用する ROLS 呼び出しを発行するときに、ロールバックするメッセージに IMS TM が処理する非高速メッセージが含まれる場合、メッセージ・キューの位置変更が行われる場合があります。位置変更には初期のメッセージ・セグメントが含まれるので、元の入力トランザクションを、IMS TM アプリケーション・プログラムに再度送ることができます。

入力および出力の位置設定は、標準または変更済みの DL/I アプリケーション・プログラムの SETS/SETU 呼び出しによって判別されます。入力および出力の位置設定は、CPI-C ドリブン・アプリケーション・プログラムには適用されません。

アプリケーション・プログラムは、リモート・トランザクション・プログラムに、ROLS について通知しなければなりません。

トークンなしの ROLS では、IMS は、トランザクション・プログラム・インスタンス (TPI) を指定する APPC/MVS verb の ATBCMTP TYPE (ABEND) を出します。これにより、アプリケーション・プログラムに関係するすべての会話が DEALLOCATED TYPE (ABEND_SVC) になります。元のトランザクションが LU 6.2 装置から入力され、IMS TM が APPC/MVS からメッセージを受け取った場合、廃棄可能なトランザクションは廃棄されます。廃棄不能なトランザクションは、延期キューに入れられます。

関連資料: LU 6.2 の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

制約事項

IMS TM の外部のリソースを含む会話型アプリケーション・プログラム (例えば、CPI-C ドリブン・アプリケーション・プログラム) の実行中に ROLS を出すと、IMS TM リソースだけがロールバックされます。アプリケーション・プログラムは、リモート・トランザクションに ROLS 呼び出しについて通知します。

スプール API 機能では、SETS/SETU および ROLS 呼び出しの使用は制限されません。 これらの呼び出しは、アプリケーション・プログラムから印刷データ・セットの処理外で使用できるため、制限がありません。これらのコマンドの発行時は、スプール API サポートで行われるアクションはありません。これは、これらのコマンドは印刷データ・セットの部分的なバックアウトには使用できないためです。スプール API を使用しているアプリケーションによって SETS/SETU 呼び出しまたは

ROLS 呼び出しが出されたことを示すような、特別な状況コードがアプリケーション・プログラムに返されることはありません。

ROLS 呼び出しは、PSB に DEDB または MSDB PCB が入っているとき、または DB2 データベースに対して呼び出しが発行されるときは無効です。

関連資料:

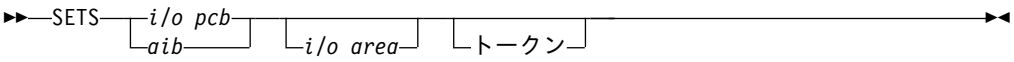
『SETS/SETU 呼び出し』

SETS/SETU 呼び出し

バックアウト・ポイント設定 (SETS) 呼び出しは、中間バックアウト・ポイントの設定または既存のバックアウト・ポイントの取り消しに使用します。

無条件設定 (SETU) 呼び出しの動作は、SETS 呼び出しの動作と同様です。 ただし、PSB の中にサポートされない PCB がある場合、またはプログラムが外部サブシステムを 使用する場合は、SETU 呼び出しは拒否されません。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
SETS/SETU	X	X	X	X	X

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。

i/o area
プログラムの区域を指定します。ここには、IMS により保存され、対応する ROLS 呼び出しで返されるデータが入ります。このパラメーターは入力パラメーターです。

token
プログラムの区域の名前を指定します。ここには、4 バイトの ID が入ります。このパラメーターは入力パラメーターです。

使用法

呼び出し名自体を除いては、SETS と SETU の形式およびパラメーターは同じです。

SETS および SETU 呼び出しは、IMS が ROLS 呼び出しで使用するバックアウト・ポイントを提供します。 ROLS 呼び出しは、SETS および SETU 呼び出しのバックアウト・ポイントを使って動作します。

SETS または SETU の状況コード SC の意味は、以下のとおりです。

SETS SETS 呼び出しは拒否されます。入出力 PCB の状況コード SC は、PSB にサポートされないオプションが入っているか、またはアプリケーション・プログラムが外部サブシステムに呼び出しを発行したことを示します。

SETU SETU 呼び出しは拒否されません。状況コード SC は、PSB 内にサポートされない PCB があるか、またはアプリケーションが外部サブシステムに呼び出しを発行したことを示します。

制約事項

SETS 呼び出しは、PSB に DEDB または MSDB PCB が入っているとき、または DB2 データベースに対して呼び出しが発行されるときは無効です。

CPI-C ドリブン・トランザクション・プログラムでは、SETS/SETU 呼び出しを発行できません。

スプール API 機能では、SETS/SETU および ROLS 呼び出しの使用は制限されません。これらの呼び出しは、アプリケーションから印刷データ・セットの処理外で使用できるため、制限がありません。これらのコマンドの発行時は、スプール API サポートで行われるアクションはありません。これは、これらのコマンドは印刷データ・セットの部分的なバックアウトには使用できないためです。

ROLS 呼び出しの前に、同一のトークンを使用して最大 255 の SETS 呼び出しを指定でき、しかも、正しいメッセージ・レベルにバックアウトすることができます。 255 の SETS 呼び出しの後で、メッセージのバックアウトが続行されますが、255 番目の SETS 呼び出しと同じメッセージ・レベルにのみバックアウトされます。 SETS のトークン・カウントは、同期点処理時にゼロにリセットされます。

ROLS 呼び出しの前に同一のトークンを使用して最大 255 の SETS 呼び出しを指定でき、しかも、正しいメッセージ・レベルにバックアウトすることができます。 255 の SETS 呼び出しの後も、メッセージは引き続き 255 番目の SETS 呼び出しと同じメッセージ・レベルにバックアウトされます。 SETS のトークン・カウントは、同期点処理時にゼロにリセットされます。

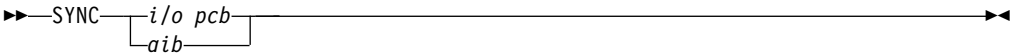
関連資料:

180 ページの『ROLS 呼び出し』

SYNC 呼び出し

同期点 (SYNC) 呼び出しは、コミット・ポイント処理を要求するときに使用します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
SYNC	X	X	X		

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbbb を指定しなければなりません。

使用法

SYNC 呼び出しを発行して、IMS TM がアプリケーション・プログラムのコミット・ポイントを使用してアプリケーション・プログラムを処理するよう要求します。

制約事項

SYNC 呼び出しは、バッチ指向 BMP にのみ有効です。

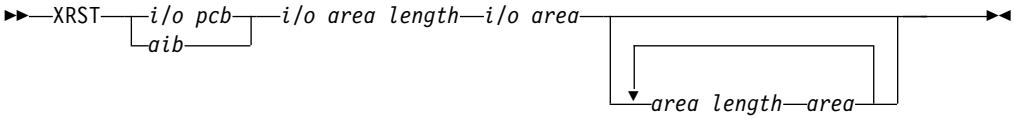
CPI 通信ドリブン・アプリケーション・プログラムから SYNC 呼び出しを発行することはできません。

XRST 呼び出し

拡張再始動 (XRST) 呼び出しは、プログラムを再始動するときに使用します。

プログラムで記号チェックポイント呼び出しを使用する場合は、XRST 呼び出しを使用する必要があります。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
XRST	X	X	X	X	X

パラメーター

i/o pcb
入出力 PCB を指定します。これは、プログラムに渡されるリスト内の最初の PCB アドレスです。このパラメーターは入出力パラメーターです。

aib
この呼び出しに使用するアプリケーション・インターフェース・ブロック (AIB) を指定します。このパラメーターは入出力パラメーターです。

以下のフィールドは、AIB 内で初期設定しなければなりません。

AIBID
目印。この 8 バイトのフィールドには DFSAIBbb を入れる必要があります。

AIBLEN
AIB の長さ。このフィールドには、アプリケーション・プログラムが入手した AIB の実際の長さを入れます。

AIBRSNM1
リソース名。この 8 バイトの左寄せフィールドには PCB 名 IOPCBbb を指定しなければなりません。

AIBOALEN
入出力域の長さ。このフィールドには、呼び出しリストに指定した入出力域の長さを入れます。このパラメーターは、XRST 呼び出しの間は使用されません。互換性の理由により、このパラメーターは引き続きコーディングする必要があります。

i/o area length
このパラメーターは、IMS では使用されなくなりました。互換性の理由により、引き続きこのパラメーターはこの呼び出しに組み込む必要があり、また有効なアドレスを入れなければなりません。プログラム内の任意の区域の名前を指定することにより、有効なアドレスを獲得することができます。

i/o area
プログラムに 14 バイトの区域を指定します。この区域は、プログラムを通常に開始する場合はブランクに設定され、拡張再始動を実行する場合はチェックポイント ID を持ちます。

area length
復元する区域の長さが 2 進数で入るプログラムの 4 バイトのフィールドを指定します。この入力パラメーターは任意指定です。最大 7 つの区域長を指定することができます。各区域長に、区域パラメーターも指定しなければなりません。XRST 呼び出しで指定する区域の数は、プログラムが出す CHKP 呼び出しに指定する区域の数以下でなければなりません。プログラムを再始動すると、IMS TM は、CHKP 呼び出しに指定した区域だけを復元します。

area
IMS TM にリストアさせたいユーザーのプログラム内の区域を指定します。区域は 7 つまで指定できます。 指定する各区域には、区域長 が先頭にあります。このパラメーターは入力パラメーターです。

使用法

記号チェックポイント呼び出し (CHKP) を発行したいプログラムは、拡張再始動呼び出し (XRST) も発行する必要があります。XRST 呼び出しは、一度だけ、プログラム実行の早い時期に発行される必要があります。この呼び出しは、プログラムの最初の呼び出しでなくてもかまいません。ただし、CHKP 呼び出しよりは先に出さなければなりません。XRST 呼び出しより前に出されるすべてのデータベース呼び出しは、再始動の有効範囲内には入りません。

IMS は、正常始動を行うのか、または再始動を行うのかを、XRST 呼び出しで提供される入出力域に基づいて、あるいはプログラムの JCL の EXEC ステートメントの PARM フィールドの CKPTID= 値に基づいて判断します。

プログラムの正常始動

プログラムを正常に始動するときは、 XRST 呼び出しで指し示す入出力域にブランクが入り、かつ、PARM フィールドの CKPTID= 値がヌルである必要があります。これが、以降の CHKP 呼び出しは、基本チェックポイントではなくシンボリック・チェックポイントであることを、IMS に指示します。 XRST 呼び出しを発行した後、プログラムで入出力域を検査する必要があります。プログラムを正常に始動するとき、IMS はこの区域を変更しません。

ユーザー・プログラムの再始動

プログラムの直前の実行時にとられたシンボリック・チェックポイントから、プログラムを再始動することができます。再始動の再始動に使用されるチェックポイントは、以下の方法で識別されます。すなわち、XRST 呼び出し (左寄せで、残りの区域にはブランク) により示される入出力域に、チェックポイント ID を入力するか、プログラムの JCL の EXEC ステートメントの PARM パラメーターの CKPTID= フィールドに ID を指定します。(両方指定された場合は、IMS は、EXEC ステートメントのパラメーター・フィールドの CKPTID= 値を使用します。)

ID 指定は、次のいずれかが可能です。

- 1 文字から 8 文字の拡張チェックポイント ID。
- メッセージ DFS05401 からの 14 文字の「タイム・スタンプ」ID。ここで、

  IIII は、領域 ID。

  DDD は、年間通算日。

HHMMSST は、時、分、秒、秒の十分の一の値による時刻。

- 4 文字の定数 "LAST"。(BMP のみ : これは、BMP で出した、最後のチェックポイントを、プログラムの再始動に使用するように IMS に指示します。)

システム・メッセージ DFS05401 は、チェックポイント ID とタイム・スタンプを提供します。

システム・メッセージ DFS6821 は、最後に行われたチェックポイントの ID を提供します。これらは、異常終了したバッチ・プログラムまたはバッチ・メッセージ処理プログラム (BMP) の再始動に使用することができます。

再始動されるプログラムが DL/I バッチ領域にある場合、ログ・データ・セットを定義する IMSLOGR DD ステートメントを JCL に提供する必要があります。IMS は、これらのデータ・セットを読み取り、指定された ID を持つチェックポイント・レコードを検索します。

ただし、再始動されるプログラムが BMP 領域にあり、以下の条件がすべて満たされる場合は、IMSLOGR DD ステートメントは必要ありません。

- BMP プログラムが CKPTID=LAST を指定して再始動される。
- BMP プログラムが同じ IMS システム上で、同じジョブ名、同じ PSB、および異常終了時に使用されていたのと同じプログラム名で再始動される。
- BMP プログラムの異常終了以後に IMS のコールド・スタートが行われていない。
- プログラムの再始動に必要なチェックポイント・レコードがあるデータ・セットは、異常終了時以後にアーカイブも再使用もされていない OLDS データ・セットである。あるいは、SLDSREAD ロガー機能が IMS でアクティブになっている。

上記の条件のいずれかが存在しない場合、必要なチェックポイント・レコードが入っているデータ・セットを指す DD ステートメントを提供する必要があります。

IMSLOGR DD ステートメントを提供する場合、必要なチェックポイント・ログ・レコードを入れる必要があります。IMSLOGR DD ステートメントがある場合、IMS は BMP のチェックポイント・レコードを自動的に探して取り出すことはしません。 IMSLOGR DD ステートメントのみが検索され、レコードが見つからない場合は BMP プログラムは異常終了 U0102 で終了します。

注**:** IMSLOGR DD ステートメント用の DD DUMMY ステートメントを提供することもできます。DD DUMMY ステートメントは IMSLOGR DD ステートメントが存在しないかのように処理されます。

XRST 呼び出しの完了時には、入出力域には、再始動に使用される 8 文字のチェックポイント ID が必ず入ります。チェックポイント ID が 8 個のブランク文字のときは例外です。入出力域には 14 文字のタイム・スタンプ (IIIIDDDHHMMSST) が入ります。

入出力 PCB の状況コードも検査してください。XRST 呼び出しの唯一の成功状況コードは、ブランクです。

制約事項

プログラムを正常に始動する場合には、入出力域の先頭の 5 バイトはブランクに設定しなければなりません。

プログラムが再始動され、EXEC ステートメントの PARM フィールドの CKPTID= 値が使用されなかった場合、入出力域に使われる、チェックポイント ID を超える右端のバイトは、すべてブランクに設定する必要があります。

XRST 呼び出しは、バッチおよび BMP アプリケーションからのみ使用可能です。

関連資料:

148 ページの『CHKP (シンボリック) 呼び出し』

EXEC DLI コマンド

EXEC DLI コマンドは、EXEC DLI で使用可能な唯一のコマンドです。これらのコマンドは、バッチ・プログラム、BMP 領域 (DBCTL または DB/DC を実行する)、または DBCTL を使用した CICS プログラムで、DL/I データベースの読み取りと更新を行う場合に使用できます。

システム・サービス・コマンド

次のシステム・サービス・コマンドを使用するには、最初に、SYSSERVE キーワードを指定した SCHD コマンドを出してください。

- ACCEPT コマンド
- DEQ コマンド
- LOG コマンド
- QUERY コマンド
- REFRESH コマンド
- ROLS コマンド
- SETS コマンド
- SETU コマンド
- STAT コマンド

次のシステム・サービス・コマンドは、SYSSERVE キーワードを指定した SCHD コマンドを事前に出さなくても、バッチ、BMP 領域、またはプログラム内で有効です。

- CHKP コマンド
- ROLB コマンド
- ROLL コマンド
- SYMCHKP コマンド
- XRST コマンド

次のシステム・サービス・コマンドは、DBCTL を使用するオンライン CICS プログラムで使用できます。

- ACCEPT

- DEQ
- LOG
- QUERY
- REFRESH
- ROLS
- SETS
- STAT

システム・サービス・コマンドを出すには、入出力 PCB (I/O PCB) が必要です。

以下のトピックの例では、PL/I 区切り文字を使用しています。コマンドはフリー・フォームでコーディングします。示されているキーワード、オペランド、パラメーターがコンマで区切られている場合、コンマの前後にはブランクを入れないでください。 キーワード、オペランド、およびパラメーターがブランクで区切られている場合は、ブランクをいくつでも入れることができます。コマンドの形式は、COBOL、PL/I、またはアセンブラー言語のユーザーにとって同一です。

関連資料:

143 ページの『IMS TM システム・サービスのための DL/I 呼び出し』

41 ページの『IMS DB システム・サービスのための DL/I 呼び出し』

1 ページの『データベース管理のための DL/I 呼び出し』

94 ページの『トランザクション管理のための DL/I 呼び出し』

PCB および PSB (アプリケーション・プログラミング)

EXEC DLI コマンド一覧

以下の表に、すべての EXEC DLI コマンドを要約します。

この表では、EXEC DLI コマンドを示し、各コマンドがバッチ、バッチ型 BMP、または DBCTL 使用 CICS 環境で有効かどうかを示します。

表 *51. EXEC DLI* コマンド一覧：

	プログラム特性		
要求タイプ	バッチ	バッチ型 **BMP**	**DBCTL** を使用する **CICS**[1]
ACCEPT コマンド[4]	あり	あり	あり
CHKP コマンド[4]	あり	あり	なし
DEQ コマンド[4]	あり	あり	あり
DLET コマンド[4]	あり	あり	あり
読み取りコマンド (GU, GHU, GN, GHN, GNP, GHNP)[4]	あり	あり	あり
GMSG コマンド[5]	なし	あり	あり
ICMD コマンド[5]	なし	あり	あり
ISRT コマンド[5]	あり	あり	あり
LOAD コマンド	あり	なし	なし
LOG コマンド[4]	あり	あり	あり
POS コマンド[4]	なし	あり	あり

表 51. EXEC DLI コマンド一覧 (続き):

要求タイプ	プログラム特性		
	バッチ	バッチ型 **BMP**	**DBCTL** を使用する **CICS**[1]
QUERY コマンド[4]	あり	あり	あり
RCMD コマンド[5]	なし	あり	あり
REFRESH コマンド[4]	あり	あり	あり
REPL コマンド[4]	あり	あり	あり
RETRIEVE コマンド	あり	あり	なし
ROLB コマンド	あり	あり	なし
ROLL コマンド	あり	あり	なし
ROLS コマンド[2、4]	あり	あり	あり
SCHD コマンド	なし	なし	あり
SETS コマンド[2、4]	あり	あり	あり
SETU コマンド	あり	あり	なし
STAT コマンド[2、4]	あり	あり	あり
SYMCHKP コマンド	あり	あり	なし
TERM コマンド	なし	なし	あり
XRST コマンド	あり	あり	なし

注:

1. CICS リモート DL/I 環境では、DBCTL を使用するリモート CICS への機能シップを行っている場合、「DBCTL 使用 CICS」の欄のコマンドがサポートされます。
2. ROLS コマンドと SETS コマンドは、PSB に DEDB が入っているときは有効ではありません。
3. STAT は、プロダクト・センシティブ・プログラミング・インターフェースです。
4. これらのコマンドは AIB 形式でサポートされます。
5. これらのコマンドは AOI 資料で説明されています。

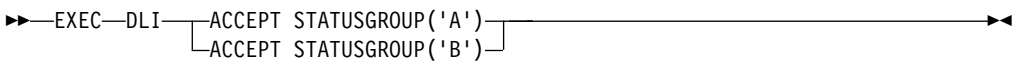
関連概念:

IMS 自動化操作プログラム・インターフェース (AOI) (オペレーションおよびオートメーション)

ACCEPT コマンド

受入れ (ACCEPT) コマンドは、トランザクションを異常終了させるのではなく、状況コードをプログラムに戻すように IMS に指示するために使用します。

フォーマット



オプション

STATUSGROUP('A')
使用できない理由についての状況コードをアプリケーションが受け入れる準備ができていることを IMS に知らせます。IMS はこれを知らされると、以降に出

された呼び出しが使用不能データへのアクセスを必要とする場合に、疑似異常終了させる代わりに、状況コードを戻します。

これは、必須指定のオプションです。

STATUSGROUP('B')
使用できない理由およびデッドロックの発生についての状況コードをアプリケーションが受け入れる準備ができていることを IMS に知らせます。IMS はこれを知らされると、以降に出された呼び出しが使用不能データまたはデッドロックの発生へのアクセスを必要とする場合に、疑似異常終了させる代わりに、状況コードを戻します。

使用法

ACCEPT コマンドを使用して、プログラムを異常終了させるのではなく状況コードを戻すように IMS に指示します。そのような状況コードは、PSB のスケジュールが完了したが、参照されたデータベースのすべてを使用することができなかった場合に戻されます。

例

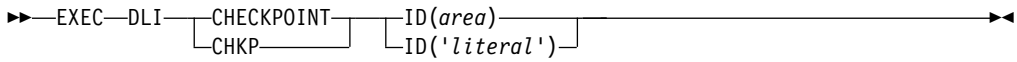
```
EXEC DLI ACCEPT STATUSGROUP('A');
```

上記の例では、ACCEPT コマンドを指定する方法を示しています。

CHKP コマンド

チェックポイント (CHKP) コマンドは、基本チェックポイントを出して論理作業単位を終了するために使用されます。このコマンドは CICS プログラムでは使用できません。

フォーマット



オプション

ID(area)
チェックポイント ID が入ります。 チェックポイント ID が入っている、プログラムの区域の名前です。 示された区域は 8 バイトです。 PL/I を使用している場合、このオプションを、大構造、配列、または文字ストリングを指すポインターとして指定してください。

ID('literal')
'literal' は、引用符で囲まれた 8 バイトのチェックポイント ID です。CHKP コマンドでは、示された区域の長さは 8 バイトです。

使用法

チェックポイントを指定する場合に使用できるコマンドは、基本チェックポイント・コマンドの CHKP、およびシンボリック・チェックポイント・コマンドの SYMCHKP です。

バッチ・プログラムでは、記号コマンドか基本コマンドのいずれかを使用することができます。

これらのチェックポイント・コマンドを使用すると、ユーザーが、プログラムの変更内容をデータベースにコミットし、プログラムが異常終了した場合はこのプログラムが再始動できる位置を設定できるようになります。

IMS チェックポイントを取るため、どの DD ステートメントにも CHKPT=EOV パラメーターは指定しないでください。

どちらのコマンドの場合も、コマンドが出された時点でデータベースの位置がなくなります。 GU コマンドまたはその他の位置設定方法により、位置を再設定しなければなりません。

非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。

基本 CHKP コマンドを出して、データベースに対してプログラムが行った変更をコミットし、プログラムが再始動できる位置を設定できます。 基本 CHKP コマンドを出すときには、プログラムを再始動するためのコードをユーザーが指定する必要があります。

CHKP コマンドを出すときには、チェックポイントの ID を指定する必要があります。この場合、ID が入っているプログラム内のデータ域の名前を指定するか、または実際の ID を単一引用符で囲んで指定することができます。

例

```
EXEC DLI CHKP ID(chkpid);
EXEC DLI CHKP ID('CHKP0007');
```

説明

上記の例では、CHKP コマンドを指定する方法を示しています。

制約事項

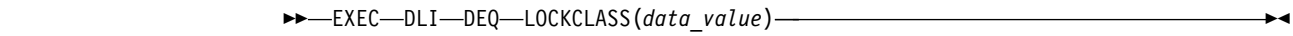
CHKP コマンドの制約事項は、次のとおりです。

- このコマンドは CICS プログラムでは使用できません。
- CHKP コマンドを使用するには、その前にプログラムの I/O PCB を定義する必要があります。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。

DEQ コマンド

デキュー (DEQ) コマンドは、LOCKCLASS オプションを使用して検索されたセグメントを解放するために使用されます。

フォーマット



Option

LOCKCLASS(data_value)

同じ data_value の LOCKCLASS オプションを持つ、以前の GU、GN、または GNP コマンドの結果として保持されているロックを解除することを指定します。Data_value は B から J の範囲内の 1 バイトの英字でなければなりません。

全機能の場合は、LOCKCLASS オプションを指定し、その後にセグメントのロック・クラスを指定します (例えば LOCKCLASS('B'))。 このオプションに文字 (B-J) が指定されていない場合、EXECDLI は状況コード GL を設定し、ABENDU1041 を開始します。

DEQ コマンドは高速機能ではサポートされません。

使用法

LOCKCLASS オプションを使用して検索されたセグメントのロックを解除するときに、DEQ コマンドを使用します。読み取りコマンドに LOCKCLASS を指定すると、ユーザーのトランザクションで排他的に使用するためにセグメントを予約することができます。ユーザーのトランザクションが同期点に達するか、あるいは DEQ コマンドが出されて予約セグメントのロックが解除されるまで、他のトランザクションがこれらの予約セグメントを更新することはできません。LOCKCLASS オプションを指定すると、アプリケーション・プログラムはこれらの予約セグメントを処理せず、変更をまったく加えていない状態でこれらのセグメントを検索します。

例

プログラムにおいて LOCKCLASS オプションは次のように使用できます。

```
EXEC DLI DEQ LOCKCLASS(data_value)
EXEC DLI GU SEGMENT(PARTX)
       SEGMENT(ITEM1) LOCKCLASS('B') INTO(PTAREA1);
EXEC DLI GU SEGMENT(PARTX)
       SEGMENT(ITEM2) LOCKCLASS('C') INTO(PTAREA2);
EXEC DLI DEQ LOCKCLASS('B');
```

説明

この例は、DEQ コマンドの形式を示しています。ここで、data_value は B から J の 1 バイトの英字です。DEQ コマンドは、最初の GU の結果 PARTX セグメントの LOCKCLASS 'B' で保持されているロックを解除します。2 番目の GU の間に PARTX セグメントの LOCKCLASS 'C' のロックはそのまま保持されます。

制約事項

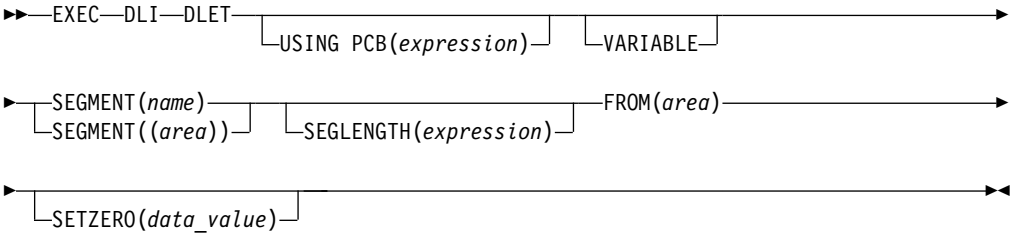
DEQ コマンドの制約事項は、次のとおりです。

- このコマンドを使用するには、最初にプログラムの入出力 PCB を定義しなければなりません。

DLET コマンド

削除 (DLET) コマンドは、データベースからセグメントとその従属セグメントを削除するために使用されます。

フォーマット



オプション

USING PCB(expression)
コマンドのために使用する DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

VARIABLE
セグメントが可変長であることを示します。

SEGMENT(name)
検索、挿入、削除、あるいは置き換えを行いたいセグメント・タイプの名前を指定することにより、コマンドを修飾します。

SEGMENT((area))
プログラムの中でセグメント・タイプの名前を含んでいる区域の参照です。 コマンドでセグメントの名前を指定する代りに、区域を指定することができます。

SEGLENGTH(expression)
検索されたセグメントが入る入出力域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (COBOL プログラムでは、INTO オプションまたは FROM オプションを指定しているすべての SEGMENT レベルに必要です。)

要件: SEGLENGTH に指定する値は、この呼び出しで処理する最長セグメントの長さ以上でなければなりません。

FROM(area)
追加、置き換え、または削除されるセグメントを含んでいる区域を指定します。 1 つのコマンドで 1 つ以上のセグメントを挿入するには、FROM を使用してください。

SETZERO(data_value)
サブセット・ポインターをゼロに設定することを指定します。

使用法

DLET コマンドを使用して、データベースから 1 つのセグメントとその従属セグメントを削除します。セグメントを置き換える場合と同様に、まず、削除するセグメントを検索します。最初に DLET コマンドは検出されたセグメントとさらに その従属セグメント (もしあれば) をデータベースから削除します。

例

「Evelyn Parker は、この地域から転出しました。 彼女の患者番号は 10450 です。彼女のレコードをデータベースから削除してください。」

説明

Evelyn Parker に関するすべての情報をデータベースから削除する必要があります。すべての情報を削除するには、PATIENT セグメントを削除する必要があります。このようにすると、DL/I はそのセグメントのすべての従属セグメントを削除します。これこそユーザーが DL/I に実行させたい作業です。Evelyn Parker がこの診療所の患者でなくなったのであれば、彼女の ILLNESS や TREATMNT などのセグメントを保存しておく理由がないからです。

PATIENT セグメントを削除するには、その前に、次のようなコマンドを使用して、そのセグメントを検索しておかなければなりません。

```
EXEC DLI GU
     SEGMENT(PATIENT) INTO(PATAREA) WHERE (PATNO=PATNO1);
```

この患者のデータベース・レコードを削除するには、DLET コマンドを出し、この時に FROM オプションを使用して、削除するセグメントが入っている入出力域の名前を指定します。

```
EXEC DLI DLET SEGMENT(PATIENT) FROM(PATAREA);
```

このコマンドを出すと、PATIENT セグメントとその従属セグメント (ILLNESS、TREATMNT、BILLING、PAYMENT、および HOUSHOLD の各セグメント) が削除されます。

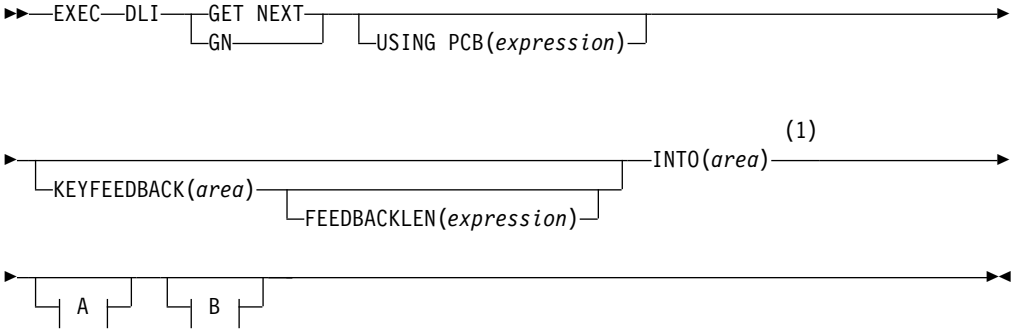
制約事項

検索コマンドと DLET コマンドとの間で、同じ PCB を使用するコマンドを出すことはできません。また、1 つの**読み取り (GET)** コマンドにつき 1 つの DLET コマンドしか出すことができません。

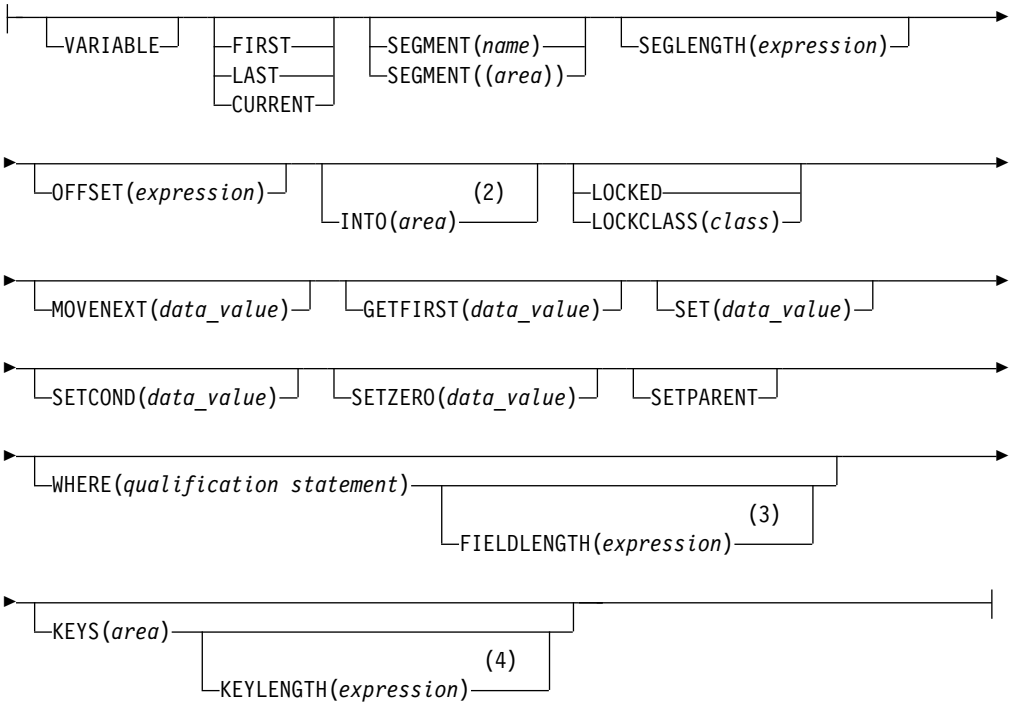
GN コマンド

Get Next (GN) コマンドは、セグメントを順に検索するために使用されます。

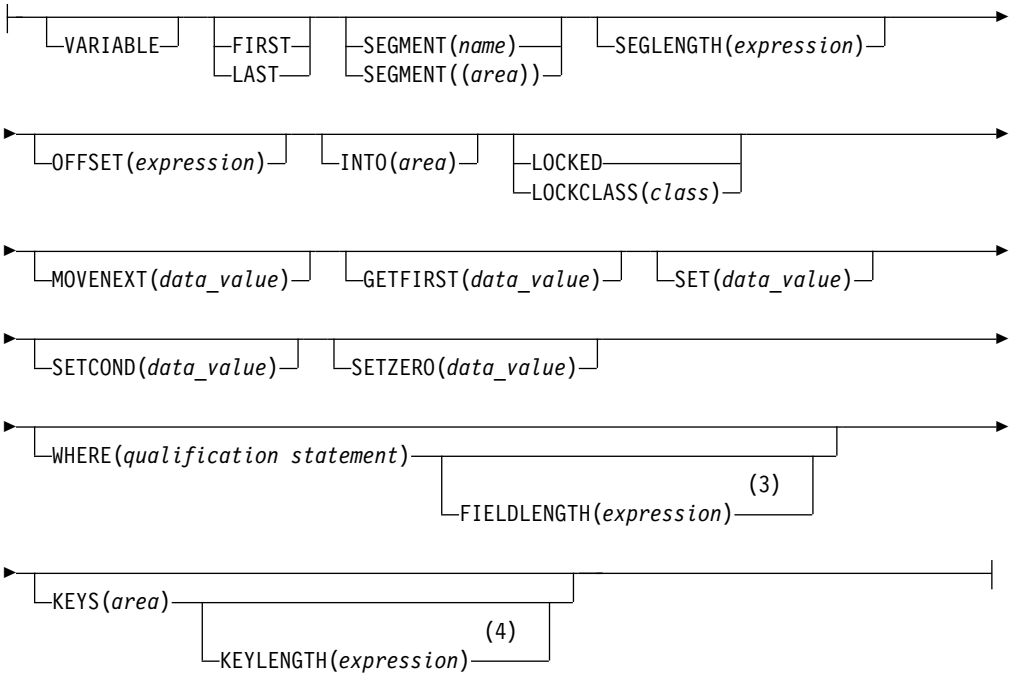
フォーマット



A それぞれの親セグメントについては、次のようになります **(**オプション**)**：



B オブジェクト・セグメントについては、次のようになります **(**オプション**)**：



注**:**

1 SEGMENT オプションを省略する場合、示されているように INTO オプションを指定してください。

2 パス・コマンドでは、親セグメントで INTO を指定してください。

3 複数の修飾ステートメントを使用する場合には、FIELDLENGTH を使用してそれぞれの長さを指定してください。例えば、**FIELDLENGTH(24,8)** と指定してください。

4 1 つのセグメント・レベルで、KEYS オプションか WHERE オプションのいずれかを使用することができますが、両方は使用できません。

オプション

USING PCB(expression)
コマンドのために使用する DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

KEYFEEDBACK(area)
セグメントの連結キーが入る区域を指定します。 区域の長さが足りない場合、キーは切り捨てられます。

FEEDBACKLEN(expression)
検索された連結キーを入れたいキー・フィードバック域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (これは、COBOL プログラムでは必須ですが、PL/I およびアセンブラー言語の各プログラムでは任意指定です。)

INTO(area)
セグメントが読み込まれる区域を指定します。

VARIABLE
セグメントが可変長であることを示します。

FIRST
あるセグメント・タイプの最初に現れるセグメントを検索すること、もしくはセグメントを最初に現れるように挿入することを指定します。

LAST
あるセグメント・タイプの最後に現れるセグメントを検索すること、もしくはセグメントを最後に現れるように挿入することを指定します。

CURRENT
コマンドを修飾し、現在位置のレベル以上のレベルをこのセグメントの修飾として使用することを示します。

SEGMENT(name), SEGMENT((area))
検索するセグメント・タイプの名前、もしくはプログラムの中でその名前を含んでいる区域を指定することにより、コマンドを修飾します。

GN コマンドには、データベースの階層レベル数と同数の修飾レベルを指定できます。WHERE または KEYS オプションが指定された完全修飾コマンドを使用することにより、必要な階層パスとセグメントを明示することができるので、コマンドについて記述するときに便利です。 ただし、GN コマンドは SEGMENT オプションを使用せずに指定できるため、GN コマンドを修飾する必要はありません。

データベース・レコード内の位置を設定したあと、SEGMENT オプションを指定せずに GN コマンドを出すと、次のセグメントが現れる位置が順次に検索されます。

KEYS または WHERE オプションを指定せずに SEGMENT オプションだけを指定すると、IMS DB は現在位置から順方向に検索して、検出されるセグメント・タイプが最初に現れる位置を検索します。GN コマンドを修飾せずに使用する場合、予期していたセグメント・タイプが検索されないことがあるため、プログラムがアクセスできる最大セグメントが入るだけの大きさの入出力域を指定してください。 (検索コマンドが成功したあとで、DIB から検索したセグメント・タイプを調べることができます。)

WHERE または KEYS オプションを使用してコマンドを完全に修飾すると、オプションの記述に従って、次のセグメントが順次に検索されます。

親セグメントに対して WHERE または KEYS オプションを指定すると、ユーザーが検索したいセグメントへのパスの一部であるセグメント・オカレンスが定義されます。 レベルに対する SEGMENT オプションを省略するか、または SEGMENT オプションを指定して WHERE オプションを指定しない場合、SEGMENT オプションへのパスをどのように指定しても、このコマンドを実行できることになります。 DL/I は、修飾された親セグメントと最低レベルの SEGMENT オプションだけを使用してコマンドを正しく実行します。 DL/I は、不在レベルの修飾を使用しません。

SEGLENGTH(expression)
検索されたセグメントが入る入出力域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (COBOL プログラムでは、INTO オプションまたは FROM オプションを指定しているすべての SEGMENT レベルに必要です。)

要件: SEGLENGTH に指定する値は、この呼び出しで処理する最長セグメントの長さ以上でなければなりません。

OFFSET(expression)
目標親に対するオフセットを指定します。 整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定するか、あるいはプログラムの中で数値を含んでいるハーフワードへの参照を指定します。 論理関係にある連結セグメントを処理するときには、OFFSET を使用します。目標親が可変長セグメントの場合には、OFFSET の指定は必須です。

LOCKED
チェックポイントまたは同期点に達するまで、ユーザーのプログラムが排他的に使用するセグメントを検索することを指定します。このオプションは Q コマンド・コードと同様に機能し、高速機能と全機能の両方に適用されます。 Q コマンド・コードのクラスとして、1 バイトの英字「A」が自動的に付加されます。

LOCKCLASS(class)
DEQ コマンドが出されるまで、あるいはチェックポイントまたは同期点に達するまで、ユーザーのプログラムが排他的に使用するセグメントを検索することを指定します。(DEQ コマンドは高速機能ではサポートされません。) Class は、検索セグメントのロック・クラスを表す 1 バイトの英字 (B-J) です。

全機能コードの場合は、LOCKCLASS オプションの後に文字 (B-J) を続けて、セグメントのロック・クラスを指定します。例えば、LOCKCLASS('B') です。LOCKCLASS の後ろに B から J の範囲の文字が指定されていない場合、EXECDLI は状況コード GL を設定し、ABENDU1041 を開始します。

高速機能では LOCKCLASS はサポートされていませんが、全機能と高速機能との一貫性をとるため、LOCKCLASS('x') を指定してください。この x は B から J の範囲の文字です。例えば、LOCKCLASS('B') となります。 LOCKCLASS の後ろに B から J の範囲の文字が指定されていない場合、EXECDLI は状況コード GL を設定し、ABENDU1041 を開始します。

MOVENEXT(data_value)
サブセット・ポインターを、現行セグメントの次のセグメントが現れる位置に移動させることを指定します。

GETFIRST(data_value)
サブセット内のセグメントが最初に現れる位置から検索を開始したいことを指定します。

SET(data_value)
無条件でサブセット・ポインターを現行セグメントに設定することを指定します。

SETCOND(data_value)
条件付きでサブセット・ポインターを現行セグメントに設定することを指定します。

SETZERO(data_value)
サブセット・ポインターをゼロに設定することを指定します。

SETPARENT
必要なレベルで親子関係を設定します。

FIELDLENGTH(expression)
WHERE オプションのフィールド値の長さを指定します。

KEYLENGTH(expression)
KEYS オプションを使用する際の、連結キーの長さを指定します。整数データ型に変換することができる、ホスト言語で書かれた式であればどのような式でも可能です。変数を指定する場合、バイナリー・ハーフワード値で宣言しなければなりません。 IBM COBOL for z/OS & VM (または VS COBOL II)、PL/I、アセンブラー言語の各プログラムの場合、KEYLENGTH はオプションです。IBM COBOL for z/OS & VM (または VS COBOL II) コンパイラーでコンパイルしていない COBOL プログラムの場合、KEYS オプションに KEYLENGTH を指定してください。

KEYS(area)
セグメントの連結キーでコマンドを修飾します。セグメント・レベルに対して KEYS または WHERE のいずれかを使用することができますが、両方を使用することはできません。

「area」は、セグメントの連結キーが入っているプログラム内の区域を指定します。

WHERE(qualification statement)
セグメントが現れる位置を指定することで、コマンドを修飾します。その引数

は、1 つ以上の修飾ステートメントから成り、それぞれがセグメントのフィールド内の値と、ユーザーが指定した値を比較します。各修飾ステートメントは次のものから構成されています。

- セグメント中のフィールドの名前
- どのように 2 つの値を比較するかを示す関係演算子
- フィールドの値と比較される値を含んでいる、プログラム中のデータ域の名前

使用法

GN コマンドは、データベースからセグメントを順次検索するために使用します。GN コマンドを出すたびに、IMS DB はこのコマンドに指定されているオプションに従って次のセグメントを検索します。 GN コマンドを出す前に、 GU コマンドを出してデータベース・レコード内の位置を設定しておかなければなりません。

GN コマンドにはセグメント・オプションを指定する必要はありません。 ただし、SEGMENT オプションのあとにできるだけ KEYS または WHERE オプションを指定して、GN コマンドを修飾してください。

例

例 ***1***

「診療所に来たことのある患者全員のリストが必要です。」

説明: この要求に対応するには、DL/I が状況コード GB をプログラムに戻すまで、プログラムがセグメント名 PATIENT で修飾されているコマンドを発行する必要があります。 (GB は、DL/I がコマンドを正しく実行することができないうちにデータベースの終わりに達したことを意味します) このコマンドは次のようになります。

```
EXEC DLI GN
   SEGMENT(PATIENT) INTO(PATAREA);
```

プログラムがこのコマンドを出すたびに、データベース・レコード内の現在位置が次のデータベース・レコードに順方向に移動します。

例 ***2***

「今月初めからこれまでに診察した患者の名前を挙げてください。」

説明: 1 つ以上の WHERE オプションまたは KEYS オプションが指定されている GN コマンドは、コマンドの要求を満たすために指定のセグメント・タイプが次に現れる位置を検索します。 この要求に対応するために、プログラムは、DL/I が GB 状況コードを戻すまで、次に示す GN コマンドを出します。 次の例では、1988 年 4 月末に使用するコマンドを示します (ILLDATE1 に 198804010 が指定されるものと想定しています)。

```
EXEC DLI GN
   SEGMENT(PATIENT) INTO(PATAREA)
   SEGMENT(ILLNESS) INTO(ILLAREA) WHERE(ILLDATE>=ILLDATE1);
```

例 ***3***

```
EXEC DLI GN INTO(PATAREA);
```

説明: 患者 04124 の PATIENT セグメントを検索した直後にこのコマンドを出した場合、患者 04124 の最初の ILLNESS セグメントが検索されます。

制約事項

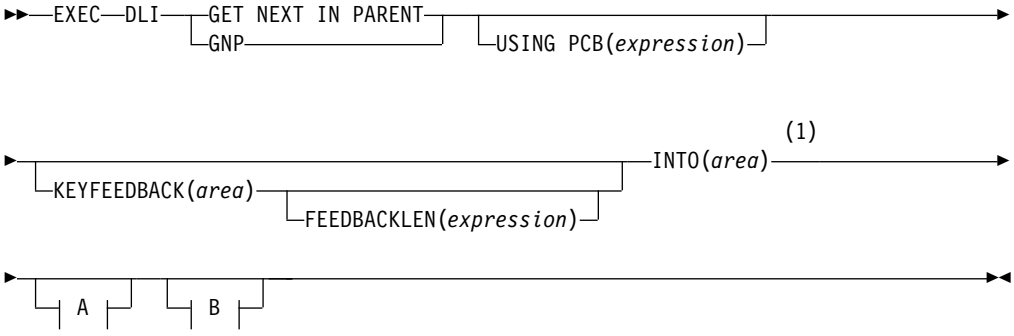
GN コマンドを修飾せずに使用すると、目的のセグメント・タイプが検索されない可能性があります。そのため、プログラムがアクセスできる最大セグメントが入るだけの大きさをもつ入出力域を指定してください。

セグメント・レベルに対して KEYS オプションまたは WHERE オプションのいずれかを使用できますが、両方は使用できません。

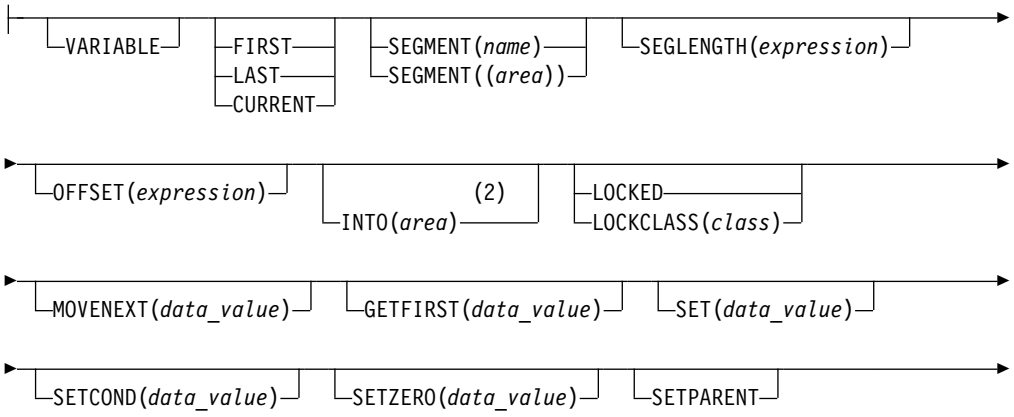
GNP コマンド

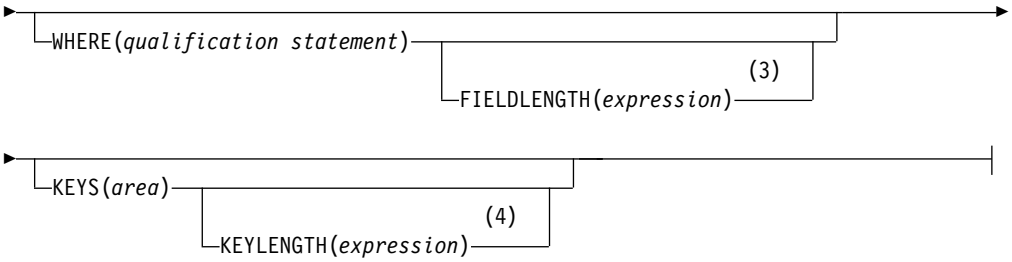
Get Next in Parent (GNP) コマンドは、従属セグメントを順次に検索するために使用されます。

フォーマット

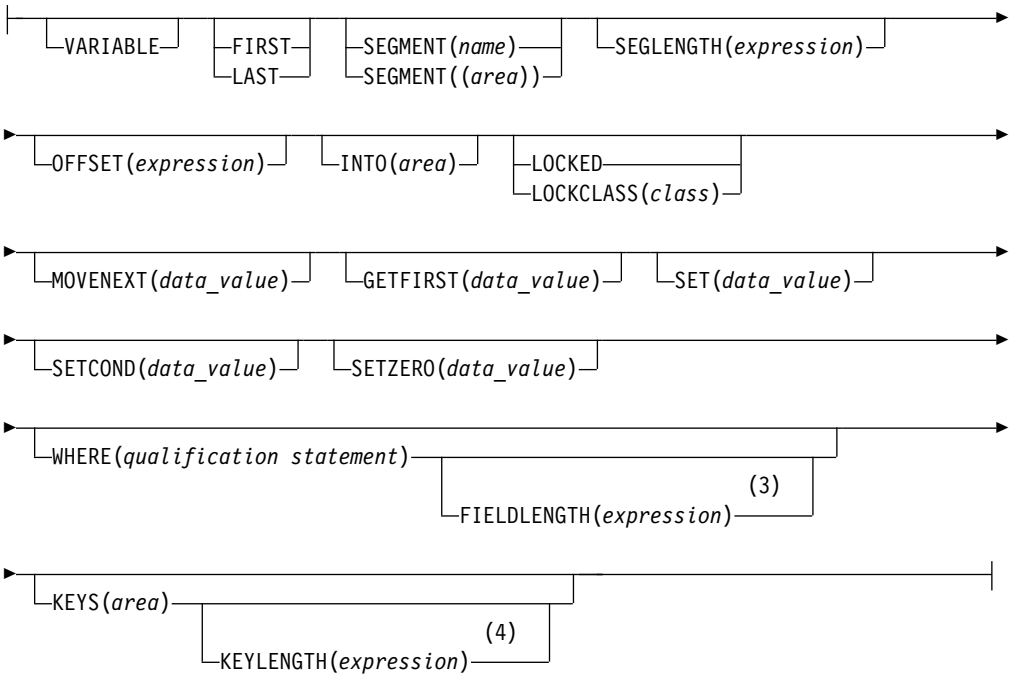


A それぞれの親セグメントについては、次のようになります **(**オプション**) :**





B オブジェクト・セグメントについては、次のようになります **(**オプション**)**：



注**:**

1 SEGMENT オプションを省略する場合、示されているように INTO オプションを指定してください。

2 パス・コマンドでは、親セグメントで INTO を指定してください。

3 複数の修飾ステートメントを使用する場合には、FIELDLENGTH を使用してそれぞれの長さを指定してください。例えば、**FIELDLENGTH(24,8)** と指定してください。

4 1 つのセグメント・レベルで、KEYS オプションか WHERE オプションのいずれかを使用することができますが、両方は使用できません。

オプション

SEGMENT オプションと WHERE オプションを使用して GNP コマンドを修飾することができます。

コマンドを修飾しないと、IMS DB は設定された親の下で次の順次セグメントを検索します。SEGMENT オプションを指定すると、IMS DB は設定された親の下で順方向に検索をして、指定のセグメント・タイプが最初に現れる位置を検索します。

データベースの階層レベル数と同数の修飾レベルを GNP コマンドに指定できます。ただし、コマンドを修飾するときに、このコマンドの親として設定されたセグメント・タイプから DL/I が出ないような方法で行わなければなりません。

USING PCB(expression)
コマンドのために使用する DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

KEYFEEDBACK(area)
セグメントの連結キーが入る区域を指定します。 区域の長さが足りない場合、キーは切り捨てられます。これを使用して、セグメントの連結キーを検索してください。

FEEDBACKLEN(expression)
検索された連結キーを入れたいキー・フィードバック域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (これは、COBOL プログラムでは必須ですが、PL/I およびアセンブラー言語の各プログラムでは任意指定です。)

INTO(area)
セグメントが読み込まれる区域を指定します。 このオプションを使用すると、1 つのコマンドで 1 つ以上のセグメントを検索できます。

VARIABLE
セグメントが可変長であることを示します。

FIRST
あるセグメント・タイプの最初に現れるセグメントを検索すること、もしくはセグメントを最初に現れるように挿入することを指定します。これを使用して、あるセグメント・タイプのセグメントが最初に現れる位置を検索してください。

LAST
あるセグメント・タイプの最後に現れるセグメントを検索すること、もしくはセグメントを最後に現れるように挿入することを指定します。これを使用して、あるセグメント・タイプのセグメントが最後に現れる位置を検索してください。

CURRENT
コマンドを修飾し、現在位置のレベル以上のレベルをこのセグメントの修飾として使用することを示します。これを使用して、現在位置に基づいてセグメントを検索してください。

SEGLENGTH(expression)
検索されたセグメントが入る入出力域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (COBOL プログラムでは、INTO オプションまたは FROM オプションを指定している SEGMENT レベルには SEGLENGTH が必須です。)

要件: SEGLENGTH に指定する値は、この呼び出しで処理する最長セグメントの長さ以上でなければなりません。

OFFSET(expression)
目標親に対するオフセットを指定します。 引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定するか、またはプログラム内の数値を含むハーフワードへの参照を指定することができます。論理関係にある連結セグメントを処理するときには、OFFSET を使用します。目標親が可変長セグメントの場合には、OFFSET の指定は必須です。

LOCKED
チェックポイントまたは同期点に達するまで、ユーザーのプログラムが排他的に使用するセグメントを検索することを指定します。これを使用して、プログラムを排他的に使用するためにセグメントを予約してください。このオプションは Q コマンド・コードと同様に機能し、高速機能と全機能の両方に適用されます。 Q コマンド・コードのクラスとして、1 バイトの英字「A」が自動的に付加されます。

LOCKCLASS(class)
DEQ コマンドが出されるまで、あるいはチェックポイントまたは同期点に達するまで、ユーザーのプログラムが排他的に使用するセグメントを検索することを指定します。(DEQ コマンドは高速機能ではサポートされません。) Class は、検索セグメントのロック・クラスを表す 1 バイトの英字 (B-J) です。

全機能コードの場合は、LOCKCLASS オプションの後に文字 (B-J) を続けて、セグメントのロック・クラスを指定します。例えば、LOCKCLASS('B') です。LOCKCLASS の後ろに B から J の範囲の文字が指定されていない場合、EXECDLI は状況コード GL を設定し、ABENDU1041 を開始します。

高速機能では LOCKCLASS はサポートされていませんが、全機能と高速機能との一貫性をとるため、LOCKCLASS('x') を指定してください。この x は B から J の範囲の文字です。例えば、LOCKCLASS('B') となります。 LOCKCLASS の後ろに B から J の範囲の文字が指定されていない場合、EXECDLI は状況コード GL を設定し、ABENDU1041 を開始します。

MOVENEXT(data_value)
サブセット・ポインターを、現行セグメントの次のセグメントが現れる位置に移動させることを指定します。

GETFIRST(data_value)
サブセット内のセグメントが最初に現れる位置から検索を開始したいことを指定します。

SET(data_value)
無条件でサブセット・ポインターを現行セグメントに設定することを指定します。

SETCOND(data_value)
条件付きでサブセット・ポインターを現行セグメントに設定することを指定します。

SETZERO(data_value)
サブセット・ポインターをゼロに設定することを指定します。

SETPARENT
必要なレベルで親子関係を設定します。

WHERE(qualification statement)
セグメントが現れる位置を指定することで、コマンドを修飾します。その引数は、1 つ以上の修飾ステートメントから成り、それぞれがセグメントのフィールド内の値と、ユーザーが指定した値を比較します。各修飾ステートメントは次のものから構成されています。

- セグメント中のフィールドの名前
- どのように 2 つの値を比較するかを示す関係演算子
- フィールドの値と比較される値を含んでいる、プログラム中のデータ域の名前

FIELDLENGTH(expression)
WHERE オプションのフィールド値の長さを指定します。

KEYS(area)
セグメントの連結キーでコマンドを修飾します。セグメント・レベルに対して KEYS または WHERE のいずれかを使用することができますが、両方を使用することはできません。

「area」は、セグメントの連結キーが入っているプログラム内の区域を指定します。

KEYLENGTH(expression)
KEYS オプションを使用する際の、連結キーの長さを指定します。整数データ型に変換することができる、ホスト言語で書かれた式であればどのような式でも可能です。変数を指定する場合、バイナリー・ハーフワード値で宣言しなければなりません。 IBM COBOL for z/OS & VM (または VS COBOL II)、PL/I、アセンブラー言語の各プログラムの場合、KEYLENGTH はオプションです。IBM COBOL for z/OS & VM (または VS COBOL II) コンパイラーでコンパイルしていない COBOL プログラムの場合、KEYS オプションに KEYLENGTH を指定してください。

SEGMENT(name), SEGMENT((area))
検索、挿入、削除、または置き換えを行いたいセグメント・タイプの名前、もしくはその名前が入っているプログラムの区域を指定することにより、コマンドを修飾します。

データベースの階層レベル数と同数の修飾レベルを GNP コマンドに指定できます。WHERE または KEYS オプションが指定された完全修飾コマンドを使用することにより、必要な階層パスとセグメントを明示することができるので、コマンドについて記述するときに便利です。 ただし、GNP コマンドを修飾する必要はまったくありません。これは、SEGMENT オプションがなくても GNP コマンドを指定できるためです。

データベース・レコード内の位置を設定したあと、SEGMENT オプションを指定せずに GNP コマンドを出すと、次のセグメントが現れる位置が順次に検索されます。

KEYS または WHERE オプションを指定せずに SEGMENT オプションだけを指定すると、IMS DB は現在位置から順方向に検索して、検出されるセグメント・タイプが最初に現れる位置を検索します。GNP コマンドを修飾せずに使用す

る場合、予期していたセグメント・タイプが検索されないことがあるため、プログラムがアクセスできる最大セグメントが入るだけの大きさの入出力域を指定してください。(検索コマンドが成功したあとで、DIB から検索セグメント・タイプを見つけることができます。)

WHERE または KEYS オプションを使用してコマンドを完全に修飾すると、オプションの記述に従って、次のセグメントが順次に検索されます。

親セグメントに対して WHERE または KEYS オプションを指定すると、ユーザーが検索したいセグメントへのパスの一部であるセグメント・オカレンスが定義されます。 レベルに対する SEGMENT オプションを省略するか、または SEGMENT オプションを指定して WHERE オプションを指定しない場合、SEGMENT オプションへのパスをどのように指定しても、このコマンドを実行できることになります。 DL/I は、修飾された親セグメントと最低レベルの SEGMENT オプションだけを使用してコマンドを正しく実行します。 DL/I は、不在レベルの修飾を使用しません。

使用法

従属セグメントの順次検索 (GNP) コマンドを使用すれば、セグメントの検索を限定することができます。つまり、特定の親の従属セグメントだけを検索することができます。 GNP コマンドを出す前には、親子関係を設定しておかなければなりません。

例

例 ***1***

「Kate Bailey に関するすべてのレコードが必要です。彼女の患者番号は 09080 です。」

説明: この要求に応じるには、患者番号 09080 の患者の従属セグメントだけを調べればよく、すべての患者の従属セグメントを検索する必要はありません。これを行うには、GU コマンドを使用して、Kate Bailey の PATIENT セグメントに現在の位置と親子関係を設定します。そのあと、DL/I がその PATIENT セグメントのすべての従属セグメントを戻すまで、SEGMENT または WHERE オプションの指定されていない GNP を出し続けます。(GE 状況コードは、すべての従属セグメントが検索されたことを表します。) この要求に応じるために、プログラムは次のようなコマンドを出すことができます。

```
EXEC DLI GU
     SEGMENT(PATIENT) INTO(PATAREA)
     WHERE (PATNO=PATNO1);
EXEC DLI GNP
     INTO(ILLAREA);
```

GNP コマンドは、SEGMENT または WHERE オプションが指定されていないと、現在の親の下で従属セグメントが最初に現れる位置を検索します。 現在位置がすでに現在の親の従属セグメントにある場合、このコマンドは、この親の下で次のセグメントが現れる位置を検索します。

GNP コマンドを修飾せずに使用する場合、予期していたセグメント・タイプが検索されないことがあるため、プログラムがアクセスできる最大セグメントが入るだけ

の大きさの入出力域を指定してください。(GNP コマンドが正常に出されると、DIB から検索セグメント・タイプを調べることができます。)

例 *2*

「頭痛にアセトアミノフェンを処方していた医師は誰ですか。」

説明*:* SEGMENT オプションだけが指定されている GNP コマンドは、設定した親の下で指定したセグメント・タイプの従属セグメントを順次に検索します。この例において、ILLNESS のキーは ILLNAME であり、TREATMNT のキーは MEDICINE とします。 この治療が処方された各 TREATMNT セグメントを検索する必要があります。この治療を処方した医師の名前は、TREATMNT セグメントの一部です。(データ域 ILLNAME1 には HEADACHE があり、MEDIC1 には ACETAMINOP が入っているものとします。) この要求に対応するには、次のようなコマンドを出します。

```
EXEC DLI GN
     SEGMENT(ILLNESS) WHERE (ILLNAME=ILLNAME1);
EXEC DLI GNP
     SEGMENT(TREATMNT) WHERE (MEDICINE=MEDIC1);
```

このコマンドを処理するため、プログラムは DL/I が状況コード GE (該当なし) を戻すまで GNP コマンドを発行し、この状況コードが戻されると、プログラムは次の HEADACHE セグメントを検索し、さらにこれに対する TREATMNT セグメントを検索します。プログラムは、ILLNAME が HEADACHE である ILLNESS セグメントがなくなるまでこれを行います。

制約事項

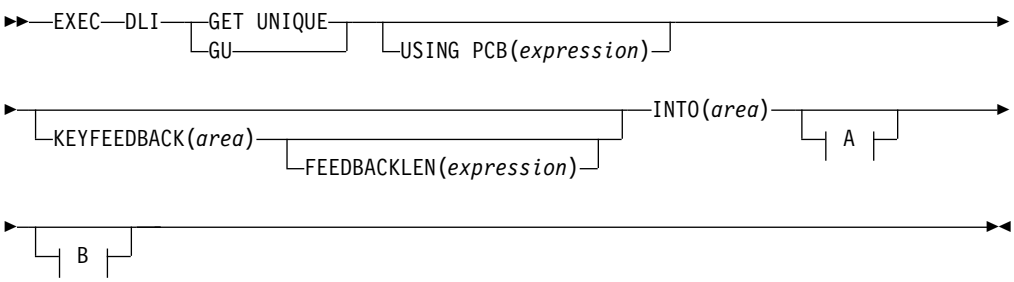
GNP コマンドの制約事項は、次のとおりです。

- このコマンドを出す前には、親子関係を設定しておかなければなりません。
- GNP コマンドを修飾するときは、このコマンドの親として設定されているセグメント・タイプから DL/I が移動するような方法では修飾できません。
- 特定の親の従属セグメントしか検索することができません。

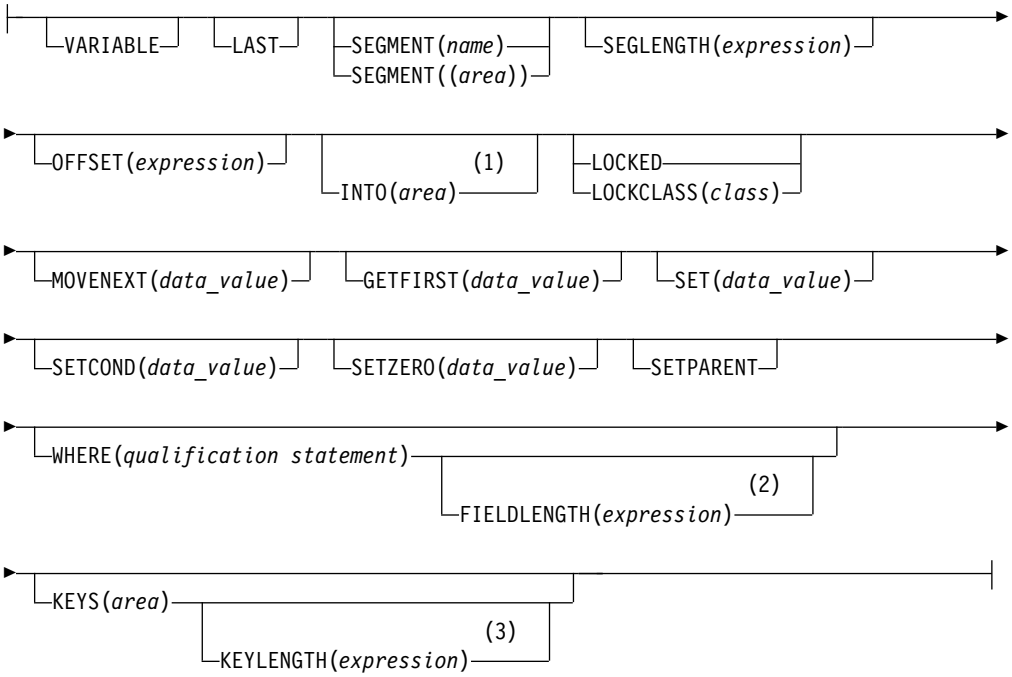
GU コマンド

セグメントの Get Unique (GU) コマンドは、特定のセグメントを直接検索するため、および順次処理を行うデータベース内の開始位置を設定するために使用されます。

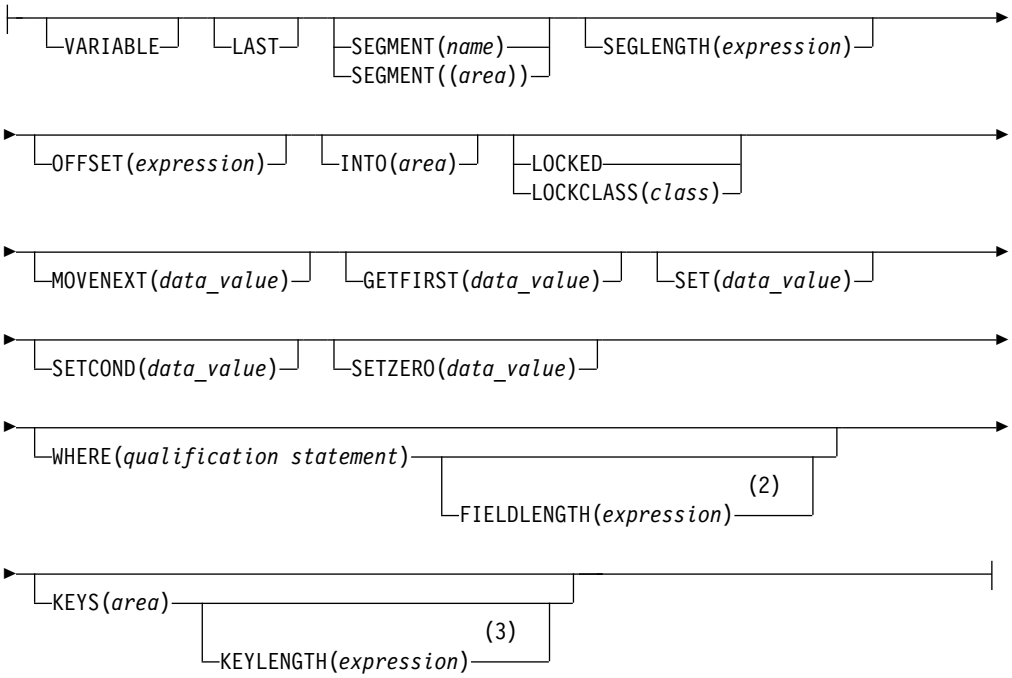
フォーマット



A:



B:



注:

1 パス・コマンドでは、親セグメントで INTO を指定してください。

2 複数の修飾ステートメントを使用する場合には、FIELDLENGTH を使用してそれぞれの長さを指定してください。例えば、**FIELDLENGTH(24,8)** と指定してください。

3 1 つのセグメント・レベルで、KEYS オプションか WHERE オプションのいずれかを使用することができますが、両方は使用できません。

オプション

USING PCB(expression)
コマンドのために使用する DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

KEYFEEDBACK(area)
セグメントの連結キーが入る区域を指定します。 区域の長さが足りない場合、キーは切り捨てられます。

FEEDBACKLEN(expression)
検索された連結キーを入れたいキー・フィードバック域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (これは、COBOL プログラムでは必須ですが、PL/I およびアセンブラー言語の各プログラムでは任意指定です。)

INTO(area)
セグメントが読み込まれる区域を指定します。

VARIABLE
セグメントが可変長であることを示します。

LAST
あるセグメント・タイプの最後に現れるセグメントを検索すること、もしくはセグメントを最後に現れるように挿入することを指定します。

SEGMENT(name), SEGMENT((area))
検索、挿入、削除、または置き換えを行いたいセグメント・タイプの名前、もしくはその名前が入っているプログラムの区域を指定することにより、コマンドを修飾します。

あるセグメント・タイプが最初に現れる位置を検索するときは、SEGMENT オプションを指定するだけでかまいません。 使用中の PCB によって定義された階層レベルと同数の修飾レベルを指定することができます。

データベースの先頭に位置を設定する場合は、ルート・セグメント・タイプを指定した SEGMENT オプションを使用した GU コマンドを出します。

1 つ以上の階層レベルに対して SEGMENT オプションを指定しないと、DL/I はそのレベルのセグメント修飾を使用します。DL/I がどの修飾を使用するかは、現在位置によって決まります。

- DL/I の位置が不在レベルに設定されている場合、DL/I はこの位置が設定されているセグメントを使用します。

- DL/I の位置が不在レベルに設定されていない場合、DL/I はそのレベルで最初に現れる位置を使用します。
- DL/I が、もっと高いレベルに設定された位置から前進する場合、DL/I は新しいパスに含まれる不在レベルの最初に現れる位置を使用します。
- ルート・レベルの SEGMENT オプションが省略されており、DL/I の位置がルートに設定されている場合、コマンドを正しく実行しようとするとき DL/I はそのルートから移動しません。

  データベースの階層レベル数と同数の修飾レベルを GU コマンドに指定できます。WHERE または KEYS オプションが指定された完全修飾コマンドを使用することにより、必要な階層パスとセグメントを明示することができるので、コマンドについて記述するときに便利です。 ただし、GU コマンドを修飾する必要はまったくありません。これは、SEGMENT オプションがなくても GU コマンドを指定できるためです。

  KEYS または WHERE オプションを指定せずに SEGMENT オプションだけを指定すると、IMS DB は現在位置から順方向に検索して、検出されるセグメント・タイプが最初に現れる位置を検索します。GU コマンドを修飾せずに使用する場合、予期していたセグメント・タイプが検索されないことがあるため、プログラムがアクセスできる最大セグメントが入るだけの大きさの入出力域を指定してください。(検索コマンドが成功したあとで、DIB から検索セグメント・タイプを見つけることができます。)

  WHERE または KEYS オプションを使用してコマンドを完全に修飾すると、オプションの記述に従って、次のセグメントが順次に検索されます。

  親セグメントに対して WHERE または KEYS オプションを指定すると、ユーザーが検索したいセグメントへのパスの一部であるセグメント・オカレンスが定義されます。 レベルに対する SEGMENT オプションを省略するか、または SEGMENT オプションを指定して WHERE オプションを指定しない場合、SEGMENT オプションへのパスをどのように指定しても、このコマンドを実行できることになります。 DL/I は、修飾された親セグメントと最低レベルの SEGMENT オプションだけを使用してコマンドを正しく実行します。 DL/I は、不在レベルの修飾を使用しません。

SEGLENGTH(expression)
検索されたセグメントが入る入出力域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (COBOL プログラムでは、INTO オプションまたは FROM オプションを指定している SEGMENT レベルには SEGLENGTH が必須です。)

要件: SEGLENGTH に指定する値は、この呼び出しで処理する最長セグメントの長さ以上でなければなりません。

OFFSET(expression)
目標親に対するオフセットを指定します。 引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定するか、またはプログラム内の数値を含むハーフワードへの参照を指定することができます。論理

関係にある連結セグメントを処理するときには、OFFSET を使用します。目標親が可変長セグメントの場合には、OFFSET の指定は必須です。

LOCKED
チェックポイントまたは同期点に達するまで、ユーザーのプログラムが排他的に使用するセグメントを検索することを指定します。このオプションは、Q コマンド・コードと同様に機能します。このオプションは高速機能と全機能の両方に適用されます。Q コマンド・コードのクラスとして、1 バイトの英字「A」が自動的に付加されます。

LOCKCLASS(class)
DEQ コマンドが出されるまで、あるいはチェックポイントまたは同期点に達するまで、ユーザーのプログラムが排他的に使用するセグメントを検索することを指定します。(DEQ コマンドは高速機能ではサポートされません。) Class は、検索セグメントのロック・クラスを表す 1 バイトの英字 (B-J) です。

全機能コードの場合は、LOCKCLASS オプションの後に文字 (B-J) を続けて、セグメントのロック・クラスを指定します。例えば、LOCKCLASS('B') です。LOCKCLASS の後ろに B から J の範囲の文字が指定されていない場合、EXECDLI は状況コード GL を設定し、ABENDU1041 を開始します。

高速機能では LOCKCLASS はサポートされていませんが、全機能と高速機能との一貫性をとるため、LOCKCLASS('x') を指定してください。この x は B から J の範囲の文字です。例えば、LOCKCLASS('B') となります。 LOCKCLASS の後ろに B から J の範囲の文字が指定されていない場合、EXECDLI は状況コード GL を設定し、ABENDU1041 を開始します。

MOVENEXT(data_value)
サブセット・ポインターを、現行セグメントの次のセグメントが現れる位置に移動させることを指定します。

GETFIRST(data_value)
サブセット内のセグメントが最初に現れる位置から検索を開始したいことを指定します。

SET(data_value)
無条件でサブセット・ポインターを現行セグメントに設定することを指定します。

SETCOND(data_value)
条件付きでサブセット・ポインターを現行セグメントに設定することを指定します。

SETZERO(data_value)
サブセット・ポインターをゼロに設定することを指定します。

SETPARENT
必要なレベルで親子関係を設定します。

FIELDLENGTH(expression)
WHERE オプションのフィールド値の長さを指定します。

KEYLENGTH(expression)
KEYS オプションを使用する際の、連結キーの長さを指定します。引数には、整数データ型に変換されるホスト言語の式であればどのようなものでも指定できます。変数はバイナリー・ハーフワード値として宣言してください。 IBM

COBOL for z/OS & VM (または VS COBOL II)、PL/I、アセンブラー言語の各プログラムの場合、KEYLENGTH はオプションです。 IBM COBOL for z/OS & VM (または VS COBOL II) コンパイラーでコンパイルしていない COBOL プログラムの場合、KEYS オプションに KEYLENGTH を指定してください。

WHERE(qualification statement)
GU コマンドをさらに修飾する場合は、SEGMENT オプションのあとに WHERE を使用します。 GU コマンドを完全に修飾すると、データベース・レコード内での現在位置に関係なく、セグメントを検索することができます。

KEYS(area)
KEYS を使用して、GU コマンドをさらに修飾し、連結キーを使用してセグメント・オカレンスを指定します。

KEYS または WHERE オプションを指定せずに SEGMENT オプションだけを指定すると、IMS DB は現在位置から順方向に検索して、検出されるセグメント・タイプが最初に現れる位置を検索します。GU コマンドを修飾せずに使用する場合、予期していたセグメント・タイプが検索されないことがあるため、プログラムがアクセスできる最大セグメントが入るだけの大きさの入出力域を指定してください。(検索コマンドが成功したあとで、DIB から検索セグメント・タイプを見つけることができます。)

WHERE または KEYS オプションを使用してコマンドを完全に修飾すると、オプションの記述に従って、次のセグメントが順次に検索されます。

親セグメントに対して WHERE または KEYS オプションを指定すると、ユーザーが検索したいセグメントへのパスの一部であるセグメント・オカレンスが定義されます。 レベルに対して SEGMENT オプションを省略するか、または WHERE オプションを指定せずに SEGMENT オプションだけを指定すると、そのオプションへのパスはすべてコマンドを正しく実行することを表します。 DL/I は、修飾された親セグメントと最低レベルの SEGMENT オプションのみを使用してコマンドを正しく実行します。 DL/I は、不在レベルの修飾を使用しません。

使用法

GU コマンドは、データベースから特定のセグメントを検索するとき、または順次処理のためにデータベース内での位置を設定するときに使用します。

検索するセグメント・タイプを示すには、GU コマンドに少なくとも SEGMENT オプションを指定しなければなりません。(IMS DB は、SEGMENT 引数で指定されたセグメントが最初に現れる位置を検索します。)

あるセグメント・タイプの特定のオカレンスを検索する必要がある場合、SEGMENT オプションのあとに WHERE または KEYS オプションを使用して、コマンドをさらに修飾することができます。

GU コマンドをさらに WHERE オプションまたは KEYS オプションで修飾し、セグメント・タイプの特定のオカレンスを指定することがあります。 GU コマンドを完全に修飾すると、データベース・レコード内での現在位置に関係なく、セグメントを検索することができます。

例

*例 **1***

「Robert James は最後にどのような病気で来院しましたか。その日、その病気に対して薬が投与されましたか。彼の患者番号は 05136 です。」

説明*:* この例では、情報を 2 つ要求しています。最初の要求に対応して、最新の ILLNESS セグメントを検索するには、次のように GU コマンドを出します (PATNO1 に 05163 が入っているものとします)。

```
EXEC DLI GU
   SEGMENT(PATIENT) WHERE(PATNO=PATNO1)
   SEGMENT(ILLNESS) INTO(AREA);
```

この患者が最後に診療所を訪れた日付の ILLNESS セグメントを検索したあと、さらに別のコマンドを発行してその日にその患者が治療を受けたかどうかを調べることができます。 患者の最後の来院日が 1988 年 1 月 5 日の場合、次に示すコマンドを出して、その患者が治療を受けたかどうかを調べることができます (PATNO1 には 05163 が入っており、 DATE1 には 19880105 が入っているものとします)。

```
EXEC DLI GU
   SEGMENT(PATIENT) WHERE(PATNO=PATNO1)
   SEGMENT(ILLNESS) WHERE(ILLDATE=DATE1)
   SEGMENT(TREATMNT) INTO(TRTAREA) WHERE(DATE=DATE1);
```

*例 **2***

「Joan Carter は現在どんな病気で治療を受けていますか。彼女の患者番号は 10320 です。」

```
EXEC DLI GU
   SEGMENT(PATIENT) WHERE(PATNO=PATNO1)
   SEGMENT(ILLNESS) INTO(ILLAREA);
```

説明*:* この例では、患者番号が 10320 の患者の ILLNESS セグメントを検索する必要があります。

*例 **3***

```
EXEC DLI GU
   SEGMENT(PATIENT)
   SEGMENT(ILLNESS)
   SEGMENT(TREATMNT) INTO(AREA);
```

説明*:* この例では、最初の TREATMNT セグメントを検索するために、3 つのレベルの修飾を指定しています。

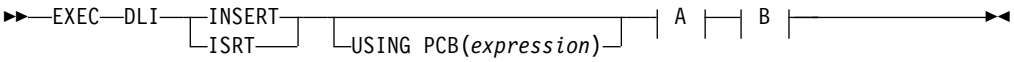
制約事項

検索したいセグメント・タイプを示すには、少なくとも SEGMENT オプションを指定しなければなりません。

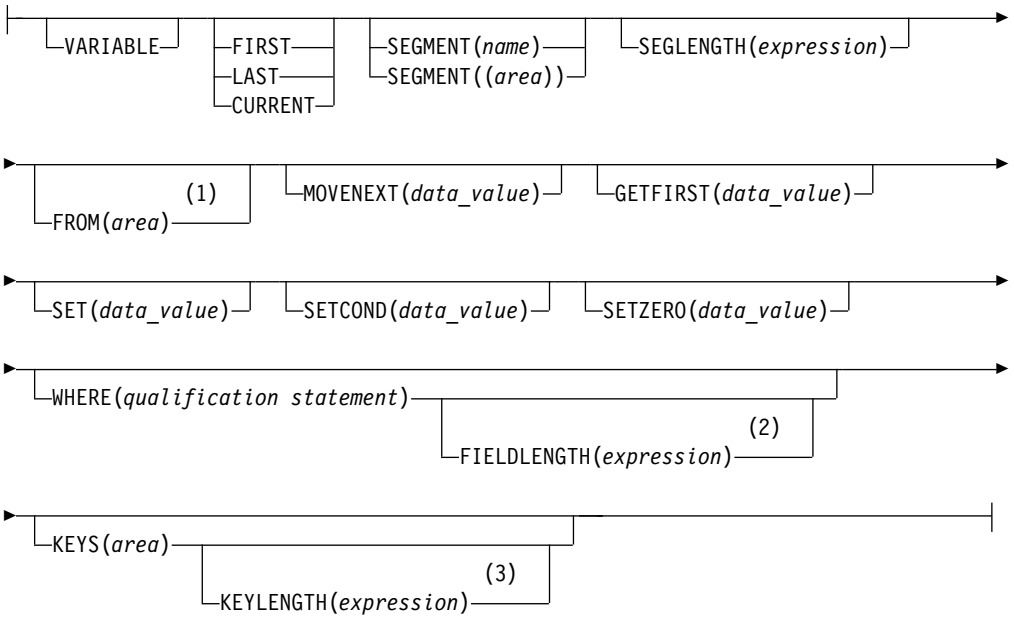
ISRT コマンド

挿入 (ISRT) コマンドは、データベースに 1 つ以上のセグメントを追加するために使用されます。

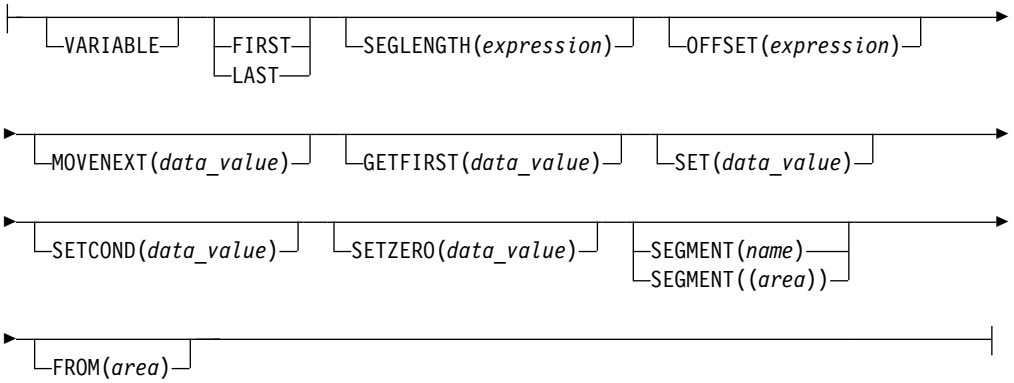
フォーマット



A それぞれの親セグメントについては、次のようになります **(**オプション**) :**



B オブジェクト・セグメントについては、次のようになります **(**必須**) :**



注**:**

1 パス・コマンドでは、親セグメントで FROM を指定してください。

2 複数の修飾ステートメントを使用する場合には、FIELDLENGTH を使用してそれぞれの長さを指定してください。例えば、**FIELDLENGTH(24,8)** と指定してください。

3 1 つのセグメント・レベルで、Keys オプションか Where オプションのいずれかを使用することができますが、両方は使用できません。

オプション

USING PCB(expression)
コマンドのために使用する DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

VARIABLE
セグメントが可変長であることを示します。

FIRST
あるセグメント・タイプの最初に現れるセグメントを検索すること、もしくはセグメントを最初に現れるように挿入することを指定します。セグメント・タイプの最初のオカレンスとしてセグメントを挿入する際には、FIRST を使用してください。

LAST
あるセグメント・タイプの最後に現れるセグメントを検索すること、もしくはセグメントを最後に現れるように挿入することを指定します。セグメント・タイプの最後のオカレンスとしてセグメントを挿入する際には、LAST を使用してください。

CURRENT
コマンドを修飾し、現在位置のレベル以上のレベルをこのセグメントの修飾として使用することを示します。現在位置にもとづいてセグメントを挿入する際には、CURRENT を使用してください。

SEGMENT(name), SEGMENT((area))
検索、挿入、削除、または置き換えを行いたいセグメント・タイプの名前、もしくはプログラムの中でその名前を含んでいる区域を指定することにより、コマンドを修飾します。

データベースに追加したいセグメントごとに、少なくとも 1 つの SEGMENT オプションがなければなりません。 ISRT がパス・コマンドでない限り、挿入されるセグメントは最も低いレベルの SEGMENT オプションによって指定されます。 このレベルには、WHERE または KEYS オプションを使用できません。

セグメントがユニーク・キーをもっていれば、DL/I はこのセグメントをそのキー・シーケンスに従って挿入します。 (セグメントにキーがないか、キーが非ユニーク・キーの場合は、DL/I は DBDGEN の実行時に RULES パラメーターに指定された値に応じてセグメントを挿入します。)

最も低いレベルのセグメントだけに SEGMENT オプションを指定し、親セグメントを SEGMENT、WHERE、または KEYS オプションで修飾しない場合は、データベース内の現在位置が、このセグメントの挿入に適切な位置にあることを確認しなければなりません。 DL/I が使用する SEGMENT オプションは、データベース・レコード内の現在位置によって決まります。

- DL/I の位置が不在レベルに設定されている場合、DL/I はこの位置が設定されているセグメントを使用します。
- DL/I の位置が不在レベルに設定されていない場合、DL/I はそのレベルで最初に現れる位置を使用します。

- DL/I が、もっと高いレベルに設定された位置から前進する場合、DL/I は新しいパスに含まれる不在レベルの最初に現れる位置を使用します。
- ルート・レベルの SEGMENT オプションが省略されており、DL/I の位置がルートに設定されている場合、コマンドを正しく実行しようとするとき DL/I はそのルートから移動しません。

セグメントの挿入位置を設定するときは、必ずより高いレベルを修飾することが賢明な方法です。

ルート・セグメントを挿入する場合は、SEGMENT オプションを指定するだけでかまいません。 DL/I は、入出力域から獲得したキーを使用して、データベースのどこにルート・セグメントを挿入しなければならないかを決定します。挿入するセグメントがルート・セグメントではないが、その直接の親をすでに挿入している場合、 ISRT コマンドでこの挿入セグメントに対して SEGMENT オプションを指定することにより、セグメントが入出力領域で作成されたあとですぐにこのセグメントを挿入することができます。 位置を設定するために親レベルのセグメントをコーディングする必要はありません。

複数の親セグメントを指定するとき、WHERE オプションを指定したものと指定しないものを混在することができます。 親セグメントに対して SEGMENT オプションしか指定しないと、DL/I はこのコマンドの要求を満たすために、各セグメント・タイプが最初に現れる位置を使用します。

SEGLENGTH(expression)
獲得するセグメントの入っている入出力域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (COBOL プログラムでは、INTO オプションまたは FROM オプションを指定しているすべての SEGMENT レベルに必要です。)

要件: SEGLENGTH に指定する値は、この呼び出しで処理する最長セグメントの長さ以上でなければなりません。

FROM(area)
追加、置き換え、または削除されるセグメントを含んでいる区域を指定します。1 つのコマンドで 1 つ以上のセグメントを挿入するには、FROM を使用してください。

MOVENEXT(data_value)
サブセット・ポインターを、現行セグメントの次のセグメントが現れる位置に移動させることを指定します。

GETFIRST(data_value)
サブセット内のセグメントが最初に現れる位置から検索を開始したいことを指定します。

SET(data_value)
無条件でサブセット・ポインターを現行セグメントに設定することを指定します。

SETCOND(data_value)
条件付きでサブセット・ポインターを現行セグメントに設定することを指定します。

SETZERO(data_value)
サブセット・ポインターをゼロに設定することを指定します。

WHERE(qualification statement)
セグメントが現れる位置を指定することで、コマンドを修飾します。その引数は、1 つ以上の修飾ステートメントから成り、それぞれがセグメントのフィールド内の値と、ユーザーが指定した値を比較します。各修飾ステートメントは次のものから構成されています。

- セグメント中のフィールドの名前
- どのように 2 つの値を比較するかを示す関係演算子
- フィールドの値と比較される値を含んでいる、プログラム中のデータ域の名前

セグメントを挿入するときは、WHERE はそのセグメントの親の上に位置を設定します。 これを行うには、もっと高いレベルの SEGMENT オプションを WHERE または KEYS で修飾します。

複数の親セグメントを指定するとき、WHERE オプションを指定したものと指定しないものを混在することができます。 親セグメントに対して SEGMENT オプションしか指定しないと、DL/I はこのコマンドの要求を満たすために、各セグメント・タイプが最初に現れる位置を使用します。

FIELDLENGTH(expression)
WHERE オプションのフィールド値の長さを指定します。

KEYS(area)
セグメントの連結キーでコマンドを修飾します。セグメント・レベルに対して KEYS または WHERE のいずれかを使用することができますが、両方を使用することはできません。

KEY は、親セグメントの修飾に使用することができます。 WHERE を使用する代りに KEYS を指定し、セグメントの連結キーを修飾として使用することができます。 それぞれのコマンドには、最も高いレベルの SEGMENT オプションの直後に、 KEYS オプションを 1 回だけ使用することができます。

「area」は、セグメントの連結キーが入っているプログラム内の区域を指定します。

KEYLENGTH(expression)
KEYS オプションを使用する際の、連結キーの長さを指定します。整数データ型に変換することができる、ホスト言語で書かれた式であればどのような式でも可能です。変数を指定する場合、バイナリー・ハーフワード値で宣言しなければなりません。 IBM COBOL (または VS COBOL II)、PL/I、またはアセンブラー言語の各プログラムの場合、KEYLENGTH はオプションです。IBM COBOL for MVS & VM (または VS COBOL II) コンパイラーでコンパイルしていない COBOL プログラムの場合、KEYS オプションに KEYLENGTH を指定してください。

使用法

既存のデータベースに新しいセグメントを追加するときに、ISRT コマンドを使用します。 ISRT コマンドを出すと、DL/I は FROM オプションで指定された入出力域

からデータを取り出し、データベースにセグメントを追加します。 (データベースの初期ロードでは、ISRT コマンドではなく LOAD コマンドを使用する必要があります。)

ISRT コマンドを使用して、既存のセグメント・タイプの新しいオカレンスを HIDAM、HISAM、HDAM データベースに追加することができます。 HSAM データベースの場合、このデータベースに新しいセグメントを追加できるのは、データベース全体を再処理するか、あるいは新しいセグメントをデータベースの最後に追加するときだけです。

ISRT コマンドを出してデータベースにセグメントを追加するには、その前に、入出力域に挿入されるセグメントがプログラムによって作成されている必要があります。このセグメントがキーをもっている場合、正しいキーを入出力域の正しい位置に入れなければなりません。 フィールド・センシティビティーを使用する場合、フィールドの順序は、セグメントに対するアプリケーションの視点について PSB が定義した順序でなければなりません。

ルート・セグメント・オカレンスを追加する場合、DL/I はユーザーが入出力域に与えたキーを使用し、データベースにこのオカレンスを正しい順序で入れます。 挿入するセグメントがルートではないが、その親を挿入したばかりだった場合には、子セグメントの名前だけで修飾された挿入要求を出すことにより、この子セグメントを挿入することができます。ISRT 要求を出す前に、入出力域に新しいセグメントを作成しておかなければなりません。 新しいルート・セグメント・オカレンスを追加するときも、挿入要求をセグメント名で修飾します。 既存のデータベースに新しいセグメント・オカレンスを追加する場合は、DBD の中にセグメント・タイプ が定義されていなければなりません。 プログラムの入出力域に新しいセグメント・オカレンスを作成したあとは、それらを直接もしくは順次に追加することができます。

挿入するセグメント・タイプに固有キー・フィールドがある場合は、DL/I が新しいセグメント・オカレンスを追加する位置は、そのキー・フィールドの値によって異なります。 セグメントがキー・フィールドをもっていない場合、またはキーが固有でない場合は、FIRST、LAST、または HERE 挿入規則のいずれかを指定することによって、新しいセグメント・オカレンスをどこに追加するかを制御することができます。 この規則は、データベースに関する SEGM ステートメントの RULES パラメーターで指定します。

例

例 ***1***

「Chris Edwards に関するレコードに、1993 年 2 月 1 日に彼が診療所を訪れたことについての情報を追加します。 彼の患者番号は 02345 です。 病気は咽頭痛でした。」

説明:まず、プログラムの入出力域に ILLNESS セグメントを作成する必要があります。ILLNESS セグメントの入った入出力域は、次のようになります。

```
19930201SORETHROAT
```

この新しいセグメント・オカレンスをデータベースに追加するには、次のコマンドを使用します。

```
EXEC DLI ISRT
     SEGMENT(PATIENT) WHERE (PATNO=PATNO1)
     SEGMENT(ILLNESS) FROM(ILLAREA);
```

例 *2*

「Chris Edwards に関するレコードに、治療についての情報、および病気についての情報を追加します。」

説明*:* セグメント入出力域に TREATMNT セグメントを作成します。TREATMNT セグメントには、日付、投薬する医薬品、投薬量、医師名が入っています。

```
19930201MYOCINbbb
0001TRIEBbbbbb
&b
```

次のコマンドは、ILLNESS セグメントと TREATMNT セグメントをデータベースに追加します。

```
EXEC DLI ISRT
     SEGMENT(PATIENT) WHERE (PATNO=PATNO1)
     SEGMENT(ILLNESS) FROM(ILLAREA)
     SEGMENT(TREATMNT) FROM(TRETAREA);
```

例 *3*

```
EXEC DLI ISRT
     SEGMENT(ILLNESS) KEYS(CONKEY)
     SEGMENT(TREATMNT) FROM(TRETAREA);
```

説明*:* このコマンドを使用することは、ILLNESS および PATIENT セグメントのキー・フィールド上に修飾された WHERE オプションを指定することと同じです。

制約事項

ISRT コマンドの制約事項は、次のとおりです。

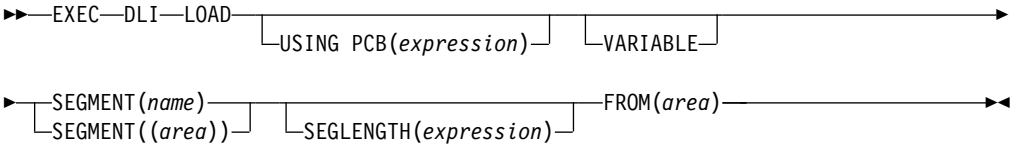
- 入出力域に新しいセグメントを作成する前に、ISRT コマンドを発行することはできません。
- データベースに追加したいセグメントごとに、少なくとも 1 つの SEGMENT オプションがなければなりません。
- セグメントを挿入するときは、そのセグメントの親の上に、あらかじめ位置を設定しておかなければなりません。
- 最も低いレベルのセグメントだけに SEGMENT オプションを指定し、親セグメントを SEGMENT、WHERE、または KEYS オプションで修飾しない場合は、データベース内の現在位置が、このセグメントの挿入に適切な位置にあることを確認しなければなりません。
- セグメントに FROM オプションを指定する場合、そのセグメントを WHERE オプションや KEYS オプションで修飾することはできません。DL/I は入出力域に指定されたキー・フィールド値を修飾として使用します。
- 追加するセグメント・タイプごとに個別の入出力域を使用しなければなりません。

- FROM オプションのある SEGMENT オプションと、FROM オプションのない SEGMENT オプションを混在させることはできません。親セグメントに FROM オプションを使用するときは、従属セグメントごとに FROM オプションを使用しなければなりません。(パスはどのレベルからでも開始できますが、いかなるレベルも省略してはなりません。)
- FIRST オプションの使用は、セグメントがキーをまったくもたないか、キーが非固有で、しかも DBD 内の SEGM ステートメントの RULES オペランドで HERE がそのセグメントに指定されている場合に限られます。
- LAST オプションの使用は、セグメントがまったくキーをもたないか、キーが非固有で、しかもセグメントの INSERT 規則が FIRST か HERE のときだけです。

LOAD コマンド

ロード (LOAD) コマンドは、データベースのロード時にセグメントを順次追加するために使用します。

フォーマット



オプション

USING PCB(expression)
使用したい DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

VARIABLE
セグメントが可変長であることを示します。

SEGMENT(name)
検索、挿入、削除、あるいは置き換えを行いたいセグメント・タイプの名前を指定します。

SEGMENT((area))
プログラムの中でセグメント・タイプの名前を含んでいる区域の参照です。 コマンドでセグメントの名前を指定する代りに、区域を指定することができます。

SEGLENGTH(expression)
獲得するセグメントの入っている入出力域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (COBOL プログラムでは、INTO オプションまたは FROM オプションを指定している SEGMENT レベルには SEGLENGTH が必須です。)

要件: SEGLENGTH に指定する値は、この呼び出しで処理する最長セグメントの長さ以上でなければなりません。

FROM(area)
追加、置き換え、または削除されるセグメントを含んでいる区域を指定します。

使用法

LOAD コマンドは、データベース・ロード・プログラムで使用するコマンドです。詳細は、「*IMS V14* データベース管理」に記載されています。

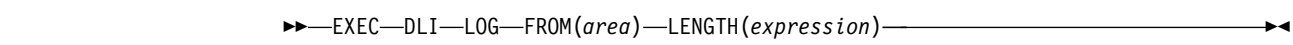
例

```
EXEC DLI LOAD
      SEGMENT(ILLNESS) FROM(ILLAREA);
```

LOG コマンド

ログ (LOG) コマンドは、情報をシステム・ログに書き込むために使用します。

フォーマット



オプション

FROM(area)
追加、置き換え、または削除されるセグメントを含んでいる区域を指定します。

LENGTH(expression)
区域の長さを指定します。

使用法

LOG コマンドを使用して、システム・ログに情報を書き込みます。

例

```
EXEC DLI LOG
      FROM(ILLAREA) LENGTH(18);
```

制約事項

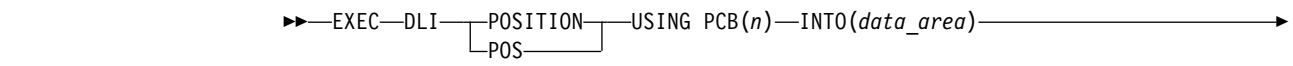
LOG コマンドの制約事項は、次のとおりです。

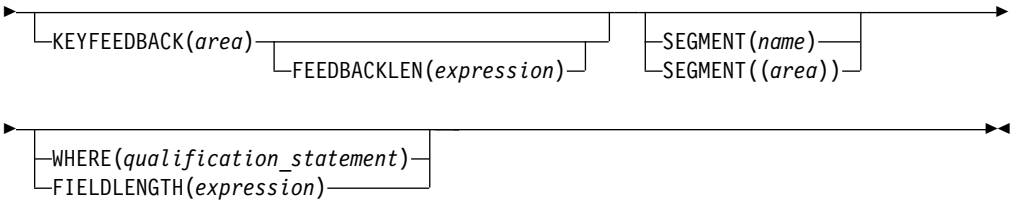
- このコマンドを使用するには、最初にプログラムの入出力 PCB を定義しなければなりません。

POS コマンド

位置 (POS) コマンドは、従属またはセグメントのいずれかの位置を検索します。

フォーマット





オプション

USING PCB(n)
コマンドのために使用する DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

INTO(data_area)
セグメントが読み込まれる区域を指定します。

KEYFEEDBACK(area)
セグメントの連結キーが入る区域を指定します。 区域の長さが足りない場合、キーは切り捨てられます。

FEEDBACKLEN(expression)
検索された連結キーを入れたいキー・フィードバック域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。
(FEEDBACKLEN は COBOL プログラムでは必須ですが、PL/I およびアセンブラー言語の各プログラムでは任意指定です。)

SEGMENT(name)
検索、挿入、削除、あるいは置き換えを行いたいセグメント・タイプの名前を指定することにより、コマンドを修飾します。

SEGMENT((area))
プログラムの中でセグメント・タイプの名前を含んでいる区域の参照です。 コマンドでセグメントの名前を指定する代りに、区域を指定することができます。

WHERE(qualification statement)
セグメントが現れる位置を指定することで、コマンドを修飾します。その引数は、1 つ以上の修飾ステートメントから成り、それぞれがセグメントのフィールド内の値と、ユーザーが指定した値を比較します。

FIELDLENGTH(expression)
WHERE オプションのフィールド値の長さを指定します。

使用法

次のことを行う際には、POS コマンドを使用してください。

- 特定の順次従属セグメント (最後に挿入されたものも含む) の位置を検索する。
- 各 DEDB エリア内で未使用のスペースの量を調べる。

POS コマンドで指定された区域が使用不能である場合、入出力域は変更されず、FH 状況コードが戻されます。

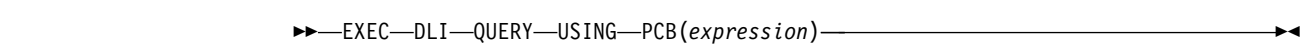
制約事項

POS コマンドは DEDB 専用です。

QUERY コマンド

照会 (QUERY) コマンドは、状況コードと DL/I インターフェース・ブロック (DIB) 内のその他の情報を取得します。 DIB は IMS PCB のサブセットです。

フォーマット



オプション

USING PCB (*expression*) は必須です。 それ以外のオプションは、QUERY コマンドでは指定できません。

使用法

全機能データベースの場合、DIB には NA、NU、TH、またはブランクが入らなければなりません。これらのコードについての説明は、「*IMS V14* メッセージおよびコード 第 *4* 巻*: IMS* コンポーネント・コード」を参照してください。

QUERY コマンドが使用されるのは、PSB のスケジュールと最初のデータベース呼び出しの間です。プログラムがすでに DB PCB を使用して呼び出しを出している場合は、REFRESH コマンドを使用して DIB 内の情報を更新してください。

例

例 ***1***

```
EXEC DLI QUERY USING PCB(expression);
```

説明*:* 上記の例では、QUERY コマンドを指定する方法を示しています。この例では、(n) は PCB を指定します。

例 ***2***

```
EXEC DLI REFRESH DBQUERY;
```

説明*:* プログラムがすでに DB PCB 名を使用して呼び出しを出している場合は、REFRESH コマンドを使用して DIB 内の情報を更新してください。 REFRESH コマンドは DB PCB をすべて更新します。 このコマンドは 1 回だけしか出すことができません。

制約事項

QUERY コマンドの制約事項は、次のとおりです。

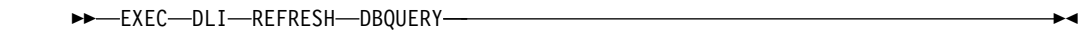
- このコマンドを使用するには、最初にプログラムの入出力 PCB を定義しなければなりません。

- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。

REFRESH コマンド

最新表示 (REFRESH) コマンドは、最後に出されたコマンドについて、DIB から最新の情報を獲得するために使用します。

フォーマット



オプション

DBQUERY は必須です。 それ以外のオプションは、REFRESH コマンドでは指定できません。

使用法

REFRESH コマンドは QUERY コマンドとともに使用します。

QUERY コマンドが発行されるのは PSB のスケジュールと最初のデータベース呼び出しの間です。プログラムがすでに DB PCB を使用して呼び出しを出している場合は、 REFRESH コマンドを使用して DIB 内の情報を更新してください。

REFRESH コマンドは DB PCB をすべて更新します。 このコマンドを出せるのは 1 回だけです。

例

```
EXEC DLI REFRESH DBQUERY;
```

説明

上記の例では、REFRESH コマンドを指定する方法を示しています。

制約事項

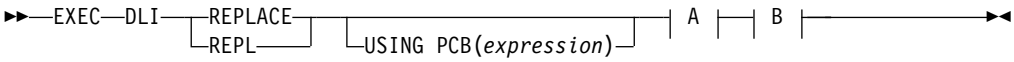
REFRESH コマンドの制約事項は、次のとおりです。

- このコマンドを使用するには、まず最初にプログラムの入出力 PCB 定義してください。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- このコマンドを出せるのは 1 回だけです。

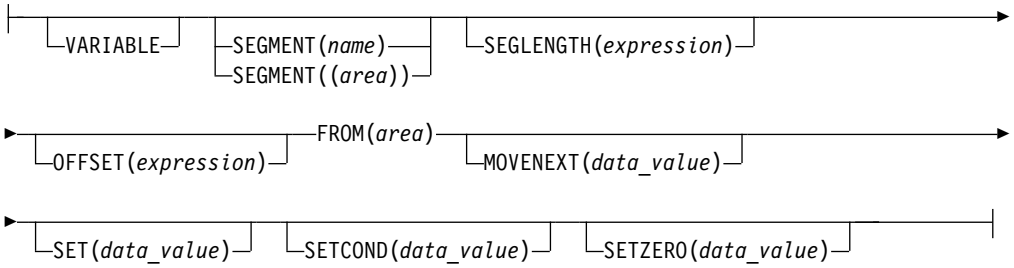
REPL コマンド

置換え (REPL) コマンドは、セグメントの置き換えに使用されますが、通常は、1 つ以上のそのフィールドの値を変更するために使用されます。

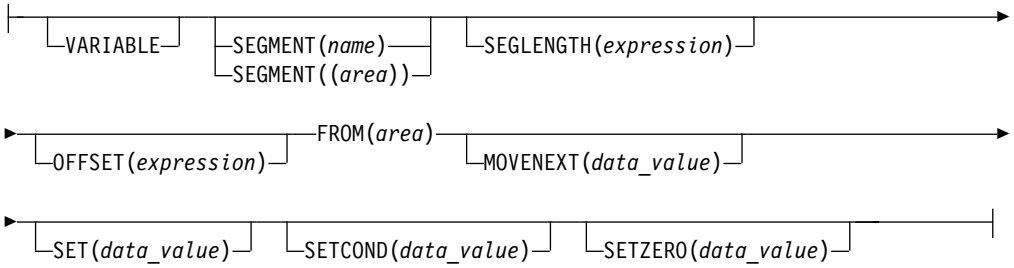
フォーマット



A それぞれの親セグメントについては、次のようになります **(**オプション**) :**



B オブジェクト・セグメントについては、次のようになります **(**必須**) :**



オプション

USING PCB(expression)
コマンドのために使用する DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

VARIABLE
セグメントが可変長であることを示します。

SEGMENT(name)
検索、挿入、削除、あるいは置き換えを行いたいセグメント・タイプの名前を指定することにより、コマンドを修飾します。

SEGMENT((area))
プログラムの中でセグメント・タイプの名前を含んでいる区域の参照です。 コマンドでセグメントの名前を指定する代りに、区域を指定することができます。

SEGLENGTH(expression)
獲得するセグメントの入っている入出力域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフ

ワードを指す参照を指定することもできます。 (COBOL プログラムでは、INTO オプションまたは FROM オプションを指定しているすべての SEGMENT レベルに必要です。)

要件: SEGLENGTH に指定する値は、この呼び出しで処理する最長セグメントの長さ以上でなければなりません。

OFFSET(expression)
目標親に対するオフセットを指定します。 整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定するか、あるいはプログラムの中で数値を含んでいるハーフワードへの参照を指定します。 論理関係にある連結セグメントを処理するときに、OFFSET を使用します。 目標親が可変長セグメントの場合には、このオプションが必須です。

FROM(area)
追加、置き換え、または削除されるセグメントを含んでいる入出力域を指定します。 置き換えたいセグメントごとに、FROM オプションを対応する SEGMENT オプションのあとに入れることによって、より多くのセグメントを置き換えることができます。 1 つ以上の親セグメントに FROM オプションを使用することを、パス・コマンドと呼びます。

FROM のあとの引数は、プログラムの中で定義された入出力域を識別します。置き換えたいセグメント・タイプごとに個別の入出力域を使用しなければなりません。

MOVENEXT(data_value)
サブセット・ポインターを、現行セグメントの次のセグメントが現れる位置に移動させることを指定します。

SET(data_value)
無条件でサブセット・ポインターを現行セグメントに設定することを指定します。

SETCOND(data_value)
条件付きでサブセット・ポインターを現行セグメントに設定することを指定します。

SETZERO(data_value)
サブセット・ポインターをゼロに設定することを指定します。

使用法

REPL コマンドは 1 つ以上の SEGMENT オプションと FROM オプションで修飾してください。この 2 つのオプションを指定することにより、検索されたセグメントのうちどのセグメントを置換するかを示すことができます。

REPL コマンドの前にあった Get コマンドがパス・コマンドであり、かつ必ずしもすべての検索済みセグメントを置き換えたくない場合、あるいは PSB が必ずしもすべての検索済みセグメントに対して置き換えを識別できない場合、SEGMENT オプションを省略することにより、どの検索セグメントを置き換えないかを指示することができます。

置き換えを識別できないセグメントのパス置き換えをプログラムで実行しようとする場合、 REPL コマンド用の入出力域に入っているセグメントのデータは、先行す

る読み取りコマンドで戻されたセグメントと同じでなければなりません。 この状況でデータを変更すると、トランザクションが異常終了するので、置き換えコマンドを実行してもデータは変更されません。

REPL パス・コマンドの規則と ISRT パス・コマンドの規則が異なることに注意してください。 ISRT コマンドでは挿入されるセグメント・レベルをスキップすることはできませんが、 REPL コマンドではこのようにスキップすることができます。

セグメント内の情報を更新するには、 REPL コマンドを使用します。 REPL コマンドはセグメント内のデータを、ユーザーがアプリケーション・プログラムに与えた情報と置き換えます。 まず、セグメントを検索して入出力域に入れなければなりません。 次に、入出力域で情報を更新し、REPL コマンドでセグメントを置き換えます。 プログラムが正常にセグメントを置き換えるようにするには、PCB の SENSEG ステートメントに PROCOPT=A または PROCOPT=R を指定することにより、PCB 内でそのセグメントが置換可能であると定義しておかなければなりません。

読み取りコマンドと REPL コマンドの間では、同じ PCB を使用するコマンドを出すことはできません。また、読み取りコマンド 1 つにつき REPL コマンドは 1 つしか出すことができません。

例

例 ***1***

```
EXEC DLI GU SEGMENT(PATIENT) INTO(PATAREA);
EXEC DLI REPL SEGMENT(PATIENT) FROM(PATAREA);
```

説明*:* この例では、読み取りコマンドと REPL コマンドの間には、同じ PCB を使用するコマンドを出すことができないこと、また、読み取りコマンド 1 つにつき REPL コマンドは 1 つしか出せないことを示しています。 このコマンドを出してセグメント内の情報を再度更新する場合は、まず GU コマンドを出してから REPL コマンドを再度出す必要があります。

例 ***2***

「ID が 08642 の患者から 65.00 ドルの支払いを受けました。この患者の請求レコードと支払いレコードをこの情報で更新し、この患者に対する現時点の請求書を印刷してください。」

説明*:* この処理要求に対応するための手順は、次の 4 つの部分に分かれています。

1. この患者の BILLING セグメントと PAYMENT セグメントを検索する。
2. BILLING セグメントの値から 65.00 ドルを引き、PAYMENT セグメントに 65.00 ドルを加えることによって、これらのセグメントの新しい値を計算する。
3. BILLING および PAYMENT セグメントの値を新しい値で置き換える。
4. この患者に対して、患者の名前、番号、住所、現在の請求金額、およびこの日までの支払い金額を示した請求書を印刷する。

BILLING および PAYMENT セグメントを検索するために、 GU コマンドを出します。請求書を印刷するときに、PATIENT セグメントも必要となるため、 PATIENT セグメントと BILLING セグメントの SEGMENT オプションの後に INTO を指定することができます。

```
EXEC DLI GU
     SEGMENT(PATIENT) INTO(PATAREA) WHERE (PATNO=PATNO1)
     SEGMENT(BILLING) INTO(BILLAREA)
     SEGMENT(PAYMENT) INTO(PAYAREA);
```

現在の請求金額と支払い金額を計算したあとで、請求書を印刷してから、データベース内の請求セグメントと支払いセグメントを置き換えます。 REPL コマンドを出す前に、入出力域内のセグメントを変更しておく必要があります。

PATIENT セグメントは変更されていないので、BILLING および PAYMENT セグメントを置き換えるときに PATIENT セグメントを置き換える必要はありません。 PATIENT セグメントを置換しないことを DL/I に指示する場合は、 REPL コマンドで PATIENT セグメントに SEGMENT オプションを指定しないでください。

```
EXEC DLI REPL
     SEGMENT(BILLING) FROM(BILLAREA)
     SEGMENT(PAYMENT) FROM(PAYAREA);
```

このコマンドは、BILLING および PAYMENT セグメントを置き換えるが、PATIENT セグメントを置き換えないことを、DL/I に指示するものです。

これらの 2 つの例は、パス・コマンド と呼ばれます。1 つのコマンドで複数のセグメントを置き換えるときに、REPL パス・コマンドを使用します。

例 ***3***

「患者番号 10250 の Steve Arons が、この町の新住所に引っ越しました。 新住所は、4638 Brooks Drive, Lakeside, California です。 新住所でデータベースを更新してください。」

説明: Steve Arons の PATIENT セグメントを検索して、そのセグメントの住所部分を置き換える必要があります。 PATIENT セグメントを検索するために、この GU コマンドを使用できます (PATNO1 に 10250 が入っているとします)。

```
EXEC DLI GU
     SEGMENT(PATIENT) INTO(PATAREA) WHERE (PATNO=PATNO1);
```

PATIENT セグメントの最初の 2 つのフィールド (PATNO と NAME) は置き換えないので、入出力域でこれらを変更する必要はありません。 PATNO フィールドと NAME フィールドのあとにある入出力域に新住所を入れます。 その後で、REPL コマンドを出します。

```
EXEC DLI REPL
     SEGMENT(PATIENT) FROM(PATAREA);
```

例 ***4***

```
EXEC DLI GU SEGMENT(PATIENT) INTO(PATAREA)
     WHERE (PATNO=PATNO1)
     SEGMENT(ILLNESS) INTO(ILLAREA)
     SEGMENT(TREATMNT) INTO(TRETAREA);
EXEC DLI REPL SEGMENT(PATIENT) FROM(PATAREA)
     SEGMENT(TREATMNT) FROM(TRETAREA);
```

説明: この例では、患者番号 10401 の PATIENT セグメントと TREATMNT セグメントを置き換えたいが、ILLNESS セグメントは変更したくない場合を想定しています。 これを行うには、このコマンドを出します (PATNO1 に 10401 が入っているものとします)。

制約事項

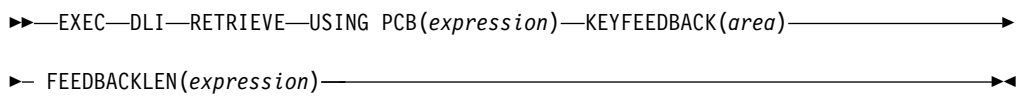
REPL コマンドの制約事項は、次のとおりです。

- 読み取りコマンドと REPL コマンドの間では、同じ PCB を使用する他のコマンドを出すことはできません。
- 1 つの読み取りコマンドに対して REPL コマンドは 1 つだけしか出せません。
- セグメント内の情報を更新する場合、まず最初に GU コマンドを再度出してから REPL コマンドを再度出すようにしなければなりません。
- REPL コマンドは、1 つ以上の SEGMENT オプションと FROM オプションで修飾しなければなりません。
- セグメントに FROM オプションを指定する場合、そのセグメントを WHERE オプションや KEYS オプションで修飾することはできません。DL/I は入出力域に指定されたキー・フィールド値を修飾として使用します。

RETRIEVE コマンド

RETRIEVE コマンドは、バッチ・プログラムと BMP プログラムにおいてデータベース内の現行位置を判別するために使用します。

フォーマット



オプション

USING PCB(expression)
コマンドのために使用する DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

expression は、検索したい連結キーの PCB を指定します。 整数データ型に変換することができる、ホスト言語で書かれた式であればどのような式でも可能です。 数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 この値は、PSB 用に生成された PCB の数より少ない、正の整数値でなければなりません。リスト内の最初の PCB (入出力 PCB) は 1 です。リスト内の最初の DB PCB は 2、2 番目は 3、...となります。

KEYFEEDBACK(area)
セグメントの連結キーが入る区域を指定します。 区域の長さが足りない場合、キーは切り捨てられます。

FEEDBACKLEN(expression)

検索された連結キーを入れたいキー・フィードバック域の長さを指定します。その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 (これは、COBOL プログラムでは必須ですが、PL/I およびアセンブラー言語の各プログラムでは任意指定です。)

expression は、キー・フィードバック入出力域の長さです。 整数データ型に変換することができる、ホスト言語で書かれた式であればどのような式でも可能です。数値を指定するか、またはそのプログラムの中で数値を含んでいるハーフワードを指す参照を指定できます。 IBM COBOL for z/OS & VM (または VS COBOL II)、PL/I、アセンブラー言語の各プログラムの場合、FEEDBACKLEN はオプションです。 IBM COBOL for z/OS & VM (または VS COBOL II) コンパイラーでコンパイルしていない COBOL プログラムの場合、KEYFEEDBACK オプションに FEEDBACKLEN を指定してください。

使用法

シンボリック・チェックポイント・コマンドを出すプログラムは、拡張 RESTART (XRST) コマンドまたは RETRIEVE コマンドも出す必要があります。 RETRIEVE コマンドは、プログラムの開始時に 1 回出されます。 RETRIEVE コマンドを使用して、プログラムを正常に開始させたり、異常終了した場合に再始動させることができます。

RETRIEVE コマンドは、特定のチェックポイント ID または時刻/日付スタンプから使用することができます。RETRIEVE コマンドがデータベースの位置変更を行うため、プログラムはこの位置が正しいかどうかを調べる必要があります。

RETRIEVE コマンドを出したあと、位置が設定される基準となるセグメント・タイプとレベルは、DIB の DIBSEGM フィールドと DIBSEGLV フィールドに戻されます。 DIBKFBL の値は、連結キーの実際の長さに設定されます。 DIBSTAT フィールドには、XRST コマンドではなく GU 位置変更の結果戻された値が入っています。

RESTART コマンドは、連結キーにより修飾された内部 GU を出して DL/I データベースの位置変更を行おうとします。 GU による位置変更によって指定されたデータベース内の位置が、 XRST コマンドで使用されるチェックポイント ID に対して正しい位置であるかどうかは、ユーザーが確認する必要があります。RETRIEVE コマンドを使用すれば、GU 位置変更で使用された連結キーを検索したり、プログラムがアクセスする、すべての PCB における現在位置を判別することができます。

例

```
EXEC DLI RETRIEVE USING PCB(2) KEYFEEDBACK(KEYAREA);
EXEC DLI RETRIEVE USING PCB(5) KEYFEEDBACK(KEYAREA);
```

説明

上記の RETRIEVE コマンドは、最初の DB PCB と 4 番目の DB PCB の連結キーを検索します。 (リストの 1 番目の PCB は入出力 PCB です。このため、最初の DB PCB はリストでは 2 番目になります) 最初の RETRIEVE コマンドを発行した後で、最初の DB PCB 内での位置を判別するには、KEYAREA の連結キーと、DIB

の DIBSEGM フィールドと DIBSEGLV フィールドの戻された値を調べます。 2 番目の RETRIEVE コマンドを出した後で、4 番目の DB PCB 内での位置を確認するには、上記と同じ 2 つのフィールドを調べます。

制約事項

RETRIEVE コマンドの制約事項は、次のとおりです。

- このコマンドは CICS プログラムでは使用できません。
- このコマンドを使用するには、まず最初にプログラムの入出力 PCB 定義してください。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- このコマンドは、システム・ログが直接アクセス・ストレージに記憶され、動的バックアウトが指定されていないかぎりは、使用することはできません。また、プログラムを実行する場合は、JCL のパラメーター・フィールドに BKO=Y と指定しなければなりません。

ROLB コマンド

ロールバック (ROLB) コマンドは、変更内容を動的にバックアウトして制御をプログラムに戻すために使用します。このコマンドは CICS プログラムでは使用できません。

フォーマット

オプション

ROLB コマンドで指定できるオプションはありません。

使用法

バッチ・プログラムまたは BMP プログラムにより、一部の処理が無効であると判別された場合、これらのプログラムがその不正な処理の影響を取り除くことができるようにするコマンドが 2 つあります。それはロールバック・コマンド ROLL と ROLB です。

ROLB コマンドがバッチ・プログラムで使用できるのは、システム・ログが直接アクセス・ストレージに保管され、 BKO 実行パラメーターによって動的バックアウトが指定されている場合です。

ROLB を出すと、IMS DB は、最後のチェックポイント以降、またはチェックポイントを出していないプログラムの場合はプログラムの開始以降に、データベースに対して行われた変更内容をすべてバックアウトします。ROLB コマンドを出すと、IMS DB は変更をバックアウトした後に制御をプログラムに戻し、ROLB コマンドの次のステートメントの処理に移ることができるようにします。

例

```
EXEC DLI ROLB;
```

説明

上記の例では、ROLB コマンドを使用して変更内容を動的にバックアウトし、制御をプログラムに戻す方法を示します。

制約事項

ROLB コマンドの制約事項は、次のとおりです。

- このコマンドは CICS プログラムでは使用できません。
- このコマンドを使用するには、その前にプログラムの I/O PCB を定義する必要があります。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- このコマンドは、システム・ログが直接アクセス・ストレージに記憶されており、動的バックアウトが指定されていない場合には使用できません。

関連資料:

『ROLL コマンド』

ROLL コマンド

ロール (ROLL) コマンドは、変更内容を動的にバックアウトするために使用します。このコマンドは CICS プログラムでは使用できません。

フォーマット

オプション

ROLL コマンドで指定できるオプションはありません。

使用法

バッチ・プログラムにより一部の処理が無効であると判別された場合、これらのプログラムがその不正な処理の影響を取り除くことができるようにするコマンドが 2 つあります。それはロールバック・コマンド ROLL と ROLB です。

バッチ・プログラムでは ROLL を使用することができます。

ROLL を出すと、CICS と DL/I は、最後のチェックポイント以降、またはチェックポイントを出していないプログラムの場合はプログラムの開始以降に、データベースに対して行われた変更内容をすべてバックアウトします。ROLL コマンドを出すと、DL/I は更新内容をバックアウトしてからプログラムを終了します。

例

```
EXEC DLI ROLL;
```

説明

上記の例では、ROLL コマンドを使用して変更内容を動的にバックアウトする方法を示しています。

ROLL コマンドを使用すると、IMS はユーザー異常終了コード U0778 でプログラムを終了します。このタイプの異常終了は、記憶ダンプなしでプログラムを終了させます。

制約事項

ROLL コマンドの制約事項は、次のとおりです。

- このコマンドは CICS プログラムでは使用できません。
- このコマンドを使用するには、その前にプログラムの I/O PCB を定義する必要があります。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- このコマンドは、システム・ログが直接アクセス・ストレージに記憶され、動的バックアウトが指定されている場合には使用することはできません。 また、プログラムを実行する場合は、JCL のパラメーター・フィールドに BKO=Y と指定しなければなりません。

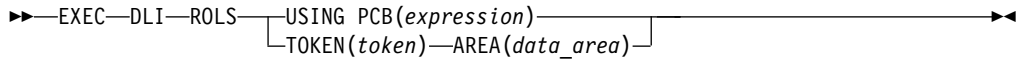
関連資料:

231 ページの『ROLB コマンド』

ROLS コマンド

SETS または SETU へのロールバック (ROLS) コマンドは、これより前に SETS コマンドで設定された処理点までバックアウトするために使用します。

フォーマット



オプション

USING PCB(expression)
使用したい DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

TOKEN(token)
現在処理中のポイントと関連付けられている 4 バイトのトークンです。
TOKEN と AREA の両方を指定すると、 ROLS コマンドは、ユーザーが指定した SETS または SETU までバックアウトします。

AREA(data_area)
ROLS コマンドを出したときにプログラムに復元される区域の名前です。 data-area フィールドの最初の 2 バイトには、その 2 バイトの長さも含めた data-area の長さが入ります。次の 2 バイトは、X'0000' に設定しなければなりません。 TOKEN と AREA の両方を指定すると、 ROLS コマンドは、ユーザーが指定した SETS までバックアウトします。

ROLS 呼び出しには、 TOKEN と AREA を指定する形式 (IOPCB のみ) と、TOKEN と AREA を指定しない形式 (IOPCB または DBPCB) の 2 つの形式があります。

使用法

SETS コマンドと ROLS コマンドを使用すると、DL/I 全機能データベースの状態を保存するポイントを複数定義し、これらのポイントへ後で戻ることができます。(例えば、これらのコマンドを使用すれば、参照された DL/I データベースのすべてが使用可能とならないのに PSB スケジューリングが終了したときに起こる状況をプログラムで処理することができます。)

SETS および ROLS コマンドの使用は、DL/I 全機能データベースだけに適用されます。つまり、作業論理単位 (LUW) が全機能データベース以外のタイプのリカバリー可能リソース (VSAM ファイルなど) を更新する場合、SETS 要求と ROLS 要求の影響は非 DL/I リソースには現れません。バックアウト・ポイントは CICS コミット・ポイントではありません。それらは、DBCTL リソースだけに適用される中間バックアウト・ポイントです。関連リソースのすべてに整合性をもたせるかどうかは、ユーザーが判断します。

ROLS コマンドは、特定の SETS または SETU 要求を出す前、あるいは最後のコミット・ポイントよりも前のすべての全機能データベースの状態にバックアウトするときに、使用できます。

例

例 ***1***

```
EXEC DLI ROLS TOKEN(token1) AREA(data_area)
```

説明: 上記の例 (IOPCB のみ) では、前の SETS 呼び出しで指定されている対応する TOKEN までのバックアウトが実行され、制御がアプリケーションに戻ります。

例 ***2***

```
EXEC DLI ROLS USING PCB(PCB5)
```

説明: この例では、IOPCB または DBPCB に対して前の同期点までのバックアウトが実行され、アプリケーションが状況コード U3033 で疑似的に異常終了します。制御はアプリケーションには戻りません。

上記の例では PCB5 はデータ使用不能状況コードを受け取った DB PCB の番号です。 このコマンドを出すと、プログラムが ACCEPT STATUSGROUPA コマンドを出さなかった場合と同じ動作になります (「IMS バージョン 14アプリケーション・プログラミング」のトピック『データ可用性の強化』を参照)。

例 *3*

```
EXEC DLI ROLS
```

説明*:* この例では、IOPCB または DBPCB で、PCB への前の参照によって使用不能な状況コードが戻される場合、前の同期点までのバックアウトが実行され、アプリケーションが U3033 で疑似的に異常終了します。 制御はアプリケーションには戻りません。

制約事項

ROLS コマンドの制約事項は、次のとおりです。

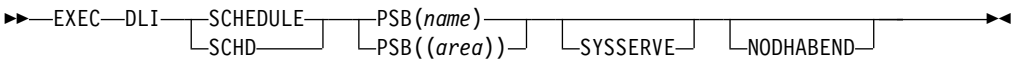
- このコマンドを使用するには、最初にプログラムの入出力 PCB を定義しなければなりません。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- このコマンドは、システム・ログが直接アクセス・ストレージに記憶され、動的バックアウトが指定されている場合には使用することはできません。 また、プログラムを実行する場合は、JCL のパラメーター・フィールドに BKO=Y と指定しなければなりません。

SCHD コマンド

スケジュール (SCHD) コマンドは、CICS オンライン・プログラムで PSB をスケジュールするために使用されます。

入出力 PCB の詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『PCB および PSB』を参照してください。

フォーマット



オプション

PSB(name)
SCHD コマンドでスケジュールするアプリケーション・プログラムが使用できる PSB の名前を指定します。

PSB((area))
SCHD コマンドでスケジュールするプログラムが使用できる PSB の名前が入っている、プログラム内の 8 バイトのデータ域を指定します。

SYSSERVE
アプリケーション・プログラムが入出力 PCB を処理でき、さらにこのプログラムが作業論理単位 (LUW) でシステム・サービス要求を出すことを指定します。

NODHABEND
DHxx 異常終了によって CICS トランザクションが失敗しないことを指定します。

EXEC DLI でスケジュールが失敗する場合は、状況コードが DIB に戻されたために、CICS トランザクションが DHxx の異常終了によって失敗している可能性があります。このオプションを指定することにより、このような状況が発生しなくなります。 SCHD コマンドが失敗すると、制御と DIB 内の状況コードがアプリケーション・プログラムに戻され、このプログラムにより適切な処置がとられます。

使用法

CICS プログラムから DL/I データベースにアクセスするには、その前に、プログラムが PSB をスケジュールすることによりデータベースにアクセスすることを、DL/I に通知しておかなければなりません。SCHD コマンドを出してこのようなスケジュールを行います。 これ以降 PSB を使うことがない場合、あるいはあとで PSB (1 つ以上) をスケジュールする場合は、 TERM コマンドを使用して既存の PSB を終了します (入出力 PCB と PSB の詳細については、「*IMS V14* アプリケーション・プログラミング」のトピック『PCB および PSB』を参照してください)。

SCHD コマンドは、以下のコード例に示すように、次の 2 つの方法で指定することができます。

例

```
EXEC DLI SCHD PSB(psbname)SYSSERVE;
EXEC DLI SCHD PSB((AREA));
```

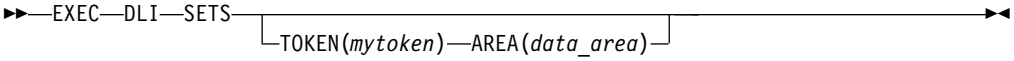
説明

上記の例では、CICS プログラム内で PSB をスケジュールする 2 つの方法を示しています。

SETS コマンド

バックアウト・ポイント設定 (SETS) コマンドは、アプリケーション内で、ある機能を実行する一連の DL/I 要求を開始する前に DL/I データベースの状態を保存しておくポイントを定義するために使用します。アプリケーションは、この機能を完了できない場合にあとから ROLS コマンドを出すことができます。

フォーマット



オプション

TOKEN(mytoken)
現在処理中のポイントと関連付けられている 4 バイトのトークンです。

AREA(data_area)
SETS コマンドが出されたときにプログラムに復元される区域の名前です。data-area フィールドの最初の 2 バイトには、その 2 バイトの長さも含めた data-area の長さが入ります。次の 2 バイトは、X'0000' に設定しなければなりません。

使用法

SETS コマンドを使用すると、DL/I データベースの状態を保存するポイントを複数定義することができます。これらのポイントへあとで戻ることができます。 例えば、SETS コマンドを使用すれば、PSB のスケジュールの完了時、参照する DL/I データベースが使用できない状態で PSB のスケジュールが完了した場合に発生する可能性のある状況を、プログラムが処理できるようになります。

SETS コマンドは、DL/I 全機能データベースだけに適用されます。 作業論理単位 (LUW) が全機能データベース以外のタイプのリカバリー可能リソース (VSAM ファイルなど) を更新する場合、 SETS コマンドの影響は非 DL/I リソースには現れません。 バックアウト・ポイントは CICS コミット・ポイントではありません。それらは、DBCTL リソースだけに適用される中間バックアウト・ポイントです。関連リソースのすべてに整合性をもたせるかどうかは、ユーザーが判断します。

例

```
EXEC DLI SETS TOKEN(mytoken) AREA(data_area)
```

説明

上記の例は、SETS コマンドを指定する方法を示しています。

制約事項

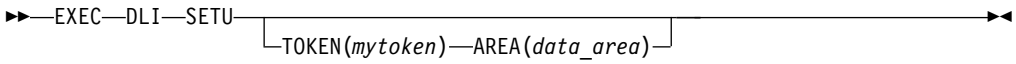
SETS コマンドの制約事項は、次のとおりです。

- このコマンドを使用するには、最初にプログラムの入出力 PCB を定義しなければなりません。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- バッチでは、このコマンドを使用できるのは、システム・ログが直接アクセス・ストレージに保管されており、動的バックアウトが指定されている場合だけです。 また、プログラムを実行する場合は、JCL のパラメーター・フィールドに BKO=Y と指定しなければなりません。
- このコマンドがリジェクトされるのは、PSB に DEDB または MSDB PCB が入っている場合、あるいは DB2 データベースに対して呼び出しが行われた場合です。
- サポートされていない PCB が PSB に入っている場合、あるいはプログラムが外部サブシステムを使用する場合に、このコマンドを使用することはできますが、機能しません。

SETU コマンド

バックアウト・ポイント無条件設定 (SETU) コマンドは、サポートされていない PCB が PSB に入っている場合、あるいはプログラムが外部サブシステムを使用する場合に拒否されることがない点を除くと、SETS コマンドと同様に機能します。

フォーマット



オプション

TOKEN(mytoken)
現在処理中のポイントと関連付けられている 4 バイトのトークンです。

AREA(data_area)
SETU コマンドが出されたときにプログラムに復元される区域の名前です。 data-area フィールドの最初の 2 バイトには、その 2 バイトの長さも含めた data-area の長さが入ります。次の 2 バイトは、X'0000' に設定しなければなりません。

使用法

SETU コマンドを使用すると、DL/I データベースの状態を保存するポイントを複数定義することができます。これらのポイントへあとで戻ることができます。 例えば、SETU コマンドを使用すれば、参照される DL/I データベースが使用できない状態で PSB のスケジュールが完了した場合に発生する可能性のある状況を、プログラムが処理できるようになります。

SETU コマンドは、DL/I 全機能データベースだけに適用されます。作業論理単位 (LUW) が全機能データベース以外のタイプのリカバリー可能リソース (VSAM ファイルなど) を更新している場合、SETU コマンドは、非 DL/I リソースに影響を与えません。バックアウト・ポイントは CICS コミット・ポイントではありません。それらは、DBCTL リソースだけに適用される中間バックアウト・ポイントです。関連リソースのすべてに整合性をもたせるかどうかは、ユーザーが判断します。

例

```
EXEC DLI SETU TOKEN(mytoken) AREA(data_area)
```

説明

上記の例は、SETU コマンドを指定する方法を示しています。

制約事項

SETU コマンドの制約事項は、次のとおりです。

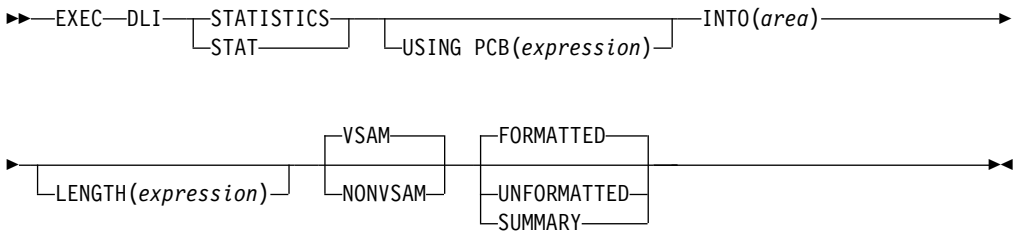
- このコマンドは CICS プログラムでは使用できません。
- このコマンドを使用するには、最初にプログラムの入出力 PCB を定義しなければなりません。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- このコマンドは、システム・ログが直接アクセス・ストレージに記憶され、動的バックアウトが指定されている場合には使用することはできません。 また、プログラムを実行する場合は、JCL のパラメーター・フィールドに BKO=Y と指定しなければなりません。

STAT コマンド

統計 (STAT) コマンドは、プログラムのデバッグに使用できる IMS データベース統計を得るために使用されます。

このトピックにはプロダクト・センシティブ・プログラミング・インターフェース情報が含まれています。

フォーマット



オプション

USING PCB(expression)
使用したい DB PCB を指定します。 その引数には、整数データ型に変換されるものであれば、どのような式でも指定できます。数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。

INTO(area)
データが読み込まれる区域を指定します。

LENGTH(expression)
区域の長さを指定します。

VSAM/NONVSAM
データベース・タイプを指定します。

FORMATTED/UNFORMATTED/SUMMARY
出力のタイプを指定します。

使用法

STAT コマンドの詳細については、「*IMS V14* アプリケーション・プログラミング」で説明します。

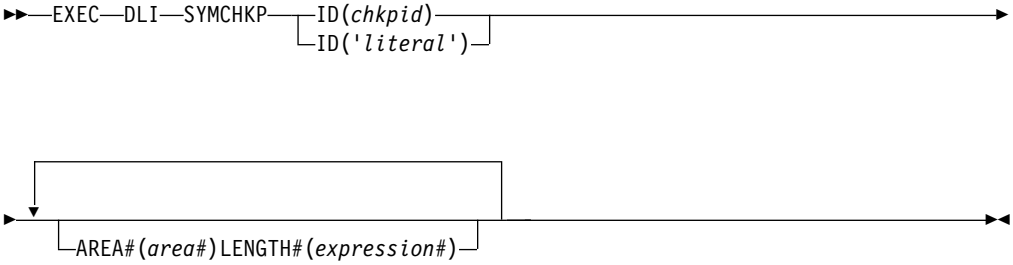
例

STAT コマンドの例については、「*IMS V14* アプリケーション・プログラミング」を参照してください。

SYMCHKP コマンド

シンボリック・チェックポイント (SYMCHKP) コマンドは、シンボリック・チェックポイントを出して論理作業単位を終了するために使用されます。

フォーマット



オプション

ID(chkpid)
チェックポイント ID が入っている、プログラム内の 8 バイトの区域の名前です。 PL/I を使用している場合、このパラメーターを、大構造、配列、または文字ストリングを指すポインターとして指定してください。

ID('literal')
引用符で囲まれた 8 バイトのチェックポイント ID です。

AREA#(area#)
IMS にチェックポイントを実行させたいプログラム内の区域を指定します。チェックポイントを実行する区域の指定は必須ではありませんが、7 つより多い区域を指定することはできません。複数の区域を指定する場合は、介在する区域をすべて含まなければなりません。 例えば、AREA3 を指定する場合、AREA1 と AREA2 も指定しなければなりません。 SYMCHKP コマンドを使用して指定する区域は、XRST コマンドで指定された区域と同じでなければなりません。

LENGTH#(expression#)
整数データ型に変換することができる、ホスト言語で書かれた式であればどのような式でも可能です。数値を指定するか、または数値を含んでいるハーフワードへの参照を指定できます。IBM COBOL for z/OS & VM (または VS COBOL II)、PL/I、アセンブラー言語の各プログラムの場合、LENGTH1 から LENGTH7 まではオプションです。IBM COBOL for z/OS & VM (または VS COBOL II) コンパイラーでコンパイルされない COBOL プログラムでは、指定するそれぞれの AREAx (x は 1 から 7 まで) ごとに LENGTHx (x は 1 から 7 まで) を指定する必要があります。

使用法

チェックポイントを指定する場合に使用できるコマンドは、基本チェックポイント・コマンドの CHKP、およびシンボリック・チェックポイント・コマンドの SYMCHKP です。

バッチ・プログラムでは、記号コマンドか基本コマンドのいずれかを使用することができます。

これらのチェックポイント・コマンドを使用すると、ユーザーが、プログラムの変更内容をデータベースにコミットし、プログラムが異常終了した場合はこのプログ

ラムが再始動できる位置を設定できるようになります。 IMS チェックポイントを取るため、どの DD ステートメントにも CHKPT=EOV パラメーターは指定しないでください。

プログラムでチェックポイントを出す時期およびその理由については、「*IMS V14* アプリケーション・プログラミング」を参照してください。どちらのコマンドの場合も、コマンドが出された時点でデータベースの位置がなくなります。 GU コマンドまたはその他の位置設定方法により、位置を再設定しなければなりません。

データベースに対してプログラムが行った変更をコミットし、プログラムが再始動できる位置を設定する以外に、シンボリック・チェックポイント・コマンドは次のことを行います。

- プログラムが異常終了した場合、拡張再始動 (XRST) コマンドを使用してプログラムを再始動します。
- プログラムの再始動時に復元される、プログラム内のデータ域を 7 つまで保管できます。 これには、変数、カウンター、状況情報を保管することができます。

例

```
EXEC DLI SYMCHKP
     ID(chkpid)
     AREA1(area1) LENGTH1(expression1)
     ...
     AREA7(area7) LENGTH7(expression7)
```

説明

上記の例では、SYMPCHKP コマンドを使用して、シンボリック・チェックポイントを出す方法と論理作業単位を終了する方法を示しています。

制約事項

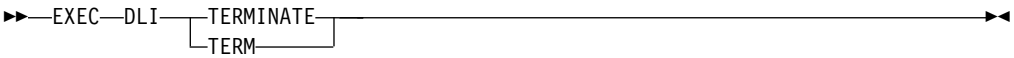
SYMCHKP コマンドの制約事項は、次のとおりです。

- このコマンドを出す場合は、XRST コマンドも出さなければなりません。
- このコマンドは CICS プログラムでは使用できません。
- SYMCHKP コマンドを使用するには、まず最初にプログラムの入出力 PCB を定義しなければなりません。
- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- SYMCHKP コマンドを使用して指定する区域とその指定順序は、 XRST コマンドで指定される区域と同じでなければなりません。
- 複数の区域を指定する場合は、介在する区域をすべて指定しなければなりません。例えば、AREA3 を指定する場合、AREA1 と AREA2 も指定しなければなりません。
- IBM COBOL for z/OS & VM (または VS COBOL II) コンパイラーでコンパイルされていない COBOL プログラムに expression1 を指定する場合は、指定する各 AREAx (x は 1 から 7 まで) に対して LENGTHx (x は 1 から 7 まで) を指定する必要があります。

TERM コマンド

終了 (TERM) コマンドは、CICS オンライン・プログラムで PSB を終了するために使用します。

フォーマット



オプション

TERM コマンドで指定できるオプションはありません。

使用法

すでにスケジュール済みの PSB 以外の PSB を使用する場合は、TERM コマンドを使用して PSB を解放します。

TERM コマンドを出すと、データベースのすべての変更がコミットされるため、バックアウトすることはできません。 CICS に戻ると PSB が終了し、変更内容がコミットされるため、CICS に戻る前に別の PSB のスケジュールを行う場合またはデータベースの変更内容をコミットする場合以外は、TERM コマンドを使用する必要はありません。

TERM コマンドで指定できるオプションはありません。終了した PSB をプログラムがあとで使用する場合、再度 SCHD コマンドを出して PSB をスケジュール変更する必要があります。

ほとんどのアプリケーションでは、TERM コマンドを使用する必要はありません。

例

```
EXEC DLI TERM
```

説明

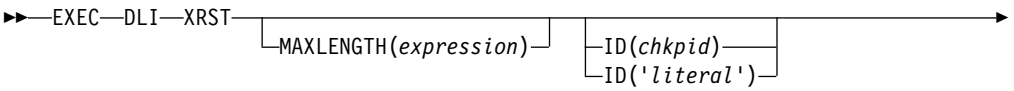
上記の例では、TERM コマンドを使用して PSB を終了する方法を示しています。

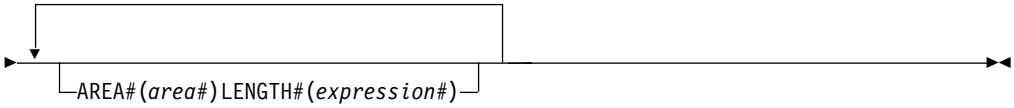
XRST コマンド

拡張再始動 (XRST) コマンドは、拡張再始動を出し、プログラムを正常に始動させるため、もしくはチェックポイント ID または時刻/日付スタンプから拡張再始動させるために使用されます。

プログラムでシンボリック・チェックポイント・コマンドを使用する場合は、 XRST コマンドを使用してください。

フォーマット





オプション

MAXLENGTH(expression)
プログラムが再始動される区域の長さを指定します。このパラメーターは、PSB の中で、あるいはプログラムでパス・コマンドを使用する場合は、パスのすべてのセグメントの中で、最長のセグメントです。 整数データ型に変換することができる、ホスト言語で書かれた式であればどのような式でも可能です。 数値を指定することもできるし、あるいはプログラムの中で数値を含んでいるハーフワードを指す参照を指定することもできます。 MAXLENGTH は必須ではなく、デフォルトの値は 512 バイトです。

ID(chkpid) ID('literal')
このパラメーターは、プログラム内の 30 バイト域の名前か、引用符で囲まれた 30 バイトのチェックポイント ID のどちらかです。このパラメーターはオプションです。このパラメーターを使用する代りに、JCL のパラメーター・フィールドに、チェックポイント ID か時刻/日付スタンプを指定できます。両方とも指定すると、IMS は、EXEC ステートメントのパラメーター・フィールドの値を使用します。プログラムを正常に始動させる場合は、チェックポイント ID をまったく指定しないか、*chkpid* で指し示されたフィールドにブランクが入っていることを確認してください。

プログラムが再始動され、EXEC ステートメントの PARM フィールドの CKPTID= 値が使用されなかった場合、入出力域に使われる、チェックポイント ID を超える右端のバイトは、すべてブランクに設定する必要があります。

XRST コマンドは、IIIIDDDHHMMSST 形式の時刻/日付スタンプを指定した後に出すか、あるいはチェックポイント ID を指定することによりプログラム内の特定のチェックポイントから出すことができます。 IIIIDDD は領域 ID と日付です。HHMMSST は時、分、秒、および 1/10 秒で表される実際の時間です。チェックポイント ID および時刻/日付スタンプは、システム・メッセージ DFS0540I で与えられます。

PL/I を使用している場合、*chkpid* を、大構造、配列、または文字ストリングを指すポインターとして指定してください。

AREA#(area#)
area# は、復元させたいプログラムの最初の区域を指定します。区域は 7 つまで指定できます。 区域はまったく指定しなくてもかまいませんが、複数の区域を指定する場合は、介在する区域をすべて指定しなければなりません。例えば、AREA3 を指定する場合、AREA1 と AREA2 も指定しなければなりません。XRST コマンドに指定する区域とその指定順序は、SYMCHKP コマンドで指定する区域と同じでなければなりません。 プログラムを再始動させると、SYMCHKP コマンドで指定した区域だけが復元されます。

LENGTH#(expression#)
プログラムが再始動される区域の長さを指定します。その引数には、整数データ型に変換することができる、ホスト言語で書かれた式であれば、どのような式でも指定できます。数値を指定することもできるし、あるいは数値を含んでいるハ

ーフワードを指す参照を指定することもできます。IBM COBOL for z/OS & VM (または VS COBOL II)、PL/I、アセンブラー言語の各プログラムの場合、LENGTH1 から LENGTH7 まではオプションです。 IBM COBOL for z/OS & VM (または VS COBOL II) コンパイラーでコンパイルされない COBOL プログラムでは、指定するそれぞれの AREAx (x は 1 から 7 まで) ごとに LENGTHx (x は 1 から 7 まで) を指定する必要があります。各修飾ステートメントは次のものから構成されています。

- セグメント中のフィールドの名前
- どのように 2 つの値を比較するかを示す関係演算子
- フィールドの値と比較される値を含んでいる、プログラム中のデータ域の名前

使用法

プログラムがシンボリック・チェックポイント・コマンドを出す場合には、拡張再始動 (XRST) コマンドも出さなければなりません。 XRST は、プログラムの開始時に 1 回出されます。 XRST コマンドを使用して、プログラムを正常に開始させたり、プログラムが異常終了した場合にこのプログラムを拡張再始動することができます。

特定のチェックポイント ID、または時刻/日付スタンプからプログラムを拡張再始動することができます。 XRST がデータベースの位置変更を行おうとするため、位置が正しいかどうかもプログラムで検査する必要があります。

XRST コマンドを出したあとで、DIB の中の DIBSEGM フィールドをテストしてください。 正常始動のあとは、DIBSEGM フィールドにブランクが入っていなければなりません。 拡張再始動が完了した時、 DIBSEGM フィールドにはチェックポイント ID が入っていなければなりません。 通常 XRST は、8 バイトのシンボリック・チェックポイント ID と、それに続けて 4 個のブランクを返します。 8 バイトの ID がすべてブランクの場合には、XRST は 14 バイトのタイム・スタンプ ID を戻します。 XRST コマンドが成功したことを示す状況コードは、ブランクの状況コードだけです。 XRST コマンドの処理時に DL/I がエラーを検出した場合、プログラムは異常終了します。

例

```
EXEC DLI XRST MAXLENGTH(expression)
   ID(chkpid)
   AREA1(area1) LENGTH1(expression1)
   ...
   AREA7(area7) LENGTH7(expression7)
```

説明

上記の例は、XRST コマンドを指定する方法を示しています。

制約事項

XRST コマンドの制約事項は、次のとおりです。

- このコマンドは CICS プログラムでは使用できません。
- このコマンドを使用するには、最初にプログラムの入出力 PCB を定義しなければなりません。

- 非ユニーク・キーをもつセグメントまたはキーなしセグメントでは、位置の再設定はできません。
- このコマンドは、システム・ログが直接アクセス・ストレージに記憶され、動的バックアウトが指定されていないかぎりは、使用することはできません。また、プログラムを実行する場合は、JCL のパラメーター・フィールドに BKO=Y と指定しなければなりません。

コマンド・コードの参照情報

以下のコマンド・コードの参照情報を使用してください。

制約事項**:** MSDB 呼び出しでコマンド・コードを使用することはできません。

制約事項**:** 以下の制約事項は、高速機能副次索引コマンド・コードおよび複数の SSA サポートに適用されます。

- C コマンド・コードは、最初の SSA 以外の SSA では指定できません。指定した場合、状況コード AJ で拒否されます。
- ISRT 呼び出しの V コマンド・コードは無視されます。
- A、G、およびサブセット・ポインター関連のコマンド・コード (M、R、S、W、および Z) はサポートされません。これらのコマンド・コードは、状況コード AJ で拒否されます。

制約事項**:** 以下の制約事項は、ターゲット・セグメントがルート・セグメントではない場合に、ターゲット・セグメントの物理親セグメントに対するすべての DL/I 呼び出しに適用されます。

- P、Q、U、および V コマンド・コードは無視されます。
- SSA に修飾ステートメントが含まれる場合、フィールド名は親セグメントのシーケンス・フィールド名でなければなりません。シーケンス・フィールド名以外のフィールド名が指定された場合、状況コード AK で拒否されます。

表 *52.* コマンド・コードの要約

コマンド・コード	説明
A	位置決めをクリアし、データベースの始まりから呼び出しを開始する。
C	セグメントの連結キーを使用して、セグメントを識別する。
D	セグメントごとに別々の (経路) 呼び出しを使用する代わりに、1 つの呼び出しだけを使用して、階層経路内のセグメントのシーケンスを検索または挿入する。
F	特定セグメント・オカレンスを検索するとき、親のもとにあるセグメントの最初のオカレンスまでバックアップする。ルート・セグメントに対しては無視される。
G	ランダム化や HALDB 区画選択出口ルーチンの呼び出しを回避し、データベースを順次に検索する。
L	親のもとにあるセグメントの最後のオカレンスをリトリーブする。
M	サブセット・ポインターを、現在位置以降の次のセグメント・オカレンスに移動する。 (DEDB でのみ使用。)

表 52. コマンド・コードの要約 (続き)

コマンド・コード	説明
N	Get Hold 呼び出しのあとにセグメントを置き換えるときに、置き換えたくないセグメントを指定する。通常、セグメントの経路を置き換える時に使用する。
O	フィールド名、またはセグメントの位置と長さを、結合フィールド位置の SSA 修飾に入れることができる。
P	通常のレベル (呼び出しの最低 SSA レベル) よりも高いレベルで親子関係を設定する。
Q	ユーザーがセグメントの処理と更新を終えるまで他のプログラムがそのセグメントを更新できないように、セグメントを予約する。
R	サブセット内の最初のセグメントのオカレンスを検索する。(DEDB でのみ使用。)
S	サブセット・ポインターを、無条件で現行位置に設定する。 (DEDB でのみ使用。)
U	セグメントの検索対象を、位置が設定されているセグメント・オカレンスの従属セグメントだけに限定する。
V	セグメントの修飾として、現在位置およびそれ以上の位置に階層レベルを使用する。
W	サブセット・ポインターがまだ設定されていない場合に、現在位置にサブセット・ポインターを設定する。 (DEDB でのみ使用。)
Z	サブセット・ポインターをゼロに設定し、再使用できるようにする。 (DEDB でのみ使用。)
-	ヌル。 コマンド・コードを指定せずにコマンド・コード形式で SSA を使用する。 実行中に、ユーザーの希望するコマンド・コードで置き換えることができる。

以下の表は、コマンド・コードおよびそれらを適用できる呼び出しを示します。

表 53. コマンド・コードおよび関連呼び出し

コマンド・コード	GU	GHU	GN	GHN	GNP GHNP	REPL	ISRT	DLET
A			X					
C		X	X		X		X	
D		X	X		X		X	
F		X	X		X		X	
G			X					
L		X	X		X		X	
M		X	X		X	X	X	
N						X		
O		X	X		X		X	
P		X	X		X		X	
Q		X	X		X		X	
R		X	X		X		X	

表 53. コマンド・コードおよび関連呼び出し (続き)

コマンド・コード	GU GHU	GN GHN	GNP GHNP	REPL	ISRT	DLET
S	X	X	X	X	X	
U	X	X	X		X	
V	X	X	X		X	
W	X	X	X	X	X	
Z	X	X	X	X	X	X
-	X	X	X	X	X	X

A コマンド・コード

A コマンド・コードを使用してデータベース内の位置をクリアすることができます。その結果、呼び出しはデータベースの始まりから開始されます。

アプリケーションがデータベースの全探索を行って、あるパスを下降しても要求データが見つからなかった場合、そのアプリケーションはコマンド・コード A を使用して修飾 GN または GHN 呼び出しを発行して、データベースの始まりに位置をリセットし、異なるパスでデータを検索できます。

C コマンド・コード

C コマンド・コードを使用することにより、修飾ステートメントを提供する代りに、そのステートメントを識別する手段としてセグメントを連結したキーを提供することを、IMS に知らせることができます。C コマンド・コードまたは修飾ステートメントのどちらを使用することもできますが、両方を使用することはできません。

C コマンド・コードは、すべての Get 呼び出しおよび ISRT 呼び出しで使用できます。連結されたキーをコーディングするときには、*C の後に括弧で囲んだキーを続け、修飾ステートメントを指定する位置と同じ場所に *C 以下の部分を指定してください。

例えば、この要求を満たす場合を考えてみます。

> Joan Carter が 2009 年 3 月 3 日に来院したかどうか。 彼女の患者番号は 07755 である。

PATIENT セグメントのキー・フィールドが患者番号であって、 ILLNESS セグメントのキー・フィールドが日付フィールドであるため、連結されたキーは 0775520090303 になります。この番号は、4 桁の暦年、各 2 桁の月と日から構成されています。要求を満たすためには、次の SSA を指定して GU 呼び出しを出す必要があります。

```
GU   ILLNESSb*C(0775520090303)
```

C コマンド・コードを使用するほうが、修飾ステートメントよりも便利なことがあります。これは、連結されたキーだけを使用するほうが、プログラムの実行中に修飾ステートメントの各部分を SSA 域に移動させるよりも簡単なためです。セグメ

ントの連結されたキーを使用することは、それらのキーで修飾されたセグメントまでの経路にあるすべての SSA を指定することと同じ意味を持ちます。

例えば、この要求に答える場合を考えてみます。

> 患者番号 07755 の Joan Carter は、2009 年 3 月 3 日にどのような処置を受けたか。

修飾ステートメントを使用する場合、GU 呼び出しで次のような SSA を指定することになります。

```
GU    PATIENTb(PATNObbbEQ07755)
      ILLNESSb(ILLDATEbEQ20090303)
      TREATMNTb
```

C コマンド・コードを使用すると、GU で次の SSA を指定することによって上記の要求に答えることができます。

```
GU    ILLNESSb*C(0775520090303)
      TREATMNTb
```

キー・フィールド以外のフィールドを使用してセグメントを修飾する必要がある場合、C コマンド・コードの代わりに修飾ステートメントを使用してください。

連結されたキーを持つ SSA は、1 つの呼び出しごとに 1 つだけ指定できます。連結されたキーで指定されたセグメントまでの経路にあるセグメントをプログラムに戻すためには、D コマンド・コードを含む非修飾 SSA を使用できます。

例えば、Joan Carter に関する PATIENT セグメントを ILLNESS セグメントとともに入出力域に戻す場合について考えてみます。次の呼び出しを使用してください。

```
GU    PATIENTb*Db

      ILLNESSb*C(0775520090303)
```

オブジェクト・セグメントを指定した C コマンド・コードは、Get 呼び出しでは使用することができますが、ISRT 呼び出しでは使用できません。 ISRT 呼び出しのオブジェクト・セグメントは、非修飾でなければなりません。

D コマンド・コード

D コマンド・コードを使用すると、個別の呼び出しで各セグメントを検索または挿入する代わりに、1 つの呼び出しで階層経路内の一連のセグメントを検索または挿入できます。 D コマンド・コードを使用する呼び出しは、パス呼び出し と呼ばれます。

プログラムで D コマンドを使用するには、PCB で P 処理オプションを指定しなければなりません。あるいは、DEDB を処理するとき、コマンド・コード D を使用します。

関連資料: P 処理オプションの使用方法の詳細は、「*IMS V14* システム・ユーティリティー」にある PSB 生成の説明を参照してください。

一連のセグメントの検索

検索呼び出しで D コマンド・コードを使用すると、IMS はそれらのセグメントを入出力域に入れます。入出力域内のセグメントは、ユーザーが指定した最初の SSA から順に、左から右の順番で続けて配置されます。 IMS が経路内の各セグメントを戻すようにするためには、各 SSA で D コマンド・コードを指定する必要があります。ただし、D コマンド・コードのない SSA をその間に介在させることができます。経路内の最後のセグメントは、常に IMS によって入出力域に戻されるため、経路内の最後のセグメントには D コマンド・コードを含む必要はありません。

D コマンド・コードは、IMS の検索論理には影響を与えません。このコマンド・コードが行うことは、各セグメントをユーザーの入出力域に移動させることだけです。 PCB 内のセグメント名は、検索された最低レベルのセグメントであるか、または GE (検出不能) 状況コードの場合には、呼び出しの条件を満たした最後のレベルのセグメントです。 これよりも高いレベルのセグメントで D コマンド・コードが指定されているものが、入出力域に入ります。

ユーザーのプログラムが要求した最低レベルのセグメントを IMS が検出できなかった場合には、GE (検出不能) 状況コードが戻されます。これは、D コマンド・コードを指定しなかったときに、ユーザーが要求したセグメントを IMS が検出できなかった場合と同じです。これは、ユーザーのプログラムが要求した最低レベルのセグメントを検出する前に IMS がデータベースの終わりに達した場合であっても、あてはまります。IMS は、パス呼び出しのどのレベルも満たさないうちにデータベースの終わりに達した場合には、GB (データベースの終了)状況コードを戻します。ただし、IMS が入出力域に 1 つ以上のセグメント (現行呼び出しの開始時点では現在位置が存在していなかった新規のセグメント) を戻した場合、および、IMS が要求された最低レベルのセグメントを検出できない場合には、データベースの最後に達している場合であっても、IMS は GE 状況コードを戻します。

D コマンド・コードによって戻される従属セグメントが必要かどうか明らかでない場合でも、D コマンド・コードを使用する利点は顕著です。例えば、従属セグメントを検査した後で、そのセグメントの使用が必要になる可能性があるとします。この場合も、D コマンドを使用すると、セグメントが必要になったときでも、そのセグメントを入手済みのため、別の呼び出しを出さずに済みます。

D コマンド・コードの例では、ユーザーのプログラムが次の要求を出す場合を考えてみます。

> 病院の患者ごとに、請求金額から受領金額を差し引いて未払い残高を計算し、それぞれの患者に郵送する請求書を印刷する。

それぞれの患者についてこの要求を処理するには、プログラムではその患者の名前と住所、患者が負担すべき金額、およびすでに支払われた金額が分かっていなければなりません。プログラムが患者セグメントがこれ以上ないことを示す GE 状況コードを受け取るまで、この呼び出しを出してください。

```
GN    PATIENTb*Db
      BILLINGb*Db
      PAYMENTbb
```

この呼び出しを出すたびに、入出力域に特定の人の患者セグメント、請求額セグメント、および支払セグメントが入ります。

一連のセグメントの挿入

ISRT 呼び出しで D コマンド・コードを使えば、1 経路分のセグメント群を同時に挿入できます。プログラムでは、パス内の各 SSA に D を挿入する必要はありません。 IMS に挿入させたい最初のセグメントに D を指定するだけで済みます。パス内のそれ以降のセグメントは、IMS によって挿入されます。

例えば、ユーザーのプログラムがこの要求を出す場合を考えてみます。

> Judy Jennison が初診で来院した。したがって、PATIENT、ILLNESS、および TREATMNT セグメントを含むレコードを追加する。

入出力域にこれらのセグメントを作成した後で、次の SSA を指定した ISRT 呼び出しを出してください。

```
ISRT    PATIENTb*Db
        ILLNESSbb
        TREATMNTb
```

PATIENT セグメントが追加されるだけではなく、その後の ILLNESS セグメントと TREATMNT セグメントもデータベースに追加されます。

経路内に論理子セグメントが存在している場合には、D コマンド・コードを使用してセグメントを挿入することはできません。

F コマンド・コード

F コマンド・コードを使用すると、特定セグメント・タイプの最初のオカレンスから検索を開始したり、新規セグメントをチェーン内の最初のオカレンスとして挿入したりできます。

最初のオカレンスとしてのセグメントの検索

F コマンド・コードは、GN 呼び出しと GNP 呼び出しで使用できます。 GU 呼び出しは、それ自体でデータベースを後戻りできます。したがって、GU 呼び出しで F コマンド・コードを使用するのは冗長であり、無視されます。 F を使用すると、このレベルの呼び出しを満たすために、親セグメントのもとで指定したセグメント・タイプの最初のオカレンスから検索を開始するように指示が行われます。

GN および GNP 呼び出しで F コマンド・コードを使用すると、データベース内で後戻りすることができます。現在位置があるセグメント・タイプの最初のオカレンスまで戻ることも、階層内で現在位置よりも前にあるセグメント・タイプまで戻ることもできます。

制約事項**:** 後戻りする前のセグメントの親は、戻り先のセグメントと同じ階層経路内になければなりません。 F をルート・レベルで指定した場合、あるいは GU または GHU で指定した場合、IMS は F を無視します。

検索は、ユーザーが親セグメントのもとで指定したセグメント・タイプの最初のオカレンスから開始する必要があります。そのレベルでの検索が満たされると、そのレベルは、あるセグメントの新規オカレンスが検索を満たしたときと同じように扱われます。 F コマンド・コードが指定されている SSA を満たすセグメントが、呼

び出しを処理する前に DL/I が位置付けられていたものと同じセグメント・オカレンスである場合でも、このことは当てはまります。

新規セグメント・オカレンスが SSA を満たすと、すべての従属セグメントの位置がリセットされます。従属セグメントに関する新しい検索は、その親セグメントのもとにある当該セグメント・タイプの最初のオカレンスから開始されます。

最初のオカレンスとしてのセグメントの挿入

ISRT 呼び出しで F を使用するということは、そのセグメントを、当該セグメント・タイプの最初のセグメント・オカレンスとして挿入するよう、IMS に指示することを意味します。F は、キーがないか、または非ユニーク・キーが指定されていて、しかも DBD における SEGM ステートメントの RULES オペランドで HERE が指定されているセグメントで使用してください。 DBD で HERE を指定する場合、F コマンド・コードがこれをオーバーライドし、IMS が新規のセグメント・オカレンスをそのセグメント・タイプの最初のオカレンスとして挿入します。

DBD の RULES 指定を変更するために使用する F コマンド・コードは、 ISRT 呼び出しのためにセグメントにアクセスする目的で使用している (論理または物理のいずれかの) パスだけに適用されます。例えば、セグメントにアクセスするために物理パスを使用している場合には、このコマンド・コードはその物理パスに適用され、論理パスには適用されません。 RULES を指定してコマンド・コードを使用する方法については、各インストール先のデータベース管理者にお尋ねください。

例えば、TREATMNT セグメントに関して、DBD で RULES=HERE を指定する場合を考えてみます。この要求を満たすものとします。

> Mary Martin が今日来院し、何人かの医師の診断を受けた。最も新しい疾患に関する最初の TREATMNT セグメントとして、Smith 医師に関する TREATMNT セグメントを追加したい。

最初に、入出力域に TREATMNT セグメントを作成します。

```
19930302ESEDRIXbbb0040SMITHbbbbb
```

次に以下の SSA を指定して ISRT 呼び出しを出します。これにより、TREATMNT セグメントの新規オカレンスが、同じキーの TREATMNT セグメント・タイプのオカレンスのうちの最初のオカレンスとして追加されます。

```
ISRT    PATIENTb(PATNObbb=b06439)
        ILLNESSb*L
        TREATMNT*F
```

この例は、データベースの種類のいかんを問わず、HDAM または PHDAM ルート・セグメントと従属セグメントに適用されます。

関連資料:

22 ページの『GU/GHU 呼び出し』

G コマンド・コード

G コマンド・コードを使用すると、ランダム化や区画選択出口ルーチンの呼び出しをスキップして、データベースを順次に検索するように、IMS に指示することがで

きます。このコマンド・コードは他のデータベース・タイプでも使用できますが、HDAM/PHDAM、DEDB、および PHIDAM データベースへのアクセスにのみ影響します。

HALDB 区画選択出口ルーチンを使用してアクセスされた HDAM/PHDAM、DEDB、または PHIDAM データベースにアクセスするとき、レコードが区画境界を越えて順序付けられていない場合は、複数の修飾 SSA の要求範囲のすべてのキーが戻されない可能性があります。データベースへの最初の呼び出しまたはコマンド A を使用する場合、コマンド・コード G を使用して、SSA が満たされるまでデータベースを順次に読み取ることができます。

L コマンド・コード

L コマンド・コードを使用して、特定のセグメント・タイプの最後のオカレンスを検索したり、セグメントをセグメント・タイプの最後のオカレンスとして挿入したりできます。

最後のオカレンスとしてのセグメントの検索

L コマンド・コードは、SSA を満たす最後のセグメント・オカレンスを検索したいことを、または指定したセグメント・オカレンスをそのセグメント・オカレンスの最後のオカレンスとして挿入したいことを表します。 F と同じように、必要なオカレンスが最後のセグメント・オカレンスであることが分かっている場合には、それ以前のオカレンスをプログラム論理で検査しなくても、そのセグメント・タイプの最後のオカレンスに直接アクセスすることができるため、L を使用するとプログラミングを単純化できます。 GU 呼び出しを使用した場合、通常 IMS は最初のオカレンスを戻すため、L は GU または GHU で使用できます。IMS は、ルート・レベルでの L を無視します。

GU、GN、および GNP で L を使用すると、IMS は、ユーザーが指定した修飾を満たすセグメント・タイプの最後のオカレンスにアクセスします。この修飾は、セグメント・タイプまたは SSA の修飾ステートメントです。セグメント・タイプだけを指定した場合 (非修飾 SSA)、IMS はその親セグメントのもとからこのセグメント・タイプの最後のオカレンスを検索します。

例えば、医療の階層を使用して次の要求を出す場合を考えてみます。

> 患者番号 10345 の Jennifer Thompson が最後に来院したのは、どのような病気のためか。

この例では、データベースの DBD で ILLNESS に RULES=LAST が指定されているものとします。この情報を検索するには、次の呼び出しを発行します。

```
GU    PATIENTb(PATNObbb=b10345)
      ILLNESSb*L
```

1 番目の SSA は、この患者の番号を IMS に提供します。2 番目の SSA は、この患者に関する ILLNESS セグメントの最後のオカレンス (この例では、時間的に最初のオカレンス) を要求します。

最後のオカレンスとしてのセグメントの挿入

ISRT で L を使用するのは、キーがないか、一意でないキーが含まれていて、そのセグメントの挿入規則が FIRST または HERE のいずれかである場合のみにしてください。 L コマンド・コードを使用すると、データベースの種類を問わず、HDAM または PHDAM ルート・セグメントと従属セグメントの FIRST と HERE がともにオーバーライドされます。

DBD の RULES 指定を変更するために使用する L コマンド・コードは、 ISRT 呼び出しのためにセグメントにアクセスする目的で使用している (論理または物理のいずれかの) パスだけに適用されます。例えば、セグメントにアクセスするために物理パスを使用している場合には、このコマンド・コードはその物理パスに適用され、論理パスには適用されません。 RULES を指定してコマンド・コードを使用する方法については、データベース管理者にお尋ねください。

N コマンド・コード

N コマンド・コードを使用すると、パス呼び出しでセグメントが置き換えられなくなります。 N コマンド・コードは D コマンド・コードとともに使用され、アプリケーション・プログラムが 1 つの呼び出しで複数のセグメントを処理できるようにします。 D コマンド・コードを単独で使用すると、セグメントの経路が検索されて入出力域に入りますが、 N コマンド・コードとともに使用した場合には、D コマンド・コードは、置き換えたいセグメントを区別できます。

例えば、次のコードは、TREATMNT セグメントの置き換えだけを行います。

```
GHU    PATIENT*D(PATNObbb=b06439)
       ILLNESSb*D(ILLDATEb=19930301)
       TREATMNT

REPL   PATIENT*N(PATNObbb=b06439)
       ILLNESSb*N(ILLDATEb=19930301)
       TREATMNT
```

制約事項: D コマンド・コードと N コマンド・コードを同時に使用すると、IMS はセグメントを検索しますが、置き換えはしません。

N コマンド・コードは REPL 呼び出しにだけ適用され、別の呼び出しで使用しても IMS には無視されます。

O コマンド・コード

O コマンド・コードを使用すると、DBD 定義のフィールド名の代わりにターゲット・データの位置および長さが含まれた SSA 修飾を指定することができます。

このコマンド・コードは、全機能データベース・タイプ
(HDAM、HIDAM、PHIDAM、および PHDAM) および高速機能 DEDB に有効です。

このコマンド・コードは、以下の DL/I 呼び出しでサポートされます。

- GU SSA
- GHU SSA
- GN SSA

- GNP SSA
- GHNP SSA
- ISRT SSA

コマンド・コード O を指定した場合、SSA 修飾には通常のフィールド名を含めることも、リトリーブしたいデータの開始オフセットと長さを含めることもできます。

オフセットおよび長さは、フィールド名を指定する場合に使用する通常の 8 バイトの文字値ではなく、2 つの 4 バイトの 2 進数値として指定する必要があります。オフセットの開始位置は 1 で、オフセットはセグメント定義の物理開始とのオフセットです。サポートされる最大長は、データベース・タイプの最大セグメント・サイズです。最小長は 1 です。

例えば、セグメントに対して、以下のオフセットおよび長さを持ついくつかのフィールドが DBD で定義されているとします。

```
Field    Offset Length
Labname  1      5
Street   10     20
State    30     2
```

アプリケーション・プログラムには、以下のマップを持つ COBOL コピーブックがあります。

```
Field    Offset Length
Labname  1      5
Type     6      3
Street   10     20
State    30     2
```

データベースには、以下のデータを持つ 2 つのレコードが含まれています。

```
           I          1111111122222222233
           I 12345678901234567901235678901
___________I ___________________________
Segment #1 I SVL  DEV 555 BAILEY AVE    CA
Segment #2 I ARC  RSC 650 HARRY RD      CA
___________I ___________________________
```

以下の形式で O コマンド・コードを使用して GU 呼び出しを指定することで、DBD でフィールドを指定する必要なくデータをリトリーブすることができます。以下の例では、16 進数編集モードを使用して DFSDDLT0 テスト・アプリケーションでオフセットおよび長さの値を指定する方法を示しています。

```
              00000000
GU IBMLABS*O ('00010005'x=SVL  )
              00000000
GU IBMLABS*O ('00010005'x=ARC  )
              00030000
GU IBMLABS*O ('00000002'x=CA)
              00000000
GU IBMLABS*O ('000060003'x=DEV)
```

最初の GU 呼び出しでは、オフセットは 1 で、ターゲット・データの長さは 5 です。

P コマンド・コード

通常 IMS は、呼び出し中にアクセスされた最低セグメントのレベルで親子関係を設定します。これよりも高いレベルで親子関係を設定するには、GU、 GN、または GNP 呼び出しで P コマンド・コードを使用できます。

P を使用して設定した親子関係は、IMS が設定した親子関係と同じように機能します。すなわち、後続の GNP 呼び出しにも有効であり、ISRT、DLET、または REPL 呼び出しによる影響も受けません。GNP 呼び出しで P コマンド・コードを使用した場合に、その GNP によって影響を受けるだけです。親子関係は、後続の GU、GHU、GN、または GHN 呼び出しによって取り消されます。

P コマンド・コードは、呼び出しの 1 レベルだけで使用してください。誤って、ある呼び出しの複数レベルで P コマンド・コードを使用すると、IMS は、P を含む呼び出しの最低レベルで親子関係を設定します。

P を使用する呼び出しを IMS が完全に満たすことができない場合 (例えば、IMS が GE 状況コードを戻した場合) でも、P を含むレベルが満たされる場合には、P は有効です。 IMS が P を含むレベルの呼び出しを完全に満たすことができない場合、IMS は、親子関係の設定を行いません。そのあとで GNP を出すと、GP (親子関係未設定) を受け取ります。

GNP 呼び出しで P を使用すると、IMS は、前の呼び出しですでに設定されている親子関係を使用して GNP 呼び出しを処理します。その GNP 呼び出しを処理した後で、IMS は親子関係をリセットして、ユーザーが P を使用して指定した親子関係を設定します。

例えば、その月に来院したすべての患者に対して現在の請求書を送付したい場合には、ILLNESS セグメントの値が判別値です。 ILLNESS セグメントの中の日付が当月の最初の日以降の値になっている患者だけを調べる必要があります。その月の間に来院した患者について、その住所と、BILLING セグメント内の支払うべき金額を調べ、請求書を印刷できるようにする必要があります。この例では、日付を 1993 年 3 月 31 日と仮定しています。必要な情報を処理するために、次の 2 つの呼び出しを出します。

```
GN    PATIENTb*PD
      ILLNESSb(ILLDATEb>=19930301)
GNP   BILLINGbb
```

その月に来院した患者が検出された後で、その患者の BILLING セグメントを検索するために GNP 呼び出しを出します。 IMS が GB 状況コードを戻すまで GN 呼び出しを繰り返して、その月に来院した患者を検出します。

Q コマンド・コード

ユーザーのプログラムがコミット・ポイントに達するまで別のプログラムによってセグメントが更新されないようにしたい場合には、Q コマンド・コードを使用できます。 Q コマンド・コードは IMS に対して、ユーザーのアプリケーション・プログラムがセグメントを処理する必要があり、プログラムが終了するまでは別のタスクがそのセグメントを変更できないことを指示します。

すなわち、Q コマンド・コードを使用してセグメントを検索すると、その後でそのセグメントを再び検索するときにも、別のプログラムによってそのセグメントが変

更されないようにすることができます。 (ただし、ユーザーのプログラムが排他的に使用するようにセグメントを予約することにより、システムのパフォーマンスが影響を受ける可能性がありますので、ご注意ください。)

Q コマンド・コードは、データ共用環境のバッチ・プログラムと、CICS および IMS オンライン・プログラムで使用できます。IMS は、非データ共用バッチ・プログラムでは Q を無視します。

データベース呼び出しの数の限度

全機能の場合、プログラムで Q コマンド・コードを使用する前に、 PSBGEN 時に MAXQ 値を指定しておかなければなりません。その結果、同期点間に設定できるデータベース呼び出し (Q コマンド・コード付き) の最大数が確立されます。

関連資料: PSBGEN については、「*IMS V14* システム・ユーティリティー」を参照してください。

高速機能は、MAXQ パラメーターをサポートしていません。したがって、高速機能では、Q コマンド・コードを使用するデータベース呼び出しを必要な数だけ出すことができます。

セグメント・ロック・クラスの使用

全機能の場合、Q コマンド・コードを使用してセグメントを検索する場合、文字 Q と、その後に続けてセグメントのロック・クラスを指示する文字 (A-J) を指定します (例えば、QA)。ロック・クラスが文字 (A-J) でない場合、IMS は状況コード GL を戻します。

高速機能は Q コマンド・コードを単独で (つまり、ロック・クラスを指定する文字なしで) サポートします。ただし、高速機能と全機能の整合性を保つため、高速機能は Q コマンド・コードを 2 バイトのストリングとして取り扱います。2 バイト・ストリングの 2 番目のバイトは英字 (A-J) でなければなりません。 2 バイト目が英字 (A-J) でない場合、IMS は状況コード AJ を戻します。

例えば、品目 1 が 50 個、品目 2 が 75 個、品目 3 が 100 個ある場合に限り、品目 1、2、 3 を発注したい場合を考えます。この注文を出す前に、プログラムは、3 つの品目セグメントをすべて調べ、各品目の在庫数量が十分であるかどうかを判別する必要があります。ユーザーのプログラムがこれを判別し、(発注が可能な場合に) 注文を出すまでは、これらのどのセグメントも別のアプリケーション・プログラムによって変更されないようにする必要があります。

全機能の場合、ユーザーのプログラムでは、この要求を処理するために品目セグメントに関する Get 呼び出しを出すときに Q コマンド・コードを使用します。 SSA で Q コマンド・コードを使用するときには、SSA のコマンド・コードの直後にロック・クラスを割り当てます。

```
GU    PART X
      ITEM 1    *QA
GU    PART X
      ITEM 2    *QA
GU    PART X
      ITEM 3    *QA
```

例外: 高速機能の場合、ロック・クラスの 2 番目のバイトは、ロック・クラス 'A' とは解釈されません。

品目セグメントを検索した後、ユーザーのプログラムはそれらを検査し、発注に十分な数だけ在庫があるかどうかを決定します。各品目の在庫数量が 100 であるものと仮定します。ユーザーのプログラムは注文を出し、それにもとづいてデータベースを更新します。このプログラムでは、セグメントを更新するために各セグメントについて GHU 呼び出しを出し、その直後に REPL 呼び出しを出します。

```
GHU   ITEM 1
REPL  ITEM 1 with the value 50
GHU   ITEM 2
REPL  ITEM 2 with the value 25
GHU   ITEM 3
REPL  ITEM 3 with the value 0
```

DEQ 呼び出しで Q コマンド・コードを使用する

Q コマンド・コードを DEQ 呼び出しと併用すると、セグメントの予約と解放を実行できます。

全機能の場合、セグメントを解放するために入出力 PCB に対する DEQ 呼び出しを出す場合には、そのセグメントのロック・クラスを指定する文字を入出力域の最初のバイトに入れてください。その後で、その文字を含む入出力域の名前を指定して DEQ 呼び出しを出します。

DEDB DEQ 呼び出しは、DEDB PCB に対して出されます。高速機能はロック・クラスをサポートしないので、 DEDB DEQ 呼び出しでは、入出力域にロック・クラスを指定する必要はありません。

制約事項: EXEC DL/I は DEQ 呼び出し用の PCB を認めていないので、EXEC DL/I インターフェースは、DEDB DEQ 呼び出しをサポートしません。

全機能 DEQ 呼び出しによるセグメントの検索

DEQ 呼び出しは、Q コマンド・コードを使用して検索されるすべてのセグメントを解放します。ただし、次の場合は除きます。

- ユーザーのプログラムによって修正されたセグメントは、そのプログラムがコミット・ポイントに達するまで解放されません。
- 階層内でユーザーの位置を保持するために必要なセグメントは、ユーザーのプログラムが別のデータベース・レコードに移動するまで解放されません。
- 別のクラスとして再びロックされたセグメントのクラスは解放されません。

ユーザーのプログラムがセグメントの読み取りだけを行う場合には、プログラムで DEQ 呼び出しを出してセグメントを解放できます。ユーザーのプログラムが DEQ 呼び出しを出さない場合にも、IMS は、そのプログラムがコミット・ポイントに達したときに予約済みのセグメントを解放します。ただし、ユーザーのプログラムがコミット・ポイントに達する前に DEQ 呼び出しを使用してセグメントを解放すると、他のプログラムがより早くそのセグメントを使用できるようになります。

高速機能 DEQ 呼び出しでバッファーを検索する

DEQ 呼び出しにより、高速機能は以下の条件の 1 つを満足するバッファーを解放します。

- バッファーは修正されていません。あるいは、バッファーは有効なルート位置を保護しません。
- バッファーは Q コマンド・コードにより保護されてきました。

DEQ 呼び出し用にバッファーを解放できない場合、高速機能は FW 状況コードを戻します。

DEQ 呼び出しが出されるまで、あるいはコミット・ポイントに到達するまで、Q コマンド・コードで実行されるすべての CI ロックまたはセグメント・レベルのロッキングは他のアプリケーション・プログラムから保護されます。

ルート・セグメントおよび従属セグメントについての考慮事項 (全機能のみ)

ルート・セグメントで Q コマンド・コードを使用した場合、PCB が更新機能を持たない他のプログラムは、データベース・レコードにアクセスできます。 PCB が更新機能を持つプログラムは、そのデータベース・レコードのどのセグメントにもアクセスできません。従属セグメントで Q コマンド・コードを使用した場合、他のプログラムは、保留機能のない Get 呼び出しのうちの 1 つを使用して、そのセグメントを読み取ることができます。プログラムが共用データベースにアクセスしていて、そのブロックのセグメントのいずれかが Q コマンド・コードで予約されている場合、他の IMS システムのアプリケーション・プログラムは、そのブロック内については更新できません。 Q コマンド・コードは、会話の 1 つのステップから別のステップまでセグメントを保持することはありません。

関連資料: Q コマンド・コードと DEQ 呼び出しの関連について詳しくは、「*IMS V14* アプリケーション・プログラミング」のトピック『ユーザー・プログラムの排他使用に対するセグメントの予約』を参照してください。

U コマンド・コード

IMS が検索呼び出しあるいは ISRT 呼び出し内のあるレベルを満たすと、そのレベルを満たしたセグメント・オカレンスに位置が設定されます。U コマンド・コードを使用すると、階層従属セグメントの検索中にセグメントから位置が移動しないようになります。

セグメントに固有の順序フィールドがある場合、このコードを使用することは、SSA を修飾してキー・フィールドの現行値と等しくすることと同じ意味を持ちます。呼び出しが満たされるときに、U コードが出されたレベルよりも上のレベルに位置が移動すると、U コードは親セグメントの位置が変更されたセグメント・タイプに対しては効力を持たなくなります。

特に、キーのない従属セグメントまたは非ユニーク・キーのセグメントを処理するときには、U が役に立ちます。キーのないセグメントまたは非ユニーク・キーのセグメントの特定オカレンス上の位置は、このコードによって保持できます。

例: Mary Warren という名前の患者が最後に来院した時の病気、およびその病気に関して受けた処置について調べたいものとします。以下の図は、Mary Warren についての PATIENT、ILLNESS、および TREATMNT セグメントを示します。

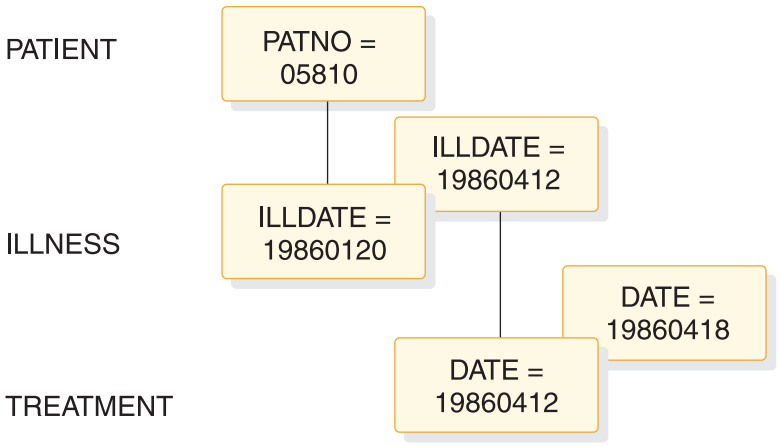


図 3. U コマンド・コードの例

この情報を検索するためには、最初の ILLNESS セグメント、およびその ILLNESS セグメントに関連した TREATMNT セグメントを検索します。最新の ILLNESS セグメントを検索するためには、次の GU 呼び出しを出すことができます。

```
GU    PATIENTb(PATNObbb=b05810
      ILLNESSb*L
```

この呼び出しの後で、IMS はルート・レベルでキー 05810 の PATIENT セグメントおよび最後の ILLNESS セグメントに、位置を設定します。これと別にキー 19860412 の ILLNESS セグメントが存在する可能性があるため、これを最新の ILLNESS セグメントと考えることができます。その ILLNESS セグメントに関連する TREATMNT セグメント・オカレンスを検索する必要が起こる場合があります。そのためには、U コマンド・コードを使用して次の GN 呼び出しを出すことができます。

```
GN     PATIENTb*U
       ILLNESSb*U
       TREATMNT
```

この例の U コマンド・コードは、IMS が位置を設定した ILLNESS および PATIENT セグメントに従属している TREATMNT セグメントだけが必要であることを IMS に指示しています。上記の GN 呼び出しを最初に出すと、19860412 というキーの TREATMNT セグメントが検索されます。 2 回目の GN 呼び出し時には、19860418 というキーの TREATMNT セグメントが検索されます。 3 回目の呼び出し時には、IMS は未検出状況コードを戻します。U コマンド・コードは、この親セグメントのもとで低位修飾を満たすセグメントが検出されないときには、別の親セグメントのもとでも検索を続行できないことを IMS に知らせます。この PATIENT SSA に U コマンド・コードが指定されていない場合、3 番目の GN 呼び出しを出すと、IMS は呼び出しを満たすためにルート・レベルで前方に移動します。修飾された SSA に U コマンド・コードを指定したときには、IMS は U を無視します。

コマンド・コード F または L とともに使用された場合、その呼び出しに関する SSA のレベルおよびそれよりも下位のすべての SSA レベルで、U コマンド・コードは無視されます。

V コマンド・コード

SSA で V コマンド・コードを使用すると、その SSA および先行のすべての SSA で U コマンド・コードを使用した場合と同じ効果が得られます。あるセグメント・レベルに関して V コマンド・コードを指定すると、そのレベルおよびそれよりも上位のレベルで設定された位置を、その呼び出しの修飾として使用するように IMS に対して指示します。

V コマンド・コードを使用すると、IMS の現在位置を指定して修飾された SSA で要求を修飾した場合と類似の効果が得られます。

例えば、この要求に答える場合を考えてみます。

> 患者番号 07755 の Joan Carter が 2009 年 3 月 3 日に処置を受けたかどうか。

修飾された SSA を使用して、次の呼び出しを出してください。

```
GU     PATIENTb(PATNObbb=b07755)
       ILLNESSb(ILLDATEb=20090303)
       TREATMNT
```

患者番号 07755 の PATIENT セグメントおよび 2009 年 3 月 3 日の ILLNESS セグメントに位置を設定した場合、この位置を使用して調べたい TREATMNT セグメントを検索できます。そのためには、次のように V コマンド・コードを指定してください。

```
GN     PATIENTbb
       ILLNESSbb*V
       TREATMNT
```

呼び出しで V コマンド・コードを使用することは、親子関係を設定したうえで後続 GNP 呼び出しを出した場合と同じような効果がありますが、V コマンド・コードで設定される親子関係は、そのコードが使用される呼び出しに関する親子関係であって、後続の呼び出しに関する親子関係は設定されません。例えば、上記の要求を満たすためには、ILLNESS セグメント・レベルで親子関係を設定し、GNP を出してその親のもとにある TREATMNT セグメントを検索することもできます。 V コマンド・コードを使用した場合には、その呼び出しの親子関係として ILLNESS セグメントが使用されるようになります。

V コマンド・コードは、どの親セグメントに対しても使用できます。修飾された SSA で V コマンド・コードを使用すると、そのレベルおよび修飾された SSA を含む上位レベルでは、V コマンド・コードは無視されます。

NULL コマンド・コード

NULL コマンド・コード (-) を使用すると、プログラムが実行中に必要となるコマンド・コードを保管するための 1 つ以上の位置を SSA 内に予約できます。

例えば、次のように、2 つのコマンド・コードのための位置を予約します。

```
GU     PATIENTb*--(PATNObbb=b07755)
       ILLNESSbILLDATEb=19930303)
       TREATMNT
```

NULL コマンド・コードを使用すると、同一の SSA のセットを複数の目的に使用できます。ただし、SSA を動的に修正すると、それだけデバッグが困難になります。

DL/I 用の DEDB コマンド・コード

M、R、S、W、および Z コマンド・コードは DEDB だけで使用されます。

サンプル・アプリケーション・プログラム

以下の例は、通帳口座 (普通預金口座) の銀行取引を記録するサンプル・アプリケーション・プログラムに基づいています。取引は、顧客の通帳に記入されたかどうかに応じて、記帳取引または未記帳取引としてデータベースに書き込まれます。

例えば、Bob Emery という人が通帳を忘れてきて銀行取引を行った場合には、アプリケーション・プログラムにより、その取引は未記帳としてデータベースに書き込まれます。アプリケーション・プログラムは、サブセット・ポインターを、最初の未記帳取引に設定するので、後で簡単にアクセスすることができます。次に Bob が通帳を持参したときに、プログラムによってそれらの取引が記帳されます。

プログラムは、前にセットしたサブセット・ポインターを使用して、最初の未記帳取引を直接検索できます。取引を記帳すると、このプログラムはサブセット・ポインターをゼロにセットします。これ以降にデータベースを更新するアプリケーション・プログラムは、未記帳取引がないことを判別できます。以下の図は、通帳が使用できる場合と使用できない場合の処理を要約しています。

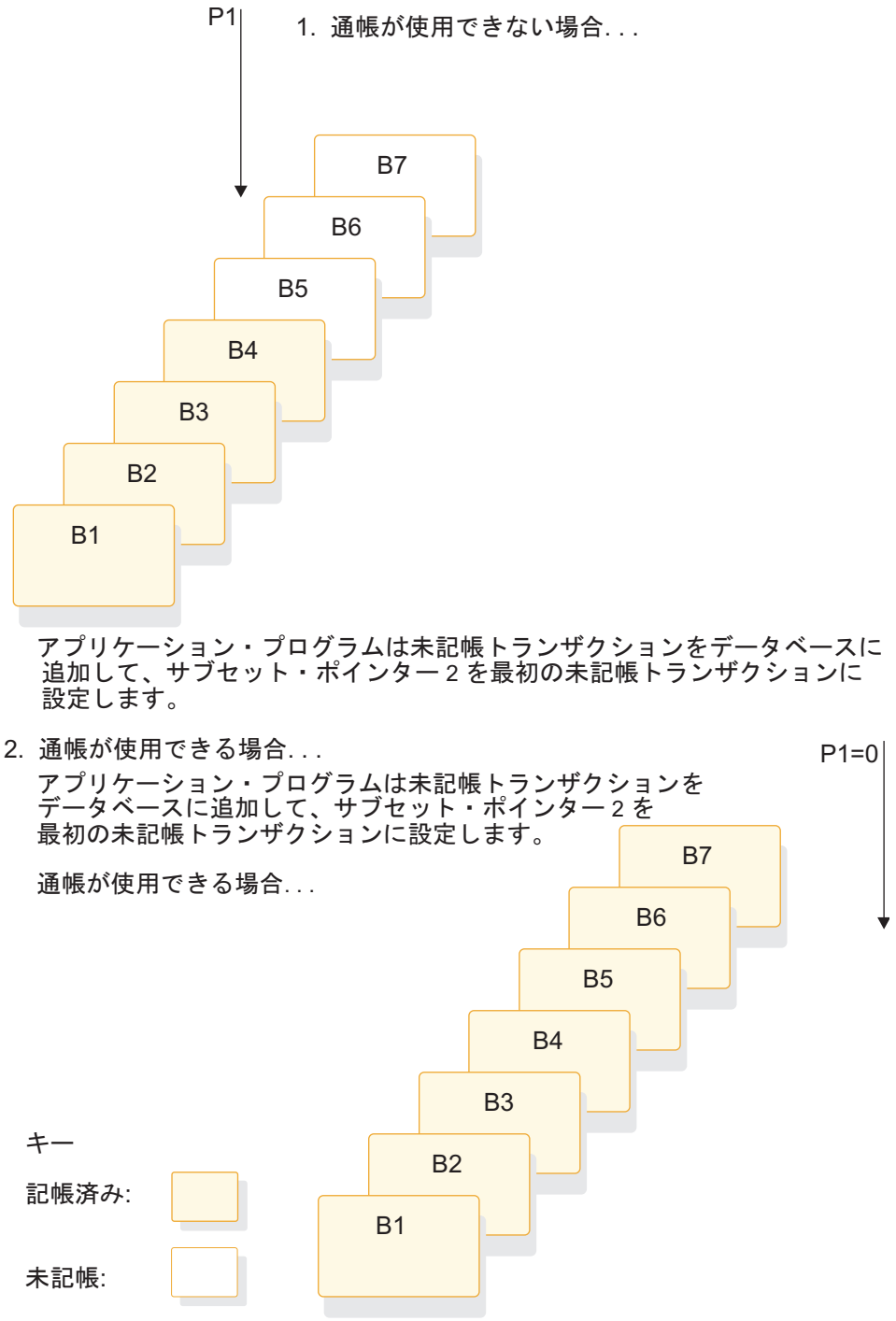


図 4. 通帳処理の例

M コマンド・コード

ユーザーのプログラムは、サブセット・ポインターを前方に移動させて、現在位置の次のセグメントに移すために、M コマンド・コードを使用した呼び出しを出します。

上記の通帳処理の例を使用して、取引のうちの全部ではなく一部を記帳した後、最初の未記帳取引にサブセット・ポインターをセットしたいものとします。以下の図に示すように、次のコマンドを使用すると、セグメント B6 にサブセット・ポインター 1 がセットされます。

```
GU      Abbbbbbb(AKEYbbb
        Bbbbbbbb*R1M1
```

現行セグメントがセグメント・チェーンの最後になっている場合に M コマンド・コードを使用すると、IMS はポインターをゼロにセットします。

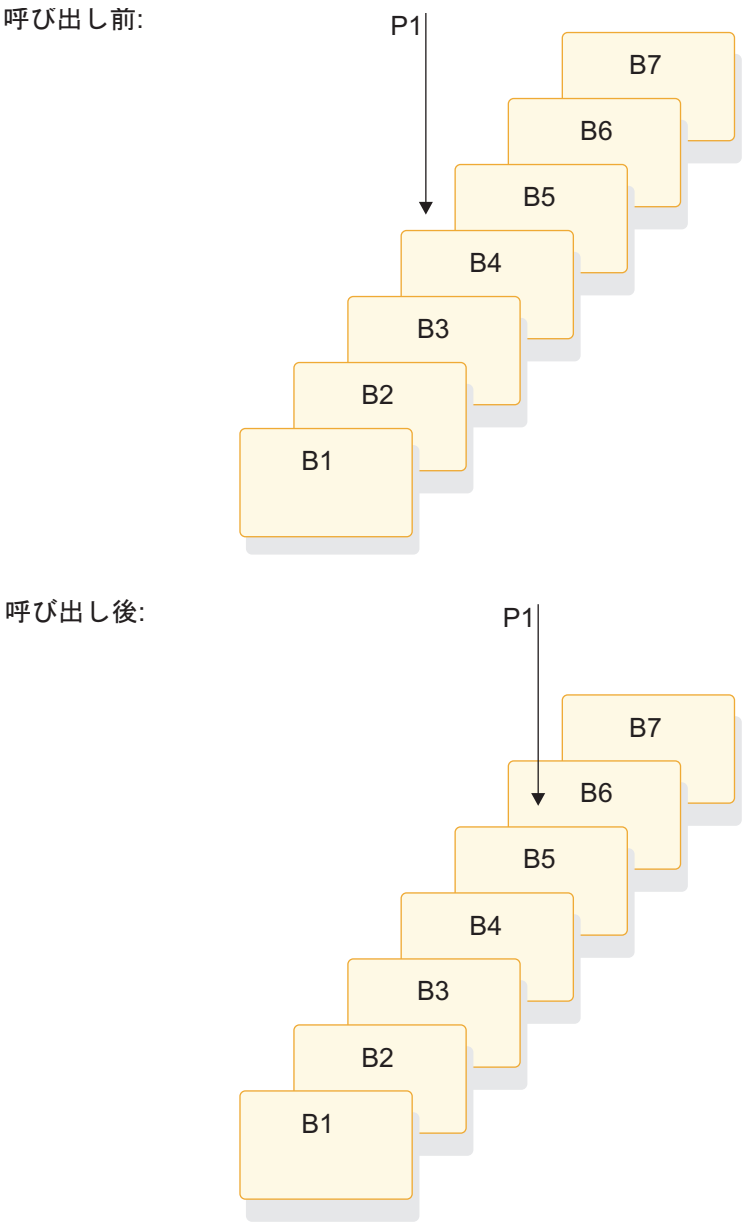


図 5. 現在位置の次のセグメントへのサブセット・ポインターの移動

R コマンド・コード

ユーザーのプログラムは、サブセット内の最初のセグメント・オカレンスを検索するために、R コマンド・コードを使用した Get 呼び出しを出します。 R コマンド・コードは、ポインターのセットまたは移動は実行しません。このコマンド・コードは、サブセット内の最初のセグメント・オカレンスに位置を設定したいことを IMS に対して指示します。 R コマンド・コードは F コマンド・コードに似ていますが、セグメント・チェーン全体ではなく、サブセットに適用される点が異なります。

預金口座の例を使用して、Bob Emery が通帳を持って銀行に来店した場合に、すべての未記帳取引を記帳したいものとします。すでにサブセット・ポインター 1 が最初の未記帳取引にセットされているため、ユーザーのプログラムでは、次の呼び出しを使用してその取引を検索できます。

```
GU      Abbbbbbb(AKEYbbbb=bA1)
        Bbbbbbbb*R1
```

以下の図に示すように、この呼び出しによってセグメント B5 が検索されます。チェーン内のセグメントの処理を続行するには、サブセット・ポインターを使用しない場合と同じように GN 呼び出しを出すことができます。

このサブセットが存在しない場合 (サブセット・ポインター 1 がゼロにセットされている場合) には、IMS が GE 状況コードを戻し、データベースにおけるユーザー位置はチェーン内の最後のセグメントの直後になります。上記の預金口座の例の場合、GE 状況コードは未記帳取引がないことを示します。

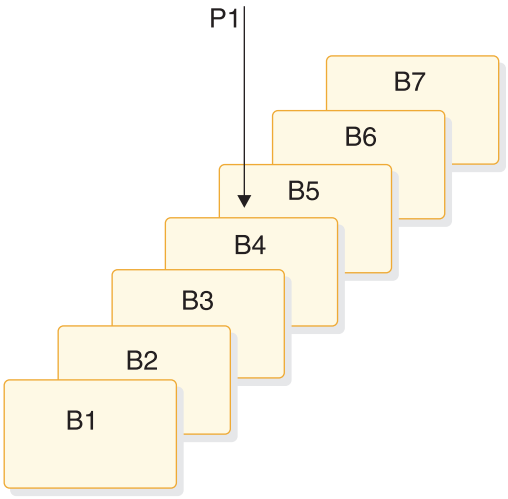


図 6. セグメント・チェーン内の最初のセグメントの検索

各 SSA には R コマンド・コードを 1 つだけ指定できます。 1 つの SSA で複数の R コマンド・コードを使用すると、IMS はユーザーのプログラムに AJ 状況コードを戻します。

R コマンド・コードを、他のコマンド・コード (ただし、F および Q 以外) と一緒に使用することができます。SSA で指定した別のコマンド・コードが有効になるのは、R コマンド・コードが処理されてサブセット内の最初のセグメントに正常に位置が設定された後です。 L コマンド・コードと R コマンド・コードを一緒に使用

すると、セグメント・チェーン内の最後のセグメントが検索されます。 (R コマンド・コードで指定されたサブセット・ポインターがセットされていない場合、IMS はセグメント・チェーン内の最後のセグメントの代わりに GE 状況コードを戻します。) R コマンド・コードと F コマンド・コードは、一緒に使用しないでください。この 2 つを一緒に使用すると、AJ 状況コードが戻されます。 R コマンド・コードは、LAST を含むすべての挿入規則を無視します。

S コマンド・コード

すでにサブセット・ポインターがセットされているかどうかにかかわらず、無条件にサブセット・ポインターをセットするために、プログラムは S コマンド・コードをつけて呼び出しを発行します。

ユーザーのプログラムが S コマンド・コードを含む呼び出しを出すと、IMS はユーザーの現在位置にポインターをセットします。

例えば、サブセット・ポインター 1 で定義されたサブセット内の最初の B セグメント・オカレンスを検索し、次の B セグメント・オカレンスにポインター 1 をリセットしたい場合には、次のようなコマンドを出します。

```
GU     Abbbbbbb(AKEYbbb=bB1)
       Bbbbbbbb*R1
GN     Bbbbbbbb*S1
```

以下の図に示すように、この呼び出しを出した後では、サブセット・ポインター 1 はセグメント B5 の代わりにセグメント B6 を指します。

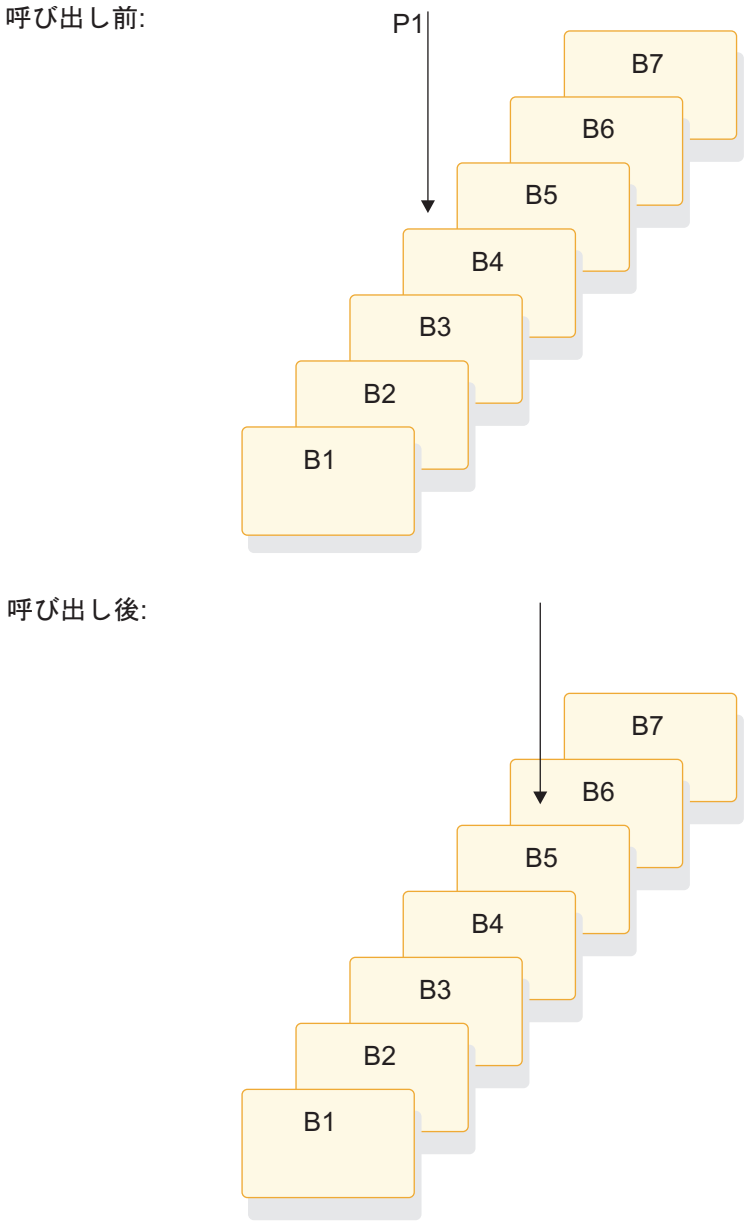


図 7. 無条件でのサブセット・ポインターの現行位置への設定

W コマンド・コード

S コマンド・コードと同じように、W コマンド・コードはサブセット・ポインターを条件付きでセットします。 W コマンド・コードを使用した場合、サブセット・ポインターを更新するのは、そのサブセット・ポインターがまだセグメントにセットされていない場合に限られるという点が S コマンド・コードを使用した場合と異なっています。

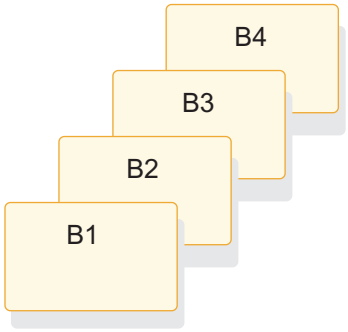
例えば、通帳処理の例を使用して、 Bob Emery が通帳を忘れて銀行に来店した場合について考えてみます。未記帳の取引をデータベースに追加します。最初の未記帳取引にポインターをセットして、後からそれらの取引を記帳するときに、最初の

未記帳取引に即時にアクセスできるようにします。次の呼び出しを出すと、ユーザーが挿入する取引が最初の未記帳取引である場合に、サブセット・ポインターがその取引にセットされます。

```
ISRT     Abbbbbbb(AKEYbbbb=bA1)
         Bbbbbbb*W1
```

以下の図に示すように、この呼び出しによってセグメント B5 にサブセット・ポインター 1 がセットされます。すでに未記帳取引がある場合には、サブセット・ポインターは変更されません。

呼び出し前:



呼び出し後:

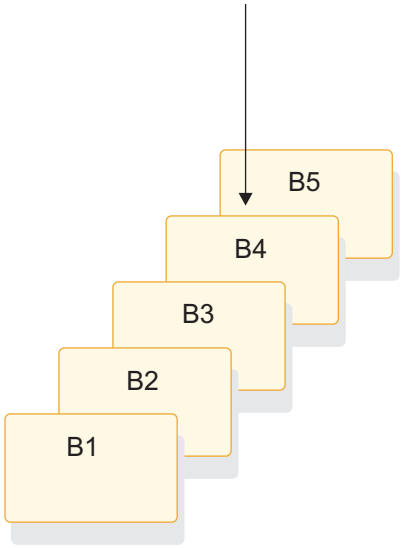


図 *8.* 条件付きでのサブセット・ポインターの現行位置への設定

Z コマンド・コード

Z コマンド・コードは、サブセット・ポインターの値をゼロにセットします。ユーザーのプログラムが Z コマンド・コードを使用した呼び出しを出すと、サブセット・ポインターはセグメントにセットされなくなり、そのポインターによって定義されたサブセットは存在しなくなります。

値がゼロのサブセット・ポインターを使用しようとすると、IMS は GE 状況コードをユーザーのプログラムに戻します。

例えば、通帳処理の例を使用して、最初の未記帳取引を検索するために R コマンド・コードを使用する場合を考えてみます。その後で、セグメントのチェーンを処理し、取引を記帳します。取引の記帳を済ませて、新しい取引をチェーンに挿入した後で、次の呼び出しのように Z コマンド・コードを使用してサブセット・ポインターをゼロにセットします。

```
ISRT     Abbbbbbb(AKEYbbbb=bA1)
         Bbbbbbbb*Z1
```

この呼び出しを出すと、サブセット・ポインター 1 がゼロにセットされ、続いてそのデータベースを更新するプログラムに、未記帳取引がないことが示されます。

呼び出し、AIB、および PCB 間の関係

以下の表は、呼び出しと全機能 (FF)、主記憶データベース (MSDB)、高速処理データベース (DEDB)、入出力、および汎用順次アクセス方式 (GSAM) の各 PCB との関係を示しています。

表 *54.* 呼び出しと *PCB* との関係

呼び出し	**AIB**	**FF PCB**	**MSDB PCB**	**DEDB PCB**	入出力 **PCB**	**GSAM PCB (**複数の場合もある**)**
APSB	X					
CHKP	X				X	
CIMS	X					
CLSE	X					X
DEQ	X			X	X	
DLET	X	X	X	X		
DPSB	X					
FLD	X		X	X		
GHN	X	X	X	X		
GHNP	X	X	X	X		
GHU	X	X	X	X		
GMSG	X					
GN	X	X	X	X	X	X
GNP	X	X	X	X		
GSCD[1]	X	X	X	X	X	
GU	X	X	X	X	X	X

表 54. 呼び出しと *PCB* との関係 (続き)

呼び出し	**AIB**	**FF PCB**	**MSDB PCB**	**DEDB PCB**	入出力 **PCB**	**GSAM PCB (**複数の場合もある**)**
ICMD	X					
INIT	X				X	
INQY	X					
ISRT	X	X	X	X	X	X
LOG	X				X	
OPEN	X					X
PCB[2]						
POS	X			X		
RCMD	X					
REPL	X	X	X	X		
ROLB	X				X	
ROLL[2]						
ROLS	X	X			X	
SETS/SETU	X				X	
SNAP[3]	X	X	X	X	X	
STAT[4]	X	X				
SYNC	X				X	
TERM[2]						
XRST	X				X	

注記:

1. GSCD は、プロダクト・センシティブ・プログラミング・インターフェースです。
2. PCB、ROLL、および TERM 呼び出しに関連する PCB はありません。
3. SNAP は、プロダクト・センシティブ・プログラミング・インターフェースです。
4. STAT は、プロダクト・センシティブ・プログラミング・インターフェースです。

DL/I テスト・プログラム (DFSDDLT0) の参照情報

DFSDDLT0 は IMS アプリケーション・プログラムのテスト・ツールで、制御ステートメント情報に基づいて IMS に呼び出しを発行します。アプリケーション・プログラムとは別に、DL/I 呼び出しの検査およびデバッグが可能です。 IMS サポート言語を使用するものを含むどのような PSB を使用しても、DFSDDLT0 を実行することができます。 DFSDDLT0 は、汎用データベース・ユーティリティー・プログラムとして使用することもできます。

DFSDDLT0 が提供する機能には、以下のものがあります。

- 以下のものを使用して、任意のデータベースに対して有効な任意の DL/I 呼び出しを発行する。
  - セグメント検索引数 (SSA) または PCB、あるいはその両方

重要: PCB を使用する呼び出しでは、PSB で LIST=YES が指定されている必要があります。

  - SSA または AIB、あるいはその両方
- 呼び出しの結果と予期された結果を比較する。選択された PCB フィールドの内容、入出力域に返されたデータ、あるいはその両方が含まれます。
- 比較の結果が等しくなかったなど出力が有効なときのみ、制御ステートメント、呼び出しの結果、および比較の結果を印刷する。
- DL/I 制御ブロック、入出力バッファー・プール、またはバッチ領域全体のダンプを作成する。
- 選択された制御ステートメントを出力ファイルに穿孔して、新規のテスト・データ・セットを作成する。新規のテスト・ケースの構造を単純化します。
- JCL で SYSIN2 DD ステートメントを使用して、複数の入力データ・セットを単一の入力データ・セットにマージする。マージされたステートメントの最終的な順序は、DFSDDLT0 制御ステートメントの 73 桁目から 80 桁目で指定することができます。
- z/OS システム・コンソールにメッセージを送る (応答あり、または応答なし)。
- 各呼び出しを最高 9,999 回繰り返す。

制御ステートメント

DFSDDLT0 は、制御ステートメントを処理してテスト環境を制御します。DFSDDLT0 では、DC 呼び出しと同様に、IMS 全機能データベースおよび高速機能データベースに呼び出しを発行することができます。

DFSDDLT0 制御ステートメントをコーディングする際には、以下の事項に留意してください。

- 各制御ステートメントの 1 桁目を埋める必要があります。 1 桁目がブランクの場合、ステートメント・タイプは、以前のステートメント・タイプがデフォルトになります。 DFSDDLT0 は、以前のステートメント・タイプに対するときと同じように、残りの文字を使用しようとします。
- 予約フィールドを使用すると、無効な出力および予期できない結果が生じる可能性があります。
- ステートメントの継続、特に CALL ステートメントにかかわるステートメントの継続は重要です。
- シーケンス番号は必須ではありませんが、いくつかの DFSDDLT0 機能に関しては大変有効です。
- DFSDDLTO ステートメントのすべてのコードとフィールドは、特に指定されないかぎり、左寄せして残りをブランクで埋めてください。

制御ステートメントの指針

制御ステートメントの順序は、正常な呼び出しを構成するために重要です。予測できない結果が生じることがないよう、以下の指針に従ってください。

- STATUS ステートメントおよび OPTION ステートメントを使用するときは、これらを使用する呼び出しの前のどこかに配置します。

- COMMENT ステートメントの 2 つのタイプは任意ですが、指定するときは記述する呼び出しの前に置きます。
- CALL FUNCTION ステートメントと必須の SSA は、すべて中断することなく連続してコーディングします。
- CALL DATA ステートメントは、最終継続の直後、CALL FUNCTION ステートメントがある場合はその継続の直後に置きます。
- COMPARE ステートメントは任意ですが、最終 CALL (FUNCTION または DATA) ステートメントに続けます。
- CALL FUNCTION ステートメント、CALL DATA ステートメント、COMPARE DATA ステートメント、COMPARE PCB ステートメント、および COMPARE AIB ステートメントが一緒にコーディングされると、呼び出しシーケンスが形成されます。他の DFSDDLT0 制御ステートメントを使用して、呼び出しシーケンスに割り込まないでください。

  例外**:** IGNORE ステートメントだけがこの規則の例外です。
- 入力ストリーム内でのステートメントの位置にかかわらず、任意のステートメントをオーバーライドするには、IGNORE ステートメント (N またはピリオド (.)) を使用します。IGNORE ステートメントは、SYSIN または SYSIN2 のいずれかの入力ストリームで使用できます。

関連資料:

314 ページの『SYSIN DD ステートメント』

314 ページの『SYSIN2 DD ステートメント』

306 ページの『PUNCH CTL ステートメント』

ABEND ステートメント

ABEND ステートメントによって、IMS は異常終了を出して DFSDDLT0 を終了させます。

以下の表は、ABEND ステートメントの形式を示しています。

表 *55. ABEND* ステートメント

桁	機能	コード	説明
1-5	制御ステートメントの識別	ABEND	異常終了 U252 を出す。(OPTION ステートメントに DUMP をコーディングしないとダンプは作成されない。)
6 から 72	予約済み	b	
73 から 80	シーケンスの指示	nnnnnnnn	SYSIN2 ステートメントがオーバーライドするため。

ABEND ステートメントの例

入力ストリームで ABEND を使用してダンプが必要な場合は、OPTION ステートメントに DUMP を指定します。OPTION ステートメントのデフォルトは NODUMP です。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
ABEND                                                              22100010
```

ダンプが作成されます。OPTION ステートメントでダンプを要求しました。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
O DUMP                                                                 22100010
```

ダンプは作成されません。OPTION ステートメントで NODUMP を要求しました。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
O NODUMP                                                               22100010
```

CALL ステートメント

CALL 制御ステートメントは、CALL FUNCTION と CALL DATA の 2 つの部分からなります。

- CALL FUNCTION ステートメントは、DL/I 呼び出し機能、セグメント検索引数 (SSA)、および呼び出しを繰り返す回数を指定する。 SSA は IMS 標準に従ってコーディングします。
- CALL DATA ステートメントを使用して、CALL FUNCTION ステートメントで指定される DL/I 呼び出しに必要なあらゆるデータ (データベース・セグメント、z/OS コマンド、チェックポイント ID) を提供する。

DFSDDLT0 呼び出し機能の例

STAK/END 呼び出し **:** 次の例は、STAK および END 呼び出し機能を示しています。

```
//BATCH.SYSIN DD *                                                     10000700
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
O SNAP= ,ABORT=0                                                       10000800
S 1 1 1 1 1                                                            10001000
L        GU    SEGA    (KEYA     =A300)                                10001100
L   0003 STAK                                                          10001150
WTO THIS IS PART OF THE STAK                                           10001200
T THIS COMMENT IS PART OF THE STAK                                     10001300
L        GN                                                            10001400
L        END                                                           10001450
U THIS COMMENT SHOULD GET PRINTED AFTER THE STAK IS DONE 3 TIMES       10001500
L   0020 GN                                                           10001600
/*
```

SKIP/START 呼び出し **:** 次の例は、SYSIN で STAK および END 呼び出し機能の処理を変更および停止するための、SYSIN2 での SKIP および START 呼び出し機能の使用を示しています。DFSDDLT0 は、SYSIN で GU 呼び出し機能を実行し、SYSIN で STACK、WTO、T コメント、GN、および END の処理をスキップし、COMMENT に進みます。

```
//BATCH.SYSIN DD *                                                     10000700
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
O SNAP= ,ABORT=0                                                       10000800
S 1 1 1 1 1                                                            10001000
L        GU    SEGA    (KEYA     =A300)                                10001100
L   0003 STAK                                                          10001150
WTO THIS IS PART OF THE STAK                                           10001200
T THIS COMMENT IS PART OF THE STAK                                     10001300
L        GN                                                            10001400
L        END                                                           10001450
U THIS COMMENT SHOULD GET PRINTED AFTER THE STAK IS DONE 3 TIMES       10001500
L   0020 GN                                                           10001600
/*
//BATCH.SYSIN2 DD *
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
```

```
L        SKIP                                                          10001150
L        START                                                         10001450
U THIS COMMENT SHOULD REPLACE THE STAK COMMENT                         10001500
U ********THIS COMMENT SHOULD GET PRINTED BECAUSE OF SYSIN2*********  10001650
/*
```

CALL FUNCTION ステートメント

以下の表は、桁番号、機能、コード、および説明などの、CALL FUNCTION ステートメントの形式を提供します。

これは、桁固有の SSA を使用していない場合に適した形式です。

表 *56. CALL FUNCTION* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	**L**	IMS 呼び出しを発行する。
2	予約済み	b	
3	SSA レベル	b	SSA レベル (オプション)。
		n	認められる 16 進文字の範囲は 1 から F。
4	予約済み	b	
5 から 8	反復カウント	b	ブランクの場合は、反復カウントのデフォルトは 1 になる。
		nnnn	'nnnn' は、この呼び出しを繰り返す回数。範囲は 1 から 9999 で、右寄せされる。先行ゼロは含めても含めなくてもよい。
9	予約済み	b	
10 から 13	DL/I 呼び出し機能の識別	b	ブランクの場合は、直前の CALL ステートメントからの機能を使用する。
		xxxx	'xxxx' は DL/I 呼び出し機能。

表 *56. CALL FUNCTION* ステートメント *(*続き*)*

桁	機能	コード	説明
	SSA の継続	CONT	単一の CALL FUNCTION ステートメントには長すぎる SSA の継続標識。先行する CALL FUNCTION ステートメントの 72 桁目に項目が必要。次の CALL ステートメントの 10 から 13 桁目に CONT が入り、SSA は 16 桁目で継続する。
14、15	予約済み	b	
16 から 23、または	SSA 名	**xxxxxxxx**	左寄せする。
16 から 23、または	トークン	**xxxxxxxx**	トークン名 (SETS/ROLS)
16 から 23、または	MOD 名	**xxxxxxxx**	MOD 名 (PURG+ISRT)
16 から 23、または	副次機能	**xxxxxxxx**	ヌル、DBQUERY、FIND、ENVIRON、PROGRAM (INQY)
16 から 19、および	統計タイプ	**xxxx**	DBAS/DBES-OSAM または VBAS/VBES-VSAM (STAT) 。[2]
20 または	統計形式	**x**	F - 定様式、U - 不定様式、S - 要約
16 から 19	SETO ID[1]	**SETx**	x は 1、2、または 3。注で定義されているように、SETO 呼び出しおよび CHNG 呼び出しに指定される。
21 から 24	SETO IOAREA SIZE	**nnnn**	0000 から 8192 の値 8192 より大きい値が指定される場合のデフォルトは 8192 になる。 値が指定されなければ、SETO サイズを指定しないで呼び出しが作成される。

表 *56. CALL FUNCTION* ステートメント (続き)

桁	機能	コード	説明
24 から 71	SSA の剰余		修飾されない SSA はブランクにする。修飾される検索引数は、24 桁目に '*' または '(' のどちらかを持ち、IMS SSA コーディング規則に従う。
72	継続表示桁	b	このステートメントは継続しない。
		x	単独では、それぞれ連続するステートメントの 16 桁目で始まる複数 SSA を示す。次のステートメントの 10 から 13 桁目に CONT があれば、続くステートメントの 16 桁目から始まる、継続された単一 SSA を示す。
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。
25 から 32	OTMA 記述子名	**xxxxxxxx**	8 バイトの文字フィールド (ICAL)。
34 から 39	同期呼び出しが処理されるまでの待ち時間	**nnnnnn**	6 バイトの文字フィールド。範囲は 1 から 999999 (ICAL)。
41-45	入力メッセージ長	**nnnnn**	要求域での入力データの長さ (ICAL)
47 から 51	応答域の長さ	**nnnnn**	出力メッセージの応答域の長さ (ICAL)

表 56. CALL FUNCTION ステートメント (続き)

桁	機能	コード	説明

注記:

1. SETO CALL :

   SETO ID (SET1、SET2、または SET3) が SETO 呼び出しで必要なのは、SETO 呼び出しに返されて、CHNG 呼び出しのオプション・パラメーター TXTU に渡されるテキスト単位アドレスを、DFSDDLT0 が記録する場合です。

   SETO ID が SETO 呼び出しで省略されると、DFSDDLT0 は返されたデータを記録しないので、CHNG 呼び出しでデータを参照することができなくなります。

   CHNG CALL :

   SETO ID (SET1、SET2、または SET3) が CHNG 呼び出しで必要なのは、SETO 呼び出しに返される SETO ID 入出力域のアドレスを DFSDDLT0 が配置する場合です。テキスト単位の SETO 呼び出しは、SETO ID を持つ SETO 呼び出しに返されます。CHNG 呼び出しと一致するこの SETO ID は、CHNG 呼び出しのオプション・パラメーター "TXTU=ADDR" フィールドに入ります。

   SETO ID が CHNG 呼び出しに指定されると、DFSDDLT0 は同じ SETO ID を使用して、SETO 呼び出しに返される該当するテキスト単位のアドレスを移動します。

   OPTION ステートメント・パラメーターの TXTU を、TXTU=xxxx とコーディングしてください。ここで、xxxx はブランクではない有効な文字です。単一引用符は有効な文字ではありません。

   xxxx に推奨する値は、SET1、SET2、または SET3 です。DFSDDLT0 はこの値を使用しません。
2. STAT は、プロダクト・センシティブ・プログラミング・インターフェースです。

次の情報は、異なるタイプの継続に適用されます。

- 3 桁目の SSA レベルは、通常はブランクである。ブランクであれば、最初の CALL FUNCTION ステートメントが SSA 1 を埋め、続く各 CALL FUNCTION ステートメントが次に低い SSA を埋めます。 3 桁目がブランクでなければ、ステートメントはそのレベルで SSA を埋め、続く CALL FUNCTION ステートメントが次に低い SSA を埋めます。
- 5 から 8 桁目は通常ブランクであるが、使用するときは右寄せする。反復呼び出し機能で指定されたように、同じ呼び出しが繰り返されます。
- 10 から 13 桁目には、DL/I 呼び出し機能が入る。呼び出し機能は、呼び出しに複数 SSA があるときの、最初の CALL FUNCTION ステートメントにのみ必要です。ブランクのままであれば、直前の CALL FUNCTION ステートメントの呼び出し機能が使用されます。
- 16 から 23 桁目には、呼び出しで SSA を使用する場合、セグメント名が入る。
- DL/I 呼び出しに複数 SSA が含まれる場合、ステートメントの 72 桁目にブランク以外の文字があり、次の SSA を次のステートメントの 16 桁目で始める必要がある。 2 番目の SSA から最終 SSA に関しては、1 桁目および 10 から 13 桁目にあるデータはブランクです。

  制約事項: ISRT 呼び出しでは、最終 SSA は修飾や継続のないセグメント名しか持ちません。

- フィールド値が 71 桁を超える場合は、72 桁目にブランク以外の文字を入れます。(この文字はフィールド値の一部分としては読み取られず、継続文字としてのみ読み取られます。) 次のステートメントでは、10 から 13 桁目にキーワード CONT を挿入し、16 桁目でフィールド値を開始して継続します。
- フィールド値の最大長は 256 バイト、SSA の最大サイズは 290 バイト、このプログラムに関する SSA の最大数は 15 で、IMS の限界と同じ。
- CALL FUNCTION ステートメントの 5 から 8 桁目に呼び出しの反復カウントが含まれる場合は、最初に GB 状況コードを検出しなければ、そのカウントに達すると呼び出しが終了する。

関連資料:

『桁固有の SSA のある CALL FUNCTION ステートメント』

桁固有の **SSA** のある **CALL FUNCTION** ステートメント**:**

この形式では、SSA のフィールドの間にブランクが存在します。24、34、および 37 桁目にはブランクが必要です。

コマンド・コードは使用できません。以下の表は、桁固有の SSA のある CALL FUNCTION ステートメントの形式を示しています。

表 *57. CALL FUNCTION* ステートメント *(*桁固有の *SSA)*

桁	機能	コード	説明
1	制御ステートメントの識別	**L**	呼び出しステートメント (10 から 13 桁目を参照)
2	予約済み	b	
3	予約済み	b	
4	予約済み	b	
5 から 8	反復カウント	b	ブランクの場合は、反復カウントのデフォルトは 1 になる。
		nnnn	'nnnn' は、この呼び出しを繰り返す回数。範囲は 1 から 9999 で、右寄せするが先行ゼロを含む必要はない。
10 から 13	DL/I 呼び出し機能の識別	b	ブランクの場合は、直前の CALL ステートメントからの機能を使用する。
		xxxx	'xxxx' は DL/I 呼び出し機能。
		CONT	単一の CALL FUNCTION ステートメントには長すぎる SSA の継続標識。先行する CALL FUNCTION ステートメントの 72 桁目に、ブランク以外の文字が必要。次の CALL ステートメントの 10 から 13 桁目に CONT があり、SSA は 16 桁目に続く必要がある。
14、15	予約済み	b	
16 から 23	SSA 名	**s-name**	呼び出しが SSA を持つ場合に必要。
24	予約済み	b	区切りフィールド
25	SSA の開始文字	**(**	セグメントが修飾されている場合に必要。
26 から 33	SSA フィールド名	**f-name**	セグメントが修飾されている場合に必要。
34	予約済み	b	区切りフィールド

表 57. *CALL FUNCTION* ステートメント (桁固有の *SSA*) (続き)

桁	機能	コード	説明
35、36	DL/I 呼び出し演算子 (複数も可)	**name**	セグメントが修飾されている場合に必要。
37	予約済み	b	区切りフィールド
38 から nn	フィールド値	**nnnnn**	セグメントが修飾されている場合に必要。 **注:** フィールド値では、'5D' および ')' を使用してはならない。
nn+1	SSA の終了文字	**)**	セグメントが修飾されている場合に必要。
72	継続表示桁	b	このステートメントは継続しない。
		x	単独では、それぞれ連続するステートメントの 16 桁目で始まる複数 SSA を示す。次のステートメントの 10 から 13 桁目に CONT があれば、続くステートメントの 16 桁目から始まる、継続された単一 SSA を示す。
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

CALL FUNCTION ステートメントに複数の SSA がある場合は、ステートメントは 72 桁目にブランク以外の文字を持ち、次の SSA は次のステートメントの 16 桁目で開始されなければなりません。フィールド値が 71 桁を超える場合は、72 桁目にブランク以外の文字を入れます。次のステートメントでは、10 から 13 桁目にキーワード CONT を挿入し、16 桁目でフィールド値を開始して継続します。フィールド値の最大長は 256 バイト、SSA の最大サイズは 290 バイト、このプログラムに関する SSA の最大数は 15 で、IMS の限界と同じ。

関連資料:

273 ページの『CALL FUNCTION ステートメント』

CALL DATA ステートメント

CALL DATA ステートメントは、通常は呼び出し機能のそのタイプでは入出力域で提供される情報を、IMS に提供します。

CALL DATA ステートメントは、最後の CALL FUNCTION ステートメントに続けてください。 1 桁目に L を、10 から 13 桁目にキーワード DATA を入力し、16 から 71 桁目には必要なデータをコーディングします。 72 桁目にブランク以外の文字を入力すれば、データを継続できます。継続ステートメントでは、1 桁目から 15 桁目をブランクにし、16 桁目からデータを再開します。以下の表は、CALL DATA ステートメントの形式を示しています。

表 58. *CALL DATA* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	**L**	CALL DATA ステートメント
2	セグメント長の増加	**K**	5 から 8 桁目に定義されるデータの長さに 2500 バイトを加える。

表 *58. CALL DATA* ステートメント *(*続き*)*

桁	機能	コード	説明
3	残りの入出力標識の伝搬	**P**	50 バイト (16 から 65 桁目) を残りの入出力域を通じて伝搬する。 **注:** これは最終の DATA ステートメントでなければならず、継続しない。
4	形式オプション	**b**	可変長セグメントではない。
		V	可変長セグメントだけを記述する最初のステートメント、または複数の可変長セグメントの最初のステートメントについては、LL フィールドがセグメント・データの前に追加される。
		M	2 番目の可変長セグメントから最終可変長セグメントまでを記述するステートメントについては、セグメント・データの前に LL フィールドが追加される。
		P	パス呼び出しの固定長セグメントを記述する最初のステートメントに対して。
		Z	メッセージ・セグメントについては、データの前に LLZZ フィールドが追加される。
		U	GSAM レコードに関する不定形式レコード形式。ISRT についてのセグメント長は、DB PCB キーのフィードバック域に入れられる。
5 から 8	セグメントのデータ長	**nnnn**	この値は右寄せするが、先行ゼロは含めなくてよい。長さを指定しない場合は、DFSDDLT0 は、読み取った DATA ステートメントの数に 56 を乗算して得た数を使用して長さを得る。
9	予約済み	**b**	
10 から 13	CALL DATA ステートメントの識別	**DATA**	これを DATA ステートメントとして識別する。
14、15	予約済み	**b**	
16 から 71 または	データ域	***xxxx***	入出力域に入るデータ
16 から 23 または	チェックポイント ID		チェックポイント ID (SYNC)
16 から 23 または	宛先名		宛先名 (CHNG)
16	DEQ オプション		DEQ オプション (A、B、C、D、E、F、G、H、I、または J)
72	継続表示桁	**b**	このセグメントがもう継続していない場合
		x	このセグメントまたは他のセグメントにさらにデータがある場合
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

可変長セグメントを挿入するか、あるいは CHKP または LOG 呼び出しのための可変長データを組み込むときには、次のようにしてください。

- CALL DATA ステートメントの 4 桁目で、V または M を使用する。
- 1 つの可変長セグメントだけが処理される場合は、V を使用する。
- 5 から 8 桁目には先行ゼロを使用し、右寄せでデータの長さを入力する。この値は 2 進数に変換され、セグメント・データの最初の 2 バイトになります。
- 72 桁目にブランク以外の文字を入力して、CALL DATA ステートメントを次の CALL DATA ステートメントに継続することができる。後続のステートメントに関しては、1 桁目から 15 桁目をブランクのままにし、16 桁目からデータを開始します。

最初のセグメントに、複数の可変長セグメントが必要な場合 (すなわち、論理子セグメントと論理親セグメントの連結で、両方とも可変長)、次のようにします。

- 4 桁目に V を入力する。
- 5 から 8 桁目に最初のセグメントの長さを入力する。
- 最初のセグメントが 56 バイトより長い場合は、可変長セグメントの挿入の説明に従ってデータを継続する。

  例外:
  - このセグメントにデータを入れるための最終 CALL DATA ステートメントには、72 桁目にブランク以外の文字を入れる必要がある。
  - 次の CALL DATA ステートメントは次の可変長ステートメントに適用され、4 桁目に M を、5 から 8 桁目にセグメント長を入れる必要がある。

このようにして、可変長セグメントをいくつでも連結することができます。 M または V、および長さ (可変長セグメントでデータを始める CALL DATA ステートメント内でのみ) を入力します。

プログラムがパス呼び出しを通じて挿入や置換を行っているときは、次のようにします。

- 4 桁目に P を入力して、セグメントがユーザー入出力域に占める長さとして使用されるよう、長さフィールドを指定する。
- 可変長セグメントと固定長セグメントの両方を含むパス呼び出し内の、固定長セグメントである CALL DATA ステートメントの最初のステートメントで、P を使用するだけでよい。
- V、M、および P を、連続する CALL DATA ステートメントで使用することができる。

INIT、SETS、ROLS、および LOG 呼び出しでは、次のようにします。

- 入出力域の形式は次のようにする。

  ```
  LLZZuser-data
  ```

  ここで、LL は入出力域内のデータの長さで、LLZZ 部分の長さを含みます。
- プログラムが入出力域でこの形式を使用するには、4 桁目に Z を入力し、5 から 8 桁目にデータの長さを入力する。5 から 8 桁目の長さはデータの長さで、LLZZ の長さの 4 バイトは含みません。

OPTION DATA ステートメント

OPTION DATA ステートメントには、SETO 呼び出しおよび CHNG 呼び出しに必要なオプションが含まれます。

以下の表は、桁数、機能、コード、および説明を含む、OPTION DATA ステートメントの形式を示しています。

表 *59. OPTION DATA* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	**L**	OPTION ステートメント
2 から 9	予約済み	**b**	
10 から 13	識別	**OPT**	これを OPTION ステートメントとして識別する。
		CONT	これをオプション入力の継続として識別する。
14、15	予約済み	**b**	
16 から 71	オプション域	**xxxx**	SETO 呼び出しおよび CHNG 呼び出しに定義されるオプション
72	継続表示桁	**b**	オプションにこれ以上継続がない場合
		x	続くステートメントにまだオプション・データが存在する場合
73 から 80	シーケンス番号	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

FEEDBACK DATA ステートメント

FEEDBACK DATA ステートメントは、フィードバック・データが入る区域を定義します。

FEEDBACK DATA ステートメントはオプションです。ただし、FEEDBACK DATA ステートメントを使用する場合は、OPTION DATA ステートメントが必要です。

以下の表は、桁数、機能、コード、および説明を含む、FEEDBACK DATA ステートメントの形式を示しています。

表 *60. FEEDBACK DATA* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	**L**	FEEDBACK ステートメント
2-3	予約済み	**b**	
4	形式オプション	**b**	フィードバック域は LLZZ を含む。
		Z	フィードバック域の長さが計算され、LLZZ がフィードバック域に加えられる。
5 から 8	フィードバック域の長さ	**nnnn**	この値は右寄せするが、先行ゼロは含めなくてよい。長さを指定しない場合は、DFSDDLT0 が、読み取られた FDBK 入力の数に 56 を乗算した数を使用して長さを得る。
2 から 9	予約済み	**b**	
10 から 13	識別	**FDBK**	これをフィードバック・ステートメントおよびフィードバック・ステートメントの継続として識別する。

表 60. FEEDBACK DATA ステートメント (続き)

桁	機能	コード	説明
14、15	予約済み	**b**	
16 から 71	フィードバック域	**xxxx**	ユーザー事前定義初期設定域を含む。
72	継続表示桁	**b**	フィードバックがこれ以上継続しない場合
		x	続くステートメントにまだフィードバック・データが存在する場合
73 から 80	シーケンス番号	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

DL/I 呼び出し機能

以下の表は、DFSDDLT0 がサポートする DL/I 呼び出し機能と、データ・ステートメントに必要なものを示しています。

表 61. DL/I 呼び出し機能：

呼び出し	AIB サポート	PCB サポート	データ・ステートメント [1]	説明
CHKP	あり	あり	R	チェックポイント
CHNG	あり	あり	R	代替 PCB の変更。
			R	代替 PCB 名のオプション・ステートメントおよびフィードバック・ステートメントのオプションを含む。
CMD	あり	あり	R	IMS コマンドを出す。この呼び出しでは入出力 PCB をデフォルトとする。
DEQ	あり	あり	R	Q コマンド・コードでロックされたセグメントをデキューする。全機能では、デキューするクラスが入っている DATA ステートメントが呼び出しの直後に続く場合、この呼び出しはデフォルトで入出力 PCB を使用する。高速機能では、DEDB PCB に対する呼び出しが出される。DEQ 呼び出しの直後に DATA ステートメントを指定してはならない。
DLET	あり	あり	O	削除。データ・ステートメントがあれば、それを使用する。なければ、呼び出しが直前の初回のセグメントの取り出し保留 (GHU) からのデータを使用する。
FLD	あり	あり	R	フィールド：FSA を使用する高速機能 MSDB 呼び出し用。この呼び出しは MSDB のみを参照する。 FSA が複数ある場合は、34 桁目にブランク以外の文字を置き、次のステートメントの 16 から 34 桁目に次の FSA を置く。FSA を含む DATA ステートメントが必要である。
GCMD	あり	あり	N	コマンド応答の取り出し。この呼び出しでは入出力 PCB をデフォルトとする。
GHN	あり	あり	O[2]	後続セグメントの取り出し保留。
GHNP	あり	あり	O[2]	親における後続セグメントの取り出し保留。
GHU	あり	あり	O[2]	初回のセグメントの取り出し保留。

表 61. DL/I 呼び出し機能 (続き):

呼び出し	AIB サポート	PCB サポート	データ・ステートメント [1]	説明
GMSG[3]	あり	なし	R	メッセージ取り出しは、自動化操作プログラム (AO) アプリケーション・プログラム使用され、AO 出口ルーチン (DFSAOE00 または別の AOIE タイプのユーザー出口) からメッセージを取り出す。データを戻す区域を準備するために、DATA ステートメントが必要。区域には、返されるデータが入るだけの大きさが必要。
GN	あり	あり	O[2]	後続セグメントの取り出し。
GNP	あり	あり	O[2]	親における後続セグメントの取り出し。
GU	あり	あり	O[2]	初回のセグメントの取り出し。
GUR	あり	なし	R	IMS カタログ・データベースから固有レコードを取得します。**ヒント:** OPTION ステートメントで LCASE=C を指定し、XML インスタンス文書として返された記録をより読みやすくします。
ICAL	あり	なし	R	IMS 呼び出しにより、IMS TM 環境で稼働するアプリケーション・プログラムは、z/OS または分散環境で稼働する非 IMS アプリケーション・プログラムやサービスに、データやサービスに対する同期要求を送信できるようになる。
ICMD[3]	あり	なし	R	この「コマンド発行」によって、自動化操作プログラム (AO) アプリケーション・プログラムは、IMS コマンドを出して、コマンド応答の最初のコマンド応答セグメントを取り出せるようになる。データを戻す区域を準備し、入力コマンドを入れるために、DATA ステートメントが必要。区域には、返されるデータが入るだけの大きさが必要。
INIT	あり	あり	R	初期設定。この呼び出しでは入出力 PCB をデフォルトとする。DATA ステートメントが必要。LLZZ 形式を使用。
INQY[3]	あり	なし	R	AIB および ENVIRON 副次機能を使用して、環境情報を要求する。データを戻す区域を準備するために、DATA ステートメントが必要。区域には、返されるデータが入るだけの大きさが必要。
			R	AIB および DBQUERY 副次機能を使用して、データベース情報を要求する。INIT DBQUERY 呼び出しと同等。データを戻す区域を準備するために、DATA ステートメントが必要。区域には、返されるデータが入るだけの大きさが必要。
ISRT	あり	あり		挿入。
			R	DB PCB、DATA ステートメントが必要。
			O	MOD 名を持つ入出力域を使用する入出力 PCB (16 から 23 桁目にある場合)
			R	代替 PCB
LOG	あり	あり	R	ログ・システム要求。この呼び出しでは入出力 PCB をデフォルトとする。DATA ステートメントが必要で、2 つの方法のいずれかで指定される。
POS	あり	あり	N	位置 : セグメントの場所を判別する DEDB 用。この呼び出しは DEDB のみを参照する。

表 *61. DL/I* 呼び出し機能 *(*続き*)*:

呼び出し	**AIB** サポート	**PCB** サポート	データ・ステートメント [1]	説明
PURG	あり	あり		パージ。
			R	この呼び出しでは、入出力 PCB をデフォルトとする。16 桁目がブランクでなければ、MOD (メッセージ出力記述子) 名が使用され、DATA ステートメントが必要である。
			O	16 桁目がブランクであれば、DATA ステートメントはオプションになる。
RCMD[3]	あり	なし	R	コマンド検索を使用すると、ICMD 呼び出しの後に自動化操作プログラム (AO) アプリケーション・プログラムで、コマンド応答セグメントの 2 番目以降のセグメントを取り出すことができる。データを戻す区域を準備するために、DATA ステートメントが必要。区域には、返されるデータが入るだけの大きさが必要。
REPL	あり	あり	R	置換 : この呼び出しは DB PCB のみを参照する。DATA ステートメントが必要。
RLSE	あり	あり	N	無変更データでアプリケーションが保持しているすべてのロックを解除する。
ROLB	あり	あり	O	ロールバック呼び出し
ROLL	なし	あり	O	ロールバック呼び出し、および U778 異常終了を出す。
ROLS	あり	あり	O	更新のバックアウト、および 3303 異常終了を出す。入出力 PCB を使用する。SETS 呼び出し機能とともに使用できる。4 番目のパラメーターとして入出力域およびトークンを使用して ROLS を出すには、CALL ステートメントの 16 桁目で 4 バイトのトークンを指定する。16 から 19 桁目をブランクにしておくと、入出力域およびトークンを使用せずにこの呼び出しが出される。 (現行 DB PCB を使用して ROLS を出すには、ROLX を使用する。)
ROLX	あり	あり	O	DB PCB に対するロール呼び出し (DFSDDLT0 呼び出し機能)。この呼び出しは DB PCB にロールバック呼び出しを要求するときに使用され、DL/I 呼び出しを作成する際に ROLS 呼び出しに変更される。
SETO	あり	あり	N	オプションの設定。OPTION ステートメントは必須。FEEDBACK ステートメントはオプション。
SETS/SETU	あり	あり	O	中間バックアウト・ポイントの作成または取り消し。入出力 PCB を使用する。 4 番目のパラメーターとして入出力域およびトークンを使用して SETS を出すには、CALL ステートメントの 16 桁目で 4 バイトのトークンを指定し、DATA ステートメントを組み込む。16 から 19 桁目をブランクにしておくと、入出力域およびトークンを使用せずにこの呼び出しが出される。

表 *61. DL/I* 呼び出し機能 *(*続き*)*:

呼び出し	AIB サポート	PCB サポート	データ・ステートメント [1]	説明
SNAP[4]	あり	あり	O	スナップ・ダンプの識別および宛先を設定する。CALL DATA ステートメントなしで SNAP 呼び出しが出される場合は、入出力バッファー・プールおよび制御ブロックのスナップが取られ、オンラインであれば LOG に、バッチであれば PRINTDD DCB に送られる。SNAP ID のデフォルトは SNAPxxxx で、xxxx は 0000 から開始され、DATA ステートメントのない SNAP 呼び出しのたびに 1 ずつ増分される。SNAP オプションのデフォルトは YYYN。CALL DATA ステートメントが使用される場合は、16 から 23 桁目で SNAP 宛先を、24 から 31 桁目で SNAP 識別を、32 から 35 桁目で SNAP オプションを指定する。 SNAP オプションのコーディングでは、スナップ・ダンプを要求するときは'Y' を、防止するときは 'N' を使用する。32 桁目が入出力バッファー・プールをスナップし、33 桁目と 34 桁目が IMS 制御ブロックをスナップし、35 桁目が領域全体をスナップする。 SNAP 呼び出し機能は、全機能データベース PCB にだけサポートされる。
STAT[5]	あり	あり	O	STAT 呼び出しは IMS システムの統計を検索する。この呼び出しは全機能 DB PCB のみを参照する必要がある。統計タイプは CALL FUNCTION ステートメントの 16 から 19 桁目にコーディングされる。 **DBAS** OSAM データベース・バッファー・プール統計用 **VBAS** VSAM データベース・サブプール統計用 統計形式は CALL FUNCTION ステートメントの 20 桁目にコーディングされる。 **F** 全統計を定様式化するために F を指定する場合、入出力域は最低 360 バイト必要。 **U** 全統計を不定様式化するために U を指定する場合、入出力域は最低 72 バイト必要。 **S** 統計の要約を定様式化するために S を指定する場合、入出力域は最低 120 バイト必要。
SYNC	あり	あり	R	同期
XRST	あり	あり	R	再始動

注:

1. R = 必須、O = オプション、N = なし
2. AIB インターフェースにはデータ・ステートメントが必要です。
3. AIB インターフェースでのみ有効です。
4. SNAP は、プロダクト・センシティブ・プログラミング・インターフェースです。
5. STAT は、プロダクト・センシティブ・プログラミング・インターフェースです。

DL/I 呼び出し機能の例

以下の例は、DL/I 呼び出し機能の使用方法を示しています。

基本 **CHKP** 呼び出し **:** CALL FUNCTION ステートメントを使用して CHKP 機能を、CALL DATA ステートメントを使用してチェックポイント ID を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        CHKP                                                          10101400
L        DATA  TESTCKPT
```

2 つのデータ域を使用する、チェックポイントへのシンボリック **CHKP** 呼び出し **:** CALL FUNCTION ステートメントを使用して CHKP 機能を、CALL DATA ステートメントを使用してチェックポイント ID データを、2 つの CALL DATA ステートメントを使用してチェックポイントが必要なデータを含めます。

シンボリック CHKP 呼び出しを使用するときは、XRST 呼び出しも使用する必要があります。 CHKP 呼び出しは、シンボリック CHKP に対する XRST 呼び出しに同調するので、シンボリック CHKP 呼び出しを使用するときには、事前の XRST 呼び出しの使用が必要です。

推奨事項**:** アプリケーション・プログラムで、最初の呼び出しとして、XRST 呼び出しを発行してください。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        XRST
L        .
L        .
L        .
L        CHKP
L        DATA  TSTCHKP2                                                 X
L      8 DATA  STRING2-                                                 X
L     16 DATA  STRING2-STRING2-
U EIGHT BYTES OF DATA (STRING2-) IS CHECKPOINTED AND
U SIXTEEN BYTES OF DATA (STRING2-STRING2-) IS CHECKPOINTED ALSO
```

CHNG 呼び出し **:** CALL FUNCTION ステートメントを使用して CHNG 機能を、CALL DATA ステートメントを使用して新規の論理端末名を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        CHNG SET1
L        OPT  IAFP=A1M,PRTO=LLOPTION1,OPTION2,
L        CONT OPTION4
L Z0023  DATA DESTNAME
```

LL は LLOPTION,.....OPTION4 の長さの 16 進数値です。

次の例では、SETO ID SET2 を使用する CHNG ステートメント、OPTION ステートメント、MODNAME を指定した DATA ステートメント、および FDBK ステートメントを使用しています。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        CHNG SET2
L        OPT  IAFP=A1M,TXTU=SET2
L Z0023  DATA DESTNAME
L Z0095  FDBK FEEDBACK AREA
```

CMD 呼び出し **:** CALL FUNCTION ステートメントを使用して CMD 機能を、CALL DATA ステートメントを使用してコマンド・データを含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        CMD
L  ZXXXX DATA  COMMAND DATA

WHERE XXXX = THE LENGTH OF THE COMMAND DATA
```

DEQ 呼び出し **:** 全機能の場合、CALL FUNCTION ステートメントを使用して DEQ 機能を含め、CALL DATA ステートメントを使用して、DEQ 値 (A、B、C、D、E、F、G、H、I、または J) を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       DEQ
L       DATA  A
```

高速機能の場合は、CALL FUNCTION ステートメントを使用して DEQ 機能を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       DEQ
```

DLET 呼び出し **:** CALL FUNCTION ステートメントを使用して DLET 機能を含めます。データ・ステートメントはオプションです。取り出し保留呼び出しと DLET 呼び出しの間に他の PCB へ介入する呼び出しがある場合は、データ・ステートメントを使用して、削除されるセグメントのある入出力域を最新表示してください。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       DLET
```

FLD 呼び出し **:** CALL FUNCTION ステートメントを使用して FLD 機能と ROOTSSA を含め、CALL DATA ステートメントを使用して FSA を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       FLD   ROOTA   (KEYA     =ROOTA)
L       DATA  ???????                                              X
L       DATA
```

GCMD 呼び出し **:** CALL FUNCTION ステートメントを使用して GCMD 機能を含めます。CALL DATA ステートメントは必要ありません。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       GCMD
```

GHN 呼び出し **:** CALL FUNCTION ステートメントを使用して GHN 機能を含めます。CALL DATA ステートメントは必要ありません。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       GHN                                                        10103210
```

GHNP 呼び出し **:** CALL FUNCTION ステートメントを使用して GHNP 機能を含めます。CALL DATA ステートメントは必要ありません。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       GHNP                                                       10103210
```

継続された **SSA** を持つ **GHU** 呼び出し **:** 2 つの CALL FUNCTION ステートメントを使用して単一 SSA を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       GHU   SEGG    (FILLRG  = G131G131G131G131G131G131G131G131G131G*
        CONT  131G131G131G131G131G131G131       )
```

GMSG 呼び出し **:** CALL FUNCTION ステートメントを使用して GMSG 機能を含めます。 CALL DATA ステートメントを使用して、AO 出口ルーチンからのメッセージを検索します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GMSG  TOKEN111 WAITAOI
L  Z0132 DATA
L        GMSG
L  Z0132 DATA
```

GN 呼び出し **:** CALL FUNCTION ステートメントを使用して GN 機能を含めます。CALL DATA ステートメントは必要ありません。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GN                                                          10103210
```

GNP 呼び出し **:** CALL FUNCTION ステートメントを使用して GNP 機能を含めます。CALL DATA ステートメントは必要ありません。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GNP                                                         10103210
```

単一 **SSA** と関係演算子を持つ **GU** 呼び出し **:** CALL FUNCTION ステートメントを使用して GU 機能を含めます。CALL DATA ステートメントは必要ありません。修飾された SSA は 24 桁目から始まり、括弧内に含まれます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU    SEGF    (KEYF    > F131*KEYF    < F400)
```

単一 **SSA** およびブール演算子で複数入力まで拡張される関係演算子を持つ **GU** 呼び出し **:** CALL FUNCTION ステートメントを使用して GU 機能を含め、CALL FUNCTION の追加継続を 3 回入力してブール演算子と継続します。CALL DATA ステートメントは必要ありません。修飾された SSA は 24 桁目から始まり、括弧内に含まれます。このタイプの SSA は、いくつものステートメントにわたって継続できます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU    SEGG    (FILLRG  > G131G131G131G131G131G131G131G131G131G131G*
         CONT  131G131G131G131G131G131G131        &FILLRG  < G400G400G4*
         CONT  00G400G400G400G400G400G400G400G400G400G400G400G400  *
         CONT        )
```

GU パス呼び出し **:** CALL FUNCTION ステートメントを使用して GU 機能を含め、CALL 機能の追加継続を 3 回入力して 2 つの追加 SSA と継続します。CALL DATA ステートメントは必要ありません。 CALL は 24 桁目と 25 桁目でコマンド・コードを使用し、パス呼び出しを構成します。このタイプの呼び出しでは、桁固有の SSA 形式を持ちません。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU    SEGA    *D(KEYA    = A200)                            *
               SEGF    *D(KEYF    = F250)                            *
               SEGG    *D(KEYG    = G251)
```

GUR 呼び出し**:** CALL FUNCTION ステートメントを使用して GUR 機能を含め、DATA ステートメントを使用して、返された XML 文書の出力域の最大サイズを指定します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
O  LCASE=C
S1111 1 1 1   1DFSCAT00      AIB
L  U0001 GUR   HEADER  (RHDRSEQ EQDBD     DBOHIDK5)
L  Z9999 DATA
```

以下の表は、GUR 呼び出しの例における主要な行およびエレメントを示しています。

表 62. 例の説明

例における行	説明
0 LCASE=C	DFSDDLT0 が XML 出力に 16 進表記ではなく文字表記を使用することを指定します。文字表記を使用しないと、返された XML 文書を読むことができません。
S1111 1 1 1 1DFSCAT00 AIB	DFSDDLT0 が AIB インターフェースを使用し、DB PCB 名が DFSCAT00 であることを指定します。これは、システム定義のカタログです。
L U0001 GUR HEADER	IMS が 1 つの GUR 呼び出しを発行することを指定します。SSA には、キー・フィールド RHDRSEQ が含まれます。これは、DBOHIDK5 という名前の DBD を検出するために使用されます。
L Z9999 DATA	DFSDDLT0 が最大データ出力域 (9999 バイト) を使用することを指定します。

GUR 呼び出しが返した XML 文書が大きすぎて DATA ステートメントによって指定された出力域に収まらない場合、GUR 呼び出しを変更して反復できるようにする必要があります。以下の 2 つの方法のいずれかで GUR 呼び出しを反復することができます。

- GUR 呼び出しに対する反復カウント (列 5 から列 8) を呼び出しの反復回数に設定します。これが推奨される方法です。以下の例の U0002 は、IMS が GUR 呼び出しを 2 回発行することを指定します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L  U0002 GUR   HEADER  (RHDRSEQ ==PSB     BMP255  )
L  Z9999 DATA
```

- 複数の GUR 呼び出しを使用します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L  U0001 GUR   HEADER  (RHDRSEQ ==PSB     BMP255  )
L  Z9999 DATA
L  U0001 GUR   HEADER  (RHDRSEQ ==PSB     BMP255  )
L  Z9999 DATA
```

どちらの方法でも同じ結果が生成されます。

ICAL 呼び出し **:** CALL FUNCTION ステートメントを使用して ICAL 機能を含めます。 CALL DATA ステートメントを使用して、IMS アプリケーションから IMS OTMA 記述子に指定されているプログラムに渡すメッセージを含めます。

次の例では、DESCRPTR という宛先に 45 バイトの要求データを指定して同期コールアウト要求メッセージを送信し、500 (5 秒) のタイムアウト値で 100 バイトの応答データが返されるように要求する方法を示しています。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ICAL  SENDRECV DESCRPTR 000500 00045 00100
L        DATA  HELLO OUT THERE. THIS IS A MESSAGE FROM IMS.
```

ICMD 呼び出し**:** CALL FUNCTION ステートメントを使用して ICMD 機能を含めます。 CALL DATA ステートメントを使用してコマンドを含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ICMD
L  Z0132 DATA  /DIS ACTIVE
```

INIT 呼び出し **:** CALL FUNCTION ステートメントを使用して INIT 呼び出しを、CALL DATA ステートメントを使用して INIT 機能の DBQUERY、STATUS GROUPA、または STATUS GROUPB を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        INIT                                                          10103210
L  Z0011 DATA  DBQUERY
```

INQY 呼び出し **:** CALL FUNCTION ステートメントを使用して INQY 呼び出しと、副次機能の DBQUERY または ENVIRON のどちらかを含めます。INIT 呼び出しと同様、副次機能はデータ入力ではなく呼び出し入力にあります。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        INQY  ENVIRON                                                 10103210
L  V0256 DATA                                                          10103211
L                                                                      10103212
```

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        INQY  DBQUERY                                                 10103210
L  V0088 DATA                                                          10103211
L                                                                      10103212
```

ISRT 呼び出し **:** 2 つの CALL FUNCTION ステートメントを使用して複数の SSA を、CALL DATA ステートメントを使用してセグメント・データを含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ISRT  STOCKSEG(NUMFIELD =20011)                              X10103210
               ITEMSSEG                                                10103211
L  V0018 DATA  3002222222222222                                        10103212
```

ただ **1** つの固定長セグメントを持つ **ISRT :** CALL FUNCTION ステートメントを使用して ISRT 機能とセグメント名を、2 つの CALL DATA ステートメントを使用して固定長セグメントを含めます。1 つの固定長セグメントだけを挿入する場合には、4 から 8 桁目はブランクにし、16 から 71 桁目にデータを入れてください。データを継続するには、72 桁目にブランク以外の文字を入れ、次のステートメントの 16 から 71 桁目にデータを続けます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ISRT  JOKESSEG                                                10103210
L        DATA  THEQUICKBLACKDOGJUMPEDONTOTHECRAZYFOXOOPSTHEQUICKBROWNFO*10103211
               XJUMPEDOVERTHELAZYDOGSIR                                10103212
```

ただ **1** つの可変長セグメントを持つ **ISRT :**CALL FUNCTION ステートメントを使用して ISRT 機能とセグメント名を、2 つの CALL DATA ステートメントを使用して可変長セグメントを含めます。ただ 1 つの可変長セグメントが処理されているときは、4 桁目に V を入力し、5 から 8 桁目にセグメント・データの長さを入れます。5 から 8 桁目の長さは 2 進数に変換され、セグメント・データの最初の 2 バイトになります。データを継続するには、72 桁目にブランク以外の文字を入れ、次のステートメントの 16 から 71 桁目にデータを続けます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ISRT  JOKESSEG                                                10103210
L  V0080 DATA  THEQUICKBLACKDOGJUMPEDONTOTHECRAZYFOXOOPSTHEQUICKBROWNFO*10103211
               XJUMPEDOVERTHELAZYDOGSIR                                10103212
```

複数の可変長セグメントを持つ **ISRT :** CALL FUNCTION ステートメントを使用して ISRT 機能とセグメント名を、4 つの CALL DATA ステートメントを使用して可変長セグメントを含めます。最初のセグメントについては、4 桁目に V を入力し、5 から 8 桁目にセグメント・データの長さを入力します。セグメントが 56 バイトより長い場合、72 桁目に非ブランクの文字を書き込み、次のステートメントへデータを継続します。このセグメントのデータを含むための最終ステートメントには、72 桁目にブランク以外の文字を入れます。

次の DATA ステートメントは次の可変長セグメントに適用され、4 桁目に M を入れ、5 から 8 桁に新セグメントの長さを入れ、データを 16 桁から入れる必要があります。この方法で、可変長セグメントをいくつでも連結することができます。72 桁目がブランクであれば、次のステートメントは以下のようになります。

- 1 桁目に L
- 4 桁目に M
- 5 から 8 桁目に新規セグメントの長さ
- 10 から 13 桁目にキーワード DATA
- 16 桁目で開始するデータ

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ISRT  AAAAASEG                                                 10103210
L  V0080 DATA  THEQUICKBLACKDOGJUMPEDONTOTHECRAZYFOXOOPSTHEQUICKBROWNFO*10103211
               XJUMPEDOVERTHELAZYDOGSIR                                *10103212
   M0107 DATA  NOWISTHETIMEFORALLGOODMENTOCOMETOTHEAIDOFTHEIRCOUNTRYNOW*10103213
               ISTHETIMEFORALLGOODMENTOCOMETOTHEAIDOFTHEIRCOUNTRY       10103214
```

PATH 呼び出しに複数セグメントを持つ **ISRT :** CALL FUNCTION ステートメントを使用して ISRT 機能とセグメント名を、7 つの CALL DATA ステートメントを使用して PATH 呼び出しに複数セグメントを含めます。

DFSDDLT0 がパス呼び出しを通じてセグメントを挿入したり置き換えたりする場合は、後続のステートメントで V および P を使用することができます。複数の可変長セグメントをコーディングする際にもこの規則が適用されますが、固定長セグメントでは、DATA ステートメントの 4 桁目に P が必要です。これによって、5 から 8 桁目の長さフィールドがセグメントの長さとして使用され、データは LL フィールドを組み込まずに入出力域で連結されます。

同じセグメントでデータを継続したり、次のステートメントで新規のセグメントを開始したりするための規則は、可変長セグメントに適用されるものと同じです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ISRT  LEV01SEG*D                                             *10103210
               LEV02SEG                                               *10103211
               LEV03SEG                                               *10103212
               LEV04SEG                                                10103213

L  V0080 DATA  THEQUICKBLACKDOGJUMPEDONTOTHECRAZYFOXOOPSTHEQUICKBROWNFO*10103214
               XJUMPEDOVERTHELAZYDOGSIR                                *10103215
   M0107 DATA  NOWISTHETIMEFORALLGOODMENTOCOMETOTHEAIDOFTHEIRCOUNTRYNOW*10103216
               ISTHETIMEFORALLGOODMENTOCOMETOTHEAIDOFTHEIRCOUNTRY      *10103217
L  P0039 DATA  THEQUICKBROWNFOXJUMPEDOVERTHELAZYDOGSIR                 *10103218
L  M0107 DATA  NOWISTHETIMEFORALLGOODMENTOCOMETOTHEAIDOFTHEIRCOUNTRYNOW*10103219
               ISTHETIMEFORALLGOODMENTOCOMETOTHEAIDOFTHEIRCOUNTRY       10103220
```

LLZZ 形式を使用する **LOG** 呼び出し **:** CALL FUNCTION ステートメントを使用して LOG 機能を、CALL DATA ステートメントを使用してログ記録されるデータの LLZZ 形式を含めます。

4 桁目に Z を入れる場合は、レコードの最初のワードは DATA ステートメントにコーディングしません。 5 から 8 桁目に指定する長さは、DATA ステートメントにない LLZZ フィールド用の 4 バイトを含める必要があります。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        LOG                                                          10103210
L  Z0016 DATA  ASEGMENT ONE                                           10103211
```

16 桁目の A がログ・レコード ID になります。

POS 呼び出し **:** CALL FUNCTION ステートメントを使用して POS 機能と SSA を含めます。CALL DATA ステートメントはオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        POS   SEGA    (KEYA     =A300)
```

MODNAME およびデータのある **PURG** 呼び出し **:** CALL FUNCTION ステートメントを使用して PURG 機能と MOD 名を含めます。CALL DATA ステートメントを使用してメッセージ・データを含めます。MOD 名が提供される場合は、DATA ステートメントが必要です。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        PURG  MODNAME1
L        DATA  FIRST SEGMENT OF NEW MESSAGE
```

データはあり **MODNAME** はない **PURG** 呼び出し **:** CALL FUNCTION ステートメントを使用して PURG 機能を含めます。DATA ステートメントはオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        PURG
L        DATA  FIRST SEGMENT OF NEW MESSAGE
```

MODNAME もデータもない **PURG** 呼び出し **:** CALL FUNCTION ステートメントを使用して PURG 機能を含めます。CALL DATA ステートメントはオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        PURG
```

RCMD 呼び出し **:** CALL FUNCTION ステートメントを使用して RCMD 機能を含めます。 CALL DATA ステートメントを使用して、ICMD 呼び出しの結果として生成されたコマンド応答の、2 番目以降のセグメントを取り出します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        RCMD
L  Z0132 DATA
```

REPL 呼び出し **:** CALL FUNCTION ステートメントを使用して REPL 機能を含めます。CALL DATA ステートメントを使用して置換データを含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        REPL
L  V0028 DATA  THIS IS THE REPLACEMENT DATA
```

RLSE 呼び出し **:** CALL FUNCTION ステートメントを使用して RLSE 機能を含めます。

```
|---+---1----+----2----+----3----+----4----+----5
L       RLSE
```

現行メッセージの最初のセグメントの戻りを要求する **ROLB** 呼び出し**:** CALL FUNCTION ステートメントを使用して ROLB 機能を組み込みます。CALL DATA ステートメントを使用して、現行メッセージの最初のセグメントを要求します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       ROLB
L       DATA  THIS WILL BE OVERLAID WITH FIRST SEGMENT OF MESSAGE
```

現行メッセージの最初のセグメントの戻りを要求しない **ROLB** 呼び出し**:** CALL FUNCTION ステートメントを使用して ROLB 機能を組み込みます。CALL DATA ステートメントはオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       ROLB
```

ROLL 呼び出し **:** CALL FUNCTION ステートメントを使用して ROLL 機能を含めます。CALL DATA ステートメントはオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       ROLL
```

トークンを使用する **ROLS** 呼び出し **:** CALL FUNCTION ステートメントを使用して ROLS 機能とトークンを含め、CALL DATA ステートメントを使用して SETS 呼び出しからのデータがオーバーレイするデータ域を提供します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       ROLS TOKEN1

L  Z0046 DATA THIS WILL BE OVERLAID WITH DATA FROM SETS
```

トークンを使用しない **ROLS** 呼び出し **:** CALL FUNCTION ステートメントを使用して ROLS 機能を含めます。CALL DATA ステートメントはオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       ROLS
```

ROLX 呼び出し **:** CALL FUNCTION ステートメントを使用して ROLX 機能を含めます。CALL DATA ステートメントはオプションです。ROLX 機能はトークンのない ROLS 呼び出しとして扱われます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L       ROLX
```

SETO 呼び出し : CALL FUNCTION ステートメントを使用して SETO 機能を含めます。DATA ステートメントはオプションですが、OPTION ステートメントが呼び出しで渡される場合は、DATA ステートメントが必要です。また、FEEDBACK ステートメントが呼び出しで渡される場合は、DATA ステートメントと OPTION ステートメントの両方が必要です。次の例では、OPTION ステートメントおよび SET1 の SETO トークンを使用する SETO ステートメントを使用しています。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        SETO  SET1 5000
L        OPT   PRTO=11OPTION1,OPTION2,
L        CONT  OPTION3,
L        CONT  OPTION4
```

11 は 11OPTION,.....OPTION4 の長さの 16 進数値です。

次の例では、OPTION ステートメントおよび SET1 の SETO トークンを使用する SETO ステートメントを使用しています。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        SETO  SET1 7000
L        OPT   PRTO=11OPTION1,OPTION2,OPTION3,OPTION4
```

11 は 11OPTION,.....OPTION4 の長さの 16 進数値です。

次の例では、OPTION ステートメント、SET2 の SETO トークンおよび FDBK ステートメントを使用する SETO ステートメントを使用しています。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        SETO  SET2 5500
L        OPT   PRTO=11OPTION1,OPTION2,OPTION3,OPTION4
L Z0099  FDBK OPTION ERROR FEEDBACK AREA
```

11 は 11OPTION,.....OPTION4 の長さの 16 進数値です。

トークンのある **SETS** 呼び出し **:** CALL FUNCTION ステートメントを使用して SETS 機能とトークンを含めます。CALL DATA ステートメントを使用して ROLS 呼び出しに返されるデータを提供します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        SETS  TOKEN1

L  Z0033 DATA  RETURN THIS DATA ON THE ROLS CALL
```

トークンのない **SETS** 呼び出し **:** CALL FUNCTION ステートメントを使用して SETS 機能を含めます。CALL DATA ステートメントはオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        SETS
```

このトピックにはプロダクト・センシティブ・プログラミング・インターフェース情報が含まれています。

SNAP 呼び出し **:** CALL FUNCTION ステートメントを使用して SNAP 機能を、CALL DATA ステートメントを使用して SNAP データを含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        SNAP                                                        10103210
L  V0022 DATA  PRINTDD 22222222                                      10103212
```

このトピックにはプロダクト・センシティブ・プログラミング・インターフェース情報が含まれています。

STAT 呼び出し **:** OSAM 統計で必要な STAT 呼び出しはただ 1 つです。VSAM 統計用の STAT 呼び出しは、最も小さいものから始めて一度に 1 つのサブプールしか検索しません。STAT が返す統計の詳細については、「IMS バージョン 14アプリケーション・プログラミング」を参照してください。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        STAT  DBASF
L        STAT  VBASS
L        STAT  VBASS
L        STAT  VBASS
L        STAT  VBASS
```

SYNC 呼び出し **:** CALL FUNCTION ステートメントを使用して SYNC 機能を含めます。CALL DATA ステートメントはオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        SYNC
```

初期 **XRST** 呼び出し **:** CALL FUNCTION ステートメントを使用して XRST 機能を含め、CALL DATA ステートメントを使用して、シンボリック・チェックポイントを使用するプログラムを通常に開始することを示すため、ブランクのチェックポイント ID を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        XRST                                                          10101400
L        DATA
L        CKPT
L        DATA  YOURID01
```

基本 **XRST** 呼び出し **:** CALL FUNCTION ステートメントを使用して XRST 機能を、CALL DATA ステートメントを使用してチェックポイント ID を含めます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        XRST                                                          10101400
L        DATA  TESTCKPT
```

シンボリック **XRST** 呼び出し **:** CALL FUNCTION ステートメントを使用して XRST 機能を、CALL DATA ステートメントを使用して、チェックポイント ID データ、およびデータが返される 1 つ以上の CALL DATA ステートメントを含めます。

XRST 呼び出しは、シンボリック CHKP 呼び出しと共に使用します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        XRST
L        DATA  TSTCHKP2                                                 X
L      8 DATA  OVERLAY2                                                 X
L     16 DATA  OVERLAY2OVERLAY2
U EIGHT BYTES OF DATA (OVERLAY2) SHOULD BE OVERLAID WITH CHECKPOINTED DATA
U SIXTEEN BYTES OF DATA (OVERLAY2OVERLAY2) IS OVERLAID ALSO
```

DFSDDLT0 呼び出し機能

DFSDDLT0 のために DFSDDLT0 呼び出し機能が作成されました。これらの機能は、「有効な」 IMS 呼び出しを表してはおらず、CTL (PUNCH) ステートメントがアクティブであるときに DFSDDLT0 で検出されても、出力として穿孔されません。

以下の表は、CALL FUNCTION ステートメントの特殊な呼び出し機能を示しています。続いて、この特殊な機能の説明および例を示しています。

表 *63. DFSDDLT0* 呼び出し機能を使用する *CALL FUNCTION* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	**L**	CALL ステートメント
2 から 4	予約済み	b	
5 から 8	反復カウント	b	ブランクの場合は、反復カウントのデフォルトは 1 になる。
		nnnn	'nnnn' は、この呼び出しを繰り返す回数。範囲は 1 から 9999 で、右寄せするが先行ゼロを含む必要はない。
9	予約済み	b	
10 から 15	特殊呼び出し機能	**STAKb**	後で実行するため制御ステートメントをスタックする。
		ENDb	スタックを停止し実行を開始する。
		SKIPb	START 機能が検出されるまでステートメントをスキップする。
		START	処理ステートメントを再度開始する。
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

STAK/END (スタック) 制御ステートメント

STAK ステートメントを使用すると、SYSIN から読み取られてメモリーに保留されている一連のステートメントを繰り返すことができます。 STAK ステートメントと END ステートメントの間の制御ステートメントは、すべて読み取られて保管されます。 DFSDDLT0 が END ステートメントを検出すると、STAK ステートメントの 5 から 8 桁目に指定された回数だけ、一連の呼び出しが実行されます。他の STAK 呼び出しに組み込まれている STAK は外側の STAK 呼び出しを異常終了させます。

SKIP/START (スキップ) 制御ステートメント

SKIP ステートメントおよび START ステートメントを使用して、DFSDDLT0 に処理させたくない一群のステートメントを識別します。これらの機能は、通常は SYSIN2 から読み取られ、確立された SYSIN 入力ストリームを一時的に変更します。 DFSDDLT0 は、SKIP ステートメントと START ステートメントの間のすべての制御ステートメントを読み取りますが、これらについては何のアクションも行いません。DFSDDLT0 は、START ステートメントを検出すると、通常は次のステートメントを読み取って処理します。

関連資料:

306 ページの『PUNCH CTL ステートメント』

COMMENT ステートメント

COMMENT ステートメントを使用して、出力データにコメントを印刷します。

条件付きおよび無条件の 2 タイプの COMMENT ステートメントが記述されます。以下の表は、COMMENT ステートメントの形式を示しています。

表 *64. COMMENT* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	**T**	条件付きコメント・ステートメント
		U	無条件コメント・ステートメント
2 から 72	コメント・データ		任意の関連コメント
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

条件付き **COMMENT** ステートメント

呼び出しごとに 5 つまでの条件付き COMMENT ステートメントを使用できます。72 桁目に継続マークは必要ありません。記述しようとする呼び出しの前に、DFSDDLT0 ストリームにステートメントをコーディングします。条件付き COMMENTS は、呼び出しが読み取られて実行されるまで、読み取られて保留されます。 (CALL の後に COMPARE ステートメントが続く場合は、条件付き COMMENTS は、比較が完了するまで保留されます。) STATUS ステートメントの 3 桁目を使用して、条件付きコメントを印刷するかどうかを制御します。DFSDDLT0 は、STATUS ステートメントの順序に従って、ステートメントを印刷します。条件付き COMMENTS、CALL、および COMPARE (複数も可) の順です。それぞれの条件付き COMMENT ステートメントを使用して、時刻および日付も印刷されます。

無条件 **COMMENT** ステートメント

無条件 COMMENT ステートメントはいくつでも使用できます。DFSDDLT0 ストリームの記述しようとする呼び出しの前にコーディングします。それぞれの無条件 COMMENT ステートメントにより、時刻および日付も印刷されます。前掲の表では、桁番号、機能、コード、および説明をリストしています。

COMMENT ステートメントの例

T/U コメント呼び出し **:** 次の例は、T および U コメント呼び出しを示しています。

```
//BATCH.SYSIN DD *                                                    10000700
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
O SNAP= ,ABORT=0                                                      10000800
S 1 1 1 1 1                                                           10001000
L        GU    SEGB    (KEYA     =A400)                               10001100
T THIS COMMENT IS A CONDITIONAL COMMENT FOR THE FIRST GN              10001300
L        GN                                                           10001400
U THIS COMMENT IS AN UNCONDITIONAL COMMENT FOR THE SECOND GN          10001500
L   0020 GN                                                           10001600
/*
```

COMPARE ステートメント

COMPARE ステートメントは、呼び出しの実際の結果と予期されていた結果を比較します。 COMPARE ステートメントの 3 タイプは、COMPARE PCB、COMPARE DATA、および COMPARE AIB です。

COMPARE PCB、COMPARE DATA、および COMPARE AIB ステートメントを使用するときは、次のようにしてください。

- CALL DATA ステートメントか別の COMPARE ステートメントがある場合は、そのどちらかの最終継続の直後に、DFSDDLT0 ストリームに COMPARE ステートメントをコーディングする。
- STATUS ステートメントの 7 桁目に、COMPARE ステートメント用の印刷オプションを指定する。

3 つの COMPARE ステートメントすべてでは、以下のようになります。

- COMPARE に返された条件コードは、比較の結果が等しくなかった場合の合計数を与える。
- 単一の固定長セグメントの長さを与えると、DFSDDLT0 が比較長を使用して比較する。長さ比較オプション (3 桁目) は適用されません。

COMPARE PCB ステートメントを使用して、比較の結果が等しくないときにスナップ・ダンプをとりたい場合は、COMPARE PCB ステートメントで要求します。SNAP ID COMPxxxx を使用するログのスナップ・ダンプは、COMPARE PCB ステートメントの 3 桁目に指定されるスナップ・ダンプ・オプションと同時に出されます。

SNAP ID の数値部分の初期値は 0000 に設定され、比較の結果が等しくないときの SNAP ごとに 1 ずつ増分されます。

COMPARE AIB ステートメント

COMPARE AIB ステートメントはオプションです。これを使用して、IMS が AIB に返す値を比較できます。

以下の表は、COMPARE AIB ステートメントの形式を示しています。

表 *65. COMPARE AIB* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	**E**	COMPARE ステートメント
2	比較オプションの保留	**H**	COMPARE ステートメントを保留する。COMPARE AIB ステートメントの注を参照。
		b	単一の COMPARE ステートメント用に保留条件をリセットする。
3	予約済み	**b**	
4 から 6	AIB 比較	**AIB**	AIB 比較を識別する。
7	予約済み	**b**	
8 から 11	戻りコード	**xxxx**	指定された戻りコードのみを認める。
12	予約済み		
13 から 16	理由コード	**xxxx**	指定された理由コードのみを認める。
17 から 72	予約済み	**b**	b
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

COMPARE AIB ステートメントの注**:** 一連の呼び出しの後で同じ COMPARE AIB を実行するには、2 桁目に H を入れます。H を指定すると、各呼び出しの後に COMPARE ステートメントが実行されます。 H COMPARE ステートメントは、繰り返される呼び出しの同じ状況コードを比較するときに特に便利です。 H COMPARE ステートメントは、他の COMPARE AIB ステートメントが読み取られるまで有効です。

COMPARE DATA ステートメント

COMPARE DATA ステートメントはオプションです。このステートメントは、IMS が返したセグメントと、ステートメント内のデータを比較して、正しいセグメントが検索されたかどうかを検査します。

以下の表は、COMPARE DATA ステートメントの形式を示しています。

表 *66. COMPARE DATA* ステートメント：

桁	機能	コード	説明
1	制御ステートメントの識別	**E**	COMPARE ステートメント
2	予約済み	**b**	
3	長さ比較オプション	**b**	固定長セグメントの場合、またはセグメントの LL フィールドが比較に組み込まれない場合、データのみが比較される。
		L	5 から 8 桁目の長さは 2 進数に変換され、セグメントの LL フィールドと比較される。
4	セグメント長さオプション	**b**	
		V	可変長セグメントのみ、またはパス呼び出しでの複数可変長セグメントの最初の可変長セグメント、または連結された論理子論理親セグメント用。
		M	パス呼び出しの 2 番目以降の可変長セグメント、または連結された論理子論理親セグメント用。
		P	パス呼び出しの固定長セグメント用。
		Z	メッセージ・セグメント用。
5 から 8	長さの比較	**nnnn**	比較に使用される長さ。 (3 桁目に L をコーディングするとき、長さオプション V、M、および P で必要)
9	予約済み	**b**	
10 から 13	ステートメントのタイプの識別	**DATA**	直前の入出力 COMPARE ステートメントからのデータが継続されない場合に、最初の COMPARE ステートメントおよび新規セグメントの最初のセグメントで必要。
14、15	予約済み	**b**	
16 から 71	データのストリング		入出力域のセグメントと比較するデータ
72	継続表示桁	**b**	ブランクであればデータは継続されない。

表 66. *COMPARE DATA* ステートメント (続き) :

桁	機能	コード	説明
		x	ブランクでなければデータは継続され、最大 3840 バイトが、後続ステートメントの 16 から 71 桁目で開始する。
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

注:

- 3 桁目に L をコーディングすると、5 から 8 桁目の値が 2 進数に変換され、返されたセグメントの LL フィールドと比較されます。 3 桁目がブランクのままでセグメントがパス呼び出しにない場合は、5 から 8 桁目の値が比較の長さとして使用されます。
- 4 桁目に V、P、または M をコーディングする場合は、5 から 8 桁目に値を入力します。
- パス呼び出しで比較する場合は、4 桁目に P をコーディングします。5 から 8 桁目の値は、パス呼び出しで使用される固定セグメントの長さと正確に一致させます。
- セグメントの長さを指定する場合、この長さは COMPARE および表示で使用します。長さを指定しない場合、DFSDDLT0 は次のうちの短い方を比較および表示用の長さに使用します。
  - IMS が入出力域に提供するデータの長さ
  - 読み取られた DATA ステートメントの数に 56 掛けた数

COMPARE PCB ステートメント

COMPARE PCB ステートメントはオプションです。これを使用して、IMS が PCB に返す値を比較したり、ブロックやバッファー・プールを印刷できます。

以下の表は、COMPARE PCB ステートメントの形式を示しています。

表 67. *COMPARE PCB* ステートメント :

桁	機能	コード	説明
1	制御ステートメントの識別	**E**	COMPARE ステートメント
2	比較オプションの保留	**H**	比較ステートメントを保留。
		b	単一の COMPARE ステートメント用に保留条件をリセットする。
3	スナップ・ダンプ・オプション (比較の結果が等しくない場合)	b	デフォルトを使用する。(OPTION ステートメントに値をコーディングして、デフォルトを変更したりオプションを停止したりできる。)
		1	完了した入出力バッファー・プール
		2	領域全体 (バッチ領域のみ)
		4	DL/I ブロック
		8	DATA または PCB の比較不一致時にジョブ・ステップを終了する。
		S	0 から 127 個のサブプールを SNAP する。複数の SNAP ダンプ・オプションの要求は、個別の 16 進値を合計すると得られる。 3 桁目にブランク、1 から 9、A から F、または S 以外のものをコーディングすると、SNAP ダンプ・オプションは無視される。
4	拡張 SNAP[1] オプション	b	拡張オプションを無視する。

表 *67. COMPARE PCB* ステートメント *(*続き*)*:

桁	機能	コード	説明
		P	完了したバッファー・プール (バッチ) を SNAP する。
		S	0 から 127 個のサブプールを SNAP する (バッチ)。 1 つの区域が 2 回スナップされることはない。 SNAP オプションは、3 桁目 (SNAP ダンプ・オプション) と 4 桁目 (拡張 SNAP オプション) の組み合わせ。
5、6	セグメント・レベル	**nn**	'nn' は COMPARE PCB のセグメント・レベル。先行ゼロが必要。
7	予約済み	b	
8、9	状況コード	b	ブランクの状況コードのみを認める。
		xx	指定された状況コードのみを認める。
		XX	状況コードを検査しない。
		OK	ブランク、GA、GC、または GK を認める。
10	予約済み	b	
11 から 18	セグメント名 ユーザー識別	**xxxxxxxx**	DB PCB 比較用のセグメント名
			入出力用の論理端末
			ALT PCB の宛先
19	予約済み	b	
20 から 23	キーの長さ	**nnnn**	'nnnn' はフィードバック・キーの長さ。
24 から 71 または	連結キー		DB PCB 比較のための連結キー・フィードバック
24 から 31	ユーザー ID		TP PCB のためのユーザー識別
72	継続表示桁	b	ブランクであれば、キー・フィードバックは継続されない。
		x	ブランクでなければ、キー・フィードバックは継続され、後続のステートメントの 16 から 71 桁目で開始される。
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

注:

1. SNAP は、プロダクト・センシティブ・プログラミング・インターフェースです。

ブランク・フィールドは、状況コード・フィールドを除いて、PCB 内の対応するフィールドとは比較されません (ブランクは有効な状況コードを表します。) 状況コードのブランク、GA、GC、または GK を 1 つのグループとして受け入れるには、8 桁と 9 桁に OK を入れます。状況コードの比較を停止するには、8 桁と 9 桁に XX を入れます。

一連の呼び出しの後で同じ比較を実行するには、2 桁に H を入れます。これによって、各呼び出しの後に COMPARE ステートメントが実行されます。これは、データベースをロードするときのみブランクの状況コードと比較する際に、特に便利です。 H COMPARE ステートメントは、他の COMPARE PCB ステートメントが読み取られるまで有効です。

関連資料:

304 ページの『OPTION ステートメント』

COMPARE DATA および COMPARE PCB ステートメントの例

以下の例は、COMPARE DATA および COMPARE PCB ステートメントの使用方法を示しています。

ブランク状況コードに対する COMPARE PCB ステートメント

COMPARE PCB ステートメントはブランクでコーディングされます。GU についてブランク状況コードを検査します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU                                                              10101100
E                                                                        10101200
```

SSA レベル、状況コード、セグメント名、連結キー長、および連結キーに対する COMPARE PCB ステートメント

COMPARE PCB ステートメントは、SSA レベル、状況コードの OK (ブランク、GA、GC、および GK を含む)、セグメント名の SEGA、連結キー長の 0004、および連結キーの A100 を比較するよう要求します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU
E   01 OK SEGA     0004A100
```

SSA レベル、状況コード、セグメント名、連結キー長、および連結キーに対する COMPARE PCB ステートメント

この COMPARE PCB ステートメントは、COMPARE PCB ステートメントのいずれかのフィールドが PCB 内の対応フィールドと一致しない場合に、3 桁目の値 8 に基づいてジョブ・ステップを終了させます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU                                                              10105100
E 8 01 OK SEGK     0004A100                                              10105200
```

状況コードを比較して比較を保持する COMPARE PCB

この COMPARE PCB ステートメントは OK の状況コード (これにはブランク、GA、GC、および GK を含む) を比較し、次の COMPARE PCB ステートメントまでその比較を保持することを要求します。OK の比較は GU に続く GN で使用され、6 回繰り返す要求を持つ GN でも使用されます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU    SEGA    (KEYA    = A300)                                  20201100
L        GN                                                              20201300
EH     OK                                                                20201400
L   0006 GN                                                              20201500
```

固定長セグメントに対する COMPARE DATA ステートメント

COMPARE DATA ステートメントは、返されるデータを比較するよう要求します。データの 72 バイトが比較されます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU
E        DATA  A100A100A100A100A100A100A100A100A100A100A100A100A100X10102200
E              A100A100A100A100                                        10102300
```

固定長データの 64 バイトに対する COMPARE DATA ステートメント

COMPARE DATA ステートメントは、返されるデータに対してデータの 64 バイトを比較するよう要求します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU                                                          10101600
E   0064 DATA  A100A100A100A100A100A100A100A100A100A100A100A100A100X10101700
E              A100A100B111B111                                        10101800
```

固定長データの 72 バイトに対する COMPARE DATA ステートメント

この COMPARE DATA ステートメントは、返されるデータに対してデータの 72 バイトを比較するよう要求します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        GU                                                          10103900
E LP0072 DATA  A100A100A100A100A100A100A100A100A100A100A100A100A100X10104000
E              A100A100A100A100                                        10104100
```

複数セグメントのデータおよび長さフィールドの可変長データに対する COMPARE DATA ステートメント

COMPARE DATA ステートメントは、セグメント 1 に対して返されたデータとデータの 36 バイトを比較し、セグメント 2 についてはデータの 16 バイトと比較するように要求します。両方のセグメントの長さフィールドを比較します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ISRT  D        (DSS      =  DSS01)                          X38005500
L              DJ       (DJSS     =  DJSS01)                         X38005600
L              QAJAXQAJ                                               38005700
L  V0036 DATA  QSS02QASS02QAJSS01QAJASS97*IQAJA**                     *38005800
L  M0016 DATA  QAJSS01*IQAJ**                                         38005850
L        GHU   D        (DSS      =  DSS01)                          X38006000
               DJ       (DJSS     =  DJSS01)                         X38006100
               QAJAXQAJ (QAJASS   =  QAJASS97)                        38006200
E LV0036 DATA  QSS02QASS02QAJSS01QAJASS97*IQAJA**                     *38006300
E LM0016 DATA  QAJSS01*2QAJ**                                         38006350
```

長さフィールド COMPARE なしの、複数セグメントの可変長データに対する COMPARE DATA ステートメント

COMPARE DATA ステートメントは、セグメント 1 に対して返されたデータとデータの 36 バイトを比較し、セグメント 2 に対してはデータの 16 バイトと比較するよう要求します。いずれのセグメントの長さフィールドも比較されません。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ISRT  D        (DSS      =  DSS01)                          X38005500
L              DJ       (DJSS     =  DJSS01)                         X38005600
L              QAJAXQAJ                                               38005700
L  V0036 DATA  QSS02QASS02QAJSS01QAJASS97*IQAJA**                     *38005800
L  M0016 DATA  QAJSS01*IQAJ**                                         38005850
L        GHU   D        (DSS      =  DSS01)                          X38006000
               DJ       (DJSS     =  DJSS01)                         X38006100
               QAJAXQAJ (QAJASS   =  QAJASS97)                        38006200
E  V0036 DATA  QSS02QASS02QAJSS01QAJASS97*IQAJA**                     *38006300
   M0016 DATA  QAJSS01*2QAJ**                                         38006350
```

複数セグメントの可変長データに対する COMPARE DATA ステートメントと 1 つの長さフィールド COMPARE

COMPARE DATA ステートメントは、セグメント 1 に対して返されたデータとデータの 36 バイトを比較し、セグメント 2 に対してはデータの 16 バイトを比較するよう要求します。セグメント 1 の長さフィールドだけを比較します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
L        ISRT  D        (DSS      =  DSS01)                              X38005500
L              DJ       (DJSS     =  DJSS01)                             X38005600
L              QAJAXQAJ                                                   38005700
L  V0036 DATA  QSS02QASS02QAJSS01QAJASS97*IQAJA**                         *38005800
L  M0016 DATA  QAJSS01*IQAJ**                                             38005850
L        GHU   D        (DSS      =  DSS01)                              X38006000
               DJ       (DJSS     =  DJSS01)                             X38006100
               QAJAXQAJ (QAJASS   =  QAJASS97)                           38006200
E LV0036 DATA  QSS02QASS02QAJSS01QAJASS97*IQAJA**                         *38006300
   M0016 DATA  QAJSS01*2QAJ**                                             38006350
```

IGNORE ステートメント

DFSDDLT0 では、1 桁目に N またはピリオド (.) があるステートメントは、どれも無視します。

N または . (ピリオド) を使用して、SYSIN または SYSIN2 入力ストリーム内のステートメントをコメント化することができます。SYSIN2 入力ストリームに N または . (ピリオド) を使用すると、SYSIN 入力ストリームも同様に無視されます。以下の表は、IGNORE ステートメントの形式を示しています。その次に、ステートメントの例を示しています。

表 *68. IGNORE* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	N または .	IGNORE ステートメント
2 から 72	無視される		
73 から 80	シーケンスの指示	nnnnnnnn	SYSIN2 ステートメントがオーバーライドするため。

N または . を使用する IGNORE ステートメントの例

```
 |---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
 . NOTHING IN THIS AREA WILL BE PROCESSED.  ONLY THE SEQUENCE NUMBER     67101010
 N WILL BE USED IF READ FROM SYSIN2 OR SYSIN.                            67101020
```

関連資料:

314 ページの『SYSIN2 DD ステートメント』

OPTION ステートメント

OPTION ステートメントを使用して、様々なデフォルト・オプションの指定を変更します。

1 つのステートメントに必要なオプションすべてを含めることができない場合には、複数の OPTION ステートメントを使用してください。継続文字は必要ありません。一度オプションを設定すれば、別の OPTION ステートメントを指定して最

初のパラメーターを変更するまでは有効です。以下の表は、OPTION ステートメントの形式を示しています。その次に、例を示しています。

表 *69. OPTION* ステートメント：

桁	機能	コード	説明
1	制御ステートメントの識別	O	OPTION ステートメント (フリー・フォームのパラメーター・フィールド)
2	予約済み	b	b
3 から 72	キーワード・パラメーター：		
	ABORT=	• 0 • 1 から 9999	• ABORT パラメーターをオフにする。 • ジョブを打ち切るまでの比較の結果が等しくない数。初期デフォルトは 5 になる。
	LINECNT=	10 から 99	ページ当たりの印刷行数。ゼロで埋める必要がある。初期デフォルトは 54 になる。
	SNAP[1]	x	比較の結果が等しくないときの SNAP オプションのデフォルト。SNAP オプションをオフにするには、'SNAP=' とコーディングする。このオプションをコーディングしない場合の初期デフォルトは 5 になる。この場合、SNAP 呼び出しを使用して、入出力バッファー・プールと DL/I ブロックのダンプが作成される。
	DUMP/NODUMP		ジョブが異常終了するときに、ダンプを作成する/しない。デフォルトは NODUMP になる。
	LCASE=	• H • C	• 小文字の 16 進数表示。これは初期デフォルト。 • 小文字の文字表示。
	STATCD/NOSTATCD		ブランクまたは GA 状況コードを受け取らない内部的なジョブの終わり STAT 呼び出しに関して、エラー・メッセージを出すか否かを指定する。NOSTATCD がデフォルト。
	ABU249/NOABU249		バッチ環境における内部的なジョブの終わり STAT 呼び出しに関する無効な状況コードが戻されたときに、DFSDDLT0 ABENDU0249 を出すか否かを指定する。NOABU249 がデフォルト。
73 から 80	シーケンスの指示	nnnnnnnn	SYSIN2 ステートメントがオーバーライドするため。

注:

1. SNAP は、プロダクト・センシティブ・プログラミング・インターフェースです。

OPTION ステートメント・パラメーターは、コンマで区切ることができます。

OPTION 制御ステートメントの例

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
O ABORT=5,DUMP,LINECNT=54,SPA=4096,SNAP=5                              67101010
```

関連資料:

300 ページの『COMPARE PCB ステートメント』

PUNCH CTL ステートメント

PUNCH CTL ステートメントを使用すると、COMPARE PCB ステートメント、COMPARE DATA ステートメント、COMPARE AIB ステートメント、他の制御ステートメント、またはこれらのステートメントの組み合わせからなる出力データ・セットを作成することができます。

以下の表は、PUNCH CTL ステートメントの形式とキーワード・パラメーターを示しています。

表 *70. PUNCH CTL* ステートメント

桁	機能	コード	説明
1-3	制御ステートメントの識別	**CTL**	PUNCH ステートメント
4 から 9	予約済み	b	
10 から 13	穿孔制御	**PUNC**	穿孔を開始 (デフォルト値ではない)。
		NPUN	穿孔を停止 (デフォルト値)。
14、15	予約済み	b	
16 から 72	キーワード・パラメーター：		
	OTHER		下記以外の入力制御ステートメントをすべて複製する。 • CTL (PUNCH) ステートメント • N または . (IGNORE) ステートメント • COMPARE ステートメント • 機能が SKIP および START の CALL ステートメント。 SKIP CALL と START CALL の間にあるどのような制御ステートメントも穿孔されない。 • 機能が STAK および END の CALL ステートメント。 STAK CALL と END CALL の間にある制御ステートメントは保管され、STAK CALL で指示される回数だけ穿孔される。

表 *70. PUNCH CTL* ステートメント *(*続き*)*

桁	機能	コード	説明
	DATAL		入出力域に返されるデータをすべて使用して、全データの COMPARE を作成する。必要に応じて、複数の COMPARE ステートメントおよび継続が作成される。
	DATAS		入出力域に返される最初の 56 バイトだけを使用して、単一データの COMPARE ステートメントを作成する。
	PCBL		PCB に返される完了キー・フィードバック域を使用して、全 PCB COMPARE を作成する。必要に応じて、複数の COMPARE ステートメントおよび継続が作成される。
	PCBS		PCB に返されるキー・フィードバック域の最初の 48 バイトだけを使用して、単一 PCB COMPARE ステートメントを作成する。
	SYNC/NOSYNC		STAK にある間に既存の STAK よりも前に GB 状況コードが高速機能呼び出しに返された場合、この機能が SYNC を出すか否かを指定する。
	START=		00000001 から 99999999 穿孔ステートメントに使用される開始シーケンス番号。8 バイトの数値をコーディングする必要がある。
	INCR=		1 から 9999 この値によって、各穿孔ステートメントのシーケンス番号を増分する。先行ゼロは必要ない。
	AIB		AIB COMPARE ステートメントを作成する。
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

単一 DFSDDLT0 入力ストリームの処理中に穿孔制御オプションを変更するには、必ず PUNC と NPUN を対にして PUNCH CTL ステートメントを使用します。

PUNCH CTL ステートメントを使用するには、次のような方法があります。

1. 新規のテスト用に CALL ステートメントだけをコーディングする。COMPARE ステートメントはコーディングしません。
2. 各呼び出しが正しく実行されたかを確認する。
3. DFSDDLT0 で適切な COMPARE ステートメントを組み合わせるために、もう一度 PUNCH CTL ステートメントを実行し、後続のレグレッション・テストで入力として使用できる新規の出力データ・セットを作成する。

既存のデータベースのセグメントを変更する場合にも、PUNCH CTL を使用することができます。この制御ステートメントを使用することにより、手作業で COMPARE ステートメントを変更するよりも、正確な COMPARE ステートメントを持つ新規のテスト・データ・セットが DFSDDLT0 で作成されるようになります。

CTL ステートメントのパラメーターは、前掲の表に示された長さと同じ長さにし、コンマで区切る必要があります。

PUNCH CTL ステートメントの例

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
CTL      PUNC  PCBS,DATAS,OTHER,START=00000010,INCR=0010               33212010
CTL      NPUN                                                         33212020
```

出力データ・セットの DD ステートメントのラベルは PUNCHDD です。データ・セットは、LRECL=80 の固定ブロックです。 DD ステートメントで指定されるブロック・サイズが使用されます。指定されなければ、ブロック・サイズは 80 に設定されます。プログラムで PUNCHDD をオープンできない場合は、DFSDDLT0 が異常終了 251 を出します。

すべてのパラメーターに関する PUNCH CTL ステートメントの例

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
CTL      PUNC  OTHER,DATAL,PCBL,START=00000001,INCR=1000,AIB           33212010
```

関連資料:

295 ページの『DFSDDLT0 呼び出し機能』

270 ページの『制御ステートメント』

STATUS ステートメント

STATUS ステートメントを使用して、印刷オプションを設定し、後続の呼び出しが出される PCB を指定します。

以下の表は、STATUS ステートメントの形式を示しています。

表 *71. STATUS* ステートメント

桁	機能	コード	説明
1	制御ステートメントの識別	**S**	STATUS ステートメント
2	出力装置オプション	**b**	DL/I 領域では PRINTDD を、MPP 領域では入出力 PCB を使用する。

表 *71. STATUS* ステートメント *(*続き*)*

桁	機能	コード	説明
		1	DD ステートメントが提供されれば、MPP 領域で PRINTDD を使用する。それ以外は入出力 PCB を使用する。
		A	1 と同様、さらに、この STATUS ステートメントの他のフィールドをすべて無視する。
3	コメント印刷オプション	**b**	印刷しない。
		1	各呼び出しを印刷する。
		2	比較の結果が等しくない場合だけ印刷する。
4	AIB 印刷オプション	**b**	印刷しない。
		1	各呼び出しを印刷する。
		2	比較の結果が等しくない場合だけ印刷する。
5	呼び出し印刷オプション	**b**	印刷しない。
		1	各呼び出しを印刷する。
		2	比較の結果が等しくない場合だけ印刷する。
6	予約済み	**b**	
7	比較印刷オプション	**b**	印刷しない。
		1	各呼び出しを印刷する。
		2	比較の結果が等しくない場合だけ印刷する。
8	予約済み	**b**	
9	PCB 印刷オプション	**b**	印刷しない。
		1	各呼び出しを印刷する。
		2	比較の結果が等しくない場合だけ印刷する。
10	予約済み	**b**	
11	セグメント印刷オプション	**b**	印刷しない。
		1	各呼び出しを印刷する。
		2	比較の結果が等しくない場合だけ印刷する。
12	タスクと実時間の設定	**b**	時間を設定しない。
		1	各呼び出しの時間を設定する。
		2	比較の結果が等しくなければ、各呼び出しの時間を設定する。
13、14	予約済み	**b**	
15	PCB 選択オプション	**1**	16 から 23 桁目で渡される PCB 名 (オプション 1 を使用)
		2	16 から 23 桁目に渡される DBD 名 (オプション 2 を使用)
		3	16 から 23 桁目に渡される相対 DB PCB (オプション 3 を使用)
		4	16 から 23 桁目に渡される相対 PCB (オプション 4 を使用)
		5	16 から 23 桁目に渡される $LISTALL (オプション 5 を使用)

表 *71. STATUS* ステートメント (続き)

桁	機能	コード	説明
		b	15 桁目がブランクであれば、16 から 23 桁目の内容に基づいて、DFSDDLT0 がオプション 2 から 5 を選択する。
オプション 1 16 から 23	PCB 選択 PCB 名	英字	この桁には、PSBGEN での PCB のラベルの名前か、PSBGEN 時に PCB 用の PCBNAME= オペランドに指定される名前を入れる必要がある。
オプション 2 16 から 23	PCB 選択 DBD 名	**b** 英字	デフォルトの PCB は PSB 内の最初のデータベース PCB になる。 16 から 23 桁目がブランクであれば、現行 PCB が使用される。DBD 名が指定されると、これが PSB 内のデータベース DBD の名前になる。
オプション 3 16 から 18 19 から 23	PCB 選択 相対位置 (PSB 内の PCB の)	**b** 数値	16 から 18 桁目がブランクである場合、このフィールドの桁 (19 から 23) が、PSB 内の DB PCB の相対番号として解釈される。この数は 23 桁目で右寄せしなければならないが、先行ゼロは必要ない。
オプション 4 16 から 18 19 から 23	PCB 選択 入出力 PCB 相対位置 (PSB 内の PCB の)	**b** 数値	16 から 18 桁目が 'TP**b**' である場合、このフィールドの桁 (19 から 23) が、PCB リストの始めからの PCB の相対番号として解釈される。この数は 23 桁目で右寄せしなければならないが、先行ゼロは必要ない。入出力 PCB は、このプログラムの PCB リストでは、常に最初の PCB になる。
オプション 5 16 から 23	PSB 内のすべての PCB のリスト	**$LISTALL**	スクリプトのテストのため、PSB 内の PCB をすべて印刷する。
24	状況印刷オプション	**b**	印刷オプションを使用して、この STATUS ステートメントを印刷する。
		1	このステートメントでは印刷オプションを使用しない。この STATUS ステートメントを印刷する。
		2	この STATUS ステートメントを印刷しないが、このステートメントで印刷オプションを使用する。
		3	この STATUS ステートメントを印刷しない。また、このステートメントで印刷オプションを使用しない。
25 から 28	PCB 処理オプション	**xxxx**	これは任意で、2 つの PCB が同じ名前を持ち、処理オプションが異なるときにのみ使用される。ブランクでなければ、PSB でどの PCB を使用するかを選択するため、16 から 23 桁目の PCB 名に加えて使用される。
29	予約済み	**b**	

表 *71. STATUS* ステートメント *(*続き*)*

桁	機能	コード	説明
30 から 32	AIB インターフェース	**AIB**	AIB インターフェースが使用され、PCB でなく AIB が渡されることを示す。(PCB を渡すのはデフォルト。) **注:** AIB インターフェースを使用するとき、PCB は、PCBNAME=name を使用して PSBGEN で定義されなければならない。IOPCB は、すべての入出力 PCB について使用される PCB 名である。 DFSDDLT0 は、15 桁目に 1 があり、16 から 23 桁目に IOPCB があるときに、この名前を認識する。
33	予約済み		
37 から 72	予約済み		
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

DFSDDLT0 が STATUS ステートメントを検出しない場合は、デフォルト印刷オプション (3 から 12 桁目) はすべて 2 であり、デフォルト出力装置オプション (2 桁目) は 1 です。入出力ストリームでは、どのような呼び出しシーケンスの前にでも STATUS ステートメントをコーディングし、参照する PCB または印刷オプションのいずれかを変更することができます。

参照された PCB は、後続の STATUS ステートメントが他の PCB を選択するまで有効です。ただし、入出力 PCB (LOG のような) に対して出さなければならない呼び出しは、この呼び出しに対して入出力 PCB を使用します。この呼び出しの後、PCB は変更されて元の PCB に戻ります。

STATUS ステートメントの例

各 **CALL** ステートメントの印刷 **:** 次の STATUS ステートメントは、DFSDDLT0 に、すべての呼び出しの COMMENTS、 CALL、 COMPARE、 PCB、および SEGMENT DATA オプションを印刷するよう指示します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
S 1 1 1 1 1
```

各 **CALL** ステートメントの印刷と **PCB** の選択**:** 次の STATUS ステートメントは、DFSDDLT0 に、すべての呼び出しの
COMMENTS、CALL、COMPARE、PCB、および SEGMENT DATA オプションを印刷し、PCB を選択することを指示します。

15 桁目の 1 は、PCBNAME で必要です。省略される場合は、PCBNAME が DBDNAME として扱われます。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
S 1 1 1 1 1   1PCBNAME
```

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
S 1 1 1 1 1   1PCBNAME       AIBb
```

各 **CALL** ステートメントの印刷と、**DBD** 名に基づく **PCB** の選択 **:** 次の STATUS ステートメントは、DFSDDLT0 に、すべての呼び出しの COMMENTS、CALL、COMPARE、PCB、および SEGMENT DATA オプションを印刷し、DBD 名によって PCB を選択することを指示します。

15 桁目の 2 はオプションです。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
S 1 1 1 1 1   2DBDNAME


|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
S 1 1 1 1 1   2DBDNAME       AIBb
```

AIB インターフェースを使用しない場合は、STATUS ステートメント入力を既存のストリームに変更する必要はありません。既存の呼び出し機能はそれまでと同様に機能します。ただし、AIB インターフェースを使用する場合は、STATUS ステートメント入力を既存のストリームに変更して、30 から 32 桁目に AIB を含めます。既存の DBD 名、相対 DB PCB、および相対 PCB は、30 から 32 桁目に AIB が含まれ、PCB が PSBGEN で PCBNAME= 名を使用して定義されているときに機能します。

WTO ステートメント

WTO (オペレーター宛メッセージ) ステートメントは、応答を待たずに z/OS コンソールにメッセージを送ります。

以下の表は、WTO ステートメントの形式を示しています。

表 *72. WTO* ステートメント

桁	機能	コード	説明
1-3	制御ステートメントの識別	**WTO**	WTO ステートメント
4	予約済み	**b**	
5 から 72	送信するメッセージ		システム・コンソールに書き出されるメッセージ
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

WTO ステートメントの例

この WTO ステートメントは、z/OS コンソールにメッセージを送り、テスト・ストリームを継続します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
WTO AT A "WTO" WITHIN TEST STREAM --WTO NUMBER 1-- TEST STARTED
```

WTOR ステートメント

WTOR (要応答オペレーター宛メッセージ) ステートメントは、z/OS システム・コンソールにメッセージを送り、応答を待ちます。

以下の表は、WTOR ステートメントの形式を示しています。

表 73. WTOR ステートメント

桁	機能	コード	説明
1 から 4	制御ステートメントの識別	**WTOR**	WTOR ステートメント
5	予約済み	**b**	
6 から 72	送信するメッセージ		システム・コンソールに書き出されるメッセージ
73 から 80	シーケンスの指示	**nnnnnnnn**	SYSIN2 ステートメントがオーバーライドするため。

WTOR ステートメントの例

この WTOR ステートメントは、DFSDDLT0 が z/OS コンソール・オペレーターから応答を受け取るまで、テスト・ストリームを保留にします。どのような応答でも有効です。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
WTOR AT A "WTOR" WITHIN TEST STREAM - ANY RESPONSE WILL CONTINUE
```

DL/I テスト・プログラム (DFSDDLT0) の JCL 要件

DFSDDLT0 は、以下の DD ステートメントを使用します。

実行 JCL は、IMS 環境 (バッチまたはオンライン)、およびインストール・システムのデータ・セット命名標準により異なります。

```
//SAMPLE JOB ACCOUNTING,NAME,MSGLEVEL=(1,1),MSGCLASS=3,PRTY=8          33001100
//GET EXEC PGM=DFSRRC00,PARM='DLI,DFSDDLT0,PSBNAME'                    33001200
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR                                33001300
//IMS      DD DSN=IMS2.PSBLIB,DISP=(SHR,PASS)                          33001400
//         DD DSN=IMS2.DBDLIB,DISP=(SHR,PASS)                          33001500
//DDCARD   DD DSN=DATASET,DISP=(OLD,KEEP)                              33001600
//IEFRDER  DD DUMMY                                                    33001700
//PRINTDD  DD SYSOUT=A                                                 33001800
//SYSUDUMP DD SYSOUT=A                                                 33001900
//SYSIN    DD *                                                        33002000
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
U THIS IS PART OF AN EXAMPLE                                          33002100
S 1 1 1 1 1    PCB-NAME                                               33002200
L        GU                                                           33002300
/*
//SYSIN2   DD *
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
ABEND                                                                 33002300
/*
```

以下のコード例は、BMP での DFSDDLT0 用の JCL のコーディング方法を示しています。プロシージャーの使用はオプションで、ここでは単に 1 つの例として示してあります。

BMP での DFSDDLT0 用の JCL コードの例

```
//SAMPLE JOB ACCOUNTING,NAME,MSGLEVEL=(1,1),MSGCLASS=A                  00010047
//*************************************************************
//*  BATCH DL/I JOB                                           *
//*************************************************************
//BMP   EXEC IMSBATCH,MBR=DFSDDLT0,PSB=PSBNAME
//BMP.PRINTDD DD SYSOUT=A
```

```
//BMP.PUNCHDD DD SYSOUT=B
//BMP.SYSIN DD *
U ***THIS IS PART OF AN EXAMPLE OF SYSIN DATA                              00010000
S 1 1 1 1 1           1                                                    00030000
L        GU                                                              00040000
L   0099 GN                                                              00050000
/*
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
//BMP.SYSIN2 DD *
U ***THIS IS PART OF AN EXAMPLE OF SYSIN2 DATA  ******************     00020000
ABEND                                                                    00050000
/*
```

SYSIN DD ステートメント

SYSIN DD ステートメントで指定するデータ・セットは、DFSDDLT0 に対する標準入力データ・セットです。直接アクセスまたはテープの入力データを処理しているときに、SYSIN 入力ストリームの中の制御ステートメントを変更したり、他の制御ステートメントを追加することが必要な場合があります。これは、SYSIN2 DD ステートメントおよび制御ステートメント・シーケンス番号を使用して行います。

SYSIN データの 73 から 80 桁目のシーケンス番号は、SYSIN2 オーバーライドが使用されない限り、オプションです。

関連資料:

『SYSIN2 DD ステートメント』

270 ページの『制御ステートメント』

SYSIN2 DD ステートメント

DFSDDLT0 に SYSIN2 DD ステートメントは必須ではありませんが、これが JCL にある場合は、DFSDDLT0 は指定されたデータ・セットを読み取って処理します。

SYSIN2 を使用する場合、次の事項が適用されます。

- SYSIN DD データ・セットは 1 次入力。 DFSDDLT0 は、SYSIN2 制御ステートメントを SYSIN DD データ・セットに挿入しようとします。
- 73 から 80 桁目に制御グループおよびシーケンス番号を正しくコーディングしないと、マージ処理が機能しない。
- 73 桁目および 74 桁目は、ステートメントの制御グループを示す。
- 75 から 80 桁目は、ステートメントのシーケンス番号を示す。
- シーケンス番号は、制御グループ内では数値順でなければならない。
- SYSIN2 内の制御グループは SYSIN 制御グループと一致していなければならないが、SYSIN2 は SYSIN 内で使用される制御グループすべてを使用する必要はない。 DFSDDLT0 では制御グループが数値順でなくてもかまいませんが、SYSIN2 内の制御グループは SYSIN の制御グループと同じ順序でなければなりません。
- DFSDDLT0 が SYSIN および SYSIN2 内の制御グループを突き合わせるときは、シーケンス番号によってステートメントを処理する。 SYSIN ステートメントの前後にある SYSIN2 ステートメントは、これに応じてマージされます。
- SYSIN2 ステートメントのシーケンス番号が、制御グループの SYSIN ステートメントのシーケンス番号と一致するときは、SYSIN2 が SYSIN を変更する。

- SYSIN2 の終了する前にプログラムが SYSIN の終わりに達すると、SYSIN の拡張であるかのように SYSIN2 のレコードを処理する。
- 置換やマージは現行の実行中にだけ起こる。元の SYSIN データは変更されません。
- マージ中に、制御ステートメントの 1 つで 73 から 80 桁目がブランクであった場合は、DFSDDLT0 はブランクを含むそのステートメントを廃棄し、PRINTDD にメッセージを送り、ファイルの終わりまでマージを続ける。

関連資料:

314 ページの『SYSIN DD ステートメント』

270 ページの『制御ステートメント』

304 ページの『IGNORE ステートメント』

PRINTDD DD ステートメント

PRINTDD DD ステートメントは、SNAP 呼び出しを使用した制御ブロックの表示を含め、DFSDDLT0 用の出力データ・セットを定義します。このステートメントは、z/OS SNAP のデータ・セット要件に従っている必要があります。

PUNCHDD DD ステートメント

出力データ・セットの DD ステートメントのラベルは PUNCHDD です。

データ・セットは、LRECL=80 の固定ブロックです。 DD ステートメントで指定されるブロック・サイズが使用され、指定されない場合は、ブロック・サイズは 80 に設定されます。プログラムで PUNCHDD をオープンできない場合は、DFSDDLT0 が異常終了 251 を出します。次は、PUNCHDD DD ステートメントの例です。

```
//PUNCHDD DD SYSOUT=B
```

DFSDDLT0 入力再始動のための PREINIT パラメーターの使用

DFSDDLT0 再始動機能を使用して、同じ従属領域で DFSDDLT0 入力ストリームを再始動します。

EXEC ステートメントの PREINIT パラメーターが再始動機能を呼び出します。DFSMPR の PREINIT パラメーターを `PREINIT=xx` とコーディングします。ここで、xx は、DFSINTxx PROCLIB メンバーの 2 文字の接尾部です。(PREINIT=DL はデフォルトの PROCLIB メンバーを参照します。)

PREINIT 処理は、アクティブ IMS 領域のそれぞれについて、チェックポイント・フィールドを確立します。このフィールドは、入出力 PCB への各 GU 呼び出しが処理されるときに、そのシーケンス番号を使用して更新されます。この理由から、使用されるこのような GU 呼び出しすべてにシーケンス番号が必要となります。再始動時にチェックポイント・フィールドにシーケンス番号が入っている場合には、DFSDDLT0 ストリームは、入出力 PCB に対するそのシーケンス番号の次の GU 呼び出しから開始されます。シーケンス番号が入っていない場合には、DFSDDLT0 ストリームは最初から開始されます。

DFSDDLSI モジュールおよびデフォルトの IMS.PROCLIB メンバーである DFSINTDL は、IMS とともに出荷され、標準 IMS インストールの一部としてインストールされます。

次は SYSIN/SYSIN2 および PREINIT のコーディングの例です。

```
//TSTPGM   JOB CARD
//DDLTTST  EXEC DFSMPR,PREINIT=DL
//MPP.SYSIN DD *
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
S11 1 1 1 1    TP     1                                                    01000000
OPTIONS SNAP= ,ABORT=9999                                                  01000010
U********************************************************************** 01000040
S11 1 1 1 1    TP     1                                                    01000050
L        GU                                                                01000060
E      OK                                                                  01000070
S11 1 1 1 1    DBPCBXXX                                                    01000080
L        GU                                                                01000090
E        DATA    A   INIT-LOAD UOW                                         01000100
E   01    ROOTSEG1 0008A  0004D                                            01000110
S11 1 1 1 1    TP     1                                                    01000120
L        ISRT                                                              01000130
L  Z0080 DATA  -SYNC INTERVAL 1    SEG 1 -MESSAGE    1                   X01000140
L P      DATA  1111111111111111111111111111111111111111111111111          01000150
L        ISRT                                                              01000160
L  Z0080 DATA  -SYNC INTERVAL 1    SEG 2 -END  EOM   1                   X01000170
L P      DATA  1111111111111111111111111111111111111111111111111          01000180
U********************************************************************** 01000190
U*   ENDING FIRST SYNC INTERVAL                                           01000200
U********************************************************************** 01000210
L        GU                                                                01000220
E      QC                                                                  01000230
L        GU                                                                01000240
E      OK                                                                  01000250
S11 1 1 1 1    DBPCBXXX                                                    01000260
WTO GETTING DATA BASE SEGMENT 1 FROM DBPCBXXX                              01000270
L  U     GHU                                                               01000280
E        DATA  INIT-LOAD UOW. 1 A.P. 1                                     01000290
E      OK                                                                  01000300
L  U0003 GN                                                                01000310
E      OK                                                                  01000320
S11 1 1 1 1    TP     1                                                    01000330
L        ISRT                                                              01000340
L  Z0080 DATA  -SYNC INTERVAL 2    SEG 1 -MESSAGE    1                   X01000350
L P      DATA  2222222222222222222222222222222222222222222222211          01000360
L        ISRT                                                              01000370
L  Z0080 DATA  -SYNC INTERVAL 2    SEG 2 -END  EOM   1                   X01000380
L P      DATA  2222222222222222222222222222222222222222222222211          01000390
U********************************************************************** 01000400
U*   ENDING SECOND SYNC INTERVAL                                          01000410
U********************************************************************** 01000420
L        GU                                                                01000430
E      QC                                                                  01000440
L        GU                                                                01000450
E      OK                                                                  01000460
S11 1 1 1 1    DBPCBXXX                                                    01000470
S11 1 1 1 1    TP     1                                                    01000480
L        ISRT                                                              01000490
L  Z0080 DATA  -SYNC INTERVAL 3    SEG 1 -MESSAGE    1                   X01000500
L P      DATA  3333333333333333333333333333333333333333333333311          01000510
L        ISRT                                                              01000520
L  Z0080 DATA  -SYNC INTERVAL 3    SEG 2 -END  EOM   1                   X01000530
L P      DATA  3333333333333333333333333333333333333333333333311          01000580
U********************************************************************** 01000590
U*   ENDING THIRD SYNC INTERVAL                                           01000600
U********************************************************************** 01000610
L        GU                                                                01000620
E      QC                                                                  01000630
```

```
//MPP.SYSIN2 DD *
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----<
ABEND                                                                01000430
/*
```

SYSIN/SYSIN2 および **PREINIT** の例に関する注**:**

1. EXEC ステートメントにコーディングされる PREINIT= パラメーターが、再始動プロセスを呼び出します。
2. DFSDDLT0 は処理を開始すると、SYSIN 内の同じシーケンス番号を持つステートメントを SYSIN2 ABEND ステートメントで置き換えます。(これは、シーケンス番号 01000430 の GU 呼び出しです。)
3. DFSDDLT0 はステートメント 01000000 で始まり、ABEND ステートメント (ステートメント番号 01000430) を検出するまで処理を続けます。入出力 PCB への GU 呼び出しは、すでにチェックポイント・フィールド (ステートメント 01000060、01000220、および 01000240) に記録されています。
4. DFSDDLT0 はスケジュール変更が行われると、チェックポイント・フィールドを検査し、01000240 を検出します。 DFSDDLT0 は、入出力 PCB への次の GU 呼び出し、ステートメント 01000450 から処理を開始します。

   現行の番号が 01000240 であるステートメントにシーケンス番号がない場合は、DFSDDLT0 がスケジュール変更の際にステートメント 01000000 から再始動します。

IMS 領域での DFSDDLT0 の実行

DFSDDLT0 は、DL/I または BMP 領域で操作するように設計されていますが、IFP または MPP 領域でも実行できます。 BMP または DL/I 領域では、EXEC ステートメントには PSB 名と異なるプログラム名を使用できます。 BMP または DL/I 領域でどのようなデータベースを対象とする呼び出しを実行する際にも、問題はありません。

MPP 領域では、プログラム名は PSB 名と同じでなければなりません。 MPP 領域で DFSDDLT0 プログラムを実行するには、DFSDDLT0 に、IMS 定義で指定された PSB 名または PSB の別名を付ける必要があります。一時ステップ・ライブラリーを使用できます。

MPP 領域、または EXEC ステートメントに入力トランザクション・コードを指定した BMP 領域では、通常、DFSDDLT0 が入出力 PCB に対する GU および GN を出して入力を入手します。 DFSDDLT0 は、「メッセージはもうありません」という状況コード QC を受け取るまで、GU および GN 呼び出しを発行します。従属領域に SYSIN DD ステートメントおよび PRINTDD DD ステートメントがある場合は、DFSDDLT0 が SYSIN および SYSIN2 から入力 (ある場合) を読み取り、PRINTDD に出力を送ります。従属領域が MPP 領域であり、入力ストリームで SYSIN からファイルの終わりが検出される前に GU を入出力 PCB に出さない場合、プログラムは暗黙的に GU を入出力 PCB に出し、プログラムをスケジュールするメッセージを獲得します。入力ストリームで GU を入出力 PCB に出し、『メッセージはもうありません』という状況コードが受け取られると、ファイルの終わりとして扱われます。入力が入出力 PCB からであれば、STATUS ステートメントの 2 桁目に 1 または A をコーディングして、PRINTDD に出力を送ることができます。

入力は固定形式なので、端末装置からキーで入力するのは困難です。メッセージ領域で、DFSDDLT0 を使用して DL/I をテストする場合は、区分セットのメンバーとして保管された制御ステートメントを読み取る別のメッセージ・プログラムを実行します。これらの制御ステートメントを入力トランザクション・キューに挿入します。こうすると、IMS が、トランザクションを処理するようにプログラムをスケジュールします。この方式では、どのような領域タイプでも同じ制御ステートメントを使用することができます。

DFSDDLT0 戻りコードの説明

DFSDDLT0 からの非ゼロの戻りコードは、その時点で発生した比較結果の不一致の数を示します。

DFSDDLTO からの戻りコード 0 は、必ずしも DFSDDLT0 がエラーなしに実行されたことを意味するわけではありません。 DSFDDLT0 によって発行されるメッセージはいくつかあり、戻りコードは変わりませんが、ある種のエラー状態を示します。これにより、比較結果の不一致のカウント用に戻りコード・フィールドが保持されます。

実行中にエラー・メッセージが発行されると、メッセージ `ERRORS WERE DETECTED WITHIN THE INPUT STREAM. REVIEW OUTPUT TO DETERMINE ERRORS.`が DFSDDLT0 出力の最後に表示されます。この出力を調べて、DFSDDLT0 が要求どおりに実行されたかどうかを確認してください。

DFSDDLT0 操作

DFSDDLT0 を使用して、データベース、印刷、検索、置換、および削除のセグメントをロードできます。また、リグレッション・テストの実行、デバッグ援助機能としての使用、および呼び出しの実行状況の検査ができます。

データベースのロード

ごく小さいデータベースをロードする場合のみ、DFSDDLT0 を使用します。その理由は、呼び出しとデータを、生成させるのではなく、すべて提供しなければならないためです。次の例は、データベースのロードに使用される CALL FUNCTION および CALL DATA ステートメントを示しています。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7---+-----<
0 SNAP= ,ABORT=0
S 1 2 2 1 1
L        ISRT COURSE
L        DATA FRENCH
L        ISRT COURSE
L        DATA COBOL
L        ISRT CLASS
L        DATA 12
L        ISRT CLASS
L        DATA 27
L        ISRT STUDENT
L        DATA SMITH          THERESE
L        ISRT STUDENT
L        DATA GRABOWSKY      MARION
```

データベースのセグメントの印刷

データベースのセグメントを印刷するには、次のどちらかの制御ステートメントのシーケンスを使用します。

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7---+-----<
.* Use PRINTDD, print call, compare, and PCB if compare unequal
.* Do 1 Get Unique call
.* Hold PCB compare, End step if status code is not blank, GA, GC, GK
.* Do 9,999 Get Next calls
S   2 2 2 1    DBDNAME
L        GU
EH8    OK
L   9999 GN
```

```
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7---+-----<
.* Use PRINTDD, print call, compare, and PCB if compare unequal
.* Do 1 Get Unique call
.* Hold PCB compare, Halt GN calls when status code is GB.
.* Do 9,999 Get Next calls
S   2 2 2 1    DBDNAME
L        GU
EH     OK
L   9999 GN
```

上のどちらの例も、GN を 9999 回繰り返すことを要求しています。最初の例では EH8 の COMPARE PCB を使用し、2 番目の例では EH の COMPARE PCB を使用していることに注意してください。

この 2 つの例の違いは次のとおりです。1 番目の例では、初めて状況コードがブランク、GA、GC、または GK でなくなったときに、ジョブ・ステップを停止します。 2 番目の例では、GB 状況コードが戻されるか、 GN が 9999 回繰り返されたとき、 GN の繰り返しを停止し、残りの DFSDDLT0 制御ステートメントの処理を行います。

セグメントの検索および置換

セグメントを検索および置換するには、次の制御ステートメントのシーケンスを使用します。

```
|----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----
S 1 1 1 1 1    COURSEDB
L        GHU   COURSE  (TYPE     =FRENCH)                              X
               CLASS   (WEEK     =27)                                  X
               STUDENT (NAME     =SMITH)
L        REPL
L        DATA  SMITH            THERESE
```

セグメントの削除

セグメントを削除するには、次の制御ステートメントのシーケンスを使用します。

```
|----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----
S 1 1 1 1 1           4
L        GHU   COURSE  (TYPE     =FRENCH)                              X
               CLASS   *L                                              X
               INSTRUC (NUMBER   =444)
L        DLET
```

リグレッション・テストの実施

リグレッション・テストの実施には、DFSDDLT0 が最適です。既知のデータベースを使用して、DFSDDLT0 は呼び出しを発行し、呼び出しの結果と期待される結果を COMPARE ステートメントを使用して比較することができます。こうして、プログラムは DL/I 呼び出しが正常に実行されたかどうかを判別することができます。印刷オプションをすべて 2 (比較を行い、かつ不一致の場合のみ印刷) としてコーディングすると、適切に要件を満たさなかった呼び出しのみが表示されます。

デバッグ援助機能の使用

プログラムをデバッグするときは、通常は DL/I ブロックの印刷が必要です。比較の結果が 1 回不一致になる COMPARE ステートメントを使用して、適切な回数だけログ・データ・セットにブロックをスナップすることができます。その後、ログからブロックを印刷します。失敗した呼び出しの前の呼び出しのように、呼び出しが正常に実行されてもブロックが必要である場合は、入力ストリームの CALL ステートメントに SNAP 機能を挿入します。

呼び出しの実行状況の検査

特定の呼び出しを実行することは非常に簡単であるため、DFSDDLT0 を使用して、特定の呼び出しがどのように取り扱われるかを検査することがきます。 DL/I が特定の状況で正しく操作されていないようであれば、この方法が有効です。適切に実行されていない可能性のある呼び出しを発行して、結果を検査することができます。

第 2 章 DRDA DDM コマンド・アーキテクチャー参照

IMS は、Distributed Relational Database Architecture™ (DRDA) の分散データ管理 (DDM) アーキテクチャーをサポートします。 IMS ターゲット DDM サーバーと通信して、DBCTL および DB/TM IMS システムの IMS DB が管理するデータベースにアクセスできる、独自のソース DDM サーバーを作成できます。

DDM アーキテクチャー向けの IMS 資料には、IMS との接続および通信に必要な DDM 構造、および IMS が変更または定義している DDM 構造についてのみ記載されています。

DDM の詳細な資料については、www.opengroup.org の The Open Group サイトで入手できる、「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

DDM アーキテクチャーには、以下の要素または項が含まれています。

- コマンド
- コマンド・オブジェクト
- 応答オブジェクト
- 応答メッセージ

それぞれの項は、コマンド、コマンド・オブジェクト、応答オブジェクト、パラメーター、またはメッセージのいずれであるかに関係なく、ソース・サーバーとターゲット・サーバー間で通信するコンポーネントを表現したり示したりする、コード・ポイント という 16 進数の値で表されます。例えば、EXCSAT コマンドは X'1041' で、EXCSATRD 応答オブジェクトは X'1443' で、また SRVNAM パラメーターは X'116D' でそれぞれ表されます。

DRDA 仕様はオープン・スタンダードであるため、その仕様を用いる製品は、そのアーキテクチャーの規則、プロトコル、標準などに準拠していることが要求されます。ただし、DRDA 仕様の一部である DDM アーキテクチャーでは、製品ごとに製品固有の拡張機能 を作成することが許可され、IMS などの製品は、DDM 定義の既存のコマンド、パラメーター、およびメッセージのサブセットと、製品が定義する製品固有の構造を使用できます。製品固有の構造を持つ拡張機能を作成する場合、その製品は DDM アーキテクチャーに準拠している必要があります。

IMS の製品固有の拡張機能は、DDM アーキテクチャーおよび DRDA 仕様の両方に準拠しています。IMS は、DDM 定義の既存のコマンド、パラメーター、およびメッセージのサブセットだけでなく、DDM アーキテクチャーに準拠しながら IMS に固有である、さまざまな IMS 定義の構造を使用します。

関連概念:

DRDA のための IMS サポート を使用したプログラミング (アプリケーション・プログラミング)

IMS がサポートする DDM 項の構文についての概要

IMS は、分散データ管理 (DDM) アーキテクチャーに定義されている項の一般構文規則をサポートします。

すべての DDM コマンド、応答メッセージ、およびチェーン・オブジェクトは、6 バイトのデータ・ストリーム構造ヘッダー (DSSHDR) で始まり、項の長さ (LL) を定義する 2 バイトの 2 進整数、DDM 項を一意的に識別する 2 バイトの 16 進数のコード・ポイント (CP)、および存在する場合はデータが、その後に順に続きます。

コマンド、メッセージ、およびオブジェクトのパラメーターは LL で始まり、CP、およびデータが、その後に順に続きます。パラメーターはインスタンス変数とも呼ばれ、これには DSSHDR は含まれません。

IMS 製品固有データ構造である aibStream、dbpcbStream、および iopcbStream などのデータ構造には、DSSHDR、LL、または CP は含まれません。

関連資料:

330 ページの『DEALLOCDB コマンド (X'C801')』

DSSHDR 構文規則

DSSHDR は、分散データ管理 (DDM) アーキテクチャーで定義されている項のデータ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダーです。

DSSHDR のフォーマットは以下のとおりです。

LL コマンド、応答、またはオブジェクト全体 (6 バイトの DSS HEADER を含む) の、2 バイト仕様の長さ。可能な最小値は 6 で、最大値は 32,767 です。

DDMID
汎用データ・ストリーム (GDS) ID に登録されている 1 バイトのシステム・ネットワーク体系 (SNA)。 DDM コマンドの DDMID フィールドは常に D0 です。

FORMAT ID
DSS が次の DSS にチェーンされているかどうか、およびエラー発生時の処理方法を示す、1 バイトの標識。このバイトには、以下のビットが、0 から 7 まで、左から右に入ります。

ビット **0**
未使用。

ビット **1**
フラグ。1 は、DSS 構造が次の DSS 構造にチェーンされていることを示します。 0 はチェーンがないことを示します。

ビット **2**
フラグ。1 はエラーが発生した場合に続行することを、0 は続行しないことを示します。

ビット **3**
フラグ。1 は、次の DSS に同じ要求相関関係子があることを示します。0 は、同じ要求相関関係子がないことを示します。ビット 1 が 0 の場合、ビット 3 も 0 です。

ビット **4** からビット **7**
以下の DSS タイプを示します。
- 1: 要求 DSS
- 2: 応答 DSS
- 3: オブジェクト DSS
- 4: 暗号化オブジェクト DSS

RQSDRR
要求と、その要求データ、要求に対する応答、および要求に対して返すデータとを関連付ける、生成される 2 バイト・フィールド。

DDM コミットおよびロールバック処理

分散データ管理 (DDM) アーキテクチャーの IMS 実装環境には、コミットおよびロールバック処理のサポートが組み込まれています。

XA サポートおよびグローバル・トランザクション処理は、DDM コマンド SYNCCTL および SYNCCRD により制御されます。

ローカル・トランザクション処理は、DDM コマンド RDBCMM および RDBRLLBCK により制御されます

IMS はこれらの DDM コマンドを、DRDA による本来の仕様の範囲を超えて拡張することはありません。

これらのコマンドの資料は、「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」にあります。

DDM コマンドおよびコマンド・オブジェクト

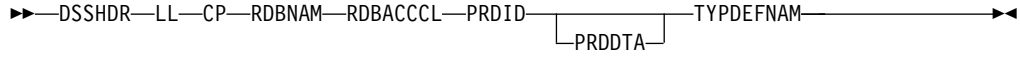
IMS は、分散データ管理 (DDM) アーキテクチャーのコマンドおよびコマンド・オブジェクトのサブセットをサポートし、その他の IMS 製品固有の DDM コマンドを定義しています。

ACCRDB コマンド (X'2001')

分散データ管理 (DDM) アーキテクチャーの ACCRDB コマンドは、ソース・サーバーに代わってプログラム仕様ブロック (PSB) を割り振ります。 PSB は DDM ソース・サーバーと IMS データベースとの間の接続を表します。

PSB は、データベース接続を閉じて通信会話が終了するまで割り振られた状態を維持します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2001'、ACCRDB コマンドの 2 バイトのコード・ポイント。

RDBNAM

ターゲット・データベースを識別する IMS PSB 名が入る必須パラメーター (X'2110')。PSB 名は、最大 8 バイト長の文字ストリングです。 RDBNAM には、オプションで IMS データ・ストアの別名を入れることができます。

RDBACCCL

データベースにアクセスするアプリケーション・マネージャーを指定する必須パラメーター。 RDBACCCL のコード・ポイントは X'210F' です。 RDBACCCL の値は予約されており、X'2407' でなければなりません。

PRDID

ソース DDM サーバーのリリース・レベルを指定する必須パラメーター。 PRDID のコード・ポイントは X'112E' です。

PRDDTA

ACCRDB コマンドの発行時にターゲット・サーバーの SRVCLSNM が不明の場合、ターゲットに渡される製品固有の情報を指定するオプション・パラメーター。 PRDDTA のコード・ポイントは X'2104' です。このパラメーターはターゲット・サーバーにより無視される場合があります。

TYPDEFNAM

データ型定義の名前を指定する必須パラメーター (X'002F')。 TYPDEFNAM は、2 バイト仕様の長さ (LL)、2 バイトのコード・ポイント (CP)、および値で構成されます。この値は予約されており、QTDSQL370 である必要があります。これは、EBCDIC ストリング、IEEE 浮動小数点数、およびバイト逆順でない浮動小数点数と整数を使用するマシンのための、一般的な EBCDIC SQL 型定義です。

使用法

ACCRDB コマンドの処理中にエラーが発生しなかった場合、IMS ターゲット・サーバーは、データベースが割り振り済みであることを示す ACCRDBRM 応答メッセージを返します。

チェーンされるコマンド・オブジェクト

ACCRDB コマンドにチェーンできるコマンド・オブジェクトはありません。

正常の応答メッセージ

ACCRDB コマンドに応答して、IMS ターゲット DDM サーバーはソース・サーバーに、以下の正常の応答メッセージを返します。

ACCRDBRM

データベースへのアクセスが完了しました。

コード・ポイント: X'2201'

以前の ACCRDB コマンドで指定されたデータベースが、クライアントの処理に使用可能であることを示します。

エラー応答メッセージ

ACCRDB コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、ACCRDB コマンドに固有の以下のエラー応答メッセージを返すことがあります。

表 *74. ACCRDB* コマンドに固有の、想定されるエラー応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'2203'	RDBATHRM	データベースへのアクセスが許可されていません。
X'2211'	RDBNFNRM	データベースが見つかりません。
X'221A'	RDBAFLRM	RDB アクセス失敗の応答メッセージ。 RDBNAM パラメーターが ACCRDB コマンドに指定されていた場合、RDBAFLRM 応答メッセージは、データベース (RDB) に接続しようとして失敗したことを示します。

関連資料:

372 ページの『ACCRDBRM 応答メッセージ (X'2201')』
406 ページの『RDBNAM パラメーター (X'2110')』
389 ページの『RDBAFLRM 応答メッセージ (X'221A')』
390 ページの『RDBATHRM 応答メッセージ (X'2203')』
391 ページの『RDBNACRM 応答メッセージ (X'2204')』

ACCSEC コマンド (X'106D')

ACCSEC DDM コマンドは、ソース・サーバーにあるアプリケーション・プログラムが、IMS ターゲット・サーバーにあるデータベースに接続する場合に実行するセキュリティー検査のタイプを決定するために使用します。

ソース・サーバーは ACCSEC コマンドを使用して、DDM アーキテクチャーで定義されているどのタイプのセキュリティー・メカニズムを識別および認証に使用するかについて、IMS ターゲット・サーバーとネゴシエーションします。 IMS は、DDM アーキテクチャーのユーザー ID とパスワードによるセキュリティー・メカニズム (USRIDPWD) のみをサポートします。 ACCSEC コマンドは、有効ないずれかのセキュリティー・メカニズムがアクティブの場合は、必ず SECCHK コマンドに先行して実行する必要があります。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'106D'、ACCSEC コマンドの 2 バイトのコード・ポイント。

SECMEC
ソース・サーバーが IMS ターゲット・サーバーとの対話に使用するセキュリティー・メカニズムを指定する必須パラメーター。IMS は、DDM アーキテクチャーの USRIDPWD セキュリティー・メカニズムのみをサポートします。USRIDPWD を指定するには、2 バイトの 2 進数で表した 3 を SECMEC パラメーターに入力します。

RDBNAM

ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。PSB 名は、最大 8 バイト長の文字ストリングです。RDBNAM には、オプションで IMS データ・ストアの別名を入れることができます。

使用法

ソース・サーバーとターゲット DRDA サーバーとの間の初期ハンドシェーク中は、ソース・サーバーは、ACCSEC コマンドにチェーンされた EXCSAT コマンドを発行する必要があります。

交換が成功すると、IMS ターゲット・サーバーは、ACCSEC コマンドへの応答として ACCSECRD 応答データ・オブジェクトを返します。ACCSECRD 応答オブジェクトにより、IMS ターゲット・サーバーが使用するセキュリティー・メカニズムはソース・サーバーに識別されます。交換が成功すると、ACCSECRD 応答オブジェクトに返される値は、ACCSEC コマンドの SECMEC パラメーターの値と同じになります。

ACCSEC コマンドの処理中に IMS ターゲット・サーバーがエラーを検出した場合、ACCSECRD 応答オブジェクトには SECCHKCD パラメーターが含まれます。ACCSECRD 応答オブジェクトには、SECCHKCD パラメーターに暗黙の重大度コード ERROR があります。エラーが発生した後は、ACCSEC コマンドを再送信してから、SECCHK コマンドを送信して接続を認証する必要があります。

チェーンされるコマンド・オブジェクト

ACCSEC コマンドにチェーンできるコマンド・オブジェクトはありません。

応答データ・オブジェクト

ACCSEC コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下の応答データ・オブジェクトを返すことがあります。

ACCSECRD (X'14AC')
アクセス・セキュリティー応答データ。

エラー応答メッセージ

ACCSEC コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下の応答メッセージを返すことがあります。

表 *75. ACCSEC* コマンドで想定される応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'121C'	CMDATHRM	許可されていないコマンド
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'1233'	RSCLMTRM	限度に達したリソース
X'123C'	INVRQSRM	要求が無効
X'124C'	SYNTAXRM	データ・ストリーム構文エラー
X'1250'	CMDNSPRM	サポートされていないコマンド
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値
X'1254'	CMDCHKRM	コマンド検査応答メッセージ
X'125F'	TRGNSPRM	サポートされていないターゲット

関連資料:

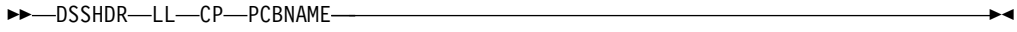
375 ページの『ACCSECRD 応答オブジェクト (X'14AC')』

406 ページの『RDBNAM パラメーター (X'2110')』

CLSQRY コマンド (X'2005')

分散データ管理 (DDM) アーキテクチャーの CLSQRY コマンドは、OPNQRY 呼び出しによって前に開いた照会を閉じます。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2005'、CLSQRY コマンドの 2 バイトのコード・ポイント。

PCBNAME

DL/I 呼び出しが実行した照会を一意的に識別する PCB 名を指定する必須パラメーター。 PCB 名は文字ストリングで指定します。この値は最初は元の OPNQRY コマンドで送信されます。続いて、元の OPNQRY 呼び出しと正しく相関させるために、CNTQRY、CLSQRY、および RLSE などのコマンドで同じ値を送信する必要があります。 PCBNAME パラメーターのコード・ポイントは X'C907' です。

使用法

DDM コマンド CLSQRY (照会のクローズ) を使用し、OPNQRY 呼び出しによって前に開いた照会を閉じます。

チェーンされるコマンド・オブジェクト

CLSQRY コマンドにチェーンできるコマンド・オブジェクトはありません。

エラー応答メッセージ

CLSQRY コマンド処理中にエラーが発生した場合、IMS ターゲット DDM サーバーは、ソース DDM サーバーに以下のエラー応答メッセージを返すことがあります。

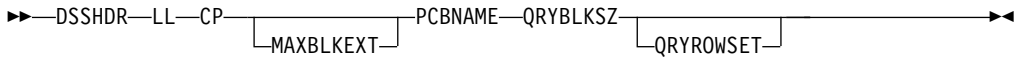
表 *76. CLSQRY* コマンドの、想定されるエラー応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'121C'	CMDATHRM	許可されていないコマンド
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'1233'	RSCLMTRM	限度に達したリソース
X'1245'	PRCCNVRM	会話型プロトコル・エラー
X'124C'	SYNTAXRM	データ・ストリーム構文エラー
X'1250'	CMDNSPRM	サポートされていないコマンド
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値
X'1254'	CMDCHKRM	コマンド検査応答メッセージ
X'125F'	TRGNSPRM	サポートされていないターゲット

CNTQRY コマンド (X'2006')

分散データ管理 (DDM) アーキテクチャーの CNTQRY コマンドは、前の OPNQRY 呼び出しで生成された結果セット・データの戻りを再開することで、照会を続行します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2006'、CNTQRY コマンドの 2 バイトのコード・ポイント。

MAXBLKEXT
CNTQRY コマンドに応じて、要求側が応答データとして受信できる結果セットごとに、追加ブロックの最大数を指定するオプション・パラメーター。数値は 2 バイトの 2 進数で指定します。値が 0 の場合は、要求側が応答セット・データの追加照会ブロックを受信できないことを示します。値が -1 の場合は、要求側が結果セット全体を受信できることを示します。 MAXBLKEXT のコード・ポイントは X'2141' です。

PCBNAME

DL/I 呼び出しが実行した照会を一意的に識別する PCB 名を指定する必須パラメーター。 PCB 名は文字ストリングで指定します。この値は最初は元の OPNQRY コマンドで送信されます。続いて、元の OPNQRY 呼び出しと正しく相関させるために、CNTQRY、CLSQRY、および RLSE などのコマンドで同じ値を送信する必要があります。 PCBNAME パラメーターのコード・ポイントは X'C907' です。

QRYBLKSZ
ソース・アプリケーション・プログラムに最適な照会ブロックのサイズを指定する必須パラメーター。照会ブロックは、応答セット・データを返すためにターゲット・サーバーが使用します。ターゲット・サーバーは必要に応じてこのパラメーターを指定変更できます。照会ブロック・サイズは、4 バイトの符号なし 2 進数で指定します。照会ブロックの最小サイズは 0.5 KB です。最大サイズは 10 MB です。 QRYBLKSIZ のコード・ポイントは X'2114' です。

QRYROWSET
1 回のネットワーク応答で返すデータの行数を指定するオプション・パラメーター。行数は 4 バイトの 2 進数で指定します。 QRYROWSET の最小値は 0 です。指定可能な最大値は 32,767 です。 QRYROWSET のコード・ポイントは X'2156' です。

使用法

OPNQRY 呼び出しで前に生成した結果セット・データの戻りを再開するには、DDM コマンド CNTQRY (照会の続行) を使用します。

チェーンされるコマンド・オブジェクト

CNTQRY コマンドにチェーンされるコマンド・オブジェクトはありません。

応答データ・オブジェクト

CNTQRY コマンドに応答して、以下の応答データ・オブジェクトが返されます。

QRYDTA (X'241B')
照会応答セット・データ。

エラー応答メッセージ

CNTQRY コマンド処理中にエラーが発生した場合、IMS ターゲット DDM サーバーは、ソース DDM サーバーに以下のエラー応答メッセージを返すことがあります。

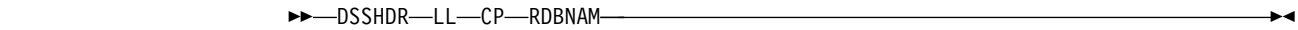
表 *77. CNTQRY* コマンドの、想定されるエラー応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'121C'	CMDATHRM	許可されていないコマンド
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'1233'	RSCLMTRM	限度に達したリソース
X'1245'	PRCCNVRM	会話型プロトコル・エラー
X'124C'	SYNTAXRM	データ・ストリーム構文エラー
X'1250'	CMDNSPRM	サポートされていないコマンド
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値
X'1254'	CMDCHKRM	コマンド検査応答メッセージ
X'125F'	TRGNSPRM	サポートされていないターゲット
X'2204'	RDBNACRM	アクセスされていないデータベース
X'220B'	ENDQRYRM	照会終了
X'220D'	ABNUOWRM	作業単位の異常終了状態
X'2213'	SQLERRRM	SQL エラー状態
X'2218'	RDBUPDRM	データベース更新応答メッセージ。

DEALLOCDB コマンド (X'C801')

分散データ管理 (DDM) アーキテクチャーの DEALLOCDB コマンドは、RDBNAM パラメーターで指定されている PSB を割り振り解除することにより、PSB に関連付けられているすべてのリソースを終了させます。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C801'、DEALLOCDB コマンドの 2 バイトのコード・ポイント。

RDBNAM

ターゲット・データベースを識別する IMS PSB 名が入る必須パラメーター (X'2110')。PSB 名は、最大 8 バイト長の文字ストリングです。 RDBNAM には、オプションで IMS データ・ストアの別名を入れることができます。

使用法

DEALLOCDB コマンド処理中にエラーが発生しなかった場合、IMS ターゲット・サーバーは、データベースの割り振り解除が正常に実行されたことを示す DEALLOCDBRM 応答メッセージを返します。

チェーンされるコマンド・オブジェクト

DEALLOCDB コマンドにチェーンできるコマンド・オブジェクトはありません。

正常の応答メッセージ

DEALLOCDB コマンドに応答して、IMS ターゲット DDM サーバーはソース・サーバーに、以下の正常の応答メッセージを返します。

DEALLOCDBRM (X'CA01')
データベースの割り振り解除が完了しました。

指定された PSB が割り振り解除されていることを示します。

エラー応答メッセージ

DEALLOCDB コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下のエラー応答メッセージを返すことがあります。

表 *78. DEALLOCDB* コマンドの、想定されるエラー応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'124C'	SYNTAXRM	データ・ストリーム構文エラー

関連資料:

406 ページの『RDBNAM パラメーター (X'2110')』

377 ページの『DEALLOCDBRM 応答メッセージ (X'CA01')』
322 ページの『IMS がサポートする DDM 項の構文についての概要』

DLIFUNC コマンド・オブジェクト (X'CC05')

分散データ管理 (DDM) アーキテクチャーの DLIFUNC (DL/I 関数) コマンド・オブジェクトを使用して、呼び出す DL/I 関数を指定します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC05'、DLIFUNC コマンド・オブジェクトの 2 バイトのコード・ポイント。

BYTSTRDR
バイト・ストリング・データ表記。データベースで実行する DL/I 呼び出しが入る必須の文字ストリングです。以下の文字ストリング値を、DLIFUNC コマンド・オブジェクトに指定できます。

ISRT
挿入呼び出し

DLET
削除呼び出し

REPL
置換呼び出し

GHU
Get Hold Unique 呼び出し

GU Get Unique 呼び出し

GHN
Get Hold Next 呼び出し

GN Get Next 呼び出し

GHNP
Get Hold Next Within Parent 呼び出し

GNP
Get Next Within Parent 呼び出し

DELETE
バッチ削除呼び出し

UPDATE
バッチ置換呼び出し

RETRIEVE
バッチ検索呼び出し

関連資料:

336 ページの『EXCSQLIMM コマンド (X'200A')』

347 ページの『OPNQRY コマンド (X'200C')』

DLIFUNCFLG コマンド・オブジェクト (X'CC09')

分散データ管理 (DDM) アーキテクチャーの DLIFUNCFLG (DL/I 機能フラグ) コマンド・オブジェクトを使用して、DL/I バッチ処理操作が GU または GN 呼び出しで始まるかどうか、および各呼び出しにどの SSA リストが関連付けられているかを指定します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC09'、DLIFUNCFLG コマンド・オブジェクトの 2 バイトのコード・ポイント。

FFFF
必須の 4 バイト・フラグ値。フラグの各バイトは、次のように異なる DL/I バッチ処理オプションを指定します。

最初のバイト

X'00' GHN 呼び出しでバッチ処理を開始します。

X'80' GHU 呼び出しでバッチ処理を開始します。

2 番目のバイト
2 番目のバイトの最初の 4 ビットは、位置取り出し呼び出しに関連付けられる SSAList を指定します。2 番目のバイトの 2 番目の 4 ビットは、位置取り出し呼び出しに続くオプションの REPL 呼び出しに関連付けられる SSAList を指定します。

B'0000'
SSA なし

B'1000'
リストの最初の SSA

B'0100'
リストの 2 番目の SSA

B'0010'
リストの 3 番目の SSA

B'0001'
リストの 4 番目の SSA

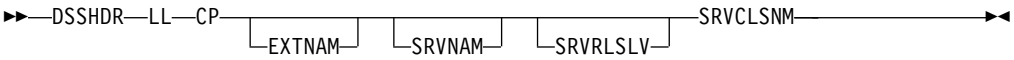
3 番目のバイト
3 番目のバイトは 2 番目のバイトと同じ形式で設定しますが、初回の位置取り出し呼び出しに続く GHN 呼び出しおよびオプションの REPL 呼び出しに使用されます。

4 番目のバイト
予約済み。

EXCSAT コマンド (X'1041')

分散データ管理 (DDM) アーキテクチャーの EXCSAT コマンドは、ソース・アプリケーション・サーバーと IMS ターゲット・サーバーとの間の属性の交換を開始して、各サーバーの DDM サポートのサーバー・クラス名とレベルを識別します。EXCSAT コマンドは必ず、ソース・サーバーから IMS ターゲット・サーバーに送信する最初のコマンドである必要があります。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'1041'、EXCSAT コマンドの 2 バイトのコード・ポイント。

EXTNAM
オプション。IMS データベースへのアクセスを要求する、可変長のプロセス名またはスレッド名。指定した名前は、トレースおよび問題判別のためのアプリケーション・スレッドを示します。ジョブ名に埋め込みブランクが含まれている場合、その名前は引用符で囲む必要があります。 EXTNAM の最大長は 255 バイトです。コード・ポイントは X'115E' です。

SRVNAM
オプション。ソース DDM サーバーの可変長の名前。指定した名前は、トレースおよび問題判別のために、ソース・アプリケーション・プログラムを実行しているコンピューターのホスト名を示します。サーバー名に埋め込みブランクが含まれている場合、その名前は引用符で囲む必要があります。最大長は 255 バイトです。コード・ポイントは X'116D' です。

SRVCLSNM
IMS が使用する DDM サーバー・クラス名 (DFS) を指定します。DFS は、現在 IMS がサポートしている唯一のクラス名です。SRVCLSNM は、IMS が使う DRDA 製品固有の拡張機能を使用できます。

SRVCLSNM のコード・ポイントは X'1147' です。可変長の DDM サーバー・クラス名は文字ストリングで指定します。

使用法

EXCSAT DDM コマンドは、IMS データベースにアクセスする要求を開始して、要求側、つまり DDM ソース・サーバーを IMS の DDM ターゲット・サーバーに識別させるために使用します。

ソース・サーバーとターゲット DRDA サーバーとの間の初期ハンドシェーク中は、ソース・サーバーは、ACCSEC コマンドにチェーンされた EXCSAT コマンドを発行する必要があります。

交換が成功すると、IMS ターゲット・サーバーは、EXCSAT コマンドの応答として EXCSATRD 応答データ・オブジェクトを返します。 EXCSATRD 応答オブジェクトは、IMS ターゲット・サーバーをソース・サーバーに識別させます。

チェーンされるコマンド・オブジェクト

EXCSAT コマンドにチェーンされるコマンド・オブジェクトはありません。

応答データ・オブジェクト

EXCSAT コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下の応答データ・オブジェクトを返すことがあります。

EXCSATRD (X'1443')
サーバー属性を交換します。

エラー応答メッセージ

EXCSAT コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下のエラー応答メッセージを返すことがあります。

表 *79. EXCSAT* コマンドの、想定されるエラー応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'1210'	MGRLVLRM	マネージャー・レベルの競合
X'124C'	SYNTAXRM	データ・ストリーム構文エラー
X'1250'	CMDNSPRM	サポートされていないコマンド
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値
X'1254'	CMDCHKRM	コマンド検査応答メッセージ
X'125F'	TRGNSPRM	サポートされていないターゲット

関連資料:

381 ページの『EXCSATRD 応答オブジェクト (X'1443')』

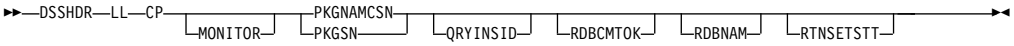
EXCSQLIMM コマンド (X'200A')

分散データ管理 (DDM) アーキテクチャーの EXCSQLIMM コマンドは、挿入、更新、または削除操作を IMS データベース上で実行します。

フォーマット

DLI フロー:

SQL フロー:



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'200A'、EXCSQLIMM コマンドの 2 バイトのコード・ポイント。

QRYINSID
8 バイトの照会インスタンス ID。

制約事項: EXCSQLIMM コマンドが位置指定 delete/update SQL ステートメントで実行されており、照会に関連付けられたセクションに対して複数の照会インスタンスが存在している場合、このパラメーターは必須です。

PCBNAME

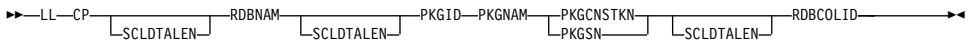
DL/I 呼び出しが実行した照会を一意的に識別する PCB 名を指定する必須パラメーター。 PCB 名は文字ストリングで指定します。この値は最初は元の OPNQRY コマンドで送信されます。続いて、元の OPNQRY 呼び出しと正しく相関させるために、CNTQRY、CLSQRY、および RLSE などのコマンドで同じ値を送信する必要があります。 PCBNAME パラメーターのコード・ポイントは X'C907' です。

PKGNAMCSN(X'2113')
SQL の実行に使用するパッケージ内の完全修飾パッケージ名、整合性トークン、およびセクション番号を指定します。 PKGNAMCSN は、含まれている RDBNAM、RDBCOLID、および PKGID の長さに応じて以下のいずれかの形式にすることができます。

- RDBNAM、RDBCOLID、および PKGID の長さがそれぞれ 18 である場合。この形式の PKGNAMCSN は、DDM レベル 7 より前で使用されていた唯一の形式 (長さ 68 に固定) と同一です。この形式で SCLDTALEN を使用することは許可されません。
- RDBNAM、RDBCOLID、または PKGID の 1 つ以上の長さが 18 より大きい場合。この形式の PKGNAMCSN では、各 RDBNAM、RDBCOLID、お

よび PKGID の前に SCLDTALEN が必要です。この形式を使用する場合、PKGNAMCSN の最小長は 75、最大長は 785 です。

形式:



パラメーター:

RDBNAM
リレーショナル・データベース名を表す 18 から 255 バイトの文字フィールド。

PKGID
リレーショナル・データベース・パッケージ ID を表す 18 から 255 バイトの文字フィールド。

PKGNAM
リレーショナル・データベース・パッケージの完全修飾名を指定する 18 から 255 バイトの文字フィールド。

PKGCNSTKN
要求側のアプリケーションとリレーショナル・データベース・パッケージが同期化されているかを検証する 8 バイトの文字フィールド。 PKGSN とは相互に排他的です。

PKGSN
セクション番号を表す 2 バイトの短フィールド。PKGCNSTKN とは相互に排他的です。

SCLDTALEN
インスタンス変数の長さを指定します。直後に以下が続きます。

- RDB コレクション ID (RDBCOLID)
- リレーショナル・データベース名 (RDBNAM)
- RDB パッケージ ID (PKGID)

このトークンは、リストされているパラメーターの 1 つ以上の長さが 18 バイトより大きい場合にのみ許可されます。

RDBCOLID
リレーショナル・データベースに含まれているオブジェクトの固有コレクションを識別する 18 から 255 バイトの文字フィールド。これは、ユーザー定義のグループ化に使用されます。

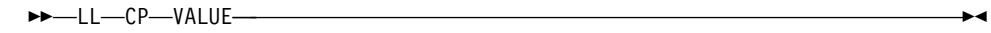
注: RDBNAM、RDBCOLID、または PKGID のいずれかの長さが 18 バイトを超えている場合、SCLDTALEN が必須で、RDBCOLID の前になければなりません。そうしないと、SCLDTALEN は許可されません。

RDBCMTOK
データベースがコミットおよびロールバック操作の処理を許可するかどうかを指定するオプション・パラメーター (X'2105')。値を X'F1' (TRUE) に設定します。これはデータベースがコミットおよびロールバック処理を許可することを示します。

注: IMS Universal ドライバーは、常に値 TRUE を送信します。

RTNSETSTT(X'210E')
コマンドの実行中にいずれかの特殊レジスター設定が変更された場合、Return SET ステートメントは、コマンドの処理が正常に完了したときにターゲット・サーバーが 1 つ以上の SQLSTT 応答データ・オブジェクトを返す必要があるかどうかを制御します。各 SQLSTT 応答データ・オブジェクトには、現行接続で設定が変更された特殊レジスターの SQL SET ステートメントが含まれます。

いずれの特殊レジスターの設定も変更されていない場合は、RTNSETSTT 設定に関係なく、SQLSTT 応答データ・オブジェクトは返されません。
形式:



パラメーター:

VALUE

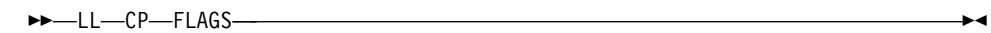
X'00' ターゲット・サーバーは、SQL SET ステートメントを返してはなりません。

X'01' ターゲット・サーバーは、設定が変更された特殊レジスターに対して 1 つ以上の SQL SET ステートメントを返す必要があります。

–

注: IMS は、常に IMS Universal ドライバーから X'01' を送信します。

MONITOR(X'1900')



FLAGS
4 バイト・フラグ値。

使用法

DDM コマンド EXCSQLIMM (即時 SQL の実行) は、置換、挿入、または削除操作を IMS データベースで実行します。

EXCSQLIMM コマンド処理中にエラーが発生しなかった場合、IMS ターゲット・サーバーはデータベース更新応答メッセージ RDBUPDRM (X'2218') を返します。

チェーンされるコマンド・オブジェクト

以下のコマンド・オブジェクトは、EXCSQLIMM コマンドにチェーンされます。

INAIB (X'CC01')
AIB データが入ります。 DLIFUNC の値が DELETE または UPDATE の場合、AIB パラメーターが必要です。

DLIFUNC (X'CC05')
データベースで実行するための DL/I 呼び出し。 DL/I 呼び出しは文字ス

トリングで指定し、データベースで実行するアクションを定義します。 DLIFUNC に指定可能な値の説明については、DLIFUNC の説明を参照してください。

FLDENTRY (X'CC03')
DLIFUNC が ISRT、REPL、または UPDATE に設定されている場合、FLDENTRY パラメーターが必要です。

SSALIST (X'CC06')
セグメント検索引数をリストします。 DLIFUNC が UPDATE または DELETE に設定されている場合、SSALIST パラメーターが必要です。 DLIFUNC が DLET、ISRT、または REPL に設定されている場合は、SSALIST パラメーターはオプションです。

正常の応答メッセージ

EXCSQLIMM コマンドに応答して、IMS ターゲット DDM サーバーはソース・サーバーに、以下の正常の応答メッセージを返します。

RDBUPDRM (X'2218')
データベース更新応答メッセージ。

応答データ・オブジェクト

EXCSQLIMM コマンドの応答として、応答データ・オブジェクトは返されません。

エラー応答メッセージ

EXCSQLIMM コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下のエラー応答メッセージを返すことがあります。

表 *80. EXCSQLIMM* コマンドの、想定されるエラー応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'121C'	CMDATHRM	許可されていないコマンド
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'1233'	RSCLMTRM	限度に達したリソース
X'1245'	PRCCNVRM	会話型プロトコル・エラー
X'124C'	SYNTAXRM	データ・ストリーム構文エラー
X'1250'	CMDNSPRM	サポートされていないコマンド
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値
X'1253'	OBJNSPRM	サポートされていないオブジェクト
X'1254'	CMDCHKRM	コマンド検査応答メッセージ

表 *80. EXCSQLIMM* コマンドの、想定されるエラー応答メッセージ *(*続き*)*

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'125F'	TRGNSPRM	サポートされていないターゲット
X'2204'	RDBNACRM	アクセスされていないデータベース
X'220D'	ABNUOWRM	作業単位の異常終了状態
X'220E'	DTAMCHRM	データ記述子の不一致
X'2213'	SQLERRRM	SQL エラー状態
X'2225'	CMMRQSRM	コミットメント要求

EXCSQLIMM の例

以下の例は、OPNQRY 呼び出しに対する要求の一部である EXCSQLIMM を示しています。

```
[ibm][ims][drda][t4]       SEND BUFFER: EXCSQLIMM             (ASCII)          (EBCDIC)
[ibm][ims][drda][t4] 0000  0060D0510002005A  200A00442113E2C1  .`.Q...Z ..D!...  .-}....!......SA
[ibm][ims][drda][t4] 0010  D4D7D3C540404040  4040404040404040  ....@@@@@@@@@@@@  MPLE
[ibm][ims][drda][t4] 0020  D5E4D3D3C9C44040  4040404040404040  ......@@@@@@@@@@  NULLID
[ibm][ims][drda][t4] 0030  4040E2E8E2E2C8F2  F0F0404040404040  @@........@@@@@@    SYSSH200
[ibm][ims][drda][t4] 0040  4040404053595340  564C303100410005  @@@@SYSLVL01.A..      ...<.<......
[ibm][ims][drda][t4] 0050  2105F10005210E01  0008190080000000  !....!..........  ..1.............
```

関連資料:

332 ページの『DLIFUNC コマンド・オブジェクト (X'CC05')』

343 ページの『FLDENTRY コマンド・オブジェクト (X'CC03')』

371 ページの『SSALIST コマンド・オブジェクト (X'CC06')』

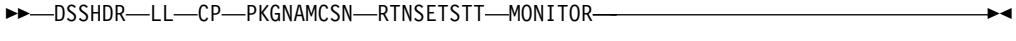
345 ページの『INAIB コマンド・オブジェクト (X'CC01')』

393 ページの『RDBUPDRM 応答メッセージ (X'2218')』

EXCSQLSET コマンド (X'2014')

分散データ管理 (DDM) アーキテクチャーの SQL SET の実行 (EXCSQLSET) コマンドは、1 つ以上の SET ステートメントを実行してアプリケーション環境を確立します。

フォーマット



パラメーター

DSSHDR
DSS に関する情報が入る 6 バイトのヘッダー・フィールド。

LL EXCSQLSET コマンドの長さが入る 2 バイトのフィールド。

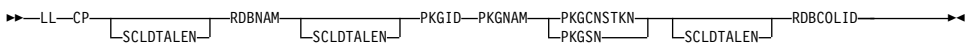
CP(X'2014')
EXCSQLSET コマンドの 2 バイトのコード・ポイント。

PKGNAMCSN(X'2113')

SQL の実行に使用するパッケージ内の完全修飾パッケージ名、整合性トークン、およびセクション番号を指定します。 PKGNAMCSN は、含まれている RDBNAM、RDBCOLID、および PKGID の長さに応じて以下のいずれかの形式にすることができます。

- RDBNAM、RDBCOLID、および PKGID の長さがそれぞれ 18 である場合。この形式の PKGNAMCSN は、DDM レベル 7 より前で使用されていた唯一の形式 (長さ 68 に固定) と同一です。この形式で SCLDTALEN を使用することは許可されません。
- RDBNAM、RDBCOLID、または PKGID の 1 つ以上の長さが 18 より大きい場合。この形式の PKGNAMCSN では、各 RDBNAM、RDBCOLID、および PKGID の前に SCLDTALEN が必要です。この形式を使用する場合、PKGNAMCSN の最小長は 75、最大長は 785 です。

形式:



パラメーター:

RDBNAM
リレーショナル・データベース名を表す 18 から 255 バイトの文字フィールド。

PKGID
リレーショナル・データベース・パッケージ ID を表す 18 から 255 バイトの文字フィールド。

PKGNAM
リレーショナル・データベース・パッケージの完全修飾名を指定する 18 から 255 バイトの文字フィールド。

PKGCNSTKN
要求側のアプリケーションとリレーショナル・データベース・パッケージが同期化されているかを検証する 8 バイトの文字フィールド。 PKGSN とは相互に排他的です。

PKGSN
セクション番号を表す 2 バイトの短フィールド。PKGCNSTKN とは相互に排他的です。

SCLDTALEN
インスタンス変数の長さを指定します。直後に以下が続きます。

- RDB コレクション ID (RDBCOLID)
- リレーショナル・データベース名 (RDBNAM)
- RDB パッケージ ID (PKGID)

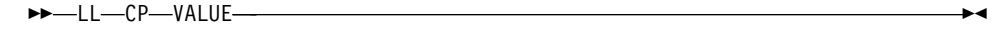
このトークンは、リストされているパラメーターの 1 つ以上の長さが 18 バイトより大きい場合にのみ許可されます。

RDBCOLID
リレーショナル・データベースに含まれているオブジェクトの固有コレクションを識別する 18 から 255 バイトの文字フィールド。これは、ユーザー定義のグループ化に使用されます。

注: RDBNAM、RDBCOLID、または PKGID のいずれかの長さが 18 バイトを超えている場合、SCLDTALEN が必須で、RDBCOLID の前になければなりません。そうしないと、SCLDTALEN は許可されません。

RTNSETSTT(X'210E')
Return SET ステートメントは、いずれかの特殊レジスターの設定がコマンドの実行時に変更されていた場合に、コマンドの処理が正常に完了したときにターゲット・サーバーが 1 つ以上の SQLSTT 応答データ・オブジェクトを返す必要があるかどうかを制御します。各 SQLSTT 応答データ・オブジェクトには、現行接続で設定が変更されている特殊レジスター用の SQL SET ステートメントが含まれています。いずれの特殊レジスターの設定も変更されていない場合は、RTNSETSTT 設定に関係なく、NO SQLSTT 応答データ・オブジェクトは返されません。

形式:



パラメーター:

VALUE
X'00' – ターゲット・サーバーは、SQL SET ステートメントを返してはなりません。

X'01' – ターゲット・サーバーは、設定が変更された特殊レジスターに対して 1 つ以上の SQL SET ステートメントを返す必要があります。

注: IMS は、必ず、Universal ドライバーから 0x'01' を送信します。

MONITOR(X'1900')

FLAGS
4 バイト・フラグ値。

EXCSQLSET の例

以下の例は、OPNQRY 呼び出しに対する要求の一部である EXCSQLSET を示しています。

```
[ibm][ims][drda][t4]        SEND BUFFER: EXCSQLSET             (ASCII)          (EBCDIC)
[ibm][ims][drda][t4]        0 1 2 3 4 5 6 7   8 9 A B C D E F  0123456789ABCDEF  0123456789ABCDEF
[ibm][ims][drda][t4] 0000   004ED05100010048  2014004421134BC9  .N.Q...H ..D!.K.  .+}............I
[ibm][ims][drda][t4] 0010   D4E2F14040404040  4040404040404040  ...@@@@@@@@@@@@@  MS1
[ibm][ims][drda][t4] 0020   D5E4D3D3C9C44040  4040404040404040  ......@@@@@@@@@@  NULLID
[ibm][ims][drda][t4] 0030   4040E2E8E2E2D5F2  F0F0404040404040  @@........@@@@@@    SYSSN200
[ibm][ims][drda][t4] 0040   404040405359534C  564C30310041      @@@@SYSLVL01.A        ...<.<....
```

注: RTNSETSTT および MONITOR は、この例では示されていません。

FLDENTRY コマンド・オブジェクト (X'CC03')

分散データ管理 (DDM) アーキテクチャーの FLDENTRY (フィールド・エントリー) コマンド・オブジェクトを使用して、挿入または更新するフィールドを指定します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC03'、FLDENTRY コマンド・オブジェクトの 2 バイトのコード・ポイント。

RECOFF
階層経路の入出力域の中にあるフィールドのオフセットが入る、必須の 4 バイトの符号付き整数。

FLDVAL
位置 RECOFF から始まる ISRT または REPL DL/I 呼び出しの入出力域に配置される、バイト配列を含む必須ストリング。

使用法

複数の FLDENTRY コマンド・オブジェクトを EXCSQLIMM コマンドにチェーンすることができます。

関連資料:

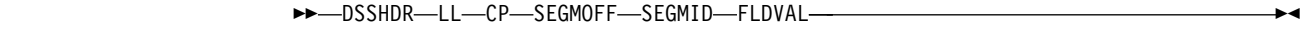
336 ページの『EXCSQLIMM コマンド (X'200A')』

FLDENTRYREL コマンド・オブジェクト (X'CC0C')

分散データ管理 (DDM) アーキテクチャーの FLDENTRYREL (相対フィールド・エントリー) コマンド・オブジェクトを使用して、挿入または更新するフィールドを指定します。

制約事項: FLDENTRYREL コマンド・オブジェクトは、ODBM DDM レベル 1、2、3 または 1、3 でサポートされます。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC0C'、FLDENTRYREL コマンド・オブジェクトの 2 バイトのコード・ポイント。

SEGMOFF
親セグメントの最初からのターゲット・フィールドのオフセットを指定する必須の 4 バイト符号付き整数。

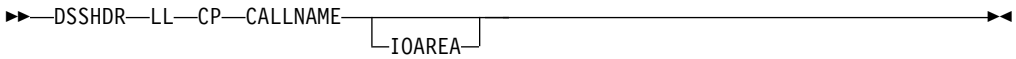
SEGMID
フィールドの参照元となる SEGMLIST 内のセグメントを指定する必須の 1 バイト符号付き整数。この値は 0 にではなく 1 に相対的です。

FLDVAL
更新または挿入されるフィールドの値。

IMSCALL コマンド (X'C803')

分散データ管理 (DDM) アーキテクチャーの IMSCALL コマンドを使用して、IMS DB システム・サービスのための DL/I 呼び出しを発行します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C803'、IMSCALL コマンドの 2 バイトのコード・ポイント。

CALLNAME
実行された DL/I 呼び出しのタイプを表す必須の文字ストリング (コード・ポイントは X'C90C')。

IOAREA
入出力域を指定するバイト配列形式のオプション・パラメーター (コード・ポイントは X'C90B')。

使用法

IMSCALL コマンドは、IMS DB システム・サービスのための DL/I 呼び出しを以下のフォーマットで発行します。

チェーンされるコマンド・オブジェクト

INAIB (X'CC01')
ソース・サーバーからターゲット・サーバーに送信する AIB データ。

正常の応答メッセージ

IMSCALL コマンドに応答して、IMS ターゲット DDM サーバーはソース・サーバーに、以下の応答メッセージを返します。

IMSCALLRM (X'CA04')
IMSCALL コマンドの実行結果が入ります。結果により、IMS DB システム・サービスの DL/I 呼び出しの成功または失敗が示されます。

エラー応答メッセージ

OPNQRY コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下のエラー応答メッセージを返すことがあります。

表 *81. OPNQRY* コマンドで想定される応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'124C'	SYNTAXRM	データ・ストリーム構文エラー
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値

関連資料:

382 ページの『IMSCALLRM 応答メッセージ (X'CA04')』

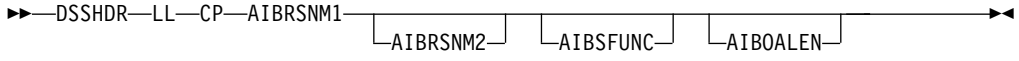
『INAIB コマンド・オブジェクト (X'CC01')』

41 ページの『IMS DB システム・サービスのための DL/I 呼び出し』

INAIB コマンド・オブジェクト (X'CC01')

分散データ管理 (DDM) アーキテクチャーの INAIB (入力 AIB) コマンド・オブジェクトを使用して、ソース・サーバーからターゲット・サーバーに送信する AIB データを入れます。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC01'、INAIB コマンド・オブジェクトの 2 バイトのコード・ポイント。

AIBRSNM1
リソース (PCB) 名が入る必須のストリング。ストリングは左寄せで、ブランクを埋め込んで合計 8 バイトにする必要があります。コード・ポイントは X'C901' です。

AIBRSNM2
ODBA 始動テーブル DFS*xxxx*0 の 4 文字の ID が入るオプションのストリング。ここで *xxxx* は 4 文字の ID です。コード・ポイントは X'C902' です。

AIBSFUNC
副次機能コードが入っているオプションのストリング。ストリングは左寄せで、ブランクを埋め込んで合計 8 バイトにする必要があります。コード・ポイントは X'C903' です。

AIBOALEN
出力の最大長を指定するオプションの 4 バイト整数。このフィールドは、データを返すすべての呼び出しで使用します。コード・ポイントは X'C904' です。

使用法

この AIB コマンド・オブジェクトには、ソース・サーバーからターゲット・サーバーに送信する AIB データのみが入ります。ターゲット・サーバーからソース・サーバーに送信する AIB および DBPCB データは、OUTAIBDBPCB オブジェクト内部の aibStream および dbpcbStream データ構造の中に入ります。

関連資料:

399 ページの『AIBOALEN パラメーター (X'C904')』
399 ページの『AIBRSNM1 パラメーター (X'C901')』
400 ページの『AIBRSNM2 パラメーター (X'C902')』
400 ページの『AIBSFUNC パラメーター (X'C903')』
336 ページの『EXCSQLIMM コマンド (X'200A')』
344 ページの『IMSCALL コマンド (X'C803')』
347 ページの『OPNQRY コマンド (X'200C')』

MONITORRD コマンド (X'1C00')

分散データ管理 (DDM) アーキテクチャーの MONITORRD により、ターゲット・エージェントはソース・エージェントにモニター・データを返すことができます。返される値は、データベース呼び出しのために経過した CPU 時間を判別するために使用されます。

フォーマット

パラメーター

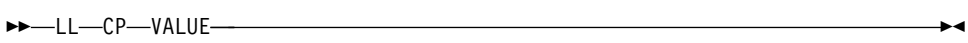
DSSHDR
DSS に関する情報が入る 6 バイトのヘッダー・フィールド。

LL MONITORRD コマンドの長さが入る 2 バイトのフィールド。

CP(X'1C00')
MONITORRD コマンドの 2 バイトのコード・ポイント。

ETIME(X'1901')
経過時間は、64 ビットの 2 進数で表され、マイクロ秒で測定されます。 2 バイト長のフィールド (LL) と 2 バイトのコード・ポイント (CP) から構成され、その後にデータが続きます。長さは 12 バイトです。

形式:



パラメーター:

VALUE
経過時間を表す 8 バイトのフィールド。

MONITORRD の例

以下の例では、通信交換に関するすべての MONITORRD ETIME 値を合計することで、さかのぼってサーバー時間を計算しています。

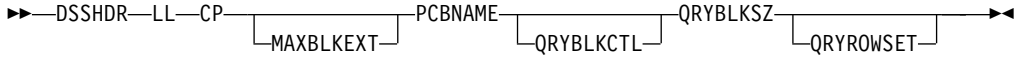
```
[ibm][ims][drda][t4]        RECEIVE BUFFER: MONITORRD          (ASCII)          (EBCDIC)
[ibm][ims][drda][t4] 0000   0016D04300020010  1C00000C19010000  ...C.............  ..}.............
[ibm][ims][drda][t4] 0010   000000036B39                        ....k9           ....,.
[ibm][ims][drda][SystemMonitor:stop] core: 283.09152ms | network: 256.137805ms | server: 254.816ms
```

OPNQRY コマンド (X'200C')

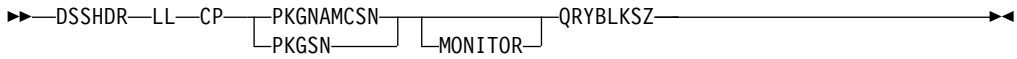
分散データ管理 (DDM) アーキテクチャーの OPNQRY コマンドは、読み取り要求のためのデータベースへの照会をオープンします。

フォーマット

DLI フロー:



SQL フロー:



パラメーター

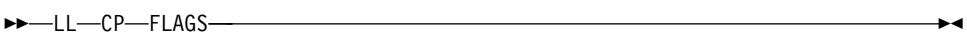
DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'200C'、OPNQRY コマンドの 2 バイトのコード・ポイント。

MAXBLKEXT
OPNQRY または CNTQRY コマンドに応じて、要求側が応答データとして受信できる結果セット単位での、追加ブロックの最大数を指定するオプション・パラメーター。数値は 2 バイトの 2 進数で指定します。値が 0 の場合は、要求側が応答セット・データの追加照会ブロックを受信できないことを示します。値が -1 の場合は、要求側が結果セット全体を受信できることを示します。MAXBLKEXT のコード・ポイントは X'2141' です。

MONITOR(X'1900')



FLAGS
4 バイト・フラグ値。

PCBNAME

DL/I 呼び出しが実行した照会を一意的に識別する PCB 名を指定する必須パラメーター。 PCB 名は文字ストリングで指定します。この値は最初は元の OPNQRY コマンドで送信されます。続いて、元の OPNQRY 呼び出しと正しく相関させるために、CNTQRY、CLSQRY、および RLSE などのコマンドで同じ値を送信する必要があります。 PCBNAME パラメーターのコード・ポイントは X'C907' です。

PKGNAMCSN(X'2113')
SQL の実行に使用するパッケージ内の完全修飾パッケージ名、整合性トークン、およびセクション番号を指定します。 PKGSN とは相互に排他的です。

PKGSN
セクション番号を表す 2 バイトの短フィールド。PKGCNSTKN とは相互に排他的です。

QRYBLKCTL
照会をオープンした場合に使用する照会ブロック・プロトコルのタイプを指定するオプション・パラメーター。IMS は、DDM アーキテクチャーの限定ブロック照会プロトコルのみをサポートします。 QRYBLKCTL パラメーターを OPNQRY コマンドで指定する場合、QRYBLKCTL パラメーターの 2 バイト・データ部分は、限定ブロック照会プロトコル (LMTBLKPRC) のコード・ポイントである 16 進値の X'2417' を指定する必要があります。QRYBLKCTL パラメーターを OPNQRY コマンドから省略した場合でも、IMS ターゲット・サーバーは限定ブロック照会プロトコルを使用します。QRYBLKCTL パラメーターのコード・ポイントは X'2132' です。

QRYBLKSZ
ソース・アプリケーション・プログラムに最適な照会ブロックのサイズを指定する必須パラメーター。照会ブロックは、応答セット・データを返すためにターゲット・サーバーが使用します。ターゲット・サーバーは必要に応じてこのパラメーターを指定変更できます。照会ブロック・サイズは、4 バイトの符号なし 2

進数で指定します。照会ブロックの最小サイズは 0.5 KB です。最大サイズは 10 MB です。 QRYBLKSIZ パラメーターのコード・ポイントは X'2114' です。

QRYROWSET
1 回のネットワーク応答で返すデータの行数を指定するオプション・パラメーター。行数は 4 バイトの 2 進数で指定します。 QRYROWSET の最小値は 0 です。最大値は 32 767 です。QRYROWSET パラメーターのコード・ポイントは X'2156' です。

使用法

OPNQRY 処理中にエラーが発生しなかった場合、IMS ターゲット・サーバーは OPNQRYRM 応答メッセージを返し、照会のオープンが正常に実行されたことを示します。

コマンド・オブジェクト

以下のコマンド・オブジェクトは、OPNQRY コマンドにチェーンされます。

INAIB (X'CC01')
AIB データが入る必須のコマンド・オブジェクト。

注: OPNQRY の INAIB オブジェクトは、ネイティブ SQL 実装環境での DRDA DDM コマンド・サポートでは使用されません。

DLIFUNC (X'CC05')
データベースで取るアクションを指定する、必須のコマンド・オブジェクト。 DLIFUNC のデータ・フィールドは、文字ストリングで指定するデータベース関数です。DLIFUNC の有効な値は、OPNQRY コマンドにチェーンされている場合、RETRIEVE、GHU、GU、GHN、GN、GNP、または GHNP です。

注: OPNQRY の DLIFUNC オブジェクトは、ネイティブ SQL 実装環境での DRDA DDM コマンド・サポートでは使用されません。

RTRVLFD (X'CC04')
クライアントが取り出すフィールドを表す、オプションのスカラー・データ・オブジェクト。複数の RTRVFLD オブジェクトを OPNQRY コマンドにチェーンできます。 RTRVFLD オブジェクトが OPNQRY コマンドに含まれていない場合、取り出したセグメントのすべてのフィールドが返されます。

注: OPNQRY の RTRVLFD オブジェクトは、ネイティブ SQL 実装環境での DRDA DDM コマンド・サポートでは使用されません。

SSALIST (X'CC06')
セグメント検索引数をリストするオプションのチェーン・オブジェクト。 SSALIST が OPNQRY コマンドに含まれていない場合、IMS ターゲット・サーバーは、IMS データベースの非修飾ステップスルーにおいて、RTRVFLD チェーン・オブジェクトおよび照会結果を無視します。

注: OPNQRY の SSALIST オブジェクトは、ネイティブ SQL 実装環境での DRDA DDM コマンド・サポートでは使用されません。

正常の応答メッセージ

OPNQRY コマンドに応答して、IMS ターゲット DDM サーバーはソース・サーバーに、以下の正常の応答メッセージを返します。

OPNQRYRM (X'2205')
照会のオープン応答メッセージ。

応答データ・オブジェクト

CNTQRY コマンドに応答して、以下の応答データ・オブジェクトが返されます。

QRYDSC (X'241A')
照会応答セット記述。

QRYDTA (X'241B')
照会応答セット・データ。

エラー応答メッセージ

OPNQRY コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下のエラー応答メッセージを返すことがあります。

表 *82. OPNQRY* コマンドで想定される応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'121C'	CMDATHRM	許可されていないコマンド
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'1233'	RSCLMTRM	限度に達したリソース
X'1245'	PRCCNVRM	会話型プロトコル・エラー
X'124C'	SYNTAXRM	データ・ストリーム構文エラー
X'1250'	CMDNSPRM	サポートされていないコマンド
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値
X'1253'	OBJNSPRM	サポートされていないオブジェクト
X'1254'	CMDCHKRM	コマンド検査応答メッセージ
X'125F'	TRGNSPRM	サポートされていないターゲット
X'2204'	RDBNACRM	アクセスされていないデータベース
X'220A'	DSCINVRM	無効な記述
X'220B'	ENDQRYRM	照会終了
X'220D'	ABNUOWRM	作業単位の異常終了状態
X'220E'	DTAMCHRM	データ記述子の不一致

表 82. *OPNQRY* コマンドで想定される応答メッセージ (続き)

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'220F'	QRYPOPRM	オープン済み照会
X'2212'	OPNQFLRM	照会のオープン失敗
X'2218'	RDBUPDRM	データベース更新応答メッセージ

OPNQRY の例

```
OPNQRY only example:
[ibm][ims][drda][t4]        SEND BUFFER: OPNQRY                (ASCII)           (EBCDIC)
[ibm][ims][drda][t4] 0000   005BD00100030055  200C004421134BC9  .[.....U ..D!.K.  .$}............I
[ibm][ims][drda][t4] 0010   D4E2F14040404040  4040404040404040  ...@@@@@@@@@@@@@  MS1
[ibm][ims][drda][t4] 0020   D5E4D3D3C9C44040  4040404040404040  ......@@@@@@@@@@  NULLID
[ibm][ims][drda][t4] 0030   4040E2E8E2E2D5F2  F0F0404040404040  @@........@@@@@@    SYSSN200
[ibm][ims][drda][t4] 0040   404040405359534C  564C303100010008  @@@@SYSLVL01....      ...<.<......
[ibm][ims][drda][t4] 0050   2114000000000005  215D01            !.......!].       .........).

OPNQRY complete chained request example for SQL SELECT:
[ibm][ims][drda][t4]        SEND BUFFER: EXCSQLSET             (ASCII)           (EBCDIC)
[ibm][ims][drda][t4]        0 1 2 3 4 5 6 7   8 9 A B C D E F  0123456789ABCDEF  0123456789ABCDEF
[ibm][ims][drda][t4] 0000   004ED05100010048  2014004421134BC9  .N.Q...H ..D!.K.  .+}............I
[ibm][ims][drda][t4] 0010   D4E2F14040404040  4040404040404040  ...@@@@@@@@@@@@@  MS1
[ibm][ims][drda][t4] 0020   D5E4D3D3C9C44040  4040404040404040  ......@@@@@@@@@@  NULLID
[ibm][ims][drda][t4] 0030   4040E2E8E2E2D5F2  F0F0404040404040  @@........@@@@@@    SYSSN200
[ibm][ims][drda][t4] 0040   404040405359534C  564C30310041      @@@@SYSLVL01.A        ...<.<....
[ibm][ims][drda][t4]
[ibm][ims][drda][t4]        SEND BUFFER: SQLSTT                (ASCII)           (EBCDIC)
[ibm][ims][drda][t4] 0000   0031D0430001002B  2414002353455420  .1.C...+$..#SET   ..}.............
[ibm][ims][drda][t4] 0010   434C49454E542057  524B53544E4E414D  CLIENT WRKSTNNAM  .<..+.......++.(
[ibm][ims][drda][t4] 0020   452027392E36352E  3137342E32352700  E '9.65.174.25'.  ................
[ibm][ims][drda][t4] 0030   00                                  .                 .
[ibm][ims][drda][t4]
[ibm][ims][drda][t4]        SEND BUFFER: PRPSQLSTT             (ASCII)           (EBCDIC)
[ibm][ims][drda][t4] 0000   0058D05100020052  200D004421134BC9  .X.Q...R ..D!.K.  ..}............I
[ibm][ims][drda][t4] 0010   D4E2F14040404040  4040404040404040  ...@@@@@@@@@@@@@  MS1
[ibm][ims][drda][t4] 0020   D5E4D3D3C9C44040  4040404040404040  ......@@@@@@@@@@  NULLID
[ibm][ims][drda][t4] 0030   4040E2E8E2E2D5F2  F0F0404040404040  @@........@@@@@@    SYSSN200
[ibm][ims][drda][t4] 0040   404040405359534C  564C303100010005  @@@@SYSLVL01....      ...<.<......
[ibm][ims][drda][t4] 0050   2116F10005214604                    !....!F.          ..1.....
[ibm][ims][drda][t4]
[ibm][ims][drda][t4]        SEND BUFFER: SQLATTR               (ASCII)           (EBCDIC)
[ibm][ims][drda][t4] 0000   001CD05300020016  2450000E464F5220  ...S....$P..FOR   ..}......&...|..
[ibm][ims][drda][t4] 0010   52454144204F4E4C  59200000          READ ONLY ..      .....|+<....
[ibm][ims][drda][t4]
[ibm][ims][drda][t4]        SEND BUFFER: SQLSTT                (ASCII)           (EBCDIC)
[ibm][ims][drda][t4] 0000   0029D04300020023  2414001B53656C65  .).C...#$...Sele  ..}...........%.
[ibm][ims][drda][t4] 0010   6374202A2066726F  6D20504844414D56  ct * from PHDAMV  .......?_.&...(.
[ibm][ims][drda][t4] 0020   41522E7761726400  00                AR.ward..         ..../....
[ibm][ims][drda][t4]
[ibm][ims][drda][t4]        SEND BUFFER: OPNQRY                (ASCII)           (EBCDIC)
[ibm][ims][drda][t4] 0000   005BD00100030055  200C004421134BC9  .[.....U ..D!.K.  .$}............I
[ibm][ims][drda][t4] 0010   D4E2F14040404040  4040404040404040  ...@@@@@@@@@@@@@  MS1
[ibm][ims][drda][t4] 0020   D5E4D3D3C9C44040  4040404040404040  ......@@@@@@@@@@  NULLID
[ibm][ims][drda][t4] 0030   4040E2E8E2E2D5F2  F0F0404040404040  @@........@@@@@@    SYSSN200
[ibm][ims][drda][t4] 0040   404040405359534C  564C303100010008  @@@@SYSLVL01....      ...<.<......
[ibm][ims][drda][t4] 0050   2114000000000005  215D01            !.......!].       .........).
```

関連資料:

383 ページの『OPNQFLRM 応答メッセージ (X'2212')』

332 ページの『DLIFUNC コマンド・オブジェクト (X'CC05')』

345 ページの『INAIB コマンド・オブジェクト (X'CC01')』
356 ページの『RTRVFLD コマンド・オブジェクト (X'CC04')』
371 ページの『SSALIST コマンド・オブジェクト (X'CC06')』
384 ページの『OPNQRYRM 応答メッセージ (X'2205')』
388 ページの『QRYPOPRM 応答メッセージ (X'220F')』

PRPSQLSTT コマンド (X'200D')

分散データ管理 (DDM) アーキテクチャーの SQL ステートメントの準備 (PRPSQLSTT) コマンドは、SQL ステートメントを既存のデータベース (RDB) パッケージ内のセクションに動的にバインドします。

フォーマット

パラメーター

DSSHDR
DSS に関する情報が入る 6 バイトのヘッダー・フィールド。

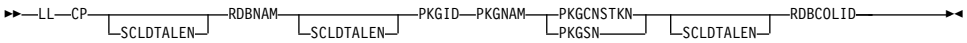
LL PRPSQLSTT コマンドの長さが入る 2 バイトのフィールド。

CP(X'200D')
PRPSQLSTT コマンドの 2 バイトのコード・ポイント。

PKGNAMCSN(X'2113')
SQL の実行に使用するパッケージ内の完全修飾パッケージ名、整合性トークン、およびセクション番号を指定します。 PKGNAMCSN は、含まれている RDBNAM、RDBCOLID、および PKGID の長さに応じて以下のいずれかの形式にすることができます。

- RDBNAM、RDBCOLID、および PKGID の長さがそれぞれ 18 である場合。この形式の PKGNAMCSN は、DDM レベル 7 より前で使用されていた唯一の形式 (長さ 68 に固定) と同一です。この形式で SCLDTALEN を使用することは許可されません。
- RDBNAM、RDBCOLID、または PKGID の 1 つ以上の長さが 18 より大きい場合。この形式の PKGNAMCSN では、各 RDBNAM、RDBCOLID、および PKGID の前に SCLDTALEN が必要です。この形式を使用する場合、PKGNAMCSN の最小長は 75、最大長は 785 です。

形式:



パラメーター:

RDBNAM
リレーショナル・データベース名を表す 18 から 255 バイトの文字フィールド。

PKGID
リレーショナル・データベース・パッケージ ID を表す 18 から 255 バイトの文字フィールド。

PKGNAM
リレーショナル・データベース・パッケージの完全修飾名を指定する 18 から 255 バイトの文字フィールド。

PKGCNSTKN
要求側のアプリケーションとリレーショナル・データベース・パッケージが同期化されているかを検証する 8 バイトの文字フィールド。 PKGSN とは相互に排他的です。

PKGSN
セクション番号を表す 2 バイトの短フィールド。PKGCNSTKN とは相互に排他的です。

SCLDTALEN
インスタンス変数の長さを指定します。直後に以下が続きます。

- RDB コレクション ID (RDBCOLID)
- リレーショナル・データベース名 (RDBNAM)
- RDB パッケージ ID (PKGID)

このトークンは、リストされているパラメーターの 1 つ以上の長さが 18 バイトより大きい場合にのみ許可されます。

注: RDBNAM、RDBCOLID、または PKGID のいずれかの長さが 18 バイトを超えている場合、SCLDTALEN が必須で、3 つの各パラメーター RDBNAM、RDBCOLID、および PKGID の前になければなりません。そうしないと、SCLDTALEN は許可されません。

RDBCOLID
リレーショナル・データベースに含まれているオブジェクトの固有コレクションを識別する 18 から 255 バイトの文字フィールド。これは、ユーザー定義のグループ化に使用されます。

RTNSQLDA(X'2116')
Return SQL Descriptor Area は、適用する SQL 記述子域をこのコマンドで指定する SQL ステートメントに返すかどうかを制御します。ターゲット SQLAM は、ステートメントの準備が完了した後にそのステートメントに対して SQL DESCRIBE 関数を実行することで、SQL 記述子域を取得します。

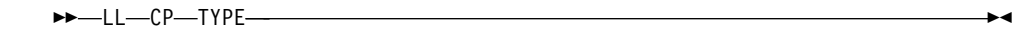
パラメーター:

VALUE
TRUE (X'F1') – SQLIMSDA が返されることを示します。

FALSE (X'F0') – SQLIMSDA が返されないことを示します。

注: IMS は、必ず、Universal ドライバーから 0x'01' を送信します。

TYPSQLDA(X'2146')
SQL 記述子域のタイプ。



パラメーター:

TYPE
コマンドに対して返される SQLIMSDA のタイプを指定する 1 バイトの符号付き数値。

0 標準出力 SQLIMSDA。このタイプは、ODBM でサポートされます。

1 標準入力 SQLIMSDA。このタイプは、ODBM でサポートされます。

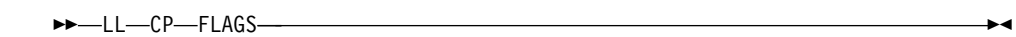
2 簡易出力 SQLIMSDA

3 簡易入力 SQLIMSDA

4 拡張出力 SQLIMSDA

5 拡張入力 SQLIMSDA

MONITOR(X'1900')



FLAGS
4 バイト・フラグ値。

PRPSQLSTT の例

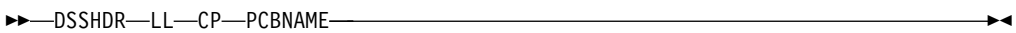
以下の例は、OPNQRY 呼び出しに対する要求の一部である PRPSQLSTT を示しています。

```
[ibm][ims][drda][t4]       SEND BUFFER: PRPSQLSTT             (ASCII)          (EBCDIC)
[ibm][ims][drda][t4] 0000  0058D05100020052  200D004421134BC9  .X.Q...R ..D!.K.  ..}............I
[ibm][ims][drda][t4] 0010  D4E2F14040404040  4040404040404040  ...@@@@@@@@@@@@@  MS1
[ibm][ims][drda][t4] 0020  D5E4D3D3C9C44040  4040404040404040  ......@@@@@@@@@@  NULLID
[ibm][ims][drda][t4] 0030  4040E2E8E2E2D5F2  F0F0404040404040  @@........@@@@@@    SYSSN200
[ibm][ims][drda][t4] 0040  4040404053595340  564C303100010005  @@@@SYSLVL01....      ...<.<......
[ibm][ims][drda][t4] 0050  2116F10005214604                    !....!F.          ..1.....
```

RLSE コマンド (X'C802')

分散データ管理 (DDM) アーキテクチャーの RLSE コマンドを使用して、アプリケーションが保持しているデータベース・ロックを解放します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C802'、RLSE コマンドの 2 バイトのコード・ポイント。

PCBNAME

DL/I 呼び出しが実行した照会を一意的に識別する PCB 名を指定する必須パラメーター。 PCB 名は文字ストリングで指定します。この値は最初は元の OPNQRY コマンドで送信されます。続いて、元の OPNQRY 呼び出しと正しく相関させるために、CNTQRY、CLSQRY、および RLSE などのコマンドで同じ値を送信する必要があります。 PCBNAME パラメーターのコード・ポイントは X'C907' です。

チェーンされるコマンド・オブジェクト

RLSE コマンドにチェーンされるコマンド・オブジェクトはありません。

正常の応答メッセージ

RLSE コマンドに応答して、IMS ターゲット DDM サーバーはソース・サーバーに、以下の正常の応答メッセージを返します。

RLSERM (X'CA03')

ロック解放応答メッセージは、RLSE コマンドが正常に完了したことを要求側に示します。

チェーンされる応答データ・オブジェクト

RLSE コマンドの応答として返される応答データ・オブジェクトはありません。

エラー応答メッセージ

RLSE コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下のエラー応答メッセージを返すことがあります。

表 *83. RLSE* コマンドの、想定されるエラー応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'121C'	CMDATHRM	許可されていないコマンド
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'1233'	RSCLMTRM	限度に達したリソース
X'124C'	SYNTAXRM	データ・ストリーム構文エラー
X'1250'	CMDNSPRM	サポートされていないコマンド
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値
X'1254'	CMDCHKRM	コマンド検査応答メッセージ

表 *83. RLSE* コマンドの、想定されるエラー応答メッセージ *(*続き*)*

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'125F'	TRGNSPRM	サポートされていないターゲット

関連資料:

394 ページの『RLSERM 応答メッセージ (X'CA03')』

RTRVFLD コマンド・オブジェクト (X'CC04')

分散データ管理 (DDM) アーキテクチャーの RTRVFLD コマンド・オブジェクトを使用して、クライアントがデータを取得しようとするフィールドを指定します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC04'、RTRVFLD コマンド・オブジェクトの 2 バイトのコード・ポイント。

RECOFF
4 バイトの符号付き整数。DL/I 呼び出しで返される階層経路の入出力域内にあるフィールドのオフセットが入ります。

FLDLEN
フィールドの長さが入る 4 バイト符号付き整数。

関連資料:

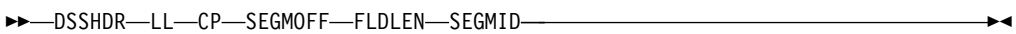
347 ページの『OPNQRY コマンド (X'200C')』

RTRVFLDREL コマンド・オブジェクト (X'CC0B')

分散データ管理 (DDM) アーキテクチャーの FLDENTRYREL (相対フィールド・エントリー) コマンド・オブジェクトを使用して、クライアントでデータを検索するフィールドを指定します。

制約事項**:** RTRVFLDREL コマンド・オブジェクトは、ODBM DDM レベル 1、2、3 または 1、3 でのみサポートされます。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC0B'、RTRVFLDREL コマンド・オブジェクトの 2 バイトのコード・ポイント。

SEGMOFF
親セグメントの最初からのターゲット・フィールドのオフセットを指定する必須の 4 バイト符号付き整数。

SEGMID
フィールドの参照元となる SEGMLIST 内のセグメントを指定する必須の 1 バイト符号付き整数。この値は 0 にではなく 1 に相対的です。

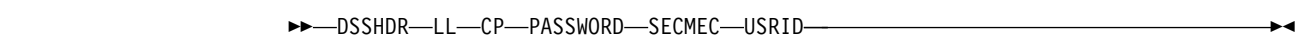
FLDLEN
ターゲット・フィールドの長さ。

SECCHK コマンド (X'106E')

分散データ管理 (DDM) アーキテクチャーの SECCHK コマンドは、ソース・サーバーから IMS ターゲット・サーバーのターゲット・セキュリティー・マネージャーにユーザー情報を渡し、RACF や他のセキュリティー製品を使用してユーザーを認証します。

セキュリティー検査が IMS ターゲット・サーバーでアクティブの場合は、SECCHK コマンドより先に ACCSEC コマンドを実行する必要があります。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'106E'、SECCHK コマンドの 2 バイトのコード・ポイント。

SECMEC
ソース・サーバーとターゲット・サーバーが承認しているセキュリティー・メカニズムを指定する必須パラメーター。 IMS の場合は、USRIDPWD を指定します。

セキュリティー・メカニズムは、ACCSEC コマンドおよび ACCSECRD 応答オブジェクトを使用して、ソース・サーバーとターゲット・サーバーの間でネゴシエーションします。

USRID
ソース・アプリケーション・プログラムのユーザー ID を文字ストリングで指定する、必須の可変長パラメーター。長さは 1 文字から 255 文字まで指定できます。

PASSWORD
ソース・アプリケーション・プログラムのパスワードを文字ストリングで指定する、必須の可変長パラメーター。長さは 1 文字から 255 文字まで指定できます。

使用法

IMS は、ユーザー ID とパスワードを使用してセキュリティーを検査します。このため、SECMEC パラメーターの値には、DDM USRIDPWD セキュリティー・メカニズムを指定します。

SECCHK コマンド処理中にエラーが発生しなかった場合、IMS ターゲット・サーバーは SECCHKRM 応答メッセージを返し、セキュリティー情報を受け入れ可能であることを示します。

SECCHK コマンドより先に ACCSEC コマンドを実行する必要があります。

チェーンされるコマンド・オブジェクト

SECCHK コマンドにチェーンできるコマンド・オブジェクトはありません。

正常の応答メッセージ

SECCHK コマンドに応答して、IMS ターゲット DDM サーバーはソース・サーバーに、以下の正常の応答メッセージを返します。

SECCHKRM (X'1219')
セキュリティー検査応答メッセージ。

エラー応答メッセージ

SECCHK コマンドに応答して、IMS ターゲット DDM サーバーはソース DDM サーバーに、以下のエラー応答メッセージを返すことがあります。

表 *84. SECCHK* コマンドの、想定されるエラー応答メッセージ

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'1218'	MGRDEPRM	マネージャー依存関係エラー
X'121C'	CMDATHRM	許可されていないコマンド
X'1232'	AGNPRMRM	永続的なエージェント・エラー
X'1233'	RSCLMTRM	限度に達したリソース
X'123C'	INVRQSRM	要求が無効
X'1245'	PRCCNVRM	会話型プロトコル・エラー
X'124C'	SYNTAXRM	データ・ストリーム構文エラー

表 *84. SECCHK* コマンドの、想定されるエラー応答メッセージ (続き)

応答メッセージのコード・ポイント	応答メッセージの名前	応答メッセージの意味
X'1250'	CMDNSPRM	サポートされていないコマンド
X'1251'	PRMNSPRM	サポートされていないパラメーター
X'1252'	VALNSPRM	サポートされていないパラメーター値
X'1253'	OBJNSPRM	サポートされていないオブジェクト
X'1254'	CMDCHKRM	コマンド検査応答メッセージ
X'125F'	TRGNSPRM	サポートされていないターゲット

SEGMLIST コマンド・オブジェクト (X'CC0A')

分散データ管理 (DDM) アーキテクチャーの SEGMLIST (セグメント・リスト) コマンド・オブジェクトは、挿入または更新する各フィールドの最小長または最大長を指定するために使用します。

制約事項: SEGMLIST コマンド・オブジェクトは、ODBM DDM レベル 1、3 または 1、2、3 でのみサポートされます。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC0A'、SEGMLIST コマンド・オブジェクトの 2 バイトのコード・ポイント。

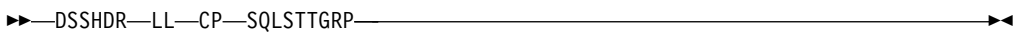
COUNT
検索または更新されるレコード内のセグメントの数を数える 1 バイトの符号付き値。1 レコードのセグメントの数は 15 に制限されます。COUNT パラメーターの値は、コマンド・オブジェクトに含まれる MINMAX パラメーターのインスタンスの数に対応します。この値は必須です。

MINMAX
2 つの 4 バイト符号付き整数に分割された 8 バイトフィールド。最初の整数はセグメントのバイトの最小数であり、2 番目の整数はバイトの最大数です。これらの整数が等しい場合、セグメントは固定長です。

SQLATTR コマンド (X'2450')

分散データ管理 (DDM) アーキテクチャーの SQL ステートメント属性 (SQLATTR) コマンドは、準備する SQL ステートメント属性を指定します。

フォーマット



パラメーター

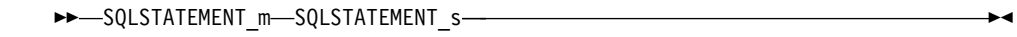
DSSHDR
DSS に関する情報が入る 6 バイトのヘッダー・フィールド。

LL SQLATTR コマンドの長さが入る 2 バイトのフィールド。

CP(X'2450')
SQLATTR コマンドの 2 バイトのコード・ポイント。

SQLSTTGRP
SQL ステートメント・グループ記述。

形式:



パラメーター:

SQLSTATEMENT_m
SQL ステートメントを含む可変長ストリング。

SQLSTATEMENT_s
SQL ステートメントを含む可変長ストリング。

SQLATTR の例

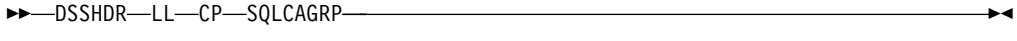
以下の例は、OPNQRY 呼び出しに対する要求の一部である SQLATTR を示しています。

```
[ibm][ims][drda][t4]        SEND BUFFER: SQLATTR               (ASCII)          (EBCDIC)
[ibm][ims][drda][t4] 0000   001CD05300020016  2450000E464F5220  ...S....$P..FOR   ..}......&...|..
[ibm][ims][drda][t4] 0010   52454144204F4E4C  59200000          READ ONLY ..      .....|+<....
```

SQLCARD コマンド (X'2408')

分散データ管理 (DDM) アーキテクチャーの SQL 連絡域を伴う SQL 記述子域の行記述 (SQLCARD) コマンドは、連絡域とともに取得する列に関するメタデータ情報を提供します。

フォーマット



パラメーター

DSSHDR
DSS に関する情報が入る 6 バイトのヘッダー・フィールド。

LL SQLCARD コマンドの長さが入る 2 バイトのフィールド。

CP(X'2408')
SQLCARD コマンドの 2 バイトのコード・ポイント。

SQLCAGRP
SQL 連絡域グループ記述。

形式:

パラメーター:

FLAG
SQLCAGRP の値が NULL であるかを判別する 1 バイトのフィールド。NULL 標識は、値 X'FF' で表示されます。

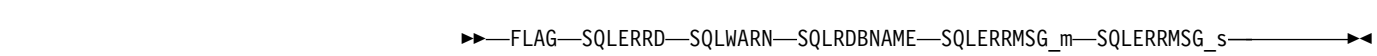
SQLCODE
各 SQL ステートメントの完了後にデータベース・マネージャーによって送信される戻りコードが入る 4 バイトの整数フィールド。

SQLSTATE
直前に実行された SQL ステートメントの結果が入る 5 バイトの文字フィールド。

SQLERRPROC
SQLIMSCODE によって報告されたエラーを検出した CSECT の名前が入る 8 バイトの文字フィールド。

SQLCAXGRP
SQL 連絡域例外グループ記述。

形式:



パラメーター:

FLAG
SQLCAXGRP の値が NULL であるかを判別する 1 バイトのフィールド。 NULL 標識は、値 X'FF' で表示されます。

SQLERRD
6 個の 1 バイトの整数フィールド。これらの値は、エラー状態の診断に使用されます。

SQLWARN
11 個の 1 バイトの文字フィールド。SQLIMSWARN0 から SQLIMSWARNA を表します。

SQLRDBNAME
リモート・データベースの名前を示す可変文字ストリング。

SQLERRMSG_m
X'FF' で区切られた 1 つ以上のトークンを含む可変文字ストリング。エラー状態の記述では変数に置き換わります。切り捨てられたトークンを含んでいる場合があります。メッセージ長が 70 バイトの場合は、切り捨てが行われた可能性があることを示しています。

SQLERRMSG_s
X'FF' で区切られた 1 つ以上のトークンを含む可変文字ストリング。エラー状態の記述では変数に置き換わります。切り捨てられたトークンを含んでいる場合があります。メッセージ長が 70 バイトの場合は、切り捨てが行われた可能性があることを示しています。

SQLDIAGGRP
SQL 記述子オプション・グループ記述

形式:

パラメーター:

FLAG
SQLCAXGRP の値が NULL であるかを判別する 1 バイトのフィールド。 NULL 標識は、値 X'FF' で表示されます。

SQLDIAGSTT
SQL 診断ステートメント・グループ記述。

SQLDIAGCI
SQL 診断条件情報配列。

SQLDIAGCN
SQL 診断接続配列。

SQLCARD の例

以下の例は、OPNQRY 呼び出しに対する要求の一部である SQLCARD を示しています。

```
[ibm][ims][drda][t4]        RECEIVE BUFFER: SQLCARD             (ASCII)           (EBCDIC)
[ibm][ims][drda][t4] 0000   0059D05300030053  2408006400000030  .Y.S...S$..d...0  ..}.............
[ibm][ims][drda][t4] 0010   3230303053514C52  4930314600010004  2000SQLRI01F....  ......<.........
[ibm][ims][drda][t4] 0020   8001000000000000  0000000000000000  ................  ................
[ibm][ims][drda][t4] 0030   0000000000202020  2020202020202020  .....             ................
[ibm][ims][drda][t4] 0040   001253414D504C45  2020202020202020  ..SAMPLE          ....(&<.........
[ibm][ims][drda][t4] 0050   2020202000000000  FF
```

SQLDARD コマンド (X'2411')

分散データ管理 (DDM) アーキテクチャーの SQL 連絡域を伴う SQL 記述子域の行記述 (SQLDARD) コマンドは、連絡域とともに取得する列に関するメタデータ情報を提供します。

フォーマット

パラメーター

DSSHDR
DSS に関する情報が入る 6 バイトのヘッダー・フィールド。

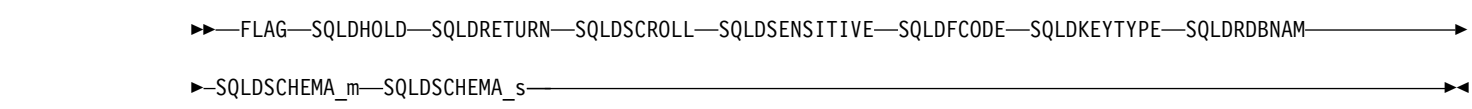
LL SQLDARD コマンドの長さが入る 2 バイトのフィールド。

CP(X'2411')
SQLDARD コマンドの 2 バイトのコード・ポイント。

SQLCARD
SQL 連絡域行記述

SQLDHGRP
SQL 記述子ヘッダー・グループ記述 (結果セットのすべてのフィールドに適用される列メタデータ)。

形式:



パラメーター:

FLAG
SQLDHGRP の値が NULL であるかを判別する 1 バイトのフィールド。NULL 標識は、値 X'FF で表示されます。

SQLDHOLD
2 バイトの短フィールド。このフィールドには、値 0 または 1 が入ります。値 1 は、このステートメントが WITH HOLD 文節を使用して定義されたカーソルに関連していることを示します。それ以外の場合、値は 0。

SQLDRETURN
2 バイトの短フィールド。

SQLDSCROLL
2 バイトの短フィールド。

SQLDSENSITIVE
2 バイトの短フィールド。

SQLDFCODE
2 バイトの短フィールド。

SQLDKEYTYPE
2 バイトの短フィールド。

SQLDRDBNAM
リモート・データベースの名前を示す可変文字ストリング。

SQLDSCHEMA_m
スキーマの名前を示す可変文字ストリング。

SQLDSCHEMA_s
スキーマの名前を示す可変文字ストリング。

SQLNUMBRP
エレメント・グループ記述の SQL 番号。

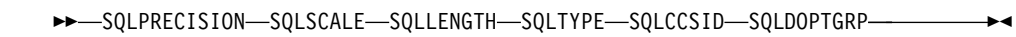
形式:

パラメーター:

Number of Columns
照会によって返される列数を表す 2 バイトの短フィールド。

SQLDAGRP
SQL データ域グループ記述 (各列に固有の列メタデータ)。

形式:



パラメーター:

SQLPRECISION
列の精度を表す 2 バイトの短フィールド。

SQLSCALE
列のスケールを表す 2 バイトの短フィールド。

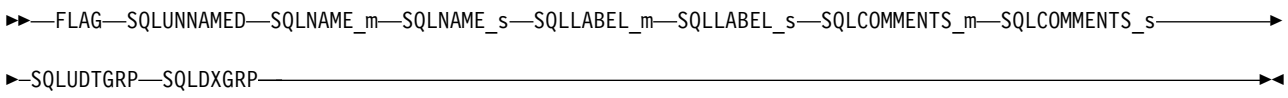
SQLLENGTH
列の長さ (バイト数) を表す 8 バイトのフィールド。

SQLTYPE
列のデータ・タイプを表す 2 バイトの短フィールド。

SQLCCSID
列の CCSID を表す 2 バイトの短フィールド。

SQLDOPTGRP
SQL 記述子オプション・グループ記述

形式:



パラメーター:

FLAG
SQLDHGRP の値が NULL であるかを判別する 1 バイトのフィールド。 NULL 標識は、値 X'FF' で表示されます。

SQLUNNAMED
2 バイトの短フィールド。

SQLNAME_m
列名を示す可変文字ストリング。

SQLNAME_s
列名を示す可変文字ストリング。

SQLLABEL_m
可変文字ストリング

SQLLABEL_s
可変文字ストリング

SQLCOMMENTS_m
可変文字ストリング

SQLCOMMENTS_s
可変文字ストリング

SQLUDTGRP:
SQL ユーザー定義データ・グループ記述

形式:

パラメーター:

FLAG
SQLDHGRP の値が NULL であるかを判別する 1 バイトのフィールド。 NULL 標識は、値 X'FF' で表示されます。

SQLUDTXTYPE
4 バイトの整数フィールド。

SQLUDTRDB
可変文字ストリング。

SQLUDTSCHEMA_m
可変文字ストリング。

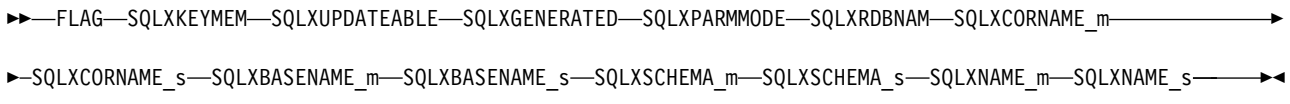
SQLUDTSCHEMA_s
可変文字ストリング。

SQLUDTNAME_m
可変文字ストリング。

SQLUDTNAME_s
可変文字ストリング。

SQLDXGRP
SQL 記述子拡張グループ記述。

形式:



パラメーター:

FLAG
SQLDHGRP の値が NULL であるかを判別する 1 バイトのフィールド。 NULL 標識は、値 X'FF' で表示されます。

SQLXKEYMEM
2 バイトの短フィールド。

SQLXUPDATEABLE
2 バイトの短フィールド。

SQLXGENERATED
2 バイトの短フィールド。

SQLXPARMMODE
2 バイトの短フィールド。

SQLXRDBNAM
リモート・データベースの名前を示す可変文字ストリング。

SQLXCORNAME_m
テーブルの名前を示す可変文字ストリング。

SQLXCORNAME_s
テーブルの名前を示す可変文字ストリング。

SQLXBASENAME_m
テーブルの名前を示す可変文字ストリング。

SQLXBASENAME_s
テーブルの名前を示す可変文字ストリング。

SQLXSCHEMA_m
スキーマの名前を示す可変文字ストリング。

SQLXSCHEMA_s
スキーマの名前を示す可変文字ストリング。

SQLXNAME_m
列の名前を示す可変文字ストリング。

SQLXNAME_s
列の名前を示す可変文字ストリング。

SQLDARD の例

以下の例は、OPNQRY 呼び出しに対する要求の一部である SQLDARD を示しています。

```
[ibm][ims][drda][t4]        RECEIVE BUFFER: SQLDARD            (ASCII)          (EBCDIC)
[ibm][ims][drda][t4] 0000   0185D0530002017F  2411000000000030  ...S....$......0  .e}...."........
[ibm][ims][drda][t4] 0010   3030303053514C30  3930373000000000  0000SQL09070....  ......<.........
[ibm][ims][drda][t4] 0020   0000000000010000  0040010000000000  .........@......  ......... ......
[ibm][ims][drda][t4] 0030   0000000000202020  2020202020202020  .....             ................
[ibm][ims][drda][t4] 0040   001253414D504C45  2020202020202020  ..SAMPLE          ....(&<.........
[ibm][ims][drda][t4] 0050   2020202000000000  FF00010000000000      ............  ................
[ibm][ims][drda][t4] 0060   0000550000000000  0000000003000000  ..U.............  ................
[ibm][ims][drda][t4] 0070   0000040000000000  0000F10100000000  ................  ..........1.....
[ibm][ims][drda][t4] 0080   000005454D504E4F  0000000000000000  ...EMPNO........  ....(&+|........
[ibm][ims][drda][t4] 0090   0000FF0000000000  0000000000065341  ..............SA  ................
[ibm][ims][drda][t4] 00A0   4D504C450006454D  504D444300000006  MPLE..EMPMDC....  (&<....(&(......
[ibm][ims][drda][t4] 00B0   454D504D44430000  0008524943485452  EMPMDC....RICHTR  .(&(............
[ibm][ims][drda][t4] 00C0   414E00000005454D  504E4F0000000000  AN....EMPNO.....  .+.....(&+|.....
```

```
[ibm][ims][drda][t4] 00D0   0004000000000000  00F1010000000000  ................  .........1......
[ibm][ims][drda][t4] 00E0   0004444550540000  0000000000000000  ..DEPT..........  ....&...........
[ibm][ims][drda][t4] 00F0   FF00000000000000  0000000653414D50  ............SAMP  ..............(&
[ibm][ims][drda][t4] 0100   4C450006454D504D  444300000006454D  LE..EMPMDC....EM  <....(&(.......(
[ibm][ims][drda][t4] 0110   504D444300000008  524943485452414E  PMDC....RICHTRAN  &(.............+
[ibm][ims][drda][t4] 0120   0000000444455054  0000000000000400  ....DEPT........  ......&.........
[ibm][ims][drda][t4] 0130   000000000000F101  0000000000000344  ...............D  ......1.........
[ibm][ims][drda][t4] 0140   4956000000000000  00000000FF000000  IV..............  ................
[ibm][ims][drda][t4] 0150   0000000000000006  53414D504C450006  ........SAMPLE..  ..........(&<...
[ibm][ims][drda][t4] 0160   454D504D44430000  0006454D504D4443  EMPMDC....EMPMDC  .(&(.......(&(..
[ibm][ims][drda][t4] 0170   0000000852494348  5452414E00000003  ....RICHTRAN....  ...........+....
[ibm][ims][drda][t4] 0180   4449560000                        DIV..             .....
```

SQLDTA コマンド (X'2412')

SQL プログラム変数データ (SQLDTA) は、リレーショナル・データベース (RDB) が実行している SQL ステートメントへの入力データで構成されます。これには、データの記述も含まれます。

フォーマット

パラメーター

DSSHDR
DSS に関する情報が入る 6 バイトのヘッダー・フィールド。

LL SQLDTA コマンドの長さが入る 2 バイトのフィールド。

CP(X'2412')
SQLDTA コマンドの 2 バイトのコード・ポイント。

FDOEXT(X'147B')
定様式データ・オブジェクト体系記述子 (FDODSC) が記述する各 SDA のエクステント・データが入るスカラー・オブジェクト。 FDODSC の SQLDTAGRP には、各フィールド定義の FDOEXT 項目があります。 FDODSC のフィールド定義に対応する FDOEXT 指定は、FDODTA オブジェクトでそのフィールドが繰り返される回数を定義します。

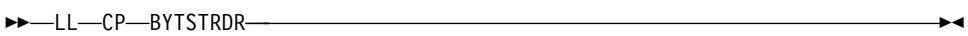
形式:

パラメーター:

BYTSTRDR
必須バイト・ストリング・データ表記。長さは偶数 (X'01010101' や X'3D2B11' など) です。

FDODSC(X'0010')
ストリング。このストリングは、値が定様式データ・オブジェクト・コンテンツ体系 (FD:OCA) 記述子であるか、FD:OCA 記述子のセグメントである DDM スカラーです。 FDODSC は、別のスカラー・オブジェクトに含まれているデータ・フィールドについて記述した 1 つ以上の FD:OCA トリプレットから構成されます。

形式:

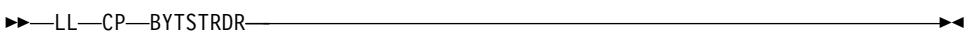


パラメーター:

BYTSTRDR
必須バイト・ストリング・データ表記。長さは偶数 (X'01010101' や X'3D2B11' など) です。

FDODTA(X'147A')
定様式データ・オブジェクト体系記述子 (FDODSC) が記述するデータが入るスカラー・オブジェクト。 FDODSC は、定様式データ・オブジェクト・データ (FDODTA) と一緒に存在する場合と、FDODTA が使用されるコマンドのコンテキストに基づいて暗黙的に定義される場合があります。

形式:



パラメーター:

BYTSTRDR
必須バイト・ストリング・データ表記。長さは偶数 (X'01010101' や X'3D2B11' など) です。

FDOOFF(X'147D')
定様式データ・オブジェクト体系記述子 (FDODSC) が記述する各 SDA のオフセット・データが入るスカラー・オブジェクト。 FDODSC の SQLDTAGRP には、各フィールド定義の FDOOFF 項目があります。 FDODSC のフィールド定義に対応する FDOOFF 指定は、FDODTA 内のデータ項目の開始に対するオフセットを定義します。最初のデータ配列のオフセット値は 0 です。

形式:

パラメーター:

BYTSTRDR
必須バイト・ストリング・データ表記。長さは偶数 (X'01010101' や X'3D2B11' など) です。

定様式データ・オブジェクト・コンテンツ体系 (FD:OCA)

定様式データ・オブジェクト・コンテンツ体系 (FD:OCA) は、データの線形ストリームを記述、編成、および操作するためのアーキテクチャーです。

DDM は、主にリレーショナル・データベース (RDB) アクセスのためのデータの記述に FD:OCA を使用します。 FD:OCA の完全な記述 (FD:OCA 記述子のビルドおよび解釈方法を含む) は、FD:OCA リファレンス内にあります。

FD:OCA の関数は、トリプレットと呼ばれるデータ構造から構成される FD:OCA 記述子を介して指定されます。属性トリプレットは、データ・ストリーム内のデー

タの表記およびレイアウトを記述します。ジェネレーター・トリプレットは、出力データ・ストリームを生成するためにどのようにデータを操作するかを指定します。

FD:OCA を含むアーキテクチャー (DDM など) は、データ・ストリームおよび FD:OCA 記述子を通信システム間で送信し、必要に応じて表示プロセスを開始します。表示プロセスは、データ・ストリームと FD:OCA 記述子の両方を入力として受け入れ、表示ストリームを出力として生成します (図 3 から 39 を参照)。FD:OCA を含むアーキテクチャーは、以降のすべての表示ストリームを処理します。例えば、DMM はストレージの表示ストリームを RDB に転送したり、表示ストリームをアプリケーション・リクエスターに渡したりすることができます。

選択済み **FD:OCA** トリプレットの概要

FD:OCA は、現行レベルの DDM アーキテクチャーで必要な属性およびジェネレーター・トリプレットの他に、多くの属性およびジェネレーター・トリプレットを定義します。以下は、トリプレットの概要を示しています。

スカラー・データ配列 **(SDA)**
: SDA は、FD:OCA が単一項目、または同じ形式を持つ単一項目の線形配列あるいは長方形配列のいずれかであるデータ値を記述するために使用するトリプレットです。 DDM は、主にデータ表記指定を DDM および SQL のデータ・タイプに関連付けるために、SDA を使用します。

グループ・データ配列 **(GDA)**
: GDA は、データ項目のグループを参照可能単位として定義するトリプレットです。 GDA のエレメントは、(ラベルによって) SDA、他の GDA、または RLO を指して、グループのデータ項目を記述します。 GDA は、各データ表記の特定の属性をオーバーライドすることができます。

行レイアウト **(RLO)**
: RLO は、以下のものを記述するトリプレットです。

  - 1 つ以上のタイプのフィールドが含まれる行
  - 1 つ以上のタイプの行が含まれるテーブル
  - マルチディメンションの混合データ構造

  RLO は、複数の無関係な構造から構成されるデータ・ストリームを記述します。 DDM は、主に SQL ステートメントが返す応答データを記述するために、RLO を使用します。

FD:OCA 記述子
: FD:OCA 記述子は、バイト・ストリームで連続的に配列された 1 つ以上のトリプレットから構成されます。他のトリプレットから参照されるトリプレットは、参照元のトリプレットより前に配置する必要があります。参照されないトリプレットは無視されます。

SQLDTA の例

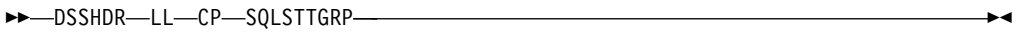
以下の例は、OPNQRY 呼び出しに対する要求の一部である SQLDTA を示しています。

```
[ibm][db2][jcc][t4]        SEND BUFFER: SQLDTA                (ASCII)          (EBCDIC)
[ibm][db2][jcc][t4] 0000   002CD00300040026  2412001300100976  .,.....&$......v  ..}.............
[ibm][db2][jcc][t4] 0010   D003000403000406  71E4D00001000F14  ........q.......  }........U}.....
[ibm][db2][jcc][t4] 0020   7A00000000000200  00000005          z...........      :...........
```

SQLSTT コマンド (X'2414')

分散データ管理 (DDM) アーキテクチャーの SQL ステートメントの行記述 (SQLSTT) コマンドには、SQL ステートメントが 1 つ含まれています。

フォーマット



パラメーター

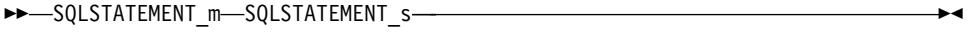
DSSHDR
DSS に関する情報が入る 6 バイトのヘッダー・フィールド。

LL SQLSTT コマンドの長さが入る 2 バイトのフィールド。

CP(X'2414')
SQLSTT コマンドの 2 バイトのコード・ポイント。

SQLSTTGRP
SQL ステートメント・グループ記述。

形式:



パラメーター:

SQLSTATEMENT_m
SQL ステートメントを含む可変長ストリング。

SQLSTATEMENT_s
SQL ステートメントを含む可変長ストリング。

SQLSTT の例

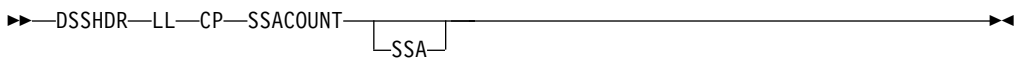
以下の例は、OPNQRY 呼び出しに対する要求の一部である SQLSTT を示しています。

```
SQLSTT for Special Registry:
[ibm][ims][drda][t4]        SEND BUFFER: SQLSTT                (ASCII)          (EBCDIC)
[ibm][ims][drda][t4] 0000   0032D0430001002C  2414002453455420  .2.C...,$..$SET   ..}.............
[ibm][ims][drda][t4] 0010   434C49454E542057  524B53544E4E414D  CLIENT WRKSTNNAM  .<..+.......++.(
[ibm][ims][drda][t4] 0020   452027392E36352E  3135302E32313827  E '9.65.150.218'  ................
[ibm][ims][drda][t4] 0030   0000                                ..                ..

SQLSTT for Select Statement:
[ibm][ims][drda][t4]        SEND BUFFER: SQLSTT                (ASCII)          (EBCDIC)
[ibm][ims][drda][t4] 0000   0029D04300020023  2414001B53656C65  .).C...#$...Sele  ..}...........%.
[ibm][ims][drda][t4] 0010   6374202A2066726F  6D20504844414D56  ct * from PHDAMV  .......?_.&...(.
[ibm][ims][drda][t4] 0020   41522E7761726400  00                AR.ward..         ..../....
```

SSALIST コマンド・オブジェクト (X'CC06')

分散データ管理 (DDM) アーキテクチャーの SSALIST コマンド・オブジェクトを使用して、DL/I 呼び出しを修飾するセグメント検索引数 (SSA) オブジェクトのリストを入れます。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC06'、SSALIST コマンド・オブジェクトの 2 バイトのコード・ポイント。

SSACOUNT
必須。2 バイト値で指定する、SSAList にある SSA の数。有効な範囲は 1 から 15 までです。

SSA
DL/I データベース呼び出しで使用する SSA オブジェクトが入る、オプションのバイト・ストリング。

関連資料:

336 ページの『EXCSQLIMM コマンド (X'200A')』

347 ページの『OPNQRY コマンド (X'200C')』

406 ページの『SSA パラメーター (X'C906')』

407 ページの『SSACOUNT パラメーター (X'C905')』

DDM 応答メッセージおよび応答オブジェクト

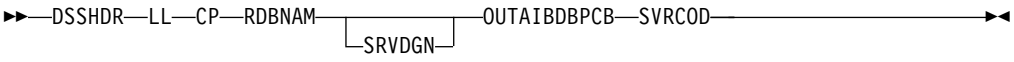
IMS ターゲット・サーバーは、分散データ管理 (DDM) アーキテクチャーに定義されている応答メッセージを使用して、ソース・サーバー・アプリケーションと通信します。 IMS が使用する一部の応答メッセージには、IMS 固有のパラメーター、値、または応答オブジェクトが入ります。

ABNUOWRM 応答メッセージ (X'220D')

分散データ管理 (DDM) アーキテクチャーの ABNUOWRM (作業単位の異常終了状態) 応答メッセージは、ターゲット・サーバーでの何らかのアクションが原因で現行の作業単位が異常終了したことを示します。

この応答メッセージは、デッドロック解決、オペレーター介入、またはデータベースが現行の作業単位をロールバックすることになる同様の状態の結果として出される場合があります。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'220D'、ABNUOWRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

SVRCOD

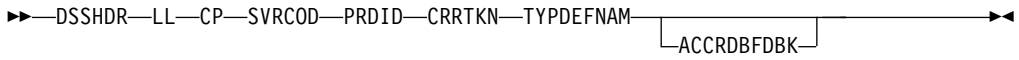
重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

ACCRDBRM 応答メッセージ (X'2201')

分散データ管理 (DDM) アーキテクチャーの ACCRDBRM (データベースへのアクセス完了) 応答メッセージは、以前に ACCRDB コマンドで指定されたデータベースが、クライアントの処理に使用可能であることを示します。

フォーマット



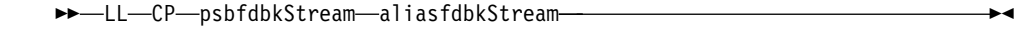
パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2201'、ACCRDBRM 応答メッセージの 2 バイトのコード・ポイント。

ACCRDBFDBK
ターゲットからソースに送信される A**psbfdbkStream** データおよび **aliasfdbkStream** データが入るオプション・パラメーター (X'CC0D')。



パラメーター

LL ACCRDBFDBK の長さ。この長さには LL および CP が含まれます。

CP X'CC0D'。ACCRDBFDBK の 2 バイトのコード・ポイント。

パラメーター **psbfdbkStream** および **aliasfdbkStream** のデータ構造は、以下のとおりです。

表 *85.* パラメーター **psbfdbkStream** および **aliasfdbkStream** のデータ構造

オフセット	長さ	名前	説明
0	1	PSB 名 NULL 標識	2 進整数 • X'00' - データの残りが後に続くことを示します。 • X'FF' - データがなく、長さの合計が 1 バイトであることを示します。
1	1	長さ	2 進整数 • 長さそのものを含めたデータの長さ。
2	8	PSB 名	文字ストリング PSB 名の長さ。

表 *85.* パラメーター **psbfdbkStream** および **aliasfdbkStream** のデータ構造 (続き)

オフセット	長さ	名前	説明
1 または 10	1	別名 NULL 標識	2 進整数 • X'00' - データの残りが後に続くことを示します。 • X'FF' - データがなく、長さの合計が 1 バイトであることを示します。
2 または 11	1	長さ	2 進整数 • 長さそのものを含めたデータの長さ。
3 または 12	4	別名	文字ストリング 別名の長さ。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

PRDID

製品固有 ID。 DDM サーバーのリリース・レベルです。

これは ACCRDB コマンドの必須パラメーターです。

CRRTKN

ソースおよびターゲット・サーバーがアプリケーションの処理を相関させるために使用する必須の相関トークン。

TYPDEFNAM

データ型定義の名前を指定する必須パラメーター (X'002F')。 TYPDEFNAM は、2 バイト仕様の長さ (LL)、2 バイトのコード・ポイント (CP)、および値で構成されます。この値は予約されており、QTDSQL370 である必要があります。これは、EBCDIC ストリング、IEEE 浮動小数点数、およびバイト逆順でない浮動小数点数と整数を使用するマシンのための、一般的な EBCDIC SQL 型定義です。

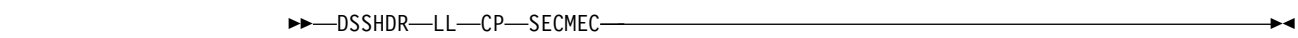
関連資料:

323 ページの『ACCRDB コマンド (X'2001')』

ACCSECRD 応答オブジェクト (X'14AC')

分散データ管理 (DDM) アーキテクチャーの ACCSECRD (アクセス・セキュリティー応答データ) 応答オブジェクトには、ターゲット・サーバーのセキュリティー・マネージャーからのセキュリティー情報が入っています。この情報は ACCSEC コマンドに応答して返されます。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'14AC'、ACCSECRD 応答オブジェクトの 2 バイトのコード・ポイント。

SECMEC
セキュリティー・メカニズム、またはターゲット・サーバーがサポートするメカニズムを指定します。ソース DDM サーバーが ACCSEC コマンドで指定している DDM セキュリティー・メカニズムをターゲット・サーバーがサポートしている場合、ACCSECRD 応答オブジェクトの SECMEC の値は、ACCSEC コマンドの SECMEC の値と同じになります。ソース・サーバーが指定しているセキュリティー・メカニズムをターゲット・サーバーがサポートしない場合、SECMEC には、ターゲット・サーバーがサポートするセキュリティー・メカニズムを識別する値が 1 つ以上入っています。

使用法

交換が成功すると、IMS ターゲット・サーバーは、ACCSEC コマンドで実行依頼されたものと同じ SECMEC パラメーターの値を持つ ACCSECRD 応答データ・オブジェクトを返すことにより、ソース・アプリケーション・プログラムが要求したセキュリティー検査のタイプを確認します。

ACCSEC コマンドの処理中にエラーが検出されると、SECCHKCD が ACCSECRD とともに返されます。 SECCHKCD パラメーターには、暗黙の重大度コード ERROR があります。 SECCHK コマンドを送信して接続の認証をする前に、ACCSEC コマンドを再送信して、ACCSECRD の新しいインスタンス変数のセットを生成する必要があります。

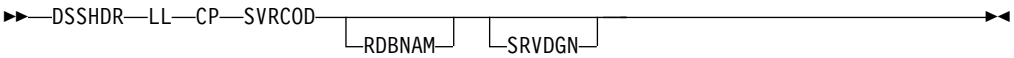
関連資料:

325 ページの『ACCSEC コマンド (X'106D')』

AGNPRMRM 応答メッセージ (X'1232')

分散データ管理 (DDM) アーキテクチャーの AGNPRMRM (永続エージェント・エラー) 応答メッセージは、ターゲット・システムが永続エラー状態を検出したために、要求したコマンドが完了できなかったことを示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'1232'、AGNPRMRM 応答メッセージの 2 バイトのコード・ポイント。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

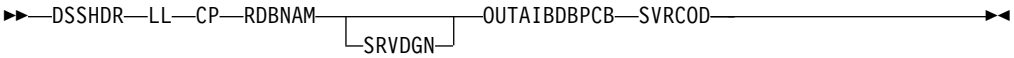
SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

CMDVLTRM 応答メッセージ (X'221D')

分散データ管理 (DDM) アーキテクチャーの CMDVLTRM (コマンド違反) 応答メッセージは、DDM コマンドが会話の処理機能に違反したことを示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'221D'、CMDVLTRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

DEALLOCDBRM 応答メッセージ (X'CA01')

分散データ管理 (DDM) アーキテクチャーの DEALLOCDBRM (データベース割り振り解除完了) 応答メッセージは、指定された PSB が割り振り解除されていることを示します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CA01'、DEALLOCDBRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

関連資料:

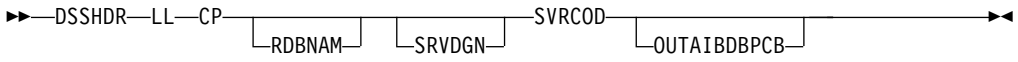
330 ページの『DEALLOCDB コマンド (X'C801')』
377 ページの『DEALLOCDBRM 応答メッセージ (X'CA01')』

ENDQRYRM 応答メッセージ (X'220B')

分散データ管理 (DDM) アーキテクチャーの ENDQRYRM (照会終了) 応答メッセージは、照会処理が終了し、照会または結果セットがクローズしたことを示します。

照会は、CNTQRY コマンドを使用して再開できません。また CLSQRY コマンドを使用してクローズできません。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'220B'、ENDQRYRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

使用法

ENDQRYRM 応答メッセージは、以下の状態の場合には、OPNQRYRM 応答メッセージにチェーンすることが必要です。

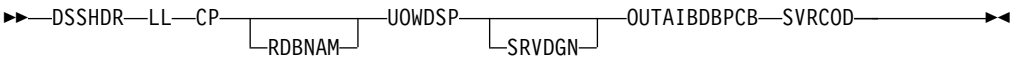
- GU、GN、またはバッチ RETRIEVE 呼び出しへの応答時にデータが返されない場合。クライアントが必要とする aibdbpcbStream データは、このオブジェクトに入ります。

- バッチ RETRIEVE 呼び出しの応答時に、QRYDTA オブジェクトに照会を満たすデータの最終行が入っている場合。これはソース・サーバーにとって、ODBM が最後の GN 呼び出しで GE/GB 状況コードを受信しており、CNTQRY を送信するべきではないことを示します。

ENDUOWRM 応答メッセージ (X'220C')

分散データ管理 (DDM) アーキテクチャーの ENDUOWRM (最終作業単位) 応答メッセージは、最終コマンドの実行結果として、作業単位が終了したことを示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'220C'、ENDUOWRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

UOWDSP
最終作業単位の処理を指定する必須パラメーター (X'2115')。処理がコミットされた場合、作業単位のすべての更新は正常に適用されています。処理がロールバックされた場合、作業単位のすべての更新は削除されます。

UOWDSP の詳細については、*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*を参照してください。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

SVRCOD

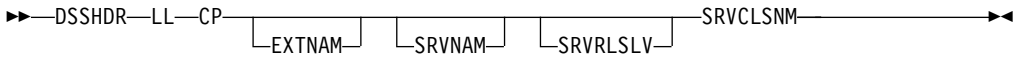
重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

EXCSATRD 応答オブジェクト (X'1443')

EXCSATRD 応答データ・オブジェクトは、サーバー名や製品リリース・レベルなどの、IMS ターゲット DDM サーバーに関する情報を、ソース DDM サーバーに返します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL EXCSATRD 応答オブジェクトの 2 バイト仕様の長さ。

CP X'1443'、EXCSATRD 応答オブジェクトの 2 バイトのコード・ポイント。

EXTNAM
オプション。ターゲット DDM サーバーの可変長の外部名。 IMS ターゲット DDM サーバーの場合、外部名は、IMS システムが DDM サーバーを実行するために作成または活動化するジョブの名前です。 EXTNAM パラメーターは、トレースおよび問題判別のために使用します。ジョブ名に埋め込みブランクが含まれている場合、その名前はデータ・フィールド内では引用符で囲む必要があります。 EXTNAM の最大長は 255 バイトです。コード・ポイントは X'115E' です。

SRVNAM
オプション。文字ストリングで指定する、ターゲット DDM サーバーの可変長の名前。トレースおよび問題判別のために返されます。サーバー名に埋め込みブランクが含まれている場合、その名前は引用符で囲む必要があります。最大長は 255 バイトです。コード・ポイントは X'116D' です。

DDM サーバー名に埋め込みブランクが含まれている場合、その名前はデータ・フィールド内では引用符で囲む必要があります。

SRVRLSLV
オプション。ターゲット DDM サーバーの製品リリース・レベルの可変長の名前。 SRVRLSLV パラメーターは、アプリケーション・プログラムのソース・

サーバーと IMS ターゲット・サーバーとの互換性を確保するために使用します。SRVRLSLV の最大長は 255 バイトです。コード・ポイントは X'115A' です。

SRVCLSNM
IMS が使用する DDM サーバー・クラス名 (DFS) を指定します。DFS は、現在 IMS がサポートしている唯一の名前です。SRVCLSNM は、IMS が使う DRDA 製品固有の拡張機能を使用できます。

SRVCLSNM のコード・ポイントは X'1147' です。可変長の DDM サーバー・クラス名は文字ストリングで指定します。

使用法

分散データ管理 (DDM) アーキテクチャーの EXCSATRD 応答オブジェクトは、EXCSAT コマンドの応答として IMS ターゲット・サーバーにより返されます。エラーが発生しなかった場合、EXCSATRD 応答オブジェクトは必ず、IMS ターゲット DDM サーバーからソース DDM サーバーに返される最初の応答コマンドになります。

サーバー属性の交換中にエラーが発生した場合、EXCSATRD 応答オブジェクトの代わりにエラー・メッセージを発行して、IMS ターゲット・サーバーは EXCSAT コマンドに応答します。

関連資料:

334 ページの『EXCSAT コマンド (X'1041')』

IMSCALLRM 応答メッセージ (X'CA04')

分散データ管理 (DDM) アーキテクチャーの IMSCALLRM (IMS 呼び出し) 応答メッセージは、IMSCALL コマンドを使用して実行される、IMS DB システム・サービスのための DL/I 呼び出しの結果を返します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CA04'、IMSCALLRM 応答メッセージの 2 バイトのコード・ポイント。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード

4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

aib

必須パラメーターであり、以下の 2 つの応答オブジェクトのいずれか 1 つためのプレースホルダー。

- OUTAIBDBPCB

  ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

- OUTAIBIOPCB

  ターゲットからソースに送信される AIB および IOPCB データが入るパラメーター (X'CC08')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

IOAREA

入出力域を指定するバイト配列形式のオプション・パラメーター。

関連資料:

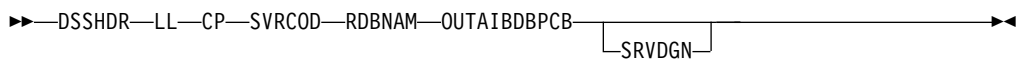
344 ページの『IMSCALL コマンド (X'C803')』

OPNQFLRM 応答メッセージ (X'2212')

分散データ管理 (DDM) アーキテクチャーの OPNQFLRM (照会のオープン失敗) 応答メッセージは、OPNQRY コマンドが照会のオープンに失敗したことを示します。

失敗理由は、OUTAIBDBPCB パラメーターで報告されます。

フォーマット



パラメーター

DSSHDR

データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2212'、OPNQFLRM 応答メッセージの 2 バイトのコード・ポイント。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

OUTAIBDBPCB には失敗の理由が入ります。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

関連資料:

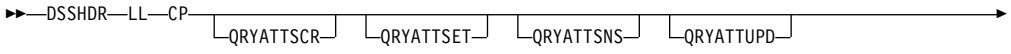
347 ページの『OPNQRY コマンド (X'200C')』

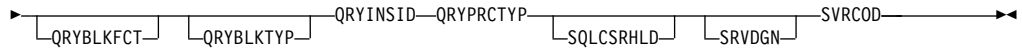
OPNQRYRM 応答メッセージ (X'2205')

分散データ管理 (DDM) アーキテクチャーの OPNQRYRM (照会のオープン) 応答メッセージは、照会のオープン (OPNQRY) コマンドまたは SQL ステートメントの実行 (EXCSQLSTT) コマンドが正常に完了し、照会処理が開始されたことを示します。

この応答メッセージは、照会で使用する照会プロトコルおよびカーソルのタイプも示します。

フォーマット





パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2205'、OPNQRYRM 応答メッセージの 2 バイトのコード・ポイント。

QRYATTSCR
スクロール可能性の照会属性を指定するオプション・パラメーター。コード・ポイントは X'2149' です。

QRYATTSET
カーソルが単一行または複数行を返すことができるかどうかを示すオプション・パラメーター。コード・ポイントは X'214A' です。

QRYATTSNS
基礎となる基本表に加えられる変更に対するオープン・カーソルの感度を指定するオプション・パラメーター。コード・ポイントは X'2157' です。

QRYATTUPD
オープン・カーソルの更新可能性を示すオプション・パラメーター。コード・ポイントは X'2150' です。

QRYBLKFCT
ブロック化因数の値が入るオプション・パラメーター。これは 1 回の照会でブロック化できる行数を指定する、ターゲット・サーバーが課す制限です。コード・ポイントは X'215F' です。

QRYBLKTYP
応答セット・データを返す照会ブロックのタイプを示すオプション・パラメーター。コード・ポイントは X'2133' です。

QRYINSID
照会のインスタンスを一意的に識別する必須パラメーター。コード・ポイントは X'215B' です。

このパラメーターは、照会をオープンした場合にターゲット・サーバーが OPNQRYRM 応答メッセージで返します。ターゲット・サーバーによるこの照会に対する後続のすべての参照には、照会の正しいインスタンスを識別するために、QRYINSID 値が含まれている必要があります。

QRYPRCTYP
使用する照会プロトコルのタイプを指定する必須のストリング・パラメーター。コード・ポイントは X'2102' です。

SQLCSRHLD
要求側がカーソル位置保持オプションを指定しているかどうかを示す、オプションのブール・パラメーター。コード・ポイントは X'211F' です。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

これらのパラメーターの詳細については、The Open Group の「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

応答データ・オブジェクト

OPNQRY コマンドに応答して、以下の応答データ・オブジェクトが OPNQRYRM メッセージにチェーンされます。

QRYDSC (X'241A')
照会応答セット記述。

QRYDTA (X'241B')
照会応答セット・データ。

関連資料:

347 ページの『OPNQRY コマンド (X'200C')』

QRYDSC 応答オブジェクト (X'241A')

分散データ管理 (DDM) アーキテクチャー QRYDSC (照会応答セット記述) 応答オブジェクトは、QRYDTA オブジェクトで返されるデータのフォーマットを定義します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'241A'、QRYDSC 応答オブジェクトの 2 バイトのコード・ポイント。

BYTSTRDR
必須のバイト・ストリング・データ表記。このバイト・ストリングには、QRYDTA オブジェクトが送信するデータの FD:OCA 記述が入ります。

使用法

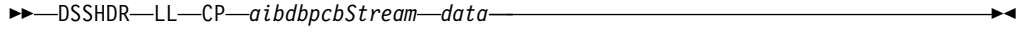
QRYDTA オブジェクトが返すデータの形式が変更されることはありません。ODBM の BYTSTRDR 値は常に以下のようになります。

0676D0260000 0671E0D000010671 F0E00000

QRYDTA 応答オブジェクト (X'241B')

分散データ管理 (DDM) アーキテクチャー QRYDTA (照会応答セット・データ) 応答オブジェクトには、照会の結果である応答セット・データの一部またはすべてが入ります。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'241B'、QRYDTA 応答オブジェクトの 2 バイトのコード・ポイント。

aibStream
IMS 定義のデータ構造。

Open Database Manager (ODBM) サービスが GU 呼び出しまたは GN 呼び出しを処理すると、aibStream データ・ストリームが QRYDTA オブジェクトの各行の最初に配置され、その後に dbpcbStream が続きます。その直後に要求した行データが続きます。 aibdbpcbStream とデータ・フィールドの連結は、照会行セット内の単一行を構成します。

dbpcbStream
IMS 定義のデータ構造。

Open Database Manager (ODBM) サービスが GU 呼び出しまたは GN 呼び出しを処理すると、aibStream データ・ストリームが QRYDTA オブジェクトの各行の最初に配置され、その後に dbpcbStream が続きます。その直後に要求した行データが続きます。 aibdbpcbStream とデータ・フィールドの連結は、照会行セット内の単一行を構成します。

データ
aibStream および dbpcbStream データ構造の後に続くデータ。

使用法

QRYDTA 応答オブジェクトの内容は、QRYDSC 応答オブジェクトにより記述されます。 IMS は、指定行のすべてのデータを「固定長バイト・ストリーム」タイプの単一カラムであるかのように送信するため、QRYDSC 情報はどの照会の場合でも同じです。各行は、aibdbpcbStream オブジェクトとそれに続くデータで構成されます。

関連資料:

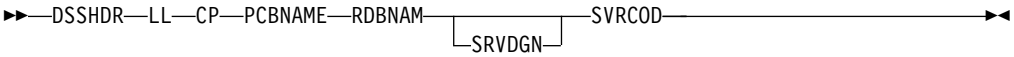
401 ページの『aibStream データ構造』

401 ページの『dbpcbStream データ構造』

QRYPOPRM 応答メッセージ (X'220F')

分散データ管理 (DDM) アーキテクチャーの QRYPOPRM (オープン済み照会) 応答メッセージは、コマンドが既にオープンしている照会に対して発行された場合に返されます。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'220F'、QRYPOPRM 応答メッセージの 2 バイトのコード・ポイント。

PCBNAME

DL/I 呼び出しが実行した照会を一意的に識別する PCB 名を指定する必須パラメーター。 PCB 名は文字ストリングで指定します。この値は最初は元の OPNQRY コマンドで送信されます。続いて、元の OPNQRY 呼び出しと正しく相関させるために、CNTQRY、CLSQRY、および RLSE などのコマンドで同じ値を送信する必要があります。 PCBNAME パラメーターのコード・ポイントは X'C907' です。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

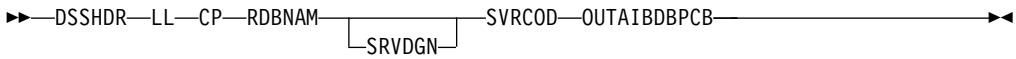
関連資料:

347 ページの『OPNQRY コマンド (X'200C')』

RDBAFLRM 応答メッセージ (X'221A')

分散データ管理 (DDM) アーキテクチャーの RDBAFLRM (データベース・アクセス失敗) 応答メッセージは、データベース・アクセスが失敗したことを示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'221A'、RDBAFLRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

OUTAIBDBPCB には失敗の理由が入ります。

使用法

RDBAFLRM 応答メッセージは、RDBNAM 値を指定してデータベースへの接続が失敗した場合のみ返されます。 RDBAFLRM 応答メッセージが返されると、ターゲットの Structured Query Language Application Manager (SQLAM) インスタンスは破棄されます。 SQLAM インスタンスの詳細については、The Open Group の「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

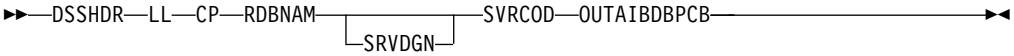
関連資料:

323 ページの『ACCRDB コマンド (X'2001')』
406 ページの『RDBNAM パラメーター (X'2110')』
404 ページの『OUTAIBDBPCB パラメーター (X'CC02')』

RDBATHRM 応答メッセージ (X'2203')

分散データ管理 (DDM) アーキテクチャーの RDBATHRM (データベース・アクセス非許可) 応答メッセージは、ユーザーにデータベース・アクセスが許可されていないことを示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2203'、RDBATHRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

OUTAIBDBPCB には失敗の理由が入ります。

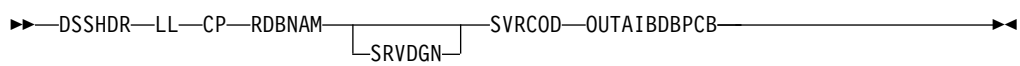
関連資料:

323 ページの『ACCRDB コマンド (X'2001')』

RDBNACRM 応答メッセージ (X'2204')

分散データ管理 (DDM) アーキテクチャーの RDBNACRM (データベース非アクセス) 応答メッセージは、データベース・サービスを要求するコマンドの前に、データベースのアクセス・コマンド (ACCRDB) が発行されていないことを示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2204'、RDBNACRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

OUTAIBDBPCB には失敗の理由が入ります。

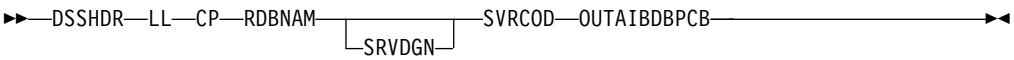
関連資料:

323 ページの『ACCRDB コマンド (X'2001')』

RDBNFNRM 応答メッセージ (X'2211')

分散データ管理 (DDM) アーキテクチャーの RDBNFNRM (データベース非検出) 応答メッセージは、要求したデータベースが見つからなかったことを示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2211'、RDBNFNRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

OUTAIBDBPCB には失敗の理由が入ります。

RDBUPDRM 応答メッセージ (X'2218')

分散データ管理 (DDM) アーキテクチャーの RDBUPDRM (データベース更新) 応答メッセージは、DDM コマンドによりターゲット・データベースが更新される結果になったことを示します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2218'、RDBUPDRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

UPDCNT

必須の 4 バイト整数パラメーター (X'C90A')。バッチあるいは singleton の INSERT、UPDATE、または DELETE 処理の影響を受ける行数が入ります。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

OUTAIBDBPCB には詳細情報が入ります。

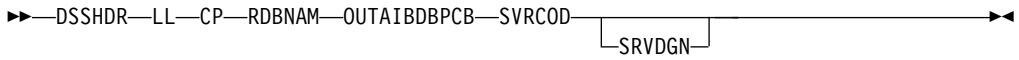
関連資料:

336 ページの『EXCSQLIMM コマンド (X'200A')』
407 ページの『UPDCNT パラメーター (X'C90A')』

RLSERM 応答メッセージ (X'CA03')

分散データ管理 (DDM) アーキテクチャーの RLSERM (ロック解除) 応答メッセージは、RLSE コマンドが正常に完了したことを要求側に示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CA03'、RLSERM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

関連資料:

354 ページの『RLSE コマンド (X'C802')』

RSCLMTRM 応答メッセージ (X'1233')

分散データ管理 (DDM) アーキテクチャーの RSCLMTRM (限度に達したリソース) 応答メッセージは、ターゲット・サーバーのリソース不足のため、要求したコマンドが完了できなかったことを示します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'1233'、RSCLMTRM 応答メッセージの 2 バイトのコード・ポイント。

PRDID
ソース DDM サーバーのリリース・レベルを指定するオプション・パラメーター。 PRDID のコード・ポイントは X'112E' です。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

RSCNAM
オプションのストリング・パラメーター。応答として、限度に達したリソースの名前を示し、RSCLMTRM 応答メッセージを送信します。 RSCNAM のコード・ポイントは X'112D' です。

RSCTYP
オプションのストリング・パラメーター。限度に達したリソースのタイプを示し、RSCLMTRM 応答メッセージを送信します。 RSCTYP のコード・ポイントは X'111F' です。

RSNCOD
理由コードを示すオプションのストリング・パラメーター。 RSNCOD のコード・ポイントは X'1127' です。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード

32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

SECCHKRM 応答メッセージ (X'1219')

分散データ管理 (DDM) アーキテクチャーの SECCHKRM (セキュリティー検査) 応答メッセージは、セキュリティー情報が受け入れ可能であることを示します。

セキュリティー・マネージャーは、セキュリティー検査コード (SECCHKCD) パラメーターを使用して、セキュリティー情報の状態を示します。

フォーマット

パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'1219'、SECCHKRM 応答メッセージの 2 バイトのコード・ポイント。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード
32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

SECCHKCD

SECCHKRM 応答メッセージのセキュリティー情報および状態を示す必須のストリング・パラメーター。 SECCHKRM 応答メッセージの、指定可能なコード値、および SECCHKCD と SVRCOD の関係についての詳細は、The Open Group の「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細につ

いては、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

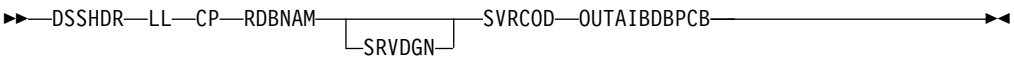
SVCERRNO
ローカル・セキュリティー・サービスのセキュリティー・サービス・エラー番号が入るオプション・パラメーター (X'11B4')。 SRVDGN には追加情報が入る場合があります。

SVCERRNO は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント、続いて 4 バイトの 2 進数データで構成されます。

SQLERRRM 応答メッセージ (X'2213')

分散データ管理 (DDM) アーキテクチャーの SQLERRRM (SQL エラー状態) 応答メッセージは、SQL エラーが発生したことを示します。

フォーマット



パラメーター

DSSHDR
データ・ストリーム構造 (DSS) に関する情報が入る 6 バイトのヘッダー。

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'2213'、SQLERRRM 応答メッセージの 2 バイトのコード・ポイント。

RDBNAM

コード・ポイント X'2110'。ターゲット・データベースを識別する IMS PSB 名が入るオプション・パラメーター (X'2110')。

SRVDGN

サーバー診断情報が入るオプション・パラメーター (X'1153')。データ部分は最大長が 32,767 バイトのストリングです。ストリングは、サーバーが問題診断の支援目的で送信する任意の情報にすることができます。 SRVDGN の詳細については、www.opengroup.org の The Open Group サイトで入手できる「*DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture*」を参照してください。

SVRCOD

重大度コードが入る必須パラメーター (X'1149')。 SVRCOD は、2 バイト長のフィールド (LL)、2 バイトのコード・ポイント (CP)、およびデータで構成されています。データは 2 バイトの 2 進数値です。以下のリストでは、指定可能な 2 バイト値を説明しています。

0 INFO - 通知のみの重大度コード
4 WARNING - 警告の重大度コード
8 ERROR - エラーの重大度コード
16 SEVERE - 重大エラーの重大度コード

32 ACCDMG - アクセス障害の重大度コード
64 PRMDMG - 永続的な障害の重大度コード
128 SESDMG - セッション障害の重大度コード

OUTAIBDBPCB

ターゲットからソースに送信される AIB および DBPCB データが入る必須パラメーター (X'CC02')。

OUTAIBDBPCB にはエラーの理由が入ります。

IMS が使用する DDM パラメーター

IMS が使用する一部の DDM の項では、IMS により製品固有のパラメーター値およびデータ構造が定義されます。

AIBOALEN パラメーター (X'C904')

AIBOALEN パラメーターは、IMS 製品固有の分散データ管理 (DDM) アーキテクチャー・パラメーターであり、データを返すすべての呼び出しの出力の最大長を示します。

フォーマット

パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C904'、AIBOALEN パラメーターの 2 バイトのコード・ポイント。

AIBOALEN
出力の最大長が入る 4 バイト整数。

関連資料:

345 ページの『INAIB コマンド・オブジェクト (X'CC01')』

AIBRSNM1 パラメーター (X'C901')

AIBRSNM1 パラメーターは、IMS 製品固有の分散データ管理 (DDM) アーキテクチャーのパラメーターであり、PCB 名が入ります。

フォーマット

パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C901'、AIBRSNM1 パラメーターの 2 バイトのコード・ポイント。

AIBRSNM1
PCB 名が入る、左寄せで 1 バイトから 8 バイトのストリング。

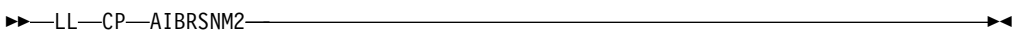
関連資料:

345 ページの『INAIB コマンド・オブジェクト (X'CC01')』

AIBRSNM2 パラメーター (X'C902')

AIBRSNM2 パラメーターは、IMS 製品固有の分散データ管理 (DDM) アーキテクチャーのパラメーターであり、ODBA 始動テーブルの ID が入ります。

フォーマット



パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C902'、AIBRSNM2 パラメーターの 2 バイトのコード・ポイント。

AIBRSNM2
ODBA 始動テーブルの 4 文字の ID を指定する 4 バイトのストリング。例えば DFSxxxx0 では、xxxx が 4 文字の ID です。

関連資料:

345 ページの『INAIB コマンド・オブジェクト (X'CC01')』

AIBSFUNC パラメーター (X'C903')

AIBSFUNC パラメーターは、IMS 製品固有の分散データ管理 (DDM) アーキテクチャーのパラメーターであり、DL/I 呼び出しの副次機能コードが (存在すれば) 入ります。

フォーマット

パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C903'、AIBSFUNC パラメーターの 2 バイトのコード・ポイント。

AIBSFUNC
DL/I 呼び出しの副次機能コードが入る、左寄せで 1 バイトから 8 バイトのストリング。

関連資料:

345 ページの『INAIB コマンド・オブジェクト (X'CC01')』

aibStream データ構造

分散データ管理 (DDM) アーキテクチャーの aibStream は、OUTAIBDBPCB DDM オブジェクトが応答メッセージで返された際には、そのオブジェクトに入っているデータ構造です。

フォーマット

表 *86. aibStream* データ構造の形式

バイト・オフセット	長さ	名前	説明
0	1	AIB NULL 標識	2 進整数。 **X'00'** aibStream データ構造の残りがあることを示します。 **X'FF'** aibStream データ構造に、AIB NULL 標識の後にデータが入っていないことを示します。 aibStream データ構造の全長は 1 バイトです。
1	4	使用される出力域	2 進整数。
5	4	戻りコード	2 進整数。
9	4	理由コード	2 進整数。
13	4	エラー・コード拡張	2 進整数。

使用法

Open Database Manager (ODBM) が GU または GN 呼び出しを処理すると、ODBM は、QRYDTA 応答オブジェクトのデータ・ストリーム内で定義されている行の形式で要求データを返します。各行は、aibStream データ構造と dbpcbStream データ構造の連結から始まり、要求データが続きます。

関連資料:

387 ページの『QRYDTA 応答オブジェクト (X'241B')』

『dbpcbStream データ構造』

404 ページの『OUTAIBDBPCB パラメーター (X'CC02')』

405 ページの『OUTAIBIOPCB パラメーター (X'CC08')』

dbpcbStream データ構造

分散データ管理 (DDM) アーキテクチャーの dbpcbStream はデータ構造であり、オブジェクトが応答メッセージで返された際には、OUTAIBDBPCB 応答オブジェクトに入っています。

フォーマット

以下の表は、dbpcbStream データ構造のフォーマットの定義を示しています。

注**:** この表では、一部のフィールドには 2 つの可能な開始バイト・オフセットがあることを示しています。それらの各フィールドでは、データベース名が dbpcbStream データ構造に含まれている場合、フィールドは大きい方のバイト・オフセットで開始されます。

表 *87. dbpcbStream* データ構造の形式

バイト・オフセット	長さ	名前	説明
0	1	DBPCB NULL 標識	2 進整数。 **X'00'** dbpcbStream データ構造の残りがあることを示します。 **X'FF'** dbpcbStream データ構造に DBPCB NULL 標識の後にデータが入っていないことを示します。 dbpcbStream データ構造の全長は 1 バイトです。
1	1	データベース名 NULL 標識	2 進整数。 **X'00'** データベース名フィールドがオフセット 2 にあることを示します。 **X'FF'** データベース名フィールドがないことを示します。
2	8	データベース名	文字ストリング。
2 または 10	2	セグメント・レベル番号	右寄せ、数字データ。
4 または 12	2	状況コード	文字データ。
6 または 14	8	セグメント名	文字ストリング。
14 または 22	1	キー・フィードバック NULL 標識	2 進整数。 **X'00'** キー・フィードバック・フィールドは、オフセット 15、またはデータベース名がオフセット 2 にある場合はオフセット 23 で開始されることを示します。 **X'FF'** キー・フィードバック・フィールドがないことを示します。
15 または 23	4	キー・フィードバック長	2 進整数。
19 または 27	可変	キー・フィードバック域	可変長文字ストリング。キー・フィードバック域の長さは、オフセット 15 のキー・フィードバックの長さフィールドで定義されます。データベース名が dbpcbStream データ構造にある場合、これはオフセット 23 になります。

使用法

Open Database Manager (ODBM) が GU または GN 呼び出しを処理すると、ODBM は、QRYDTA 応答オブジェクトのデータ・ストリーム内で定義されている行の形式で要求データを返します。各行は、aibStream データ構造と dbpcbStream データ構造の連結から始まり、要求データが続きます。

関連資料:

401 ページの『aibStream データ構造』
387 ページの『QRYDTA 応答オブジェクト (X'241B')』
404 ページの『OUTAIBDBPCB パラメーター (X'CC02')』

iopcbStream データ構造

分散データ管理 (DDM) アーキテクチャーの iopcbStream は、オブジェクトが応答メッセージで返された際には、OUTAIBIOPCB DDM オブジェクトに含まれるデータ構造です。

フォーマット

表 *88. iopcbStream* データ構造の形式

バイト・オフセット	長さ	名前	説明
0	1	IOPCB NULL 標識	2 進整数。 **X'00'** iopcbStream データ構造があることを示します。 **X'FF'** iopcbStream データ構造に IOPCB NULL 標識の後にデータが入っていないことを示します。 iopcbStream データ構造の全長は 1 バイトです。
1	1	LTERM 名 NULL 標識	2 進整数。 **X'00'** LTERM 名フィールドはオフセット 2 にあり、フィールドに論理端末名が入っていることを示します。 **X'FF'** LTERM 名フィールドがないことを示します。
2	8	LTERM 名	文字ストリング。
2 または 10	2	予約済み	予約済み。
4 または 12	2	状況コード	文字データ。
6 または 14	4	ローカル日時	バイト配列。
10 または 18	4	入力メッセージのシーケンス番号	バイト配列。
14 または 22	8	メッセージ出力記述子名	文字ストリング。
22 または 30	8	ユーザー ID	文字ストリング。
30 または 38	8	グループ名	文字ストリング。
38 または 46	12	タイム・スタンプ	バイト配列。
50 または 58	1	ユーザー ID 標識	文字データ。

使用法

先行セクションの図および説明は、iopcbStream データ構造の FD:OCA の初期記述子の定義として役立てることができます。 iopcbStream が応答メッセージで返される場合、iopcbStream データ構造と、それが含まれる OUTAIBIOPCB 応答オブジェクト (X'CC08') とを混同しないでください。

関連資料:

405 ページの『OUTAIBIOPCB パラメーター (X'CC08')』

OUTAIBDBPCB パラメーター (X'CC02')

分散データ管理 (DDM) アーキテクチャーの OUTAIBDBPCB (出力 AIBDBPCB) パラメーターは、ターゲット・サーバーからソース・サーバーに送信され、aibStream データ構造と dbpcbStream データ構造の連結が入ります。

フォーマット

パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC02'、OUTAIBDBPCB パラメーターの 2 バイトのコード・ポイント。

aibStream
必須。以下のデータが入ります。

- AIB NULL 標識
- 出力域
- 戻りコード
- 理由コード
- エラー・コード拡張

dbpcbStream
必須。以下のデータが入ります。

- DBPCB NULL 標識
- データベース名
- セグメント・レベル番号
- 状況コード
- キー・フィードバック

使用法

この OUTAIBDBPCB パラメーターはスカラー・パラメーターです。 aibStream または dbpcbStream データ構造の前には、長さ値やコード・ポイント値が置かれることはありません。

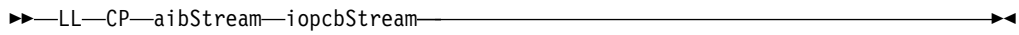
関連資料:

377 ページの『DEALLOCDBRM 応答メッセージ (X'CA01')』
401 ページの『aibStream データ構造』
401 ページの『dbpcbStream データ構造』
389 ページの『RDBAFLRM 応答メッセージ (X'221A')』

OUTAIBIOPCB パラメーター (X'CC08')

分散データ管理 (DDM) アーキテクチャーの OUTAIBIOPCB (出力 AIBIOPCB) パラメーターは、ターゲット・サーバーからソース・サーバーに送信され、aibStream データ構造と iopcbStream データ構造の連結が入ります。

フォーマット



パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'CC08'、OUTAIBIOPCB パラメーターの 2 バイトのコード・ポイント。

aibStream
必須。以下のデータが入ります。

- AIB NULL 標識
- 出力域
- 戻りコード
- 理由コード
- エラー・コード拡張

iopcbStream
必須。以下のデータが入ります。

- IOPCB NULL 標識
- LTERM 名
- 状況コード
- 入力メッセージのセグメント番号
- メッセージ出力記述子名
- グループ名
- キー・フィードバック

使用法

OUTAIBIOPCB パラメーターはスカラー・パラメーターです。 aibStream または iopcbStream データ構造の前には、長さ値やコード・ポイント値が置かれることはありません。

関連資料:

403 ページの『iopcbStream データ構造』
401 ページの『aibStream データ構造』

RDBNAM パラメーター (X'2110')

分散データ管理 (DDM) アーキテクチャーの RDBNAM パラメーターは、特定の対話のためのターゲット・データベースを示します。

フォーマット

パラメーター

LL RDBNAM パラメーターの 2 バイト仕様の長さ。

CP X'2110'、RDBNAM パラメーターの 2 バイトのコード・ポイント。

PSBNAME
IMS PSB 名。最大 8 バイト長の文字ストリングで指定します。 PSB 名はターゲット・データベースを示し、IMS に定義されている PSB 名と一致している必要があります。

ALIAS
オプション。IMS データ・ストア名の別名。ALIAS は 4 バイトで指定する必要があります。別名が 4 文字未満の場合、文字は左寄せで、残りのバイトはブランク文字のスペースを埋め込む必要があります。

ALIAS を使用する場合、PSBNAME と ALIAS をピリオドで区切る必要があります。

関連資料:

323 ページの『ACCRDB コマンド (X'2001')』

325 ページの『ACCSEC コマンド (X'106D')』

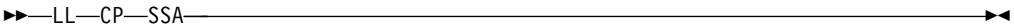
330 ページの『DEALLOCDB コマンド (X'C801')』

389 ページの『RDBAFLRM 応答メッセージ (X'221A')』

SSA パラメーター (X'C906')

SSA パラメーターは、IMS 製品固有の分散データ管理 (DDM) アーキテクチャーのパラメーターであり、DL/I 呼び出しを修飾するセグメント検索引数 (SSA) が入ります。

フォーマット



パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C906'、SSA パラメーターの 2 バイトのコード・ポイント。

SSA
SSA が入るバイト・ストリング。

関連資料:

371 ページの『SSALIST コマンド・オブジェクト (X'CC06')』

SSACOUNT パラメーター (X'C905')

SSACOUNT パラメーターは、IMS 製品固有の分散データ管理 (DDM) アーキテクチャーのパラメーターであり、SSALIST コマンド・オブジェクトに含まれるセグメント検索引数 (SSA) の数を指定します。

フォーマット

パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C905'、SSACOUNT パラメーターの 2 バイトのコード・ポイント。

SSACOUNT
SSALIST コマンド・オブジェクトに含まれる SSA の数を示す 2 バイトの 2 進数フィールド。最小値は 1 で、最大値は 15 です。

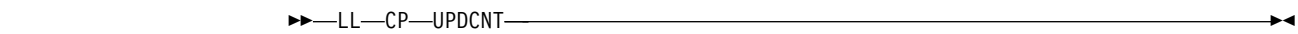
関連資料:

371 ページの『SSALIST コマンド・オブジェクト (X'CC06')』

UPDCNT パラメーター (X'C90A')

UPDCNT パラメーターは、IMS 製品固有の分散データ管理 (DDM) アーキテクチャーのパラメーターであり、個別あるいはバッチの INSERT、UPDATE、または DELETE 処理の影響を受ける行数を指定します。

フォーマット



パラメーター

LL 2 バイトの 2 進整数で指定する長さ。この長さには LL および CP が含まれます。

CP X'C90A'、UPDCNT パラメーターの 2 バイトのコード・ポイント。

UPDCNT
バッチあるいは singleton の INSERT、UPDATE、または DELETE 処理の影響を受ける行数が入る 4 バイト整数。

関連資料:

393 ページの『RDBUPDRM 応答メッセージ (X'2218')』

第 3 章 IMS アダプター (REXX 版) 参照

IMS アダプター (REXX 版) (REXXTDLI) は、IMS ユーザーが TSO/E (タイム・シェアリング・オプション拡張機能) のもとで REXX EXEC を対話式に開発し、これを IMS、MPP、BMP、IFP、またはバッチ領域で実行できる環境を提供します。

この製品は、DFSDDLT0 と競合するものではなく、付属品です。IMS アダプター (REXX 版) を使用すれば、ボリュームが小さいトランザクション・プログラムを作成したり、またはそのプロトタイプを作成したりするための、アプリケーション・プログラミング環境が整います。

IMS のもとで稼働する REXX 環境には、「*z/OS TSO/E REXX* 解説書」に記載されているのと同じ機能と制約事項があります。この制約事項は、TSO、 ISPEXEC、および ISREDIT 環境でない場合や、LISTDS のような TSO 特有の機能がない場合にあてはまります。「*z/OS TSO/E REXX* 解説書」に記載されているように、環境に独自の外部機能を追加することができます。

IMS は、IRXJCL を使用して REXX EXEC を呼び出します。この方式では、戻りコード (RC20) は制限付き戻りコードです。戻りコード 20 は、処理が成功しなかった場合、および EXEC が実行されなかった場合に、IRXJCL の呼び出し元に返されます。

REXX EXEC は、IMS アプリケーションとして実行され、COBOL などの他の IMS サポート・プログラミング言語と同様の特性を持ちます。プログラミング言語 (REXX およびサポートされる他の言語) は、MPP 領域で混合して使用することができます。例えば、COBOL トランザクションは REXX トランザクションが完了した後でも実行させることができ、逆もまた可能です。

REXX の利点は以下のとおりです。

- REXX は使いやすい解釈言語である。
- REXX では、EXEC を標準 PSB (IVPREXX またはユーザーが確立したもの) のもとで実行できるので、EXEC を追加して実行するための特殊 PSB 生成が必要ない。
- REXX インターフェースが DL/I 呼び出しをサポートし、次の機能を提供している。
  - DL/I 呼び出し、状況、およびパラメーターのトレースの呼び出し
  - 最新の DL/I 呼び出しの照会
  - 拡張データ・マッピング
  - 名前およびオフセットによる PCB の指定
  - ストレージの獲得および解放
  - `WTO`、`WTP`、`WTL`、および `WTOR` を使用するメッセージ

REXX EXEC を実行するには、次のシステム環境条件が必要です。

- DFSREXX0 および DFSREXX1 が、STEPLIB などの、IMS 従属領域またはバッチ領域にアクセス可能なロード・ライブラリーにある。

- DFSREXX0 が独立型で、RENT オプションが指定されている。
- DFSREXX1 が DFSLI000 および DFSCPIR0 (SRRCMIT と SRRBACKのため) とバインド済みであり、オプションで DFSREXXU ともバインド済みである。オプションは、RENT でなく REUS です。
- IVPREXX (DFSREXX0 プログラムのコピー) が IMS トランザクション・プログラムとしてインストールされている。このプログラムは、IVP (インストール検査プログラム) によってインストールされます。
- PSB が、アセンブラー言語または COBOL で定義されている。
- SYSEXEC DD が、IMS で実行される REXX EXEC を含むデータ・セットのリストを指す。この DD は、IMS 従属領域またはバッチ領域の JCL に入れてください。
- トレース、エラー、および SAY 命令など、SYSTSPRT DD が REXX 出力のために使用される。SYSTSPRT DD は、通常、インストール・システムに応じて SYSOUT=A または他のクラスに割り振られ、さらに、IMS 従属領域またはバッチ領域の JCL に入れられる必要があります。
- TSO のもとでは IMS 従属領域にコンソールがないので、SYSTSIN DD が REXX 入力のために使用される。 SYSTSIN では一般的に、REXX PULL ステートメントが使用されます。

関連資料: SYSTSPRT および SYSTSIN の詳細については、「*z/OS TSO/E REXX* 解説書」を参照してください。

関連資料:

444 ページの『IVPREXX サンプル・アプリケーション』

IMS アダプター (REXX版) の概要

以下の図は、高水準の REXX 環境での IMS アダプターを示しています。この図では、IMS プログラム・コントローラーのもとでの環境構成と、その環境のコンポーネント間の対話のパスが示されています。

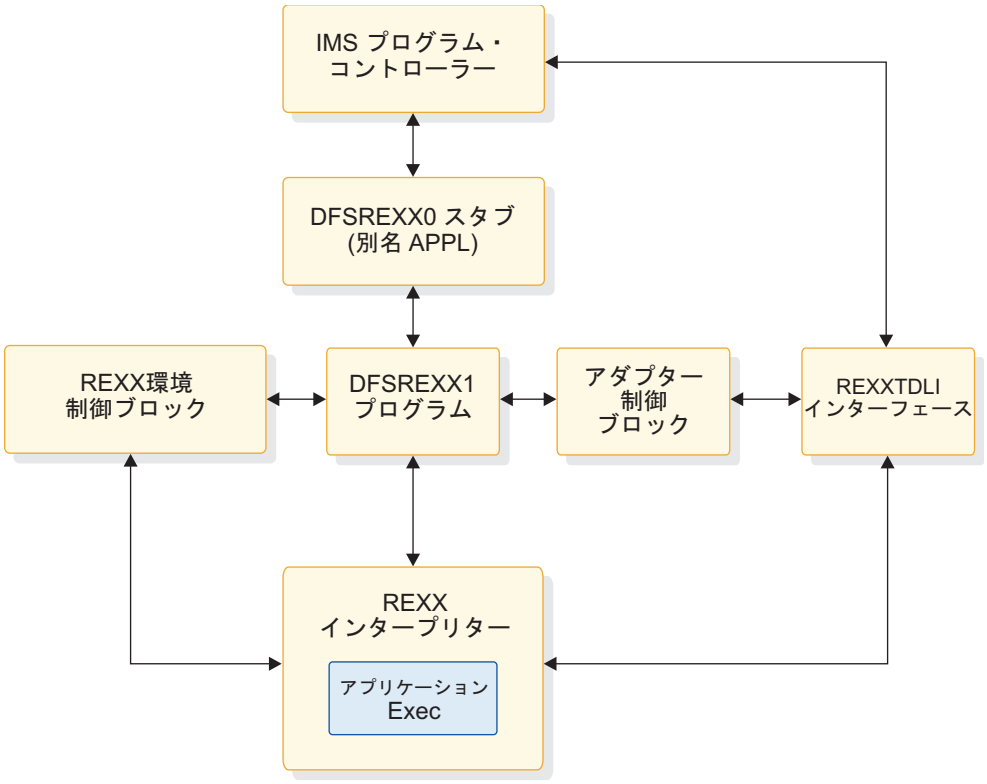


図 9. *IMS アダプター (REXX 版)* の論理総括ダイアグラム

関連資料:

IMS アダプター (REXX 版) 出口ルーチン (DFSREXXU) (出口ルーチン)

サンプル出口ルーチン (DFSREXXU)

IMS は、 IMS アダプター (REXX 版) とともに使用されるサンプルのユーザー出口ルーチンを提供しています。

ユーザー出口ルーチンの作成方法については、「*IMS V14* 出口ルーチン」を参照してください。サンプル・ユーザー出口ルーチンは、エントリー時に呼び出されているのかどうかを検査します。そうである場合、ユーザー出口ルーチンは、引数を付けずにパラメーター・リストをトランザクション・コードに設定し、始動の IMSRXTRC レベルを 2 に設定します。戻りコードは 0 に設定されます。最新バージョンの DFSREXXU ソース・コードについては、IMS.ADFSSMPL 配布ライブラリーを参照してください。メンバー名は DFSREXXU です。

その他の環境のアドレッシング

REXX ADDRESS 命令を使用して、コマンドの宛先を変更します。IMS アダプター (REXX 版) は、次の 2 つのホスト・コマンド環境を通して機能します: REXXTDLI と REXXIMS その他のホスト・コマンド環境も、IMS EXEC を使用してアクセス可能です。

z/OS 環境は、TSO および TSO 以外のアドレス・スペースで、TSO によって提供されます。この環境は、ファイル入出力用の EXECIO のような他のプログラムを実行するために使用されます。IMS は、z/OS EXECIO リソースを管理しません。したがって、IMS COMMIT または BACKOUT は、これらのリソースにはなんの影響も与えません。 EXECIO が IMS 制御のリソースでないため、保全性は維持されません。フラット・ファイル入出力で保全性が問題となる場合は、IMS 提供の保全性を確実にしている IMS GSAM を使用してください。

APPC/MVS が使用可能であれば、他の環境を使用できます。使用可能な環境は次のとおりです。

APPCMVS
z/OS に固有の APPC インターフェースで使用

CPICOMM
CPI 通信に使用

LU62 z/OS に固有の APPC インターフェースで使用

関連資料**:** 環境のアドレッシングの詳細については、「*z/OS TSO/E REXX* 解説書」を参照してください。

REXX トランザクション・プログラム

REXX トランザクション・プログラムでは、あらゆる PSB 定義を使用することができます。テスト用に IVP が設定した定義は、IVPREXX と呼ばれます。

IMS 定義のステージ 1 のセクションを次に例示します。

```
*********************************************************************
*   IVP APPLICATIONS DEFINITION FOR DB/DC, DCCTL                    *
*********************************************************************
         APPLCTN GPSB=IVPREXX,PGMTYPE=TP,LANG=ASSEM  REXXTDLI SAMPLE
           TRANSACT CODE=IVPREXX,MODE=SNGL,                          X
               MSGTYPE=(SNGLSEG,NONRESPONSE,1)
```

この例では GPSB を使用していますが、ユーザーが定義する任意の PSB を使用することができます。GPSB では、TP PCB および変更可能代替 PCB を含む、総称 PSB が提供されます。データベース PCB は含まれません。 REXX アプリケーションには特定の言語タイプがないので、ASSEM の言語タイプを指定します。

推奨事項**:** REXX アプリケーションの場合は、アセンブラー言語または COBOL を指定してください。

IMS は、MPP 領域で使用される PSB 名または他の領域タイプ用の PGM 名と同じロード・モジュール名を使用して、トランザクションをスケジュールします。アプリケーション・プログラムが REXX EXEC で構成されていても、このロード・モ

ジュールを使用してください。IMS アダプター (REXX 版) は、ユーザーが使用できるようにロード・モジュールを提供しています。このモジュールは DFSREXX0 と呼ばれます。以下のことが可能です。

- アプリケーション PSB 名と同じ名前を持つ steplib データ・セットにコピーする。ロード・モジュールをコピーするための標準ユーティリティー (IEBCOPY や SAS など) か、またはバインダーのどちらかを使用します。
- バインダーを使用して、アプリケーション PGM 名と同じ DFSREXX0 の別名を定義する。

例えば、以下のコード例は、バインダーを使用して名前 IVPREXX へのコピー機能を実施する PGM セットアップ・ジョブのセクションを示しています。この例では、IVP を使用しています。

```
//* REXXTDLI SAMPLE - GENERIC APPLICATION DRIVER
//*
//LINK    EXEC PGM=IEWL,
//          PARM='XREF,LIST,LET,SIZE=(192K,64K)'
//SYSPRINT  DD SYSOUT=*
//SDFSRESL  DD DISP=SHR,DSN=IMS.SDFSRESL
//SYSLMOD   DD DISP=SHR,DSN=IMS1.PGMLIB
//SYSUT1    DD UNIT=(SYSALLDA,SEP=(SYSLMOD,SYSLIN)),
//             DISP=(,DELETE,DELETE),SPACE=(CYL,(1,1))
//SYSLIN    DD *
  INCLUDE   SDFSRESL(DFSREXX0)
     ENTRY DFSREXX0
       NAME  IVPREXX(R)
/*
```

IMS がアプリケーション・トランザクションをスケジュールすると、ロード・モジュールがロードされ、これに制御が渡されます。ロード・モジュールは、(適当であれば) トランザクション・コードの引数を使用し、PGM 名として REXX EXEC 名を確立します。モジュールは、使用可能であればユーザー出口ルーチン (DFSREXXU) を呼び出します。ユーザー出口ルーチンで REXX EXEC (または、実行する他の EXEC) を選択し、EXEC 引数の変更、あるいはその他必要な処理を行うことができます。

ユーザー出口ルーチンからの戻り時に、ルーチンの要求するアクションがとられます。通常、このアクションは REXX EXEC 呼び出しと関係があります。EXEC ロードは、SYSEXEC DD 割り振りを使用して行われます。この割り振りは、REXX で書かれ、プログラムで使用される任意の関数と同様に実行される IMS REXX アプリケーション・プログラムを含んでいる 1 つ以上の区分データ・セットを指している必要があります。

SAY ステートメントおよびトレースなど、標準 REXX 出力は SYSTSPRT に送られます。この DD は必須で、SYSOUT=A に設定することができます。

スタックが空のときは、REXX PULL ステートメントが SYSTSIN DD から読み取られます。このようにして、バッチ入力データを BMP またはバッチ領域に簡単に提供することができます。 SYSTSIN はオプションですが、空のスタックから PULL が出され、かつ SYSTSIN が割り振られていない場合は、エラー・メッセージが表示されます。以下のコード例に、IVPREXX サンプルの EXEC を実行する MPP 領域に必要な JCL を示します。

IVPREXX サンプル EXEC を実行するために使用される JCL コード

```
//IVP32M11 EXEC PROC=DFSMPR,TIME=(1440),
//         AGN=IVP,          AGN NAME
//         NBA=6,
//         OBA=5,
//         SOUT='*',         SYSOUT CLASS
//         CL1=001,          TRANSACTION CLASS 1
//         CL2=000,          TRANSACTION CLASS 2
//         CL3=000,          TRANSACTION CLASS 3
//         CL4=000,          TRANSACTION CLASS 4
//         TLIM=10,          MPR TERMINATION LIMIT
//         SOD=,             SPIN-OFF DUMP CLASS
//         IMSID=IVP1,       IMSID OF IMS CONTROL REGION
//         PREINIT=DC,       PROCLIB DFSINTXX MEMBER
//         PWFI=Y            PSEUDO=WFI
//*
//* ADDITIONAL DD STATEMENTS
//*
//DFSCTL   DD DISP=SHR,
//            DSN=IVPSYS32.PROCLIB(DFSSBPRM)
//DFSSTAT  DD SYSOUT=*
//* REXX EXEC SOURCE LOCATION
//SYSEXEC  DD DISP=SHR,
//            DSN=IVPIVP32.INSTALIB
//         DD DISP=SHR,
//            DSN=IVPSYS32.SDFSEXEC
//* REXX INPUT LOCATION WHEN STACK IS EMPTY
//SYSTSIN  DD *
/*
//* REXX OUTPUT LOCATION
//SYSTSPRT DD SYSOUT=*
//* COBOL OUTPUT LOCATION
//SYSOUT   DD SYSOUT=*
```

関連資料:

IMS アダプター (REXX 版) 出口ルーチン (DFSREXXU) (出口ルーチン)

REXXTDLI コマンド

以下のトピックでは、REXX コマンド、および REXX コマンドの DL/I 呼び出しへの適用について説明します。

REXXTDLI 環境について説明するときに、コマンドと呼び出し という用語は、区別しないで使用することができます。ただし、コマンド という用語は、REXXIMS 環境について説明するときにだけ使用します。一貫性を持たせるために、DL/I を説明するときは呼び出し を使用し、REXX を説明するときはコマンド を使用します。

REXX 環境用 IMS アダプターでコマンドを出すには、まず最初に正しい環境にアドレッシングする必要があります。 IMS アダプター (REXX 版) は、アドレス可能な環境を 2 つ提供します。その環境は次のとおりです。

REXXTDLI

GU および ISRT といった標準 DL/I 呼び出し用に使用されます。 REXXTDLI インターフェース環境は、すべての標準 DL/I 呼び出しに使用され、REXX 固有のコマンドでは使用されません。この環境で出されるすべてのコマンドは、標準 DL/I 呼び出しと見なされ、適切に処理されます。 この環境の GU 呼び出しは、次のようになります。

```
Address REXXTDLI "GU MYPCB DataSeg"
```

REXXIMS

REXX 環境用 IMS アダプターで REXX 固有のコマンド (WTO や MAPDEF など) にアクセスする場合に使用されます。 REXXIMS インターフェース環境は、標準 DL/I 呼び出しおよび REXX 固有のコマンドの両方で使用されます。この環境でコマンドが出されると、IMS は、このコマンドが REXX に固有のものかどうかを検査します。コマンドが REXX 固有のものでなければ、IMS は、このコマンドが標準 DL/I 呼び出しであるかどうかを検査します。コマンドは、その結果に応じて処理されます。

REXX 固有のコマンドは、拡張コマンドとも呼ばれ、REXX インターフェース用に IMS アダプターによって追加された REXX 拡張機能です。この環境の WTO 呼び出しは、次のようになります。

```
Address REXXIMS "WTO Message"
```

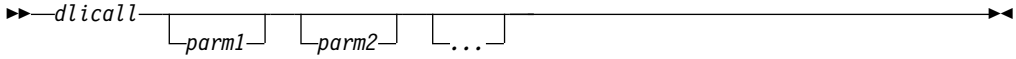
スケジュールされた EXEC に入った時点では、デフォルト環境は z/OS です。したがって、IMS アダプター (REXX 版) の呼び出しを発行するには ADDRESS REXXTDLI または ADDRESS REXXIMS を使用する必要があります。

関連資料: 環境のアドレッシングについての一般情報は、「*TSO/E Version 2 Procedures Language MVS/REXX Reference*」を参照してください。

REXXTDLI 呼び出し

以下では、REXXTDLI 呼び出しの使用時の考慮事項について説明します。

フォーマット



DL/I 呼び出しの形式は、呼び出しタイプによって異なります。サポートされる DL/I 呼び出しのパラメーターの形式は、 1 ページの『データベース管理のための DL/I 呼び出し』、 94 ページの『トランザクション管理のための DL/I 呼び出し』、および 41 ページの『IMS DB システム・サービスのための DL/I 呼び出し』に示してあります。呼び出し用のパラメーターは、大/小文字のどちらを使用してもよく、1 つ以上のブランクで分けられ、通常は REXX 変数です。詳細については、 416 ページの『パラメーター操作』を参照してください。

同期コールアウト要求の発行

REXXTDLI インターフェースを使用して同期コールアウト (ICAL 呼び出し) 要求を発行するには、入力域と出力域の前に DFSAIB キーワードを指定する必要があります。入力域と出力域は、どちらも変数、*mapname、または !token として指定できます。

REXXTDLI インターフェースからの ICAL 呼び出しの構文は以下のとおりです。

出力域の未定義または暗黙の変数には、デフォルトの長さである 1024 バイトが AIBOAUSE フィールドへの入力として渡されます。これより長いメッセージを指定するには、STORAGE コマンドを発行する必要があります。

戻りコード

AIBTDLI インターフェースを使用すると、REXX RC 変数が DL/I 呼び出しの AIB からの戻りコードに設定されます。

AIBTDLI インターフェースを使用しない場合は、シミュレートされた戻りコードが返されます。このシミュレートされた戻りコードは、PCB 状況コードが GA、GK、または bb の場合はゼロに設定されます。状況コードが他の値の場合のシミュレートされた戻りコードは、X'900' または 10 進数の 2304 です。

パラメーター操作

IMS アダプター (REXX 版) を使用して、REXX 環境にアプリケーション・プログラム用のパラメーターを設定することができます。このように設定するのは、アプリケーション・プログラムで変数やマップをパラメーターとして使用するときです。アプリケーションがストレージ・トークンを使用するときは、REXX はこのような設定をしません。アプリケーション・プログラムでは、REXX 以外のアプリケーションと同様にトークンを提供し、結果を構文解析する必要があります。パラメーターの種類とその定義については、 417 ページの表 89 を参照してください。

REXXTDLI インターフェースは、次のような設定を行います。

- 入出力 PCB 用の入出力域検索が構文解析される。 LL フィールドが除去されます。また、ZZ フィールドが除去されて、REXXIMS('ZZ')機能呼び出しによって利用可能になります。その他のデータは、指定された変数またはマップに置かれます。 REXX LENGTH() 関数を使用して、返されたデータの長さを検出します。
- 次のように、TP PCB または代替 PCB の入出力域が作成される。
  - 適切な LL フィールド
  - コマンドが使用されなかった場合は、先行 SET ZZ コマンドまたは X'0000' からの ZZ フィールド
  - パス済みの変数またはマップで指定されるデータ
- SPA 処理用の入出力域が、4 バイト長の ZZ フィールドを除いて先の 2 項目と類似している。
- CHNG および SETO 呼び出しのフィードバック域が構文解析される。LLZZLL フィールドが除去され、その他のデータは適切な長さで返されます。
- パラメーターの形式に LLZZ という部分があると、特殊な扱いを受ける。このようなパラメーターは、AUTH、CHNG、INIT、ROLS、SETO、および SETS 呼び出しにあります。LLZZ フィールドは、IMS がユーザーにデータを返すときに除去され、IMS がユーザーからデータを検索するときに追加されます (ZZ は常に X'0000')。基本的に、アプリケーションは LLZZ フィールドを無視し、これに続くデータだけを処理します。
- XRST および記号 CHKP の数値パラメーターは、必要であれば、10 進数と 32 ビット数 (フルワード) の間で変換される。

表 89. IMS アダプター (REXX 版) のパラメーターのタイプおよび定義

タイプ[1]	パラメーター定義
PCB	**重要:** REXXTDLI インターフェースを使用して同期コールアウト要求 (ICAL 呼び出し) を行う場合は、PCB パラメーターは不要です。代わりにキーワード DFSAIB を、入力パラメーターと出力パラメーターの前に指定する必要があります。 PCB ID (以下のいずれかを含む変数として指定) • PCBNAME= パラメーターの PSB 生成で定義された PCB 名。 PCB 名の定義の詳細については、「IMS バージョン 14 システム・ユーティリティー」を参照してください。名前の長さは 1 文字から 8 文字とし、ブランクで埋め込む必要はありません。名前が指定してあれば AIBTDLI インターフェースが使用され、このインターフェースから、戻りコードおよび理由コードが獲得されます。 • DFSAIB 仕様で形式設定された AIB ブロック。この変数には、更新された AIB が返されます。 • # とそのあとの PCB オフセット番号 (#1= 最初の PCB)。設定の例を次に示します。 – `IOPCB=:"#1"` – `ALTPCB=:"#2"` – `DBPCB=:"#3"` この表記を使用すれば、データベース DL/I 呼び出しで返される IOAREA のデフォルトの長さは 4096 になります。AIBTDLI インターフェースが使用されるときのみ、正確な長さを利用することができます。
In	入力変数。定数、変数、*mapname[2]、または !token[3] として指定できます。
SSA	SSA (セグメント検索引数) 付きの入力変数。定数、変数、*mapname[2]、または !token[3] として指定できます。
Out	コマンドの正常終了後に結果を保管する出力変数。変数、*mapname[2]、または !token[3] として指定できます。
In/Out	入力時は入力情報を、コマンドの正常終了後は結果を含む変数。変数、*mapname[2]、または !token[3] として指定できます。
Const	入力定数。このコマンド引数は、値が入る変数ではなく実際の値でなければなりません。

表 89. IMS アダプター (REXX 版) のパラメーターのタイプおよび定義 (続き)

タイプ[1]	パラメーター定義

注:

1. 417 ページの表 89 に示したパラメーター・タイプは、 2 ページの表 1、 95 ページの表 25、および 41 ページの表 5、また、 420 ページの表 90 にリストしてあるタイプと対応しています。

   DL/I 呼び出しに指定するすべてのパラメーターは、大/小文字のどちらを使用してもかまいません。ただし、複合変数 (配列構造に関する REXX 用語) の STEM 部分に関連する値は除きます。ピリオド (.) はすべてのパラメーターの代わりに使用することができ、ヌル (長さゼロのストリング) として読み取られ、白抜き (プレースホルダー) として書き込まれます。オプション・パラメーターをスキップしたいときは、パラメーターの代わりにピリオドを使用すると便利です。

2. *mapname の詳細については、 425 ページの『MAPGET』および 426 ページの『MAPPUT』を参照してください。
3. !token の詳細については、 429 ページの『STORAGE』を参照してください。

DL/I 呼び出しの例

次の例では、入出力 PCB に対して出される ISRT 呼び出しを示しています。『Hello World』 というメッセージが書き込まれます。

```
IO = "IOPCB"           /* IMS Name for I/O PCB */
OutMsg="Hello World"
Address REXXTDLI "ISRT IO OutMsg"
If RC¬=0 Then Exit 12
```

この例では、*IO* が PCB 名を含む変数で、この PCB 名は入出力 PCB の定数 "IOPCB" です。戻りコード (RC) がゼロ以外の場合は、EXEC の戻りコードは 12 で終了 (Exit) します。ここでは他の処理を行うこともできます。

次の例では、IMS サンプル部品データベースからある部品を入手します。部品番号は "250239" です。 実際の部品キーの接頭部は "02" で、DBD で定義されるキーの長さは 17 バイトです。

この例では、*Part_Segment* という変数にセグメントを書き出します。

```
PartNum = "250239"
DB      = "DBPCB01"
SSA     = 'PARTROOT(PARTKEY = '||Left('02'||PartNum,17)||')'
Address REXXTDLI "GU DB Part_Segment SSA"
```

注:

- 実際の EXEC では、引数から PartNum の値が求められるため、呼び出しの後で戻りコードを検査する必要があります。
- ここで使用される LEFT 関数は、REXX 組み込み関数です。この組み込み関数は、あらゆる IMS REXX EXEC で利用可能です。関数の詳細については、「*TSO/E Version 2 Procedures Language MVS/REXX Reference*」を参照してください。
- REXX では、単一引用符 (') と二重引用符 (") は、それぞれが対応していればどちらを使用してもかまいません。

IMS.SDFSISRC ライブラリーには、DFSSUT04 EXEC が組み込まれています。この EXEC を使用すれば、どのような予期しない戻りコードや状況コードも処理できます。最新の DL/I 呼び出しから状況コードを獲得するには、IMSQUERY('STATUS') 機能を実行します。2 文字の状況コードが返されます。

IMS 環境および IMS 以外の環境の両方で稼働する EXEC を使用する場合は、IMS 環境が使用可能であるかを確認してください。 IMS 環境が使用可能かどうかは、次の 2 通りの方法で調べることができます。

- z/OS SUBCOM コマンドを使用して、REXXTDLI または REXXIMS どちらかの環境を指定する。このコードは、次のようになります。

```
Address MVS 'SUBCOM REXXTDLI'
If RC=0 Then Say "IMS Environment is Available."
        Else Say "Sorry, no IMS Environment here."
```

- REXX の PARSE SOURCE 命令を使用して、アドレス・スペース名 (8 番目のワード) を調べる。 IMS 環境で稼働していれば、トークンは IMS という値になります。このコードは、次のようになります。

```
Parse Source . . . . . Token .
If Token='IMS' Then Say "IMS Environment is Available."
               Else Say "Sorry, no IMS Environment here."
```

以下の IMS REXX プログラムのサンプルでは、DL/I ICAL 呼び出しを使用して同期コールアウト要求メッセージを OTMDEST1 という OTMA 記述子に送信する方法を示します。ここでは、入力ストリングに「Hello from IMS」、タイムアウト値に 60 秒を指定します。出力データは、変数 *Output* で設定されます。

```
Address REXXIMS
Input = 'Hello from IMS';
Timer = 6000
'SET SUBFUNC SENDRECV'
'SET RSNAME1 OTMDEST1'
'SET TIMER Timer'

'ICAL DFSAIB Input Output'
Say Input
Say Output

Outlen = IMSQUERY('OUTLEN')
Say Outlen
Errxtn = IMSQUERY('ERRXTN')
Say Errxtn
```

以下のサンプルは、DL/I ICAL 呼び出しを発行した IMS REXX プログラムの出力を示しています。

```
DFS3180I INQY ENVIRON Region=BMP      Number=1
DFS3180I INQY ENVIRON Tran=TXCD255  PGM=DFSREXX0
DFS3180I Starting EXEC Name=DFSREXX0
DFS3160I IMS CMD=SET SUBFUNC SENDRECV
DFS3161I REXXIMS  Command=SET      RC=0
DFS3160I IMS CMD=SET RSNAME1 OTMDEST1
DFS3161I REXXIMS  Command=SET      RC=0
DFS3160I IMS CMD=SET TIMER timer
DFS3161I REXXIMS  Command=SET      RC=0
DFS3160I IMS CMD=ICAL DFSAIB Input Output
DFS3161I REXXTDLI Call=ICAL RC=0000 Reason=0000 Status=".."
Hello from IMS
HELLO FROM TM RA APP
50
0
```

環境の判別

IMS 環境および IMS 以外の環境の両方で稼働する EXEC を使用する場合は、IMS 環境が使用可能であるかを確認してください。 IMS 環境が使用可能かどうかは、次の 2 通りの方法で調べることができます。

- z/OS SUBCOM コマンドを使用して、REXXTDLI または REXXIMS どちらかの環境を指定する。このコードは、次のようになります。

```
Address z/OS 'SUBCOM REXXTDLI'
If RC=0 Then Say "IMS Environment is Available."
        Else Say "Sorry, no IMS Environment is here."
```

- REXX の PARSE SOURCE 命令を使用して、アドレス・スペース名 (8 番目のワード) を調べる。 IMS 環境で稼働していれば、トークンは IMS という値になります。このコードは、次のようになります。

```
Parse Source . . . . . Token . If Token='IMS' Then Say "IMS Environment
is Available."
               Else Say "Sorry, no IMS Environment here."
```

関連資料:

429 ページの『STORAGE』

421 ページの『DLIINFO』

136 ページの『SETO 呼び出し』

『REXXIMS 拡張コマンド』

REXXIMS 拡張コマンド

IMS アダプター (REXX 版) を使用すると、標準の DL/I 呼び出しにアクセスできます。また、REXX 環境用の一連の拡張コマンドも提供されます。

これらのコマンドは、以下の表にリストしてあり、ADDRESS REXXIMS を使用すると利用可能になります。 REXXIMS 環境にアドレッシングすると、DL/I 呼び出しも利用できます。

表 *90. REXXIMS* 拡張コマンド :

コマンド	パラメーターのタイプ
DLIINFO	Out [PCB]
IMSRXTRC	In
MAPDEF	Const In [Const]
MAPGET	Const In
MAPPUT	Const Out
SET	Const In
SRRBACK	Out
SRRCMIT	Out
STORAGE	Const Const [In [Const]]
WTO	In
WTP	In
WTL	In
WTOR	In Out

表 *90. REXXIMS* 拡張コマンド *(*続き*)*:

コマンド	パラメーターのタイプ

注記:

DL/I 呼び出しで指定されるすべてのパラメーターは、複合変数 (配列構造に関する REXX 用語) の STEM 部分に関連する値を除いて、大/小文字のどちらを使用してもかまいません。ピリオド (.) はすべてのパラメーターの代わりに使用することができ、ヌル (長さゼロのストリング) として読み取られ、白抜き (プレースホルダー) として書き込まれます。オプション・パラメーターをスキップしたいときは、パラメーターの代わりにピリオドを使用します。

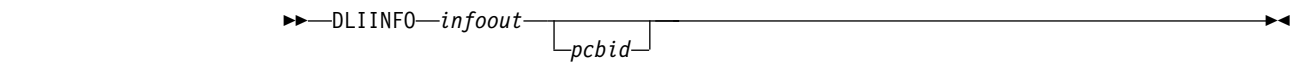
関連資料:

415 ページの『REXXTDLI 呼び出し』

DLIINFO

DLIINFO 呼び出しでは、最新の DL/I 呼び出しまたは特定の PCB の情報を要求します。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
DLIINFO	X	X	X	X	X

使用法

infoout 変数名は、DL/I 情報に割り当てられる REXX 変数です。*pcbid* 変数名は、指定された PCB およびその最新の状況コードに関連したアドレスを返します。

戻される情報の形式は、以下のとおりです。

ワード 説明

1 最新の DL/I 呼び出し (該当するものがないときは '.')

2 最新の DL/I PCB 名 (名前または # 番号、該当するものがないときは '.')

3 16 進数の最新の DL/I AIB アドレス (該当するものがないときは 00000000)

4 16 進数の最新の DL/I PCB アドレス (該当するものがないときは 00000000)

5 最新の DL/I 戻りコード (該当するものがないときは 0)

6 最新の DL/I 理由コード (該当するものがないときは 0)

7 最新の DL/I 呼び出し状況 (ブランクであるか該当するものがないときは '.')

例

```
Address REXXIMS 'DLIINFO MyInfo'         /* Get Info                  */
Parse Var MyInfo DLI_Cmd DLI_PCB DLI_AIB_Addr DLI_PCB_Addr,
          DLI_RC DLI_Reason DLI_Status .
```

構文解析を行うときは、必要に応じて発生する今後の追加に備えて、状況コード (7 番目に返されるワード) のあとに必ずピリオドをコーディングしてください。ワード 3、4、7 は、*pcbid* が DLIINFO 呼び出しで指定されるときに使用できます。

関連資料:

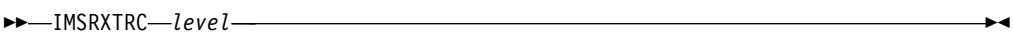
415 ページの『REXXTDLI 呼び出し』

435 ページの『PCBINFO EXEC : 現行 PSB で使用可能な PCB の表示』

IMSRXTRC

IMSRXTRC コマンドは、主にデバッグに使用されます。このコマンドは、REXX プログラムを実行している間にとられるトレース・アクション (すなわち、SYSTSPRT を通じてユーザーに送られるトレース出力の量) を制御します。

フォーマット



呼び出し名	**DB/DC**	**DBCTL**	**DCCTL**	**DB** バッチ	**TM** バッチ
IMSRXTRC	X	X	X	X	X

使用法

level 変数名は REXX 変数または数字で、有効な値は 0 から 9 です。ユーザー出口で指定変更されないかぎり、EXEC 開始の初期値は 1 です。トレース出力は、DDNAME SYSTSPRT に送られます。IMS アダプター (REXX 版) 出口ルーチンの詳細については、「IMS バージョン 14 出口ルーチン」を参照してください。

IMSRXTRC コマンドは、通常の REXX トレース (TRACE) と組み合わせて使用することも、TRACE の代わりに使用することもできます。

レベル 説明

0 エラーのみのトレース

1 前のレベルと DL/I 呼び出し、その戻りコード、および環境状況のトレース (フロー分析に有効)

2 前のレベルと変数の集合のすべて

3 前のレベルと変数の取り出しのすべて (問題の診断に有効)

4-7 前のレベルすべて

8 前のレベルすべて、および標準 IMS 言語インターフェースとの間のパラメーター・リスト。「*IMS V14* メッセージおよびコード 第 *2* 巻*: DFS* 以外メッセージ」のメッセージ DFS3179 を参照してください。

9 前のレベルすべて

例

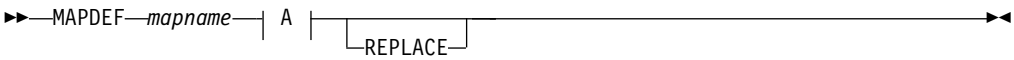
```
Address REXXIMS 'IMSRXTRC 3'
```

IMSRXTRC は、REXX TRACE 命令とは独立するものです。

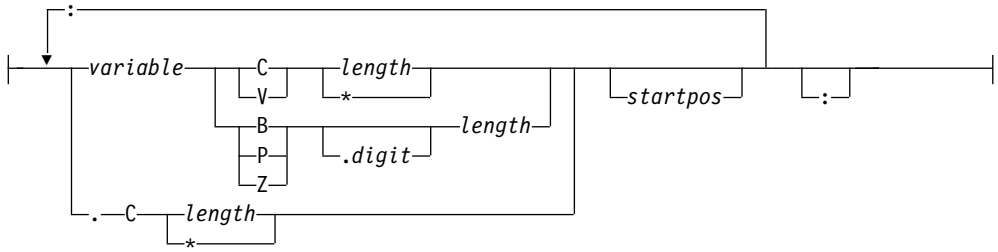
MAPDEF

MAPDEF コマンドは、データのマッピングを定義するよう要求します。

フォーマット



A:



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
MAPDEF	X	X	X	X	X

使用法

データのマッピングは、REXXIMS インターフェースに追加された機能拡張です。REXX には変数構造がないので、データ変換が必要なときは特に、データベース・セグメントや MFS 出力マップからのフィールドを構文解析するのに時間がかかります。MAPDEF、MAPGET、および MAPPUT コマンドを使用すると、ほとんどの定様式データを簡単に抽出することができます。

- *mapname* は 1 文字から 16 文字で、大/小文字のどちらを使用してもよい名前。
- definition (**A**) は、マップ定義が入る変数。
- REPLACE が指定される場合は、既存のマップ名の置換が認められる。指定されず、マップ名がすでに定義されている場合はエラーが起こり、メッセージ DFS3171E が SYSTPRT に送られます。

マップ定義 の形式は、REXX 用に単純化されており、他の言語でのデータ宣言に似ています。この定義では、構文解析するすべての変数を、適切なデータ・タイプで宣言してください。形式は、構文図の **A** に示されています。

変数名

変数名 *variable* は、構文解析した情報を入れるために使用される REXX 変数です。変数名は大/小文字のどちらを使用してもかまいません。 STEM (配列構造に関する REXX 用語) 変数を使用すれば、使用時 (明示または暗黙の `MAPGET` または `MAPPUT` 呼び出し時) に解析され、これが効果的です。複合変数の STEM 部分として索引タイプの変数を使用するときは、索引変数を変更すれば、いくつものレコードを配列に簡単にロードすることができます。マップ名またはトークンは、マップ定義内で変数名に置換されることはありません。

内部カーソルの位置変更

ピリオド (.) は内部のカーソル位置を変更するための変数プレースホルダーとして使用されます。この場合、データ・タイプは C でなければなりませんが、長さは負の数、正の数、ゼロのいずれでもかまいません。関係のないフィールドは、正の値を使用してスキップします。マップの中間にあるフィールドについては、絶対位置決めは使用せず、負の長さを使用して再定義します。

データ・タイプ値は次のとおりです。
C 文字
V 可変
B 2 進数 (数値)
Z ゾーン 10 進数 (数値)
P パック 10 進数 (数値)

すべての数値データ・タイプには、ピリオドおよび数値を続けることができます。この数値は、数値を変換する際の、10 進小数点の右側の桁数を表します。

length 値

length 値は、数値か、残りのバッファーが使用されることを表すアスタリスク (*) にします。アスタリスクはデータ・タイプ C および V にのみ指定できます。データ・タイプ V は、宣言される長さが 2 のときに 4 バイトになるように、データ・ストリングに先行する 2 バイトの長さフィールドをマップします。

データ・タイプに対応する有効な長さは、次のとおりです。
C 1 バイトから 32767 バイトまたは *
V 1 バイトから 32765 バイトまたは *
B 1 から 4 バイト
Z 1 から 12 バイト
P 1 から 6 バイト

アスタリスク (*) 以外の値の場合は、その値によってカーソル位置が移動します。

startpos 値を使用して、構文解析の位置を固定場所に再設定します。*startpos* が省略されると、直前のマップ変数定義 (カーソル位置) の右の桁が使用されます。変数定義が初めてであれば、桁 1 が使用されます。

注**:** 長さがアスタリスク (*) の場合にはカーソル位置は移動しないので、長さがアスタリスク (*) で開始桁が指定されていない変数のあとに宣言される変数は、同じ定義をオーバーレイします。

例

この例では、DBMAP という名前のマップを定義します。マップ名の前にアスタリスク (*) を置くと、DBMAP が GU 呼び出しで暗黙に使用されます。

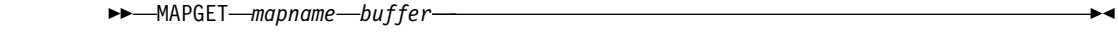
```
DBMapDef = 'RECORD         C    * :',  /* Pick up entire record     */
           'NAME           C   10 :',  /* Cols 1-10 hold the name   */
           'PRICE          Z.2  6 :',  /* Cols 11-16 hold the price */
           'CODE           C    2 :',  /* Cols 11-16 hold the code  */
           '.              C   25 :',  /* Skip 25 columns           */
           'CATEGORY       B    1'     /* Col 42 holds category     */
Address REXXIMS 'MAPDEF DBMAP DBMapDef'

.
.
.

Address REXXTDLI 'GU DBPCB *DBMAP' /* Read and decode a segment      */
If RC¬=0 Then Signal BadCall       /* Check for failure             */
Say CODE                           /* Can now access any Map Variable*/
```

GU 呼び出しで検索されるセグメント全体が、RECORD に配置されます。最初の 10 文字は NAME に配置され、次の 6 文字は、ゾーン 10 進数から小数点以下 2 桁の EBCDIC に変換されて、PRICE に配置されます。次の 2 文字は CODE に配置され、その次の 25 文字はスキップされて、次の文字は 2 進数から EBCDIC に変換されて CATEGORY に配置されます。スキップされた 25 文字は、RECORD 変数に表示されます。

MAPGET

MAPGET コマンドは、バッファーの構文解析や変換を行って、以前に MAPDEF コマンドで定義した、指定データ・マッピングにするよう要求します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
MAPGET	X	X	X	X	X

使用法

mapname 変数名は、使用するデータ・マッピングを指定します。これは、1 文字から 16 文字の名前で、大/小文字のどちらを使用してもかまいません。 *buffer* 変数名は、構文解析するデータを含む REXX 変数です。

マップ名は、変数名を設定したり書き込む代わりに、REXXTDLI 呼び出しで指定することもできます。このステップは、暗黙の MAPGET と呼ばれます。こうすると、明示の (または変数に依存した) MAPGET 呼び出しは回避されます。DL/I 呼び出しで変数の代わりにマップ名が渡されるように指示するには、例えば 'GU IOPCB *INMAP' のように、アスタリスク (*) の付いている名前を優先させます。

例

次の例では、明示サポートが使用されています。

```
Address REXXTDLI 'GU DBPCB SegVar'
If RC=0 Then Signal BadCall          /* Check for failure              */
Address REXXIMS 'MAPGET DBMAP SegVar'/* Decode Segment                 */
Say VAR_CODE                         /*Can now access any Map Variable */
```

次の例では、暗黙サポートが使用されています。

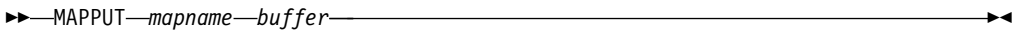
```
Address REXXTDLI 'GU DBPCB *DBMAP'   /* Read and decode segment if read*/
If RC=0 Then Signal BadCall          /* Check for failure              */
Say VAR_CODE                         /* Can now access any Map Variable*/
```

MAPGET 時にエラーが起こると、メッセージ DFS3172I が出されます。エラーは、デコードされる入力セグメントより大きい MAP が定義されるときや、パック 10 進数形式またはゾーン 10 進数形式からのデータ変換のエラー時に起こります。プログラムは継続し、明示の MAPGET は戻りコード 4 を受け取ります。一方、暗黙の MAPGET (例えば REXXTDLI 呼び出しの場合) の戻りコードは影響を受けません。どちらの場合でも、障害のある変数の値は REXX が除去します。

MAPPUT

MAPPUT コマンドは、MAPDEF コマンドで定義した指定データ・マッピングからの変数を、単一の変数にパックもしくは連結するように要求します。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
MAPPUT	X	X	X	X	X

使用法

mapname 変数名は、使用するデータ・マッピングを指定する 1 文字から 16 文字の名前で、大/小文字のどちらを使用してもかまいません。*buffer* 変数名は、結果の値が入る REXX 変数です。

マップ名は、変数名を取り出したり読み込む代わりに、REXXTDLI 呼び出しで指定することもできます。このステップは暗黙の MAPPUT と呼ばれ、明示の MAPPUT 呼び出しを回避することができます。DL/I 呼び出しで、変数の代わりにマップ名が渡されるように指示するには、例えば 'ISRT IOPCB *OUTMAP' のように、アスタリスク (*) の付いている名前を優先させます。

注: データ・マッピングが単に部分的なもので、レコード内のフィールドで REXX 変数にマップしないものがあるときは、マッピングの最初のフィールドは、以下のコード例に示されるように長さがアスタリスク (*) である文字タイプとなります。マップされない (スキップされる) フィールドが明示であるか暗黙であるかに関係なく、MAPGET 呼び出しと MAPPUT 呼び出しの間で脱落しないようにするには、このステップをとるしかありません。

次の例では、明示サポートが使用されています。

```
Address REXXTDLI
'GHU DBPCB SegVar SSA1'              /* Read segment             */
If RC¬=0 Then Signal BadCall         /* Check for failure        */
```

```
Address REXXIMS 'MAPGET DBMAP SegVar'   /* Decode Segment           */
DBM_Total = DBM_Total + Deposit_Amount  /* Adjust Mapped Variable   */
Address REXXIMS 'MAPPUT DBMAP SegVar'   /* Encode Segment           */
'REPL DBPCB SegVar'                     /* Update Database          */
If RC¬=0 Then Signal BadCall            /* Check for failure        */
```

次の例では、暗黙サポートが使用されています。

```
Address REXXTDLI
'GHU DBPCB *DBMAP SSA1'                 /* Read and decode segment if read */
If RC¬=0 Then Signal BadCall            /* Check for failure               */
DBM_Total = DBM_Total + Deposit_Amount  /* Adjust Mapped Variable          */
'REPL DBPCB *DBMAP'                     /* Update Database                 */
If RC¬=0 Then Signal BadCall            /* Check for failure               */
```

MAPPUT 時に、マップ・フィールドが変数の内容より大きく定義されている、といったエラーが起こると、そのフィールドは切り捨てられます。変数の内容がフィールドより小さければ、変数は次のように埋め込まれます。

文字 **(C)**
: 右方をブランクで埋め込む

文字 **(V)**
: 右方をゼロで埋め込む

数値 **(B、Z、P)**
: 左方をゼロで埋め込む

MAPPUT が処理されているときに MAP 変数が存在しないときは、変数および変数の位置はスキップされます。未定義でスキップされるフィールドのデフォルトは、すべて 2 進ゼロです。ヌル・パラメーターは、通常どおりに構文解析されます。非数値またはヌルのフィールドを数値フィールドに変換すると、値 0 が使用され、エラーにはなりません。

SET

SET コマンドは、DL/I 呼び出しの前に、AIB 副次機能の値および ZZ 値を再設定します。

フォーマット

呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
SET	X	X	X	X	X

使用法

SET SUBFUNC コマンドで、次の DL/I 呼び出しで使用される AIB 副次機能を設定します。この値は、次の REXXTDLI が PCB 名を渡す場合にのみ使用されます。この呼び出しで PCB 名を渡す場合は、呼び出しの前に、 IMS アダプター (REXX 版) が副次機能名 (1 個から 8 個の文字またはブランク) を AIB に入れます。この

値は最初はデフォルトとしてブランクになっており、任意の REXXTDLI DL/I 呼び出しが完了するたびにブランクにリセットされます。

SET ZZ コマンドを使用して、次に続く DL/I 呼び出しで使用される ZZ 値を設定します。このコマンドは、IMS 会話型トランザクションおよび端末装置に依存するアプリケーションで主に使用され、デフォルトの 2 進ゼロ以外の値に ZZ フィールドを設定します。デフォルトの ZZ 値以外の値を必要とする ISRT 呼び出しの前に、SET コマンドを使用してください。

ICAL 呼び出しによって同期コールアウト要求を発行する場合は、以下の使用規則が適用されます。

- SET SUBFUNC コマンドは、副次機能コードを SENDRECV に設定して発行する必要があります。
- SET RSNAME1 コマンドは、変数を OTMA 記述子の名前に設定して発行する必要があります。
- オプションで、SET TIMER コマンドを発行して ICAL タイムアウト値を設定することができます。タイムアウト値は 6 桁以内の数値で指定する必要があります。

例

この例では、INQY 呼び出しと SET SUBFUNC コマンドを使用して、環境情報を入手します。

```
IO="IOPCB"
Func = "ENVIRON"                       /* Sub-Function Value */
Address REXXIMS  "SET SUBFUNC Func"    /* Set the value      */
Address REXXTDLI "INQY IO EnviData"    /* Make the DL/I Call */
IMS_Identifier = Substr(EnviData,1,8)  /* Get IMS System Name*/
```

次の例では、SPA 処理のために SET ZZ コマンドが会話型トランザクションとともに使用されています。

```
Address REXXTDLI 'GU IOPCB SPA'           /* Get first Segment         */
Hold_ZZ = IMSQUERY('ZZ')                  /* Get ZZ Field (4 bytes)    */

.
.
.

Address REXXIMS  'SET ZZ Hold_ZZ'        /* Set ZZ for SPA ISRT       */
Address REXXTDLI 'ISRT IOPCB SPA'        /* ISRT the SPA              */
```

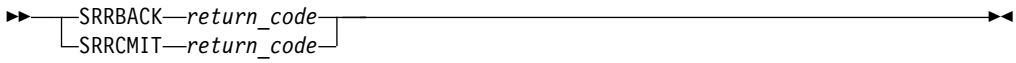
次の例では、3270 装置特性フラグを設定するために SET ZZ コマンドが使用されています。

```
Bell_ZZ = '0040'X                        /* ZZ to Ring Bell on Term   */
Address REXXIMS  'SET ZZ Bell_ZZ'        /* Set ZZ for SPA ISRT       */
Address REXXTDLI 'ISRT IOPCB Msg'        /* ISRT the Message          */
```

SRRBACK および SRRCMIT

共通プログラミング・インターフェース・リソース・リカバリー (CPI-RR) コマンドを使用すると、処理をバックアウトおよびコミットする SAA リソース・リカバリー・インターフェース機能をインターフェースで使用することができます。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
SRRBACK、SRRCMIT	X		X		

使用法

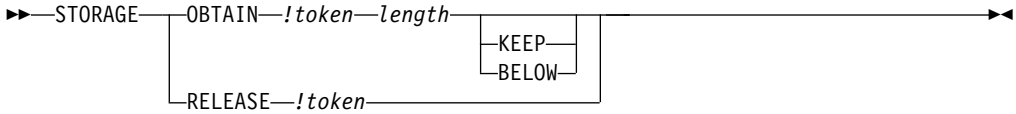
SRR コマンドからの戻りコードは、REXX 変数 RC および *return_code* 変数名に返されて入れられます。

SRRBACK および SRRCMIT の詳細については、「*IMS V14* コミュニケーションおよびコネクション」および「*SAA CPI* 資源回復解説書」を参照してください。

STORAGE

STORAGE コマンドを使用すると、REXXTDLI および REXXIMS 呼び出しにおけるパラメーターの変数の代わりに使用できるシステム・ストレージを獲得できます。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
STORAGE	X	X	X	X	X

使用法

REXX では、文字 (!) および (#) で始まる変数を使用できますが、これらの文字が特別な意味を持つコマンドもあります。 REXXTDLI インターフェースを使用するときは、この文字を変数の最初の文字に使用しないでください。

!token 変数名はストレージを識別し、1 個の感嘆符と、それに続く 1 から 16 文字の大/小文字のどちらを使用してもよいトークン名で構成されます。*length* 変数名は数値または変数で、4 から 16777216 バイト (16 MB) の範囲で獲得する (OBTAIN)、10 進数のサイズが入ります。ストレージ・クラスで指定変更できるのは BELOW および KEEP の 2 つで、どのような特別なトークンの場合でも指定できるのはこのうちの 1 つだけです。BELOW 関数は、16 MB 境界より下の専用ストレージを獲得します。KEEP 関数は、この EXEC が終了した後にトークンが保持されるようマークされます。デフォルトのアクションでは、あらゆる場所でストレージを入手し、EXEC が終了するとトークンを解放します。

STORAGE コマンドを使用して、入出力域が固定場所 (例えば、スプール API) になければならないときや、LLZZ 処理を回避するときに DL/I 呼び出しで使用するストレージを取得してください。トークンは、割り振られた後は、REXXTDLI DL/I 呼び出しや STORAGE RELEASE コマンドで使用することができます。

STORAGE を使用する際は、次のことに注意してください。

- DL/I 呼び出しで使用するときは、LLZZ フィールド用の設定はない。 REXX 以外のアプリケーションで必要とされるようにトークンを充てんし、その結果を構文解析してください。
- 単一の STORAGE コマンドに、KEEP と BELOW の両方を指定することはできない。
- RELEASE 関数は、KEEP がマークされたトークンでのみ必要である。 KEEP がマークされていないトークン、および EXEC が終了するまでに明示的に解放されないトークンは、すべて、IMS アダプター (REXX 版) が自動的に解放されます。
- OBTAIN を使用するときは、記憶ブロック全体が 0 に初期設定される。
- 受け取られるストレージの開始アドレスは、常にダブルワードの境界にある。
- RELEASE を使用するか、KEEP 以外で RELEASE を取得した EXEC が終了するまで、トークンを再取得できない。試行すると、戻りコード -9 が与えられ、エラー・メッセージ DFS3169 が出されます。
- ストレージ・トークンに KEEP を指定した場合には、その EXEC またはそのトークンの名前を認識している別の EXEC が同じ IMS 領域で開始されたときに、トークンにアクセスし直すことができる。
- KEEP がマークされたトークンは、ABEND が起こったり、領域のストレージをクリアするような他の誤動作が起こると保存されない。 IMSQUERY(*!* token) 機能を使えば、入力時に、そのブロックが存在するかどうかを調べるのは簡単です。

例

この例では、スプール API を使用した STORAGE コマンドの使用方法を示しています。

```
/* Get 4K Buffer below the line for Spool API Usage */
Address REXXIMS 'STORAGE OBTAIN !MYTOKEN 4096 BELOW'
/* Get Address and length (if curious) */
Parse Value IMSQUERY('!MYTOKEN') With My_Token_Addr My_Token_Len.
Address REXXIMS 'SETO ALTPCB !MYTOKEN SETOPARMS SETOFB'

.
.
.

Address REXXIMS 'STORAGE RELEASE !MYTOKEN'
```

関連資料:

415 ページの『REXXTDLI 呼び出し』

432 ページの『IMSQUERY 拡張機能』

WTO、WTP、および WTL

WTO コマンドは、オペレーターへのメッセージを作成するときに使用します。WTP コマンドは、プログラムへのメッセージを作成するときに使用します (WTO ROUTCDE=11)。 WTL コマンドは、コンソール・ログへのメッセージを作成するときに使用します。

フォーマット

呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
WTO、WTP、WTL	X	X	X	X	X

使用法

message 変数名は、保管されて適切な場所に表示されるテキストが入る REXX 変数です。

例

この例は、保管されている簡単なメッセージを REXX 変数 MSG に書き込む方法を示しています。

```
Msg = 'Sample output message.'          /* Build Message            */
Address REXXIMS 'WTO Msg'               /* Tell Operator            */
Address REXXIMS 'WTP Msg'               /* Tell Programmer          */
Address REXXIMS 'WTL Msg'               /* Log It                   */
```

WTOR

WTOR コマンドは、z/OS システム・オペレーターからの入力や応答を要求します。

フォーマット

呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
WTOR	X	X	X	X	X

使用法

message 変数名は、z/OS コンソールに表示されるテキストが入る REXX 変数です。オペレーターの応答は、*response* 変数名で示される REXX 変数に入れられます。

注意 : このコマンドは、オペレーターが応答するまで、コマンドが実行中の IMS 領域を停止します。

例

この例では、z/OS マスター・コンソールまたは代替コンソールで ROLL または CONT を入力するよう、オペレーターに要求します。WTOR が応答された後は、その応答は REXX 変数名 *response* に入れられ、EXEC は引き続き IF ステートメントを適切に処理します。

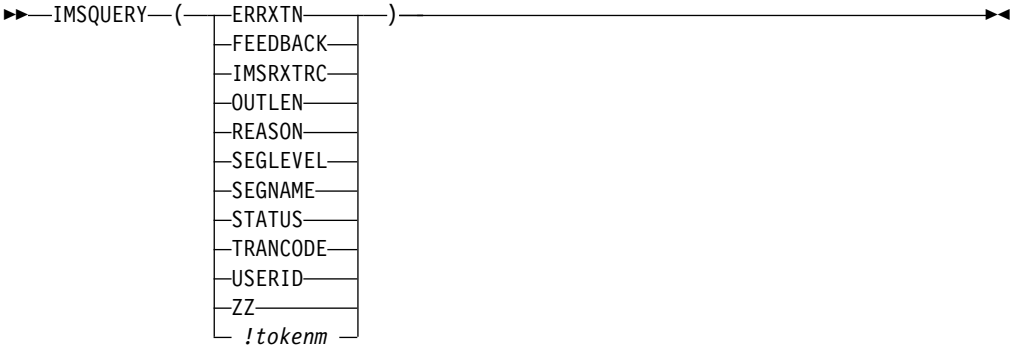
```
Msg = 'Should I ROLL or Continue. Reply "ROLL" or "CONT"'
Address REXXIMS 'WTOR Msg Resp'          /* Ask Operator     */
If Resp = 'ROLL' Then                    /* Tell Programmer  */
   Address REXXTDLI 'ROLL'               /* Roll Out of this */
```

IMSQUERY 拡張機能

IMSQUERY 機能は、環境についてまたは先行 DL/I 呼び出しについて、特定の IMS 情報を照会するために使用できます。

ICAL 呼び出しを使用して同期コールアウト要求を発行した後に、IMSQUERY 関数を使用して戻りコードと理由コードを確認することができます。戻りコードまたは理由コードに、部分的な出力データが返されたことが示されている場合は、IMSQUERY 関数を発行して出力データ長とエラー拡張コードを検索することができます。

フォーマット



呼び出し名	DB/DC	DBCTL	DCCTL	DB バッチ	TM バッチ
IMSQUERY	X	X	X	X	X

使用法

機能呼び出しの形式は IMSQUERY ('Argument') で、引数 (Argument) は以下のリストのいずれかの値です。

引数　返されたデータの説明

ERRXTN
ICAL を使用して同期コールアウト要求が発行された後にエラー応答メッセージが返された場合のエラー拡張コード

FEEDBACK
現行 PCB からの FEEDBACK 域

IMSRXTRC
現行 IMSRXTRC トレース・レベル #

OUTLEN
ICAL を使用して同期コールアウト要求を発行した後に部分的な応答が返された場合の応答メッセージの出力長

REASON
最新の呼び出し (最新の REXXTDLI タイプ呼び出しで AIB が使用された場合は、AIB) からの理由コード

SEGLEVEL
現行 PCB からのセグメント・レベル (最新の REXXTDLI 呼び出しは DB PCB に対するものでなければなりません。そうでない場合は、ヌルが返されます)

SEGNAME
現行 PCB からのセグメント名 (最新の REXXTDLI 呼び出しは DB PCB に対するものでなければなりません。そうでない場合は、ヌルが返されます)

STATUS
最後に実行された REXXTDLI 呼び出し (DL/I 呼び出し) からの IMS 状況コード。この引数は、PCB からの 2 文字の状況コードです。

TRANCODE
使用可能であれば、処理中のトランザクション・コード。

USERID
利用可能であれば入力端末のユーザー ID。非メッセージ・ドリブン領域で実行中の場合、値は DFSDCxxx PROCLIB メンバーの BMPUSID= キーワードの指定によって決まります。

- BMPUSID=USERID が指定されている場合、JOB ステートメントの USER= キーワードの値が使用されます。
- JOB ステートメントで USER= が指定されていない場合は、プログラムの PSB 名が使用される。
- BMPUSID=PSBNAME が指定されている場合、または BMPUSID= がまったく指定されていない場合は、プログラムの PSB 名が使用されます。

ZZ 最新の REXXTDLI コマンドからの (LLZZ の) ZZ。トランザクションが会話型であれば、入出力 PCB への GU 呼び出しを発行した後で、この引数を使用して ZZ 値が保管されます。

トークン
ブランクで分離された、アドレス (16 進数) と指定トークンの長さ (10 進数)

この値は、変数に配置されるか、式で解決されます。その場合、以下に示すように引用符を省略しなければなりません。

```
Token_Name="!MY_TOKEN"
AddrInfo=IMSQUERY(Token_Name)
        /*  or  */
AddrInfo=IMSQUERY("!MY_TOKEN")
```

関数引数には大文字と小文字のどちらを使用してもかまいません。関数引数の中ではブランクは使用できません。必要であれば、REXX STRIP 関数を引数で使用することができます。IMSQUERY が優先構文ですが、REXXIMS もサポートされており、同様に使用できます。

例

```
If REXXIMS('STATUS')='GB' Then Signal End_Of_DB
:
Hold_ZZ = IMSQUERY('ZZ')        /* Get current ZZ field*/
:
Parse Value IMSQUERY('!MYTOKEN') With My_Token_Addr My_Token_Len .
```

関連資料:


IMS アダプター (REXX 版) 出口ルーチン (DFSREXXU) (出口ルーチン)

414 ページの『REXXTDLI コマンド』

429 ページの『STORAGE』

REXXTDLI を使用するサンプル EXEC

以下の REXX EXEC のサンプルは、IMS サービスにアクセスするための REXXTDLI の使用方法を示しています。

これらの例は、REXX で IMS アプリケーションを作成するためのさまざまな機能を強調するように設計されています。サンプルは単純化されており、実際に使用される状態を反映していません (例えば、サンプルではデータベースは使用していません)。

PART EXEC データベース・アクセスの例は、IMS インストール検査プログラム (IVP) が作成する PART データベースにアクセスする 3 つの EXEC の集合です。この例の最初の 2 つの EXEC である PARTNUM および PARTNAME は、IVP の一部として IMS とともに提供されるプログラム DFSSAM02 を実行する PART トランザクションの拡張機能です。 3 番目の EXEC は DFSSAM01 EXEC で、IMS とともに提供され、EXEC 内で EXECIO を使用する例です。

SAY EXEC : 式の計算

以下のコード例は SAY EXEC のリストです。SAY は、引数として提供される式を計算し、結果を表示します。

REXX コマンドの INTERPRET は、指定された式を計算して変数に割り当てるために使用されます。この変数は定様式応答メッセージで使用されます。

計算を行う EXEC

```
/* EXEC TO DO CALCULATIONS */
Address REXXTDLI
Arg Args
If Args='' Then
   Msg='SUPPLY EXPRESSION AFTER EXEC NAME.'
Else Do
   Interpret 'X='Args           /* Evaluate Expression */
```

```
   Msg='EXPRESSION:' Args '=' X
End
'ISRT IOPCB MSG'
Exit RC
```

この EXEC は、IMS アダプター (REXX 版) を使用したアプリケーション開発の例です。また、実行時の数式を評価する機能である動的解釈など REXX の利点も示されています。

PDF EDIT セッションを以下の図に示します。この図は、新規の EXEC を入力して IMS のもとで実行させる方法を示しています。

```
EDIT ---- USER.PRIVATE.PROCLIB(SAY) - 01.03 ------------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> PAGE
****** ***************************** TOP OF DATA ******************************
000001 /* EXEC TO DO CALCULATIONS */
000002 Address REXXTDLI
000003 Arg Args
000004 If Args='' Then
000005    Msg='SUPPLY EXPRESSION AFTER EXEC NAME.'
000006 Else Do
000007    Interpret 'X='Args           /* Evaluate Expression */
000008    Msg='EXPRESSION:' Args '=' X
000009 End
000010
000011 'ISRT IOPCB MSG'
000012 Exit RC
****** **************************** BOTTOM OF DATA ****************************
```

図 *10. SAY EXEC* での *PDF EDIT* セッション

SAY EXEC を実行するには、IVPREXX を使用して次の式を指定してください。

```
IVPREXX SAY 5*5+7
```

この式は、以下の図に示されるような出力を作成します。

```
EXPRESSION: 5*5+7 = 32
EXEC SAY ended with RC= 0
```

図 *11. SAY EXEC* からの出力の例

PCBINFO EXEC : 現行 PSB で使用可能な PCB の表示

PCB EXEC は、EXEC で使用可能な PCB (実行中の PSB の PCB) を EXEC にマップします。

マッピングでは、PCB のタイプ (IO、TP、または DB)、関連する LTERM または DBD 名、およびその他の有用な情報を表示します。PCB マッピングを作成するには、早期に連結されたライブラリーに DFSREXX0 を入れ、PSB/DBD を生成して、これを既存のアプリケーションの名前に変更します。

```
IMS PCB System Information Exec: PCBINFO
System Date: 09/26/92    Time: 15:52:15

PCB # 1: Type=IO, LTERM=T3270LC  Status=   UserID=         OutDesc=DFSMO2
         Date=91269 Time=1552155
PCB # 2: Type=TP, LTERM=* NONE * Status=AD
PCB # 3: Type=TP, LTERM=* NONE * Status=
PCB # 4: Type=TP, LTERM=CTRL     Status=
PCB # 5: Type=TP, LTERM=T3275    Status=
EXEC PCBINFO ended with RC= 0
```

図 *12.* データベース *PCB* を使用しない *PSB* での *PCBINFO EXEC* 出力の例

```
IMS PCB System Information Exec: PCBINFO
System Date: 09/26/92    Time: 15:53:34

PCB # 1: Type=IO, LTERM=T3270LC  Status=   UserID=         OutDesc=DFSMO2
         Date=89320 Time=1553243
PCB # 2: Type=DB, DBD  =DI21PART Status=   Level=00 Opt=G
EXEC PCBINFO ended with RC= 0
```

図 *13.* データベース *PCB* を使用する *PSB* での *PCBINFO EXEC* 出力の例

PCBINFO EXEC のリスト

```
/* REXX EXEC TO SHOW SYSTEM LEVEL INFO */
Address REXXTDLI
Arg Dest .
WTO=(Dest='WTO')
Call SayIt 'IMS PCB System Information Exec: PCBINFO'
Call SayIt 'System Date:' Date('U') '   Time:' Time()
Call Sayit ' '
/* A DFS3162 message is given when this exec is run because it does */
/* not know how many PCBs are in the list and it runs until it gets */
/* an error return code.  Note this does not show PCBs that are     */
/* available to the PSB by name only, in other words, not in the PCB list.     */
Msg='PCBINFO: Error message normal on DLIINFO.'
'WTP MSG'
Do i=1 by 1 until Result='LAST'
   Call SayPCB i
End
Exit 0


SayPCB: Procedure Expose WTO
   Arg PCB
   'DLIINFO DLIINFO #'PCB    /* Get PCB Address */
   If rc<0 Then Return 'LAST'   /* Invalid PCB Number */
   Parse Var DLIInfo  . . AIBAddr PCBAddr .
   PCBINFO=Storage(PCBAddr,255) /* Read PCB */
   TPPCB=(Substr(PCBInfo,13,1)='00'x) /* Date Field, must be TP PCB */
   If TPPCB then Do
      Parse Value PCBInfo with,
         LTERM 9 . 11 StatCode 13 CurrDate 17 CurrTime 21,
         InputSeq 25 OutDesc 33 UserID 41
      If LTERM='' then LTERM='* NONE *'
      CurrDate=Substr(c2x(CurrDate),3,5)
      CurrTime=Substr(c2x(CurrTime),1,7)
      If CurrDate¬='000000' then Do
         Call SayIt 'PCB #'Right(PCB,2)': Type=IO, LTERM='LTERM,
            'Status='StatCode 'UserID='UserID 'OutDesc='OutDesc
         Call SayIt '         Date='CurrDate 'Time='CurrTime
      End
      Else
         Call SayIt 'PCB #'Right(PCB,2)': Type=TP, LTERM='LTERM,
```

```
            'Status='StatCode
   End
   Else Do
      Parse Value PCBInfo with,
         DBDName 9 SEGLev 11 StatCode 13 ProcOpt 17 . 21 Segname . 29,
         KeyLen 33 NumSens 37
      KeyLen = c2d(KeyLen)
      NumSens= c2d(NumSens)

      Call SayIt 'PCB #'Right(PCB,2)': Type=DB, DBD  ='DBDName,
               'Status='StatCode 'Level='SegLev 'Opt='ProcOpt
   End
Return '

SayIt: Procedure Expose WTO
   Parse Arg Msg
   If WTO Then
      'WTO MSG'
   Else
      'ISRT IOPCB MSG'
```

復帰文字

関連資料:

421 ページの『DLIINFO』

PART EXEC : データベースへのアクセス例

この一連の EXEC は、IMS とともに出荷される PART データベースにアクセスします。この EXEC で、固定レコード・データベース読み取り、SSA、および多くの REXX 関数について説明します。PART データベース EXEC (PARTNUM、PARTNAME、および DFSSAM01) についても説明されています。

PARTNUM EXEC を使用して、指定する数値と等しいか、これより大きい数値で始まる部品番号を表示します。出力画面の例を以下の図に示します。

『300』 またはそれより大きい番号で始まる部品番号をリストするには、次のコマンドを入力します。

```
PARTNUM 300
```

300 またはそれ以上の数で始まる部品番号は、すべてリストされます。このリストは、以下の図に示されています。

```
IMS Parts DATABASE Transaction
System Date: 02/16/92    Time: 23:28:41

Request: Display 5 Parts with Part_Number >= 300
 1 Part=3003802          Desc=CHASSIS
 2 Part=3003806          Desc=SWITCH
 3 Part=3007228          Desc=HOUSING
 4 Part=3008027          Desc=CARD FRONT
 5 Part=3009228          Desc=CAPACITOR

EXEC PARTNUM ended with RC= 0
```

図 *14. PARTNUM EXEC* 出力の例

PARTNAME を使用して、特定の文字ストリングで始まる部品名を表示します。

『TRAN』 で始まる部品名をリストするには、次のコマンドを入力します。

```
PARTNAME TRAN
```

『TRAN』 で始まる部品名がすべて画面にリストされます。この画面は、以下の図に示されています。

```
IMS Parts DATABASE Transaction
System Date: 02/16/92    Time: 23:30:09

Request: Display 5 Parts with Part Name like TRAN
 1 Part=250239           Desc=TRANSISTOR
 2 Part=7736847P001      Desc=TRANSFORMER
 3 Part=975105-001       Desc=TRANSFORMER
 4 Part=989036-001       Desc=TRANSFORMER
End of DataBase reached before 5 records shown.

EXEC PARTNAME ended with RC= 0
```

図 *15. PARTNAME EXEC* 出力の例

DFSSAM01 EXEC を使用して、部品データベースをロードします。この EXEC はバッチで実行されます。IVP の一部で、EXEC での EXECIO の使用法の例です。

関連資料: 詳細は、「*IMS V14* インストール」を参照してください。

PARTNUM EXEC: 指定された数値に近い部品の集合の表示

以下のコード例は、PSB=DFSSAM02 を指定して IVPREXX EXEC で実行するように設計されています。

PARTNUM EXEC: 指定された数値に近い部品の集合の表示

```
/* REXX EXEC TO SHOW A SET OF PARTS NEAR A SPECIFIED NUMBER    */
/* Designed to be run by the IVPREXX exec with PSB=DFSSAM02    */
/* Syntax:    IVPREXX PARTNUM string <start#>                  */

Address REXXTDLI
IOPCB='IOPCB'      /* PCB Name */
DataBase='#2'      /* PCB # */
RootSeg_Map = 'PNUM C 15 3 : DESCRIPTION C 20 27'
'MAPDEF ROOTSEG ROOTSEG_MAP'
Call SayIt 'IMS Parts DATABASE Transaction'
Call SayIt 'System Date:' Date('U') '   Time:' Time()
Call Sayit ' '

Arg  PartNum Segs .
If ¬DataType(Segs,'W') then Segs=5  /* default view amount */

PartNum=Left(PartNum,15)    /* Pad to 15 with Blanks */
If PartNum='' then
  Call Sayit 'Request: Display first' Segs 'Parts in the DataBase'
Else
  Call Sayit 'Request: Display' Segs 'Parts with Part_Number >=' PartNum
SSA1='PARTROOT(PARTKEY >=02'PartNum')'
'GU DATABASE *ROOTSEG SSA1'
Status=IMSQUERY('STATUS')
If Status='GE' then Do /* Segment Not Found */
   Call Sayit 'No parts found with larger Part_Number'
   Exit 0
End
Do i=1 to Segs While Status='  '
   Call Sayit Right(i,2) 'Part='PNum ' Desc='Description
   'GN DATABASE *ROOTSEG SSA1'
   Status=IMSQUERY('STATUS')
```

```
End
If Status='GB' then
   Call SayIt 'End of DataBase reached before' Segs 'records shown.'
Else If Status¬='  ' then Signal BadCall
Call Sayit ' '
   Exit 0

SayIt: Procedure  Expose IOPCB
   Parse Arg Msg
   'ISRT IOPCB MSG'
   If RC¬=0 then Signal BadCall
戻る

BadCall:
   'DLIINFO INFO'
   Parse Var Info Call PCB . . . . Status .
   Msg = 'Unresolved Status Code' Status,
      'on' Call 'on PCB' PCB
   'ISRT IOPCB MSG'
Exit 99
```

PARTNAME EXEC : 類似した名前の部品の集合の表示

以下のコード例に示されている REXX EXEC は、PSB=DFSSAM02 を指定して IVPREXX EXEC で実行するように設計されています。

次の PARTNAME 実行可能コードは、類似した名前のパーツを示すために使用されます。

```
/* REXX EXEC TO SHOW ALL PARTS WITH A NAME CONTAINING A STRING  */
/* Designed to be run by the IVPREXX exec with PSB=DFSSAM02     */
/* Syntax:   IVPREXX PARTNAME string <#parts>                   */

Arg  PartName Segs .
Address REXXIMS
Term    ='IOPCB'      /* PCB Name */
DataBase='DBPCB01'    /* PCB Name for Parts Database */

Call SayIt 'IMS Parts DATABASE Transaction'
Call SayIt 'System Date:' Date('U') '   Time:' Time()
Call Sayit ' '

If ¬DataType(Segs,'W') & Segs¬='*' then Segs=5
If PartName='' then Do
   Call Sayit 'Please supply the first few characters of the part name'
   Exit 0
End

Call Sayit 'Request: Display' Segs 'Parts with Part Name like' PartName
SSA1='PARTROOT '
'GU DATABASE ROOT_SEG SSA1'
Status=REXXIMS('STATUS')
i=0
Do While RC=0 & (i<Segs | Segs='*')
   Parse Var Root_Seg 3 PNum 18 27 Description 47
   'GN DATABASE ROOT_SEG SSA1'
   Status=REXXIMS('STATUS')
   If RC¬=0 & Status¬='GB' Then Leave
   If Index(Description,PartName)=0 then Iterate
   i=i+1
   Call Sayit Right(i,2)') Part='PNum ' Desc='Description
End
If RC¬=0 & Status¬='GB' Then Signal BadCall
If i<Segs & Segs¬='*' then
   Call SayIt 'End of DataBase reached before' Segs 'records shown.'
Call Sayit ' '
```

```
Exit 0

SayIt: Procedure Expose Term
   Parse Arg Msg
   'ISRT Term MSG'
   If RC¬=0 then Signal BadCall
戻る

BadCall:
   Call "DFSSUT04" Term
Exit 99
```

DFSSAM01 EXEC : 部品データベースのロード

最新版の DFSSAM01 ソース・コードは、IMS.ADFSEXEC の配布ライブラリーを調べてください。メンバー名は DFSSAM01 です。

DOCMD: IMS コマンドのフロントエンド

DOCMD は自動化操作プログラム・インターフェース (AOI) のトランザクション・プログラムで、IMS コマンドを出して出力を動的にフィルターに掛けることができます。「動的」という用語は、フィルター式でコマンド用ヘッダーをセレクター (変数名) として使用するという意味です (ブール式の結果は、行が表示されれば 1 で、そうでなければ 0 です)。

このリストは、このトピックの最後にあるコード例に表示しています。

すべてのコマンドでトランザクション AOI を使用できるわけではなく、この AOI を使用するためには何らかの設定が必要です。

DOCMD のいくつかの例を、以下の図に示します。

```
Please supply an IMS Command to execute.
EXEC DOCMD ended with RC= 0
```

図 *16. = > DOCMD* による出力

```
Headers being shown for command: /DIS NODE ALL
Variable (header) #1 = RECTYPE
Variable (header) #2 = NODE_SUB
Variable (header) #3 = TYPE
Variable (header) #4 = CID
Variable (header) #5 = RECD
Variable (header) #6 = ENQCT
Variable (header) #7 = DEQCT
Variable (header) #8 = QCT
Variable (header) #9 = SENT
EXEC DOCMD ended with RC= 0
```

図 *17. = > DOCMD /DIS NODE ALL;?* による出力

```
Selection criteria =>CID>0<= Command: /DIS NODE ALL
NODE_SUB TYPE   CID       RECD ENQCT DEQCT   QCT  SENT
L3270A   3277   01000004     5    19    19     0    26 IDLE CON
L3270C   3277   01000005   116   115   115     0   122 CON
Selected 2 lines from 396 lines.
DOCMD Executed 402 DL/I calls in 2.096787 seconds.
EXEC DOCMD ended with RC= 0
```

図 *18. = > DOCMD /DIS NODE ALL;CID>0* による出力

```
Selection criteria =>TYPE=SLU2<= Command: /DIS NODE ALL
NODE_SUB TYPE   CID       RECD ENQCT DEQCT   QCT  SENT
WRIGHT   SLU2   00000000     0     0     0     0     0 IDLE
Q3290A   SLU2   00000000     0     0     0     0     0 IDLE
Q3290B   SLU2   00000000     0     0     0     0     0 IDLE
Q3290C   SLU2   00000000     0     0     0     0     0 IDLE
Q3290D   SLU2   00000000     0     0     0     0     0 IDLE
V3290A   SLU2   00000000     0     0     0     0     0 IDLE
V3290B   SLU2   00000000     0     0     0     0     0 IDLE
H3290A   SLU2   00000000     0     0     0     0     0 IDLE
H3290B   SLU2   00000000     0     0     0     0     0 IDLE
E32701   SLU2   00000000     0     0     0     0     0 IDLE
E32702   SLU2   00000000     0     0     0     0     0 IDLE
E32703   SLU2   00000000     0     0     0     0     0 IDLE
E32704   SLU2   00000000     0     0     0     0     0 IDLE
E32705   SLU2   00000000     0     0     0     0     0 IDLE
ADLU2A   SLU2   00000000     0     0     0     0     0 IDLE
ADLU2B   SLU2   00000000     0     0     0     0     0 IDLE
ADLU2C   SLU2   00000000     0     0     0     0     0 IDLE
ADLU2D   SLU2   00000000     0     0     0     0     0 IDLE
ADLU2E   SLU2   00000000     0     0     0     0     0 IDLE
ADLU2F   SLU2   00000000     0     0     0     0     0 IDLE
ADLU2X   SLU2   00000000     0     0     0     0     0 IDLE
ENDS01   SLU2   00000000     0     0     0     0     0 IDLE
ENDS02   SLU2   00000000     0     0     0     0     0 IDLE
ENDS03   SLU2   00000000     0     0     0     0     0 IDLE
ENDS04   SLU2   00000000     0     0     0     0     0 IDLE
ENDS05   SLU2   00000000     0     0     0     0     0 IDLE
ENDS06   SLU2   00000000     0     0     0     0     0 IDLE
NDSLU2A1 SLU2   00000000     0     0     0     0     0 ASR IDLE
NDSLU2A2 SLU2   00000000     0     0     0     0     0 ASR IDLE
NDSLU2A3 SLU2   00000000     0     0     0     0     0 ASR IDLE
NDSLU2A4 SLU2   00000000     0     0     0     0     0 ASR IDLE
NDSLU2A5 SLU2   00000000     0     0     0     0     0 IDLE
NDSLU2A6 SLU2   00000000     0     0     0     0     0 ASR IDLE
OMSSLU2A SLU2   00000000     0     0     0     0     0 IDLE
Selected 34 lines from 396 lines.
DOCMD Executed 435 DL/I calls in 1.602206 seconds.
EXEC DOCMD ended with RC= 0
```

図 *19. = > DOCMD /DIS NODE ALL;TYPE=SLU2* による出力

```
Selection criteria =>ENQCT>0 & RECTYPE='T02'<= Command: /DIS TRAN ALL
  TRAN   CLS ENQCT   QCT   LCT  PLCT CP NP LP SEGSZ SEGNO PARLM RC
TACP18     1   119     0 65535 65535  1  1  1     0     0  NONE  1
Selected 1 lines from 1104 lines.
DOCMD Executed 1152 DL/I calls in 5.780977 seconds.
EXEC DOCMD ended with RC= 0
```

図 *20. = > DOCMD /DIS TRAN ALL;ENQCT>0 & RECTYPE='T02'* による出力

```
Selection criteria =>ENQCT>0<= Command: /DIS LTERM ALL
LTERM     ENQCT   DEQCT    QCT
CTRL         19      19      0
T3270LC     119     119      0
Selected 2 lines from 678 lines.
DOCMD Executed 681 DL/I calls in 1.967670 seconds.
EXEC DOCMD ended with RC= 0
```

図 *21. = > DOCMD /DIS LTERM ALL;ENQCT>0* による出力

以下のコード例は、DOCMD EXEC のソース・コードを示したものです。

DOCMD EXEC : IMS コマンドの処理

```
/*********************************************************************/
/* A REXX EXEC that executes an IMS command and parses the           */
/* output by a user supplied criteria.                               */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
/* Format:  tranname DOCMD IMS-Command;Expression                    */
/* Where:                                                            */
/*    tranname is the tranname of a command capable transaction that */
/*             will run the DFSREXX program.                         */
/*    IMS-Command is any valid IMS command that generates a table of */
/*             output like /DIS NODE ALL or /DIS TRAN ALL            */
/*    Expression is any valid REXX expression, using the header names*/
/*             as the variables, like  CID>0 or SEND=0 or more       */
/*             complex like CID>0 & TYPE=SLU2                        */
/* Example: TACP18 DOCMD DIS A               Display active          */
/*          TACP18 DOCMD DIS NODE ALL;?      See headers of DIS NODE */
/*          TACP18 DOCMD DIS NODE ALL;CID>0  Show active Nodes       */
/*          TACP18 DOCMD DIS NODE ALL;CID>0;SYSOUT  Output to SYSOUT */
/*          TACP18 DOCMD DIS NODE ALL;CID>0 & TYPE='SLU2'            */
/*********************************************************************/
Address REXXTDLI
Parse Upper Arg Cmd ';' Expression ';' SysOut
Cmd=Strip(Cmd);
Expression=Strip(Expression)
SysOut=(SysOut¬='')
If Cmd='' Then Do
   Call SayIt 'Please supply an IMS Command to execute.'
   Exit 0
End
AllOpt= (Expression='ALL')
If AllOpt then Expression=''
If Left(Cmd,1)¬='/' then Cmd='/'Cmd   /* Add a slash if necessary */
If Expression='' Then
   Call SayIt 'No Expression supplied, all output shown',
        'from:' Cmd
Else If Expression='?' Then
   Call SayIt 'Headers being shown for command:' Cmd
Else
   Call SayIt 'Selection criteria =>'Expression'<=',
        'Command:' Cmd

x=Time('R'); Calls=0
ExitRC= ParseHeader(Cmd,Expression)
If ExitRC¬=0 then Exit ExitRC
If Expression='?' Then Do
   Do i=1 to Vars.0
      Call SayIt 'Variable (header) #'i '=' Vars.i
      Calls=Calls+1
   End
End
Else Do
```

```
   Call ParseCmd Expression
   Do i=1 to Line.0
      If AllOpt then Line=Line.i
      Else Line=Substr(Line.i,5)
      If SysOut then Do
         Push Line
         'EXECIO 1 DISKW DOCMD'
      End
      Else Do
         Call SayIt Line
         Calls=Calls+1
      End
   End
   If SysOut then Do
      'EXECIO 0 DISKW DOCMD ( FINIS'
   End
   If Expression¬='' then
      Call SayIt 'Selected' Line.0-1 'lines from',
           LinesAvail 'lines.'
   Else
      Call SayIt 'Total lines of output:' Line.0-1
   Call SayIt 'DOCMD Executed' Calls 'DL/I calls in',
        Time('E') 'seconds.'
End
Exit 0

ParseHeader:
   CurrCmd=Arg(1)
   CmdCnt=0
   'CMD IOPCB CURRCMD'
   CmdS= IMSQUERY('STATUS')
   Calls=Calls+1
   If CmdS='  ' then Do
      Call SayIt 'Command Executed, No output available.'
      Return 4
   End
   Else If CmdS¬='CC' then Do
      Call SayIt 'Error Executing Command, Status='CmdS
      Return 16
   End
   CurrCmd=Translate(CurrCmd,' ','15'x)  /* Drop special characters  */
   CurrCmd=Translate(CurrCmd,'__','-/')  /* Drop special characters  */
   CmdCnt=CmdCnt+1
   Interpret 'LINE.'||CmdCnt '= Strip(CurrCmd)'
   Parse Var CurrCmd  RecType Header
   If Expression='' then Nop
   Else If Right(RecType,2)='70' then Do
      Vars.0=Words(Header)+1
      Vars.1 = "RECTYPE"
      Do i= 2 to Vars.0
         Interpret 'VARS.'i '= "'Word(CurrCmd,i)'"'
      End
   End
   Else Do
      Call SayIt 'Command did not produce a header',
                 'record, first record''s type='RecType
      Return 12
   End
Return 0

ParseCmd:
   LinesAvail=0
   CurrExp=Arg(1)
   Do Forever
      'GCMD IOPCB CURRCMD'
      CmdS= IMSQUERY('STATUS')
      Calls=Calls+1
```

```
      If CmdS¬='  ' then Leave
      /* Skip Time Stamps       */
      If Word(CurrCmd,1)='X99' & Expression¬='' then Iterate
      LinesAvail=LinesAvail+1
      CurrCmd=Translate(CurrCmd,' ','15'x)/* Drop special characters */
      If Expression='' then OK=1
      Else Do
         Do i= 1 to Vars.0
            Interpret Vars.i '= "'Word(CurrCmd,i)'"'
         End
         Interpret 'OK='Expression
      End
      If OK then Do
         CmdCnt=CmdCnt+1
         Interpret 'LINE.'||CmdCnt '= Strip(CurrCmd)'
      End
   End
   Line.0 = CmdCnt
   If CmdS¬='QD' Then
      Call SayIt 'Error Executing Command:',
                  Arg(1) 'Stat='CmdS
Return


SayIt: Procedure
   Parse Arg Line
   'ISRT IOPCB LINE'
Return RC
```

関連概念:

IMS セキュリティー (システム管理)

IVPREXX サンプル・アプリケーション

IVPREXX EXEC は、IVP の一部として IMS で提供されるフロントエンドの汎用 EXEC です。 IVPREXX は、TRANCODE (IVPREXX) の後に実行する EXEC の名前を渡すことによって他の EXEC を実行します。最新バージョンの IVPREXX ソース・コードについては、IMS.ADFSEXEC 配布ライブラリーで IVPREXX メンバーを探してください。

メッセージ・ドリブン BMP または IFP 環境で IVPREXX ドライバーのサンプル・プログラムを使用するには、IMS 領域プログラムのパラメーター・リストにプログラム名および PSB 名として IVPREXX を指定します。IVPREXX を指定すると、DFSREXX0 フロントエンド・プログラムのコピーである、IVPREXX ロード・モジュールがロードされます。IVPREXX プログラムがロードされて、IVPREXX と呼ばれる EXEC が実行されますが、これは、呼び出す EXEC または実行する関数を引き出すための引数としてトランザクションに送信される、メッセージ・セグメントを使用します。

次の例は、IMS 端末装置からの IVPREXX との対話を示しています。

IVPREXX 例 1

入力 :

```
IVPREXX execname
```

または

```
IVPREXX execname arguments
```

応答 :

```
EXEC execname ended with RC= x
```

IVPREXX 例 2

入力 :

```
IVPREXX LEAVE
```

応答 :

```
Transaction IVPREXX leaving dependent region.
```

IVPREXX 例 3

入力 :

```
IVPREXX HELLOHELLO
```

応答 :

```
One-to-eight character EXEC name must be specified.
```

IVPREXX 例 4

入力 :

```
IVPREXX
```

または

```
IVPREXX ?
```

応答 :

```
TRANCODE EXECNAME <Arguments> Run specified EXEC
TRANCODE LEAVE                Leave Dependent Region
TRANCODE TRACE level          0=None,1=Some,2=More,3=Full
TRANCODE ROLL                 Issue ROLL call
```

EXEC 名が指定されている場合は、入出力 PCB に挿入するすべてのセグメントが返されてから、完了メッセージが返されます。

通常、20000 から 20999 の範囲の REXX 戻りコード (RC) は、構文エラーまたはその他の REXX エラーです。詳しくは、z/OS システム・コンソールまたは領域出力を調べてください。

関連資料: REXX エラーおよびメッセージについて詳しくは、「*z/OS TSO/E REXX* 解説書」を参照してください。

無限ループの停止

無限ループを起こしている EXEC を停止するには、マスター端末またはシステム・コンソールから、次の IMS コマンドのいずれかを入力します。

```
/STO REGION p1 ABDUMP p2
/STO REGION p1 CANCEL
```

この例では、*p1* が領域番号で、*p2* が EXEC が実行中の TRANCODE です。/DISPLAY ACTIVE コマンドを使用して、領域番号を検索します。この手法

は、REXX EXEC に固有のものではなく、無限ループ状態にあるすべてのトランザクションに使用できます。

関連概念:

409 ページの『第 3 章 IMS アダプター (REXX 版) 参照』

第 4 章 Java プログラミング参照情報

以下のトピックでは、IMS solutions for Java developmentがサポートするクラス、インターフェース、およびメソッドの参照情報について説明します。

IMS Universal ドライバーの JDBC のサポート

IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーは、JDBC 4.0 仕様の以下のメソッドをサポートします。

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

関連資料:

466 ページの『Java API 資料 (Javadoc)』

サポートされる javax.sql.Clob のメソッド

javax.sql.Clob インターフェースは、SQL CLOB 型の Java でのマッピングを表します。IMS Universal ドライバー を使用する Java アプリケーションでは、Clob データ型は XML データの検索と保管にのみサポートされます。

次の表は、IMS Universal JDBC ドライバー および IMS Universal JCA/JDBC ドライバー によってサポートされる Clob インターフェース用のメソッドをリストしています。

表 *91. IMS Universal JDBC* ドライバー および *IMS Universal JCA/JDBC* ドライバー による *Clob* インターフェースのサポート

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
free()	なし
getAsciiStream()	あり
getCharacterStream()	あり
getCharacterStream(long pos, long length)	なし
getSubString(long pos, int length)	あり
length()	あり
position(Clob searchstr, long start)	なし
position(String searchstr, long start)	なし
setAsciiStream(long pos)	なし
setCharacterStream(long pos)	なし
setString(long pos, String str)	なし
setString(long pos, String str, int offset, int len)	なし

表 91. *IMS Universal JDBC* ドライバー および *IMS Universal JCA/JDBC* ドライバー による *Clob* インターフェースのサポート (続き)

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
truncate(long len)	なし

サポートされる java.sql.Connection のメソッド

Connection オブジェクトは、特定のデータベースとの接続 (セッション) を表します。

以下の表は、Connection インターフェース用の IMS JDBC ドライバーによりサポートされるメソッドをリストしています。

表 92. *IMS JDBC* ドライバーの *Connection* のサポート：

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
clearWarnings()	あり
close()	あり
commit()	あり
createStatement()	あり
createStatement(int resultSetType, int resultSetConcurrency)	あり
createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)	あり
getAutoCommit()	あり
getCatalog()	あり
getHoldability()	あり
getMetaData()	あり
getTransactionIsolation()	あり
getTypeMap()	あり
getWarnings()	あり
isClosed()	あり
isReadOnly()	あり
nativeSQL(String sql)	あり
prepareCall(String sql)	なし
prepareCall(String sql, int resultSetType, int resultSetConcurrency)	なし
prepareCall(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	なし
prepareStatement(String sql)	あり
prepareStatement(String sql, int autoGeneratedKeys)	なし
prepareStatement(String sql, int[] columnIndexes)	なし
prepareStatement(String sql, int resultSetType, int resultSetConcurrency)	あり
prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	あり
prepareStatement(String sql, String[] columnNames)	なし
releaseSavepoint(Savepoint savepoint)	なし
rollback()	あり
rollback(Savepoint savepoint)	なし
setAutoCommit(boolean autoCommit)	あり
setCatalog(String catalog)	あり
setHoldability(int holdability)	あり
setReadOnly(boolean readOnly)	あり

表 92. *IMS JDBC* ドライバーの *Connection* のサポート *(*続き*)*:

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
setSavepoint()	なし
setTransactionIsolation(int level)	あり
setTypeMap(Map<String,Class<?>> map)	なし

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

サポートされる java.sql.DatabaseMetaData のメソッド

DatabaseMetaData インターフェースは、データベース全体について総合的な情報を提供します。

DatabaseMetaData インターフェース用の IMS JDBC ドライバーでは、以下のメソッドがサポートされます。

表 93. *IMS JDBC* ドライバーの *DatabaseMetaData* のサポート:

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
allProceduresAreCallable()	あり
allTablesAreSelectable()	あり
dataDefinitionCausesTransactionCommit()	あり
dataDefinitionIgnoredInTransactions()	あり
deletesAreDetected(int type)	あり
doesMaxRowSizeIncludeBlobs()	あり
getAttributes(String catalog, String schemaPattern, String typeNamePattern, String attributeNamePattern)	あり
getBestRowIdentifier(String catalog, String schema, String table, int scope, boolean nullable)	あり
getCatalogs()	あり 2 番目の列 TIMESTAMP が、PSB タイム・スタンプを表すストリングとして、返された Resultset オブジェクトに追加されます。この TIMESTAMP 列には、IMS 14 APAR PI62871 (PTF UI40491) が必要です。
getCatalogSeparator()	あり
getCatalogTerm()	あり
getColumnPrivileges(String catalog, String schema, String table, String columnNamePattern)	あり
getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)	あり
getConnection()	あり
getCrossReference(String primaryCatalog, String primarySchema, String primaryTable, String foreignCatalog, String foreignSchema, String foreignTable)	あり
getDatabaseMajorVersion()	あり
getDatabaseMinorVersion()	あり
getDatabaseProductName()	あり
getDatabaseProductVersion()	あり
getDefaultTransactionIsolation()	あり
getDriverMajorVersion()	あり

表 93. *IMS JDBC* ドライバーの *DatabaseMetaData* のサポート (続き):

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
getDriverMinorVersion()	あり
getDriverName()	あり
getDriverVersion()	あり
getExportedKeys(String catalog, String schema, String table)	あり
getExtraNameCharacters()	あり
getIdentifierQuoteString()	あり
getImportedKeys(String catalog, String schema, String table)	あり
getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate)	あり
getJDBCMajorVersion()	あり
getJDBCMinorVersion()	あり
getMaxBinaryLiteralLength()	あり
getMaxCatalogNameLength()	あり
getMaxCharLiteralLength()	あり
getMaxColumnNameLength()	あり
getMaxColumnsInGroupBy()	あり
getMaxColumnsInIndex()	あり
getMaxColumnsInOrderBy()	あり
getMaxColumnsInSelect()	あり
getMaxColumnsInTable()	あり
getMaxConnections()	あり
getMaxCursorNameLength()	あり
getMaxIndexLength()	あり
getMaxProcedureNameLength()	あり
getMaxRowSize()	あり
getMaxSchemaNameLength()	あり
getMaxStatementLength()	あり
getMaxStatements()	あり
getMaxTableNameLength()	あり
getMaxTablesInSelect()	あり
getMaxUserNameLength()	あり
getNumericFunctions()	あり
getPrimaryKeys(String catalog, String schema, String table)	あり
getProcedureColumns(String catalog, String schemaPattern, String procedureNamePattern, String columnNamePattern)	あり
getProcedures(String catalog, String schemaPattern, String procedureNamePattern)	あり
getProcedureTerm()	あり
getResultSetHoldability()	あり
getSchemas()	あり 以下の列が、列 3、4、および 5 として追加されます。 • 列 3: PCB_PROCESSING_OPTIONS、PCB procopt を表すストリング • 列 4: DBD_NAME、参照される DBD 名を表すストリング • 列 5: DBD_TIMESTAMP、参照される DBD タイム・スタンプを表すストリング 列 3、4、および 5 には、IMS 14 APAR PI62871 (PTF UI40491) が必要です。
getSchemaTerm()	あり
getSearchStringEscape()	あり

表 93. *IMS JDBC* ドライバーの *DatabaseMetaData* のサポート (続き):

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
getSQLKeywords()	あり
getStringFunctions()	あり
getSuperTables(String catalog, String schemaPattern, String tableNamePattern)	あり
getSuperTypes(String catalog, String schemaPattern, String typeNamePattern)	あり
getSystemFunctions()	あり
getTablePrivileges(String catalog, String schemaPattern, String tableNamePattern)	あり
getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)	あり
getTableTypes()	あり
getTimeDateFunctions()	あり
getTypeInfo()	あり
getUDTs(String catalog, String schemaPattern, String typeNamePattern, int[] types)	あり
getURL()	あり
getUserName()	あり
getVersionColumns(String catalog, String schema, String table)	あり
insertsAreDetected(int type)	あり
isCatalogAtStart()	あり
isReadOnly()	あり
locatorsUpdateCopy()	あり
nullPlusNonNullIsNull()	あり
nullsAreSortedAtEnd()	あり
nullsAreSortedAtStart()	あり
nullsAreSortedLow()	あり
othersDeletesAreVisible(int type)	あり
othersInsertsAreVisible(int type)	あり
othersUpdatesAreVisible(int type)	あり
ownDeletesAreVisible(int type)	あり
ownInsertsAreVisible(int type)	あり
ownUpdatesAreVisible(int type)	あり
storesLowerCaseIdentifiers()	あり
storesLowerCaseQuotedIdentifiers()	あり
storesMixedCaseIdentifiers()	あり
storesMixedCaseQuotedIdentifiers()	あり
storesUpperCaseIdentifiers()	あり
storesUpperCaseQuotedIdentifiers()	あり
supportsAlterTableWithAddColumn()	あり
supportsAlterTableWithDropColumn()	あり
supportsANSI92EntryLevelSQL()	あり
supportsANSI92FullSQL()	あり
supportsANSI92IntermediateSQL()	あり
supportsBatchUpdates()	あり
supportsCatalogsInDataManipulation()	あり
supportsCatalogsInIndexDefinitions()	あり
supportsCatalogsInPrivilegeDefinitions()	あり
supportsCatalogsInProcedureCalls()	あり
supportsCatalogsInTableDefinitions()	あり
supportsColumnAliasing()	あり

表 93. *IMS JDBC* ドライバーの *DatabaseMetaData* のサポート (続き):

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
supportsConvert()	あり
supportsConvert(int fromType, int toType)	あり
supportsCoreSQLGrammar()	あり
supportsCorrelatedSubqueries()	あり
supportsDataDefinitionAnd DataManipulationTransactions()	あり
supportsDataManipulationTransactionsOnly()	あり
supportsDifferentTableCorrelationNames()	あり
supportsExpressionsInOrderBy()	あり
supportsExtendedSQLGrammar()	あり
supportsFullOuterJoins()	あり
supportsGetGeneratedKeys()	あり
supportsGroupByBeyondSelect()	あり
supportsGroupByUnrelated()	あり
supportsIntegrityEnhancementFacility()	あり
supportsLikeEscapeClause()	あり
supportsLimitedOuterJoins()	あり
supportsMinimumSQLGrammar()	あり
supportsMixedCaseIdentifiers()	あり
supportsMixedCaseQuotedIdentifiers()	あり
supportsMultipleOpenResults()	あり
supportsMultipleResultSets()	あり
supportsMultipleTransactions()	あり
supportsNamedParameters()	あり
supportsNonNullableColumns()	あり
supportsOpenCursorsAcrossCommit()	あり
supportsOpenCursorsAcrossRollback()	あり
supportsOpenStatementsAcrossCommit()	あり
supportsOpenStatementsAcrossRollback()	あり
supportsOrderByUnrelated()	あり
supportsOuterJoins()	あり
supportsPositionedDelete()	あり
supportsPositionedUpdate()	あり
supportsResultSetConcurrency(int type, int concurrency)	あり
supportsResultSetHoldability(int holdability)	あり
supportsResultSetType(int type)	あり
supportsSavepoints()	あり
supportsSchemasInDataManipulation()	あり
supportsSchemasInIndexDefinitions()	あり
supportsSchemasInPrivilegeDefinitions()	あり
supportsSchemasInProcedureCalls()	あり
supportsSchemasInTableDefinitions()	あり
supportsSelectForUpdate()	あり
supportsStatementPooling()	あり
supportsStoredProcedures()	あり
supportsSubqueriesInComparisons()	あり
supportsSubqueriesInExists()	あり
supportsSubqueriesInIns()	あり
supportsSubqueriesInQuantifieds()	あり
supportsTableCorrelationNames()	あり

表 93. *IMS JDBC* ドライバーの *DatabaseMetaData* のサポート (続き):

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
supportsTransactionIsolationLevel(int level)	あり
supportsTransactions()	あり
supportsUnion()	あり
supportsUnionAll()	あり
updatesAreDetected(int type)	あり
usesLocalFilePerTable()	あり
usesLocalFiles()	あり

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.sql.DataSource のメソッド

DataSource オブジェクトは、その DataSource オブジェクトが表す物理データ・ソースに接続するためのファクトリーです。

以下の表は、IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーによりサポートされる DataSource インターフェースのメソッドをリストしています。

表 94. *IMS Universal JDBC* ドライバーおよび *IMS Universal JCA/JDBC* ドライバーの *DataSource* のサポート:

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
getConnection()	あり
getConnection(String username, String password)	あり
getLoginTimeout()	あり
getLogWriter()	あり
setLoginTimeout(int seconds)	あり
setLogWriter(PrintWriter out)	あり

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

サポートされる java.sql.Driver のメソッド

Driver クラスは、JDBC DriverManager インターフェースを使ってデータベースに接続するために使用します。

Driver インターフェース用の IMS JDBC ドライバーでは、以下のメソッドがサポートされます。

表 95. *IMS JDBC* ドライバーの *Driver* のサポート:

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
acceptsURL(String url)	あり

表 95. IMS JDBC ドライバーの Driver のサポート (続き):

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
connect(String url, Properties info)	あり
getMajorVersion()	あり
getMinorVersion()	あり
getPropertyInfo(String url, Properties info)	あり
jdbcCompliant()	あり

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

サポートされる java.sql.ParameterMetaData のメソッド

PreparedStatement オブジェクトにあるパラメーターのタイプおよびプロパティーについて情報を取得するために使用できるオブジェクトです。

以下の表は、IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーによりサポートされる ParameterMetaData インターフェースのメソッドをリストしています。

表 96. IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーの ParameterMetaData のサポート:

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
getParameterCount	あり
isNullable	あり
isSigned	あり
getPrecision	あり
getScale	あり
getParameterType	あり
getParameterTypeName	あり
getParameterClassName	あり
getParameterMode	あり

サポートされる java.sql.PreparedStatement のメソッド

PreparedStatement オブジェクトは、プリコンパイルされた SQL ステートメントを表します。

PreparedStatement インターフェース用の IMS JDBC ドライバーでは、以下のメソッドがサポートされます。

表 97. IMS JDBC ドライバーの PreparedStatement のサポート:

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
addBatch()	なし
clearParameters()	あり
execute()	あり
executeQuery()	あり

表 97. *IMS JDBC* ドライバーの *PreparedStatement* のサポート (続き) :

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
executeUpdate()	あり
getMetaData()	あり
getParameterMetaData()	あり
setArray(int i, Array x)	あり
setAsciiStream(int parameterIndex, InputStream x, int length)	なし
setBigDecimal(int parameterIndex, BigDecimal x)	あり
setBinaryStream(int parameterIndex, InputStream x, int length)	なし
setBlob(int i, Blob x)	なし
setBoolean(int parameterIndex, boolean x)	あり
setByte(int parameterIndex, byte x)	あり
setBytes(int parameterIndex, byte[] x)	あり
setCharacterStream(int parameterIndex, Reader reader, int length)	なし
setClob(int i, Clob x)	あり
setDate(int parameterIndex, Date x)	あり
setDate(int parameterIndex, Date x, Calendar cal)	なし
setDouble(int parameterIndex, double x)	あり
setFloat(int parameterIndex, float x)	あり
setInt(int parameterIndex, int x)	あり
setLong(int parameterIndex, long x)	あり
setNull(int parameterIndex, int sqlType)	なし
setNull(int paramIndex, int sqlType, String typeName)	なし
setObject(int parameterIndex, Object x)	あり
setObject(int parameterIndex, Object x, int targetSqlType)	なし
setObject(int parameterIndex, Object x, int targetSqlType, int scale)	なし
setRef(int i, Ref x)	なし
setShort(int parameterIndex, short x)	あり
setString(int parameterIndex, String x)	あり
setTime(int parameterIndex, Time x)	あり
setTime(int parameterIndex, Time x, Calendar cal)	なし
setTimestamp(int parameterIndex, Timestamp x)	あり
setUnicodeStream(int parameterIndex, InputStream x, int length)	なし
setURL(int parameterIndex, URL x)	なし

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

サポートされる java.sql.Statement のメソッド

Statement オブジェクトは、静的 SQL ステートメントを実行し、ステートメントが作成した結果を返すために使用します。

以下の表は、IMS JDBC ドライバーによりサポートされる Statement インターフェースのメソッドをリストしています。

表 *98. IMS JDBC* ドライバーによる *Statement* のサポート：

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
addBatch(String sql)	なし
cancel()	なし
clearBatch()	なし
clearWarnings()	あり
close()	あり
execute(String sql)	あり
execute(String sql, int autoGeneratedKeys)	なし
execute(String sql, int[] columnIndexes)	なし
execute(String sql, String[] columnNames)	なし
executeBatch()	なし
executeQuery(String sql)	あり
executeUpdate(String sql)	あり
executeUpdate(String sql, int autoGeneratedKeys)	なし
executeUpdate(String sql, int[] columnIndexes)	なし
executeUpdate(String sql, String[] columnNames)	なし
getConnection()	あり
getFetchDirection()	あり
getFetchSize()	あり
getGeneratedKeys()	なし
getMaxFieldSize()	あり
getMaxRows()	あり
getMoreResults()	あり
getMoreResults(int current)	あり
getQueryTimeout()	あり
getResultSet()	あり
getResultSetConcurrency()	あり
getResultSetHoldability()	あり
getResultSetType()	あり
getUpdateCount()	あり
getWarnings()	あり
setCursorName(String name)	なし
setEscapeProcessing(boolean enable)	あり
setFetchDirection(int direction)	あり
setFetchSize(int rows)	あり
setMaxFieldSize(int max)	あり
setMaxRows(int max)	あり
setQueryTimeout(int seconds)	あり

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

サポートされる java.sql.ResultSet のメソッド

ResultSet オブジェクトは、データベースの結果セットを表すデータの表であり、通常はデータベースを照会するステートメントを実行して生成されます。

次の表は、IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバー によりサポートされる ResultSet フィールド定数を示します。

表 *99. IMS Universal JDBC* ドライバーおよび *IMS Universal JCA/JDBC* ドライバーによりサポートされる *ResultSet* フィールド定数

フィールド定数	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
ResultSet.CLOSE_CURSORS_AT_COMMIT	可[1]
ResultSet.CONCUR_READ_ONLY	あり
ResultSet.CONCUR_UPDATABLE	あり
ResultSet.FETCH_FORWARD	可[2]
ResultSet.FETCH_REVERSE	可[2]
ResultSet.FETCH_UNKNOWN	可[2]
ResultSet.HOLD_CURSORS_OVER_COMMIT	不可[3]
ResultSet.TYPE_FORWARD_ONLY	あり
ResultSet.TYPE_SCROLL_INSENSITIVE	あり
ResultSet.TYPE_SCROLL_SENSITIVE	不可[3]

注**:**

1. これは IMS DB で使用される処理モデルです。
2. これは JDBC ドライバーについてのヒントです。IMS DB により実行される特別な処理はありません。
3. IMS DB ではサポートされません。

ResultSet インターフェース用の IMS JDBC ドライバーでは、以下のメソッドがサポートされます。

表 *100. IMS JDBC* ドライバーによる *ResultSet* インターフェースのサポート

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
absolute(int row)	あり
afterLast()	あり
beforeFirst()	あり
cancelRowUpdates()	あり
clearWarnings()	あり
close()	あり
deleteRow()	あり
findColumn(String columnName)	あり
first()	あり
getArray(int i)	あり
getArray(String colName)	あり
getAsciiStream(int columnIndex)	なし
getAsciiStream(String columnName)	なし
getBigDecimal(int columnIndex)	あり

表 100. IMS JDBC ドライバーによる ResultSet インターフェースのサポート (続き)

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
getBigDecimal(int columnIndex, int scale)	あり
getBigDecimal(String columnName)	あり
getBigDecimal(String columnName, int scale)	あり
getBinaryStream(int columnIndex)	なし
getBinaryStream(String columnName)	なし
getBlob(int i)	なし
getBlob(String colName)	なし
getBoolean(int columnIndex)	あり
getBoolean(String columnName)	あり
getByte(int columnIndex)	あり
getByte(String columnName)	あり
getBytes(int columnIndex)	あり
getBytes(String columnName)	あり
getCharacterStream(int columnIndex)	なし
getCharacterStream(String columnName)	なし
getClob(int i)	可 (XML の検索のみ)
getClob(String colName)	可 (XML の検索のみ)
getConcurrency()	あり
getCursorName()	なし
getDate(int columnIndex)	あり
getDate(int columnIndex, Calendar cal)	あり
getDate(String columnName)	あり
getDate(String columnName, Calendar cal)	あり
getDouble(int columnIndex)	あり
getDouble(String columnName)	あり
getFetchDirection()	あり
getFetchSize()	あり
getFloat(int columnIndex)	あり
getFloat(String columnName)	あり
getInt(int columnIndex)	あり
getInt(String columnName)	あり
getLong(int columnIndex)	あり
getLong(String columnName)	あり
getMetaData()	あり
getObject(int columnIndex)	あり
getObject(String columnName)	あり
getObject(int i, Map<String,Class<?>> map)	なし
getRef(int i)	なし
getRef(String colName)	なし

表 *100. IMS JDBC* ドライバーによる *ResultSet* インターフェースのサポート (続き)

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
getRow()	あり
getShort(int columnIndex)	あり
getShort(String columnName)	あり
getStatement()	あり
getString(int columnIndex)	あり
getString(String columnName)	あり
getTime(int columnIndex)	あり
getTime(String columnName)	あり
getTime(String columnName, Calendar cal)	あり
getTime(int columnIndex, Calendar cal)	あり
getTimestamp(int columnIndex)	あり
getTimestamp(int columnIndex, Calendar cal)	あり
getTimestamp(String columnName)	あり
getTimestamp(String columnName, Calendar cal)	あり
getType()	あり
getUnicodeStream(int columnIndex)	なし
getUnicodeStream(String columnName)	なし
getURL(int columnIndex)	なし
getURL(String columnName)	なし
getWarnings()	あり
insertRow()	なし
isAfterLast()	あり
isBeforeFirst()	あり
isFirst()	あり
isLast()	あり
last()	あり
moveToCurrentRow()	なし
moveToInsertRow()	なし
next()	あり
previous()	あり
refreshRow()	なし
relative(int rows)	あり
rowDeleted()	なし
rowInserted()	なし
rowUpdated()	なし
setFetchDirection(int direction)	あり
setFetchSize(int rows)	あり
updateArray(int columnIndex, Array x)	あり
updateArray(String columnName, Array x)	あり

表 100. IMS JDBC ドライバーによる ResultSet インターフェースのサポート (続き)

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
updateAsciiStream(int columnIndex, InputStream x, int length)	なし
updateAsciiStream(String columnName, InputStream x, int length)	なし
updateBigDecimal(int columnIndex, BigDecimal x)	あり
updateBigDecimal(String columnName, BigDecimal x)	あり
updateBinaryStream(int columnIndex, InputStream x, int length)	なし
updateBinaryStream(String columnName, InputStream x, int length)	なし
updateBlob(int columnIndex, Blob x)	なし
updateBlob(String columnName, Blob x)	なし
updateBoolean(int columnIndex, boolean x)	あり
updateBoolean(String columnName, boolean x)	あり
updateByte(int columnIndex, byte x)	あり
updateByte(String columnName, byte x)	あり
updateBytes(int columnIndex, byte[] x)	あり
updateBytes(String columnName, byte[] x)	あり
updateCharacterStream(int columnIndex, Reader x, int length)	なし
updateCharacterStream(String columnName, Reader reader, int length)	なし
updateClob(int columnIndex, Clob x)	なし
updateClob(String columnName, Clob x)	なし
updateDate(int columnIndex, Date x)	あり
updateDate(String columnName, Date x)	あり
updateDouble(int columnIndex, double x)	あり
updateDouble(String columnName, double x)	あり
updateFloat(int columnIndex, float x)	あり
updateFloat(String columnName, float x)	あり
updateInt(int columnIndex, int x)	あり
updateInt(String columnName, int x)	あり
updateLong(int columnIndex, long x)	あり
updateLong(String columnName, long x)	あり
updateNull(String columnName)	なし
updateObject(int columnIndex, Object x)	あり
updateObject(int columnIndex, Object x, int scale)	なし
updateObject(String columnName, Object x)	あり
updateObject(String columnName, Object x, int scale)	なし
updateRef(int columnIndex, Ref x)	なし

表 *100. IMS JDBC* ドライバーによる *ResultSet* インターフェースのサポート (続き)

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
updateRef(String columnName, Ref x)	なし
updateRow()	あり
updateShort(int columnIndex, short x)	あり
updateShort(String columnName, short x)	あり
updateString(int columnIndex, String x)	あり
updateString(String columnName, String x)	あり
updateTime(int columnIndex, Time x)	あり
updateTime(String columnName, Time x)	あり
updateTimestamp(int columnIndex, Timestamp x)	あり
updateTimestamp(String columnName, Timestamp x)	あり
wasNull()	あり

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

サポートされる java.sql.ResultSetMetaData のメソッド

ResultSetMetaData オブジェクトは、ResultSet オブジェクトにあるカラムのタイプおよびプロパティーについて情報を取得するために使用できます。

ResultSetMetaData インターフェース用の IMS JDBC ドライバーでは、以下のメソッドがサポートされます。

表 *101. IMS JDBC* ドライバーによる *ResultSetMetaData* のサポート:

JDBC メソッド	**IMS Universal JDBC** ドライバーおよび **IMS Universal JCA/JDBC** ドライバーのサポート
getCatalogName(int column)	あり
getColumnClassName(int column)	あり
getColumnCount()	あり
getColumnDisplaySize(int column)	あり
getColumnLabel(int column)	あり
getColumnName(int column)	あり
getColumnType(int column)	あり
getColumnTypeName(int column)	あり
getPrecision(int column)	あり
getScale(int column)	あり
getSchemaName(int column)	あり
getTableName(int column)	あり
isAutoIncrement(int column)	あり
isCaseSensitive(int column)	あり
isCurrency(int column)	あり
isDefinitelyWritable(int column)	あり
isNullable(int column)	あり
isReadOnly(int column)	あり
isSearchable(int column)	あり

表 101. IMS JDBC ドライバーによる ResultSetMetaData のサポート (続き):

JDBC メソッド	IMS Universal JDBC ドライバーおよび IMS Universal JCA/JDBC ドライバーのサポート
isSigned(int column)	あり
isWritable(int column)	あり

関連概念:

IMS Universal JDBC ドライバーを使用したプログラミング (アプリケーション・プログラミング)

IMS Universal ドライバーの Common Client Interface のサポート

IMS Universal Database リソース・アダプターは、Java EE Connector Architecture 1.5 仕様の Common Client Interface (CCI) API をサポートします。

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.resource.cci.Connection のメソッド

javax.resource.cci.Connection インターフェースは、基礎となる物理接続にアクセスするためにクライアントが使用するアプリケーション・レベルのハンドルを表します。

以下の表は、IMS Universal Database リソース・アダプターによりサポートされる javax.resource.cci.Connection インターフェースのメソッドをリストしています。

表 102. IMS Universal Database リソース・アダプターの javax.resource.cci.Connection インターフェースのサポート

javax.resource.cci.Connection のメソッド	IMS Universal Database リソース・アダプターのサポート
close()	あり
createInteraction()	あり
getLocalTransaction()	あり
getMetaData()	あり
getResultSetInfo()	あり

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.resource.cci.ConnectionFactory のメソッド

javax.resource.cci.ConnectionFactory インターフェースは、アプリケーション・コンポーネントに EIS への Connection インスタンスを提供します。

以下の表は、IMS Universal Database リソース・アダプターによりサポートされる javax.resource.cci.ConnectionFactory インターフェースのメソッドをリストしています。

表 *103. IMS Universal Database* リソース・アダプターの *javax.resource.cci.ConnectionFactory* インターフェースのサポート

javax.resource.cci.ConnectionFactory のメソッド	**IMS Universal Database** リソース・アダプターのサポート
getConnection()	あり
getConnection(ConnectionSpec)	あり
getMetaData()	あり
getRecordFactory()	あり

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.resource.cci.ConnectionMetaData のメソッド

javax.resource.cci.ConnectionMetaData インターフェースは、Connection インスタンスを介して接続されている EIS インスタンスに関する情報を提供します。

以下の表は、IMS Universal Database リソース・アダプターによりサポートされる javax.resource.cci.ConnectionMetaData インターフェースのメソッドをリストしています。

表 *104. IMS Universal Database* リソース・アダプターの *javax.resource.cci.ConnectionMetaData* インターフェースのサポート

javax.resource.cci.ConnectionMetaData のメソッド	**IMS Universal Database** リソース・アダプターのサポート
getEISProductName()	あり
getEISProductVersion()	あり
getUserName()	あり

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.resource.cci.Interaction のメソッド

javax.resource.cci.Interaction インターフェースは、アプリケーション・コンポーネントがリレーショナル・データベース照会などの EIS 機能を実行するための手段を提供します。

以下の表は、IMS Universal Database リソース・アダプターによりサポートされる javax.resource.cci.Interaction インターフェースのメソッドをリストしています。

表 105. *IMS Universal Database* リソース・アダプターの *javax.resource.cci.Interaction* インターフェースのサポート

javax.resource.cci.Interaction のメソッド	**IMS Universal Database** リソース・アダプターのサポート
clearWarnings()	あり
close()	あり
execute(InteractionSpec, Record)	あり
execute(InteractionSpec, Record, Record)	なし
getConnection()	あり
getWarnings()	あり

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.resource.cci.LocalTransaction のメソッド

javax.resource.cci.LocalTransaction インターフェースは、リソース・マネージャーのローカル・トランザクション用のトランザクション区分インターフェースを定義します。

以下の表は、IMS Universal Database リソース・アダプターによりサポートされる javax.resource.cci.LocalTransaction インターフェースのメソッドをリストしています。

表 106. *IMS Universal Database* リソース・アダプターの *javax.resource.cci.LocalTransaction* インターフェースのサポート

javax.resource.cci.LocalTransaction のメソッド	**IMS Universal Database** リソース・アダプターのサポート
begin()	あり
commit()	あり
rollback()	あり

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.resource.cci.ResultSetInfo のメソッド

インターフェース javax.resource.cci.ResultSetInfo は、接続された EIS インスタンスが ResultSet に提供するサポートに関する情報を提供します。

以下の表は、IMS Universal Database リソース・アダプターによりサポートされる javax.resource.cci.ResultSetInfo インターフェースのメソッドをリストしています。

表 107. *IMS Universal Database* リソース・アダプターの *javax.resource.cci.ResultSetInfo* インターフェースのサポート

javax.resource.cci.ResultSetInfo のメソッド	**IMS Universal Database** リソース・アダプターのサポート
deletesAreDetected(int)	あり
insertsAreDetected(int)	あり
othersDeletesAreVisible(int)	あり
othersInsertsAreVisible(int)	あり
othersUpdatesAreVisible(int)	あり
ownDeletesAreVisible(int)	あり
ownInsertsAreVisible(int)	あり
ownUpdatesAreVisible(int)	あり
supportsResultSetType(int)	あり
supportsResultTypeConcurrency(int, int)	あり
updatesAreDetected(int)	あり

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.resource.cci.ResourceAdapterMetaData のメソッド

インターフェース javax.resource.cci.ResourceAdapterMetaData は、リソース・アダプター実装環境の機能についての情報を提供します。

以下の表は、IMS Universal Database リソース・アダプターによりサポートされる javax.resource.cci.ResourceAdapterMetaData インターフェースのメソッドをリストしています。

表 108. *IMS Universal Database* リソース・アダプターの *javax.resource.cci.ResourceAdapterMetaData* インターフェースのサポート

javax.resource.cci.ResourceAdapterMetaData のメソッド	**IMS Universal Database** リソース・アダプターのサポート
getAdapterName()	あり
getAdapterShortDescription()	あり
getAdapterVendorName()	あり
getAdapterVersion()	あり
getInteractionSpecsSupported()	あり
getSpecVersion()	あり
supportsExecuteWithInputAndOutputRecord()	あり
supportsExecuteWithInputRecordOnly()	あり
supportsLocalTransactionDemarcation()	あり

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

サポートされる javax.resource.cci.RecordFactory のメソッド

javax.resource.cci.RecordFactory インターフェースは、アプリケーション・コンポーネントに Record インスタンスを提供します。

以下の表は、IMS Universal Database リソース・アダプターによりサポートされる javax.resource.cci.RecordFactory インターフェースのメソッドをリストしています。

表 109. IMS Universal Database リソース・アダプターの javax.resource.cci.RecordFactory インターフェースのサポート

javax.resource.cci.RecordFactory のメソッド	**IMS Universal Database** リソース・アダプターのサポート
createIndexedRecord(String)	あり
createMappedRecord(String)	あり

関連概念:

IMS Universal Database リソース・アダプターを使用したプログラミング (アプリケーション・プログラミング)

Java API 資料 (Javadoc)

これらのトピックでは、IMS solutions for Java development 用の Java API 資料 (Javadoc) について説明します。

- IMS Universal ドライバー および IMS Java 依存領域リソース・アダプター の Javadoc
- IMS TM リソース・アダプター の Javadoc

IMS Universal ドライバーおよび IMS Java 依存領域リソース・アダプター

IMS Java 従属領域 (JDR) リソース・アダプターは、IMS Universal ドライバー のインターフェースまたはクラスの一部を再利用します。クラスおよびインターフェースは、1 つの .jar ファイル `imsudb.jar` として一緒にパッケージ化されています。

重要**:** IMS 14 APAR PI62871 (PTF UI40491) が必要です。

以下のパッケージは、IMS JBP、JMP、または IMS データベース・リソースと統合するための Java インターフェースまたはクラスを提供します。

表 110. *IMS Universal* ドライバー および *IMS Java* 依存領域リソース・アダプター の *IMS* データベース・アクセス用のパッケージ

パッケージ	説明	使用者
com.ibm.ims.application	IMS Java 従属領域トランザクションおよびメッセージ処理用のクラスを提供します。これには、エラー・メッセージの管理、メッセージの送受信、IMS トランザクション・サービスへのプログラム・アクセス (コミットやロールバックなど) の提供などを行うためのクラスが含まれます。	IMS Java 依存領域リソース・アダプター (imsudb.jar)
com.ibm.ims.base	Java 呼び出しを DL/I API にマップするためのクラスを提供します。	IMS Java 依存領域リソース・アダプター (imsudb.jar)
com.ibm.ims.db.cci	共通クライアント・インターフェース (CCI) アーキテクチャーを使用して IMS データベース・リソースと対話するためのクラスを提供します。	• ローカル・トランザクション・サポートがある IMS Universal Database リソース・アダプター (imsudbLocal.rar) • 追加の 2 フェーズ・コミット処理がある IMS Universal Database リソース・アダプター (imsudbXA.rar)
com.ibm.ims.dli	DL/I プログラミング・セマンティクスを使用して IMS データベースにアクセスできる Java アプリケーションを作成するための API を提供します。	imsudb.jar • IMS Universal DL/I ドライバー • IMS Universal JDBC ドライバー • IMS Java 依存領域リソース・アダプター
com.ibm.ims.dli.converters	Java データ・タイプをバイト配列との間で変換するための API を提供します。	imsudb.jar • IMS Universal DL/I ドライバー • IMS Universal JDBC ドライバー • IMS Java 依存領域リソース・アダプター
com.ibm.ims.dli.tm	IMS Java バッチ処理領域 (JBP) および Java メッセージ処理領域 (JMP) と対話するための Java インターフェースを提供します。	IMS Java 依存領域リソース・アダプター (imsudb.jar)

表 110. *IMS Universal* ドライバー および *IMS Java* 依存領域リソース・アダプター の *IMS* データベース・アクセス用のパッケージ (続き)

パッケージ	説明	使用者
com.ibm.ims.dli.types	カスタム・ユーザー・タイプ・コンバーターの作成を支援する、拡張可能な抽象タイプ・コンバーターを提供します。	imsudb.jar • IMS Universal DL/I ドライバー • IMS Universal JDBC ドライバー • IMS Java 依存領域リソース・アダプター
com.ibm.ims.jdbc	JDBC を使用して IMS データベースに接続するための、IMS 固有の拡張機能を提供します。	• IMS Universal JDBC ドライバー (imsudb.jar) • JCA サポートがある IMS Universal JDBC ドライバー (imsudbJLocal.rar または imsudbJXA.rar)
com.ibm.ims.jdbc.xa	JDBC を使用して 2 フェーズ・コミット(XA) モードで IMS データベースに接続するための、IMS 固有の拡張機能を提供します。	JCA サポートがある IMS Universal JDBC ドライバー (imsudbJXA.rar)
com.ibm.ims.jms	JMP 領域および JBP 領域から同期コールアウト要求を発行するための、IMS 固有の拡張機能が含まれます。	IMS Java 依存領域リソース・アダプター (imsudb.jar)

IMS TM リソース・アダプター の Javadoc

IMS TM リソース・アダプター Java API の資料は、IMS TM リソース・アダプター Java API リファレンスで別個に提供されています。

表 111. *IMS TM* リソース・アダプター の *Javadoc*

パッケージ	説明	使用者
com.ibm.connector2.ims.ico	IMS への接続および IMS トランザクションとの対話を管理するための拡張機能用のインターフェースおよびクラスが含まれます。	IMS TM リソース・アダプター (imsico*xxxx*.rar。ここで、*xxxx* は、1410 (V14.1.0 の場合) などのバージョン番号です)
com.ibm.connector2.ims.ico.inbound	IMS からのインバウンド通信のプロパティーを構成するためのクラスが含まれます。	IMS TM リソース・アダプター (imsico*xxxx*.rar。ここで、*xxxx* は、1410 (V14.1.0 の場合) などのバージョン番号です)

関連概念:

IMS solutions for Java developmentの概要 (アプリケーション・プログラミング)

Universal ドライバー、JDR リソース・アダプター、および JMS 用の Java API 仕様

第 5 章 メッセージ形式サービス (MFS) の参照情報

これらのトピックには、IMS メッセージ形式サービス (MFS) に関する参照情報が記載されています。

MFS アプリケーション・プログラムの設計

MFS アプリケーション・プログラムの設計目標は、装置からの独立性を確保すること、オペレーターにとって使いやすいこと、そしてアプリケーション・プログラムが単純であることを中心にして考える必要があります。効率のよい設計を行うためには、MFS の機能、そして MFS での操作やパフォーマンスに影響を与えるさまざまな要因についての基礎知識が必要です。

MFS 制御ブロック間の関係

MFS 制御ブロックの間には、複数のリンケージ・レベルが存在します。

アプリケーション環境全体を正しく設計するためには、このようなリンケージを理解しておく必要があります。

以下の図は、MFS 制御ブロック間の相互関係を示しています。その次は、4 つのリンケージ・レベルを図示したもので、それらを要約したものが最後の図です。

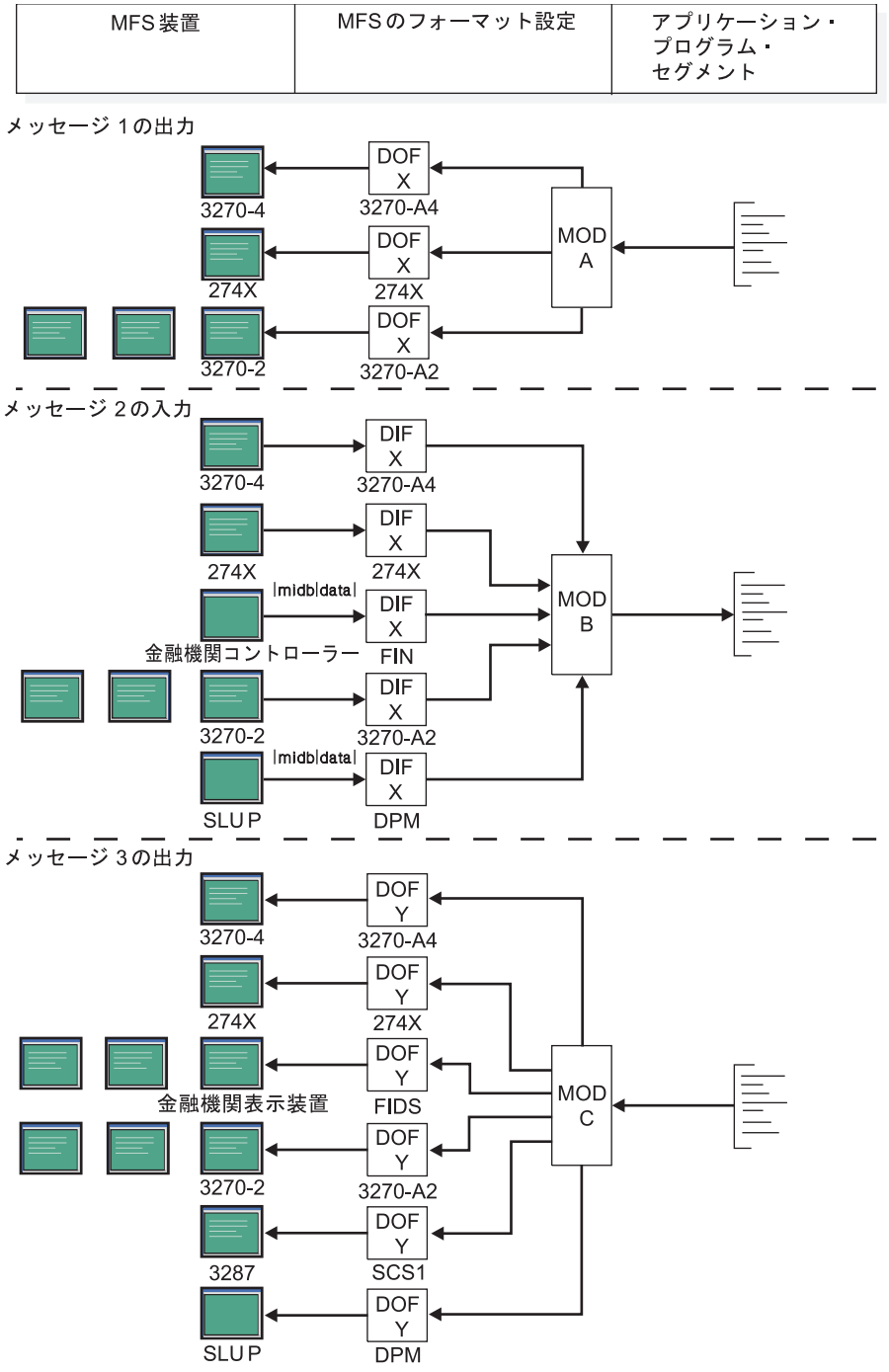


図 22. 制御ブロック間の相互関係

以下の図は、チェーン制御ブロックの最高レベルのリンケージを示しています。

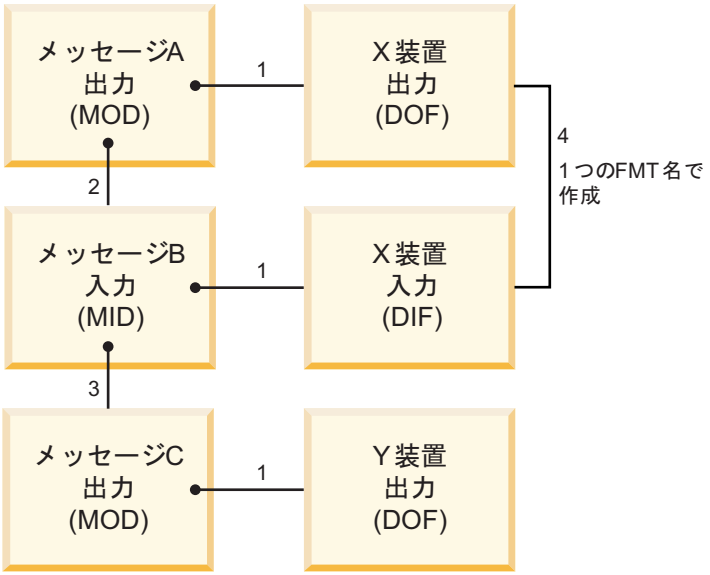


図 *23.* チェーン制御ブロックのリンケージ

上の図の注**:**

1. このリンケージは必ず存在していなければなりません。
2. このリンケージが存在していないと、MFS が 3270 装置からの装置入力データを処理しません。他の装置については、オペレーターが MID 名を入力することができます。
3. このリンケージによって、アプリケーション・プログラムをより使いやすくすることができます。アプリケーション・プログラムが DL/I の ISRT 呼び出しまたは PURG 呼び出しを行うときに、その形式名オプションで名前を指定しなくても、IMS がどの MOD 名を使用すればよいのかがわかります。この MOD 名は、入力が MFS のサポートする端末へのメッセージ通信である場合にも使用されます。
4. 1 つの出力/入力シーケンスで使用されるユーザー指定の DOF 名と DIF 名は、通常は同じです。ただし、MFS プール・マネージャーが DOF と DIF を区別できるようにするためには、MFS 言語ユーティリティーで DIF の名前を変更しなければなりません。

   リンケージの方向は決まっているので、必要であれば、複数のメッセージ記述子が同じ装置形式を使用することもできます。アプリケーション・プログラム・インターフェースの観点で見れば、出力メッセージ形式と入力メッセージ形式がまったく異なっていても、複数のアプリケーション・プログラムで 1 つの装置形式を共用することができます。

以下の図は、メッセージ・フィールドと装置フィールドとの間に存在する、別のレベルのリンケージを示しています。点は、MFS 言語ユーティリティー制御ステートメントにおける参照の方向を示すもので、データ・フローの方向を示すものではありません。つまり、1 本の線で結ばれている 2 つの項目のうち、点の付いている方の項目がもう片方の項目を参照します。

メッセージ・フィールドによる装置フィールドの参照は、特定の順序で行う必要はありません。MFLD は、DFLD のいずれも参照する必要はありません。この場合、

MFLD の定義は、MFLD が出力用の場合はアプリケーション・プログラム・セグメント内のスペースを無視し、MFLD が入力用の場合はスペースを埋め込むようにします。装置フィールドをメッセージ・フィールドで参照する必要はありません。この場合、このフィールドは装置上に設定されますが、このフィールドには出力データは送信されず、フィールドからの入力データは無視されます。

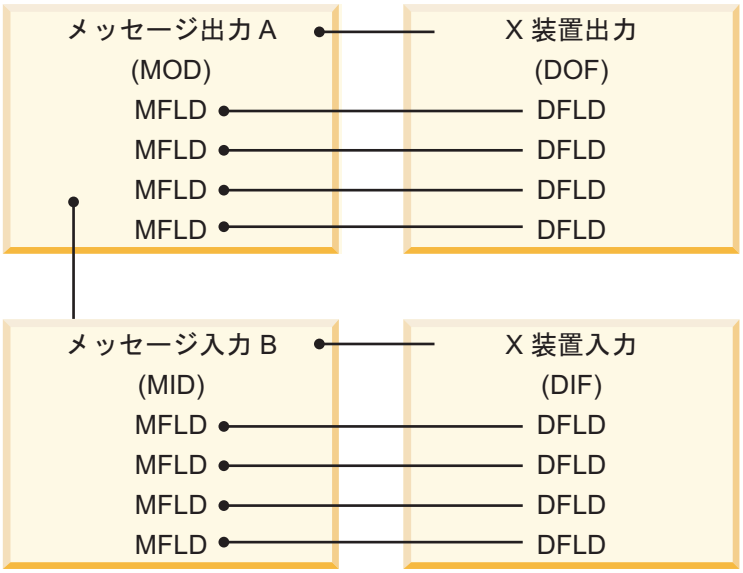


図 *24.* メッセージ・フィールドと装置フィールドのリンケージ

以下の図は、LPAGE と DPAGE 間に存在する第 3 レベルのリンケージを示しています。

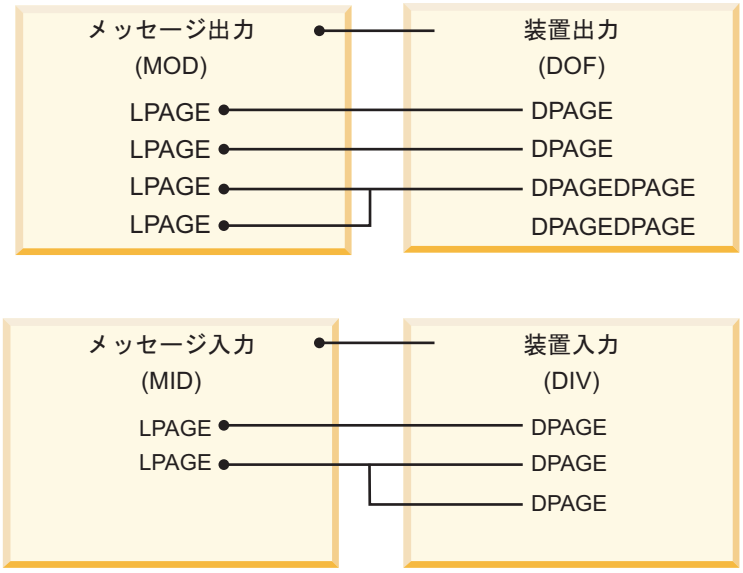


図 *25. LPAGE* と *DPAGE* の関係

MOD LPAGE は、DOF のいずれかの DPAGE を参照しなければなりません。ただし、1 つの MOD からすべての DPAGE を参照する必要はありません。

MID LPAGE が定義されていなければ、定義済みの MFLD はどの DPAGE のフィールドでも参照することができます。ただし、その装置から入力したメッセージは、どの入力データも 1 つの DPAGE に定義されているフィールドだけに対応させなければなりません。

1 つ以上の MID LPAGE を定義すれば、各 LPAGE では 1 つ以上の DPAGE を参照することができます。この場合、どの DPAGE も 1 つの LPAGE で参照しなければなりません。入力データが特定の DPAGE にもとづいて処理されるときは、その DPAGE を参照している LPAGE に基づいて、メッセージ編集が行われます。

以下の図は、第 4 レベルのリンケージを示しています。このレベルのリンケージを使用するかどうかはユーザーの任意ですが、このリンケージを使用すると、装置からの入力データを受信するときに表示される MOD LPAGE に基づいて、MID を選択することができます。

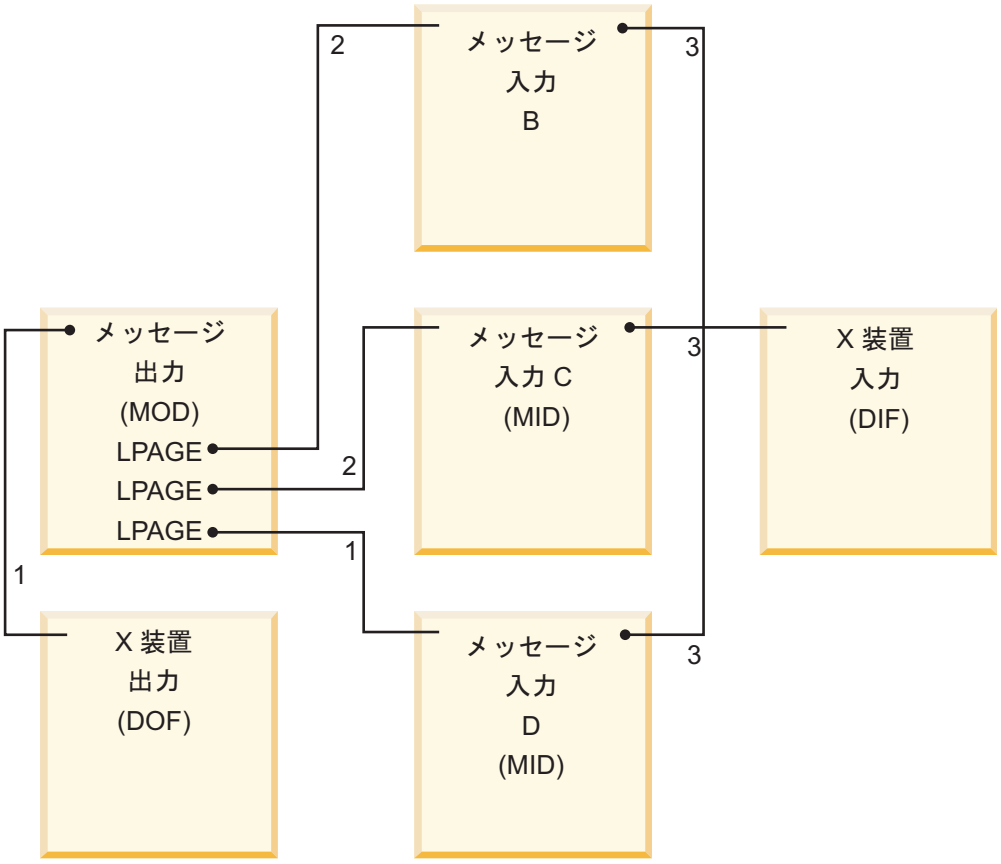


図 *26.* メッセージ記述子のリンケージ (オプション)

上の図の注:

1. 現行 LPAGE に MID 名が指定されていなければ、MSG ステートメントで指定されている次の MID 名が使用されます。
2. 現行 LPAGE に MID 名が指定されていれば、この名前を使用して入力が処理されます。

3. 形式定義に 3270 または SLU 2 装置が含まれている場合は、すべての MID が同一の DIF を参照しなければなりません。MOD を定義するときは、ユーザーが指定した同じ名前を使用して、DOF を参照しなければなりません。

以下の図は、これまでに述べてきた MFS 制御ブロックのリンケージをまとめたものです。

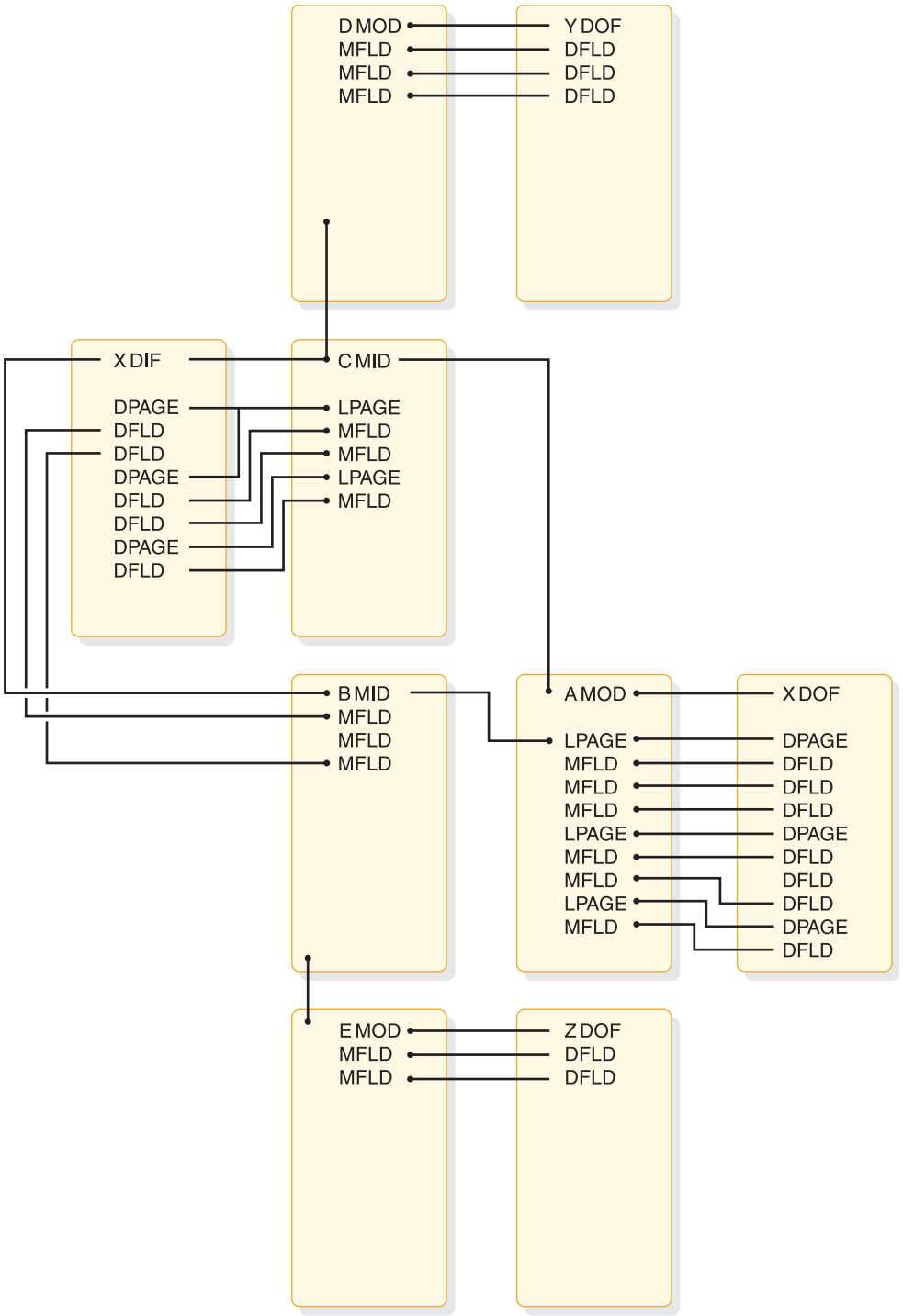


図 27. 制御ブロック・リンケージの要約

制御ブロック・リンケージは MFS 機能の基礎となるものですが、アプリケーション設計に影響を与える装置固有の条件もいくつか存在します。

3270 または SLU 2 表示装置

これらの装置では、装置への出力時にハードウェア機能によって装置にフィールドが設定され、オペレーターがフィールドの位置を変更できないため、リンケージに特別な制限があります。

フォーマット設定された入力を行えるのは、出力によってフォーマット設定されている画面からだけであるため、入力のフォーマット設定の際に使用する DPAGE および物理ページ定義は、必ずその前の出力をフォーマット設定する際に使用したものと同じでなければなりません。MFS 言語ユーティリティーは、制御ブロックのコンパイル時に、MOD、およびその MOD が参照している MID とが同じ FMT 名を参照しているかどうかを調べます。オンライン処理中に、前の DOF に対応する DIF を取り出せなかった場合は、ディスプレイにエラー・メッセージが送られます。

区画形式モードの 3290 パネル表示装置

3290 の画面は、区画 と呼ばれるいくつかの長方形の区域に分割することができます。 LPAGE/DPAGE の選択によって、出力メッセージの各論理ページは、DPAGE ステートメントで指定されている区画に送られます。

3290 が区画モードで作動しているときは、通常の制御ブロック・リンケージが働きます。ただし、MOD で記述された論理ページは別々の区画に送られることがあるので、追加の機能が用意されています。区画記述子ブロック (PDB) は、中間テキスト・ブロック (ITB) の 1 つのタイプです。 PDB では、1 つの出力メッセージに応答して画面に現れる区画セットを記述します。 PDB にはその他にも区画記述子 (PD) でコーディングされた 1 つの区画定義ステートメントも含まれています。いくつかの PD がまとまって、1 つの区画セット を定義します。

リンケージは以下のように機能します。

1. ある特定のメッセージに MOD が要求されます。この MOD で FMT を指名し、その結果、該当する DEV ステートメント (この場合は 3290 用の DEV ステートメント) と関連付けられます。DOF を作成して、メッセージが表示できるように 3290 を形式設定します。
2. DEV ステートメントそのものに PDB の名前を指定します。これによって MOD は DOF にリンクし、次に 3290 用に DEV ステートメントを介して PDB にリンクします。このリンケージにより、MOD の論理ページ (LPAGE ステートメントで定義した) は、PDB の区画記述子 (PD) にアクセスすることができます。
3. MOD の LPAGE ステートメントで、DOF の DPAGE ステートメントの名前を指定します。
4. 区画設定された 3290 では、DPAGE ステートメントに入っている PD キーワードで、PDB の区画記述子の 1 つを識別します。

このリンケージのため、各論理ページが区画記述子で記述された該当する区画と関連付けられます。メッセージ・キューから取り出された論理ページは、その該当区

画へと送られます。

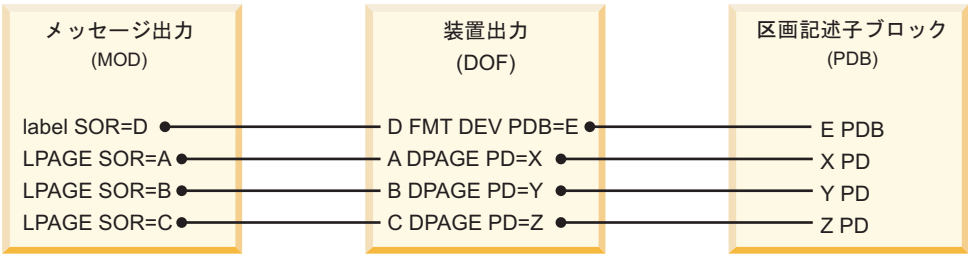


図 *28.* 区画形式モードのリンケージ

金融端末、3770、SLU 1、NTO、または SLU P 装置

これらの装置では、ハードウェアで設定するフィールド機能はないので、出力フィールドと入力フィールドの間に相関関係は必要ありません。

オペレーターが入力を行うとき、あるいはユーザーが金融端末や SLU P ワークステーション制御装置のプログラムを作成するときに、(MID 名を指定することにより) どの FMT を使用するのか、また (DPAGE に COND= で指定することにより) FMT のどの DPAGE を使用するのかを決定することができます。

金融端末または SLU P ワークステーション

金融端末および SLU P ワークステーションは非同期性の装置であるため、MFS は MOD と MID の間のチェーンを自動的に維持することができません。

したがって、MID 名は出力メッセージ・ヘッダーに入れられて装置に送信されます。リモート・コントローラーのユーザー作成アプリケーション・プログラムによって、MID 名が入力メッセージ・ヘッダーに入れられて返される場合は、オペレーターには意識されることなく、このチェーンは維持されます。

ISC サブシステム (DPM-Bn)

MSG TYPE=OUTPUT で指定した NXT=MID 名は、出力時に RDPN となり、リモート・プログラムやサブシステムによって変更されないかぎり、入力時には DPN になります。

フォーマット・ライブラリー・メンバーの選択

メッセージを入力として受信するとき、または出力に備えてメッセージを作成するときは、メッセージ記述子の中のユーザー提供の名前および端末の装置タイプと機構に基づいて、DIF または DOF いずれかが選択されます。

MFS 言語ユーティリティーは、FMT ラベル、DEV TYPE= と FEAT= の指定に基づいて、次に示すように IMS.FORMAT ライブラリーに入れる DIF および DOF のそれぞれのメンバー名を組み立てます。

バイト 内容

1 装置タイプ標識 (16 進数)。標識別の装置タイプのリストは、以下の表を参照してください。

2 装置機構標識 (16 進数)。機能別標識のリストについては、以下の表を参照してください。

3 DOF の場合、FMT ステートメントで指定したラベルの先頭文字。 DIF の場合、FMT ステートメントで指定したラベルの先頭文字を、小文字に変換したもの。

4 から **8**
FMT ステートメントのラベルの残りの文字

DEV FMT= 指定のバイト 1 の装置タイプ標識は、以下の表にリストしてあるとおりです。

表 *112. FMT=* の装置タイプ標識

装置機構	標識 **(16** 進数**)**
SLU 2-1 型表示装置	00
3284-1 型または 3286-1 型印刷装置	01
3277 または SLU 2-2 型表示装置	02
3284-2 型または 3286-2 型印刷装置	03
3604-1 または 2 (FIDS)	05
3604-3 (FIDS3)	06
3604-4 (FIDS4)	07
3600 (FIN)	08
3610、3612 ジャーナル印刷機構 (FIJP)	09
3611、3612 通帳記帳装置 (FIPB)	0A
3618 管理用印刷装置 (FIFP)	0B
SCS1: 3770; NTO; および SLU 1 (印刷データ・セット)	0C
SCS2: 3521 カード穿孔装置、3501 カード読取装置、2502 カード読取装置、および SLU 1 (送信データ・セット)	0D
3604-7 (FIDS7)	0E
それぞれ DPM-A1 から DPM-A15	11 から 1F
それぞれ DPM-B1 から DPM-B15	21 から 2F
それぞれ 3270-A1 から 3270-A15	41 から 4F

推奨事項**:** 所定のメッセージを受け取る可能性のある装置は、タイプごとに装置形式を定義する必要があります。オンライン実行中に、必要な装置タイプと機構が指定されている MOD または DOF が見つからないと、IMS のエラー時のデフォルト形式 (エラー・メッセージが入っている) を使用した出力メッセージが表示されます。また、必要な装置タイプおよび機構が指定されている MID または DIF が見つからない場合、入力は無視され、入力を行った装置にエラー・メッセージが送信されます。

ただし、各種の特定装置で同じ形式を使用することもできます。 FEAT=IGNORE が指定され、TYPE=3270,2 として定義された形式は、以下の装置のデフォルト形式として使用することができます。

- 3275
- 3276-2、3、4 型
- 3277-2 型
- 3278-2、3、4 型
- 3279-2、3 型

この端末装置を IMS に定義するには、SIZE=(n,80) (ただし n≥24) で TYPE=3270-An を指定します。

制約事項: IGNORE 機構は、MFSTEST モードではサポートされません。

端末装置を TYPE=3270,2 として IMS に定義しなければなりません。MFS は指定どおりの TYPE および FEAT の入ったブロックを探しますが、そのブロックが見つからないと、TYPE=3270,2 で FEAT=IGNORE と指定されたデフォルト形式のブロックを探します。

ETO 端末では、すでに説明してきたデフォルトの設定よりも、他のレベルでのデフォルトを優先して実行します。ETO 端末において、VTAM® PSERVIC 情報の画面サイズが 12x40 または 24x80 で定義されていて、その形式のブロックが見つからない場合は、さらに TYPE=3270,1 (12x40) または TYPE=3270,2 (24x80) を使用している同じ名前の形式や、同じ機構を使用している形式を探します。このような形式が見つからなかった場合は、すでに述べたデフォルトの検索が実行されます。この新規のデフォルト検索は MFSTEST モードのときでも使用できますが、古いデフォルト検索のときには使用できません。

装置形式は、IMS システム定義時に定められた宛先端末のさまざまな機構に基づいて選択されます。機構選択を使用してメッセージを受信するシステムでは、あらゆる機構の組み合わせごとに、装置形式を定義しておかなければなりません。機構選択は、メッセージ記述子 (MOD または MID) の指定に基づいて行われます。MOD に IGNORE オプションを指定した場合は、IGNORE 機構オプションを指定して装置形式を作成し、正しい処理を行ってください。

3270-1 型、2 型を除く 3270 または SLU 2 装置の画面サイズは、IMS システム定義時に設定するため、IMS システム定義を実行した後で MFS 言語ユーティリティーを実行させて、ユーザーが指定した形式を処理する必要があります。

異なる機能を併せ持つ装置でメッセージの受信または入力を行うとき、および各装置で特殊機構を使用するときは、機構選択を使用してください。

例えば、PF キー付きの装置のオペレーターは、PF キーを使用してフィールドにリテラルを入力することができます。PF キーが付いていない装置のオペレーターは、同じリテラルを画面上のフィールドに入力することができます。アプリケーション・プログラムにとっては、いずれのリテラルも同様に見なされます。入力方法は異なりますが、次の入力装置でアプリケーション・プログラムに同一のメッセージを入力することができます。

- 装置機構
- 印刷行 120
- 印刷行 126
- 印刷行 132

- データ入力キーボード装置
- PF キー
- 選択ライト・ペン検出
- 磁気カード読取装置 (OICR および MSR)
- 複式プラテン
- 3270、SCS1、および SCS2 装置および DPM プログラムについてユーザーが指定した機構

DEV の FEAT= に 2 バイトを指定するときは、以下の表に記載されている装置機構の標識値を使用します。

表 *113.* 装置機構の標識値の例

装置機構	標識 **(16** 進数**)**
P.L. 120 (印刷行 120)	40
P.L. 126	50
P.L. 132	60
DEK (データ入力キーボード装置)	C8
PFK (プログラム・ファンクション・キー)	C4
SLPD (選択ライト・ペン検出)	C2
OICR/MSR (磁気カード読取装置)	C1
IGNORE	7F
DEK、SLPD	4A
DEK、OICR	C9
DEK、SLPD、OICR	4B
PFK、SLPD	C6
PFK、OICR	C5
PFK、SLPD、OICR	C7
SLPD、OICR	C3
DUAL (複式プラテン)	C1
P.L. 132、DUAL	61
機構なし (3270)	40
3270、3270P、3770、SLU 1、SLU 2、SLU P、ISC (ユーザー指定の機構)	定義の際に使用できる標識 1. 01 2. 02 3. 03 4. 04 5. 05 6. 06 7. 07 8. 08 9. 09 10. 0A

3270 または SLU 2 画面のフォーマット設定

MFS は、必要なデータのみを 3270 表示装置と送受信するように設計されています。フィールドの設定やリテラルの表示のための装置副指令が送信時間の相当部分を占めることがあります。また、プログラム・データよりも副指令とリテラル・データの方が多いこともあります。

通常の操作では、表示されるフォーマットがすでに装置に存在しているときは、メッセージ内のユーザー指定データ、および変更可能フィールド属性のみが送信されます。装置に表示される現行のフォーマットは、装置出力フォーマット名、そのフォーマット内の DPAGE、およびその DPAGE 内の物理ページによって決まります。次のような状況が発生すると、MFS は装置出力に対して画面全体のフォーマット設定操作 (装置バッファーが消去され、すべてのフィールドとリテラルが送信されます) を実行します。

- 装置出力フォーマットの変更
- 装置出力フォーマット内の DPAGE の変更
- 物理ページ番号の変更
- オペレーターが CLEAR キーを押す場合
- CLEAR PARTITION キーを押して消去された区画に、画面全体のフォーマットを書き込む場合
- DEV ステートメントの DSCA オプションによってフォーマット書き込みが要求された場合
- 出力メッセージの SCA フィールドによってフォーマット書き込みが要求された場合
- MFS バイパス機能の実行後
- 永続的な入出力エラーが生じたために端末が停止した場合 - 画面は消去され、次の出力で画面全体のフォーマット操作が行われます。
- オペレーターがオペレーター識別カード読取機構を使用する場合 - 画面は消去され、次の出力で画面全体のフォーマット操作が行われます。

画面全体のフォーマット設定操作は、慎重に計画する必要があります。いくつかの要因が重なって、望まない画面表示またはプログラム入力 (またはその両方) になってしまうことがあります。

1. 非リテラル・フィールドにデータが入っていることを前提にして、プログラムが作成されている場合に、出力メッセージにデータを組み入れないと、装置がその出力メッセージを受信しても、データが画面に表示されないことがあります。この原因として、次のことが挙げられます。
   - 端末オペレーターが CLEAR キーを押してしまった。
   - 装置エラー
   - 応答前に別のメッセージが装置に送信されてしまった。
   - IMS の再始動

   このような前提のあるアプリケーションは、3270 装置依存のプログラムになります。
2. プログラムが 1 つの出力フィールドの一部分のデータしか送信しない場合、非リテラル・フィールドにすでにあったデータが原因となって、混乱が生じること

があります。すでに完全に埋まっているフィールドの部分的なデータを送信すると、そのフィールドが変更された場合に、フィールドに残っている古いデータが、新しい入力の中に混ざってしまいます。この問題を解決するためには、DPAGE ステートメントで PT (プログラム・タブ X'05') を充てん文字として指定する方法が有効です。PT を充てん文字として指定しておくと、装置フィールドに満たないデータがあれば、そのメッセージ・データ・フィールド (およびメッセージ・リテラル・フィールド) のあとにプログラム・タブ副指令が付けられて送信されます。これにより、装置フィールドの残りの部分は消去されて、ヌルになります。

プログラムがほんのわずかな出力データ・フィールドのみを所定の画面に送るのであれば、まず完全に埋まっている無保護フィールドをすべて消去することが望まれます。MFLD ステートメントのシステム制御域 (SCA) オペランド、または DEV ステートメントのデフォルト SCA (DSCA) オペランドで、アプリケーション・プログラム出力に『全無保護域消去』オプションを指定すれば、無保護フィールドを消去することができます。

3. フィールド・データの入力が確実に行われていることを確認するために、アプリケーション・プログラムによって事前変更の属性を要求することができます。事前変更の属性が要求され、メッセージがオペレーターによって論理化されずに、完全に装置へ送信されると、装置エラーや IMS の再始動が起こった場合に入力を行えなくなります。このエラーが起こるのは、端末を始動させても、IMS を再始動させても、メッセージによって画面が再設定されるわけではないからです。
4. 事前定義属性付きの装置フィールドに動的属性変更を要求すると、出力操作のたびに、そのフィールドの属性が装置に送信されます。これは、装置フィールドのデータが出力メッセージ内に入っていない場合でも送信されます。以下の属性が使用されます。
   - 出力メッセージ・フィールドに属性が備わっており、その属性が有効な場合は、動的属性変更が行われる。
   - メッセージ・フィールドが使用中の LPAGE に含まれていない場合、またはそのフィールドの属性が有効でない場合は、その装置フィールドの事前定義属性が使用される。

推奨事項**:** アプリケーションの設計は、次のような方針に沿って行ってください。

1. できるだけ多くのアプリケーションで、1 つの共通装置形式を共用します。画面全体の形式設定操作の数を減らすと、応答時間を大幅に短縮することができます。また、メッセージ形式バッファー・プールの入出力アクティビティーが減るばかりでなく、形式ブロック・プールの必要量も削減されます。
2. 形式設定操作をいつ行うかについての決定を MFS に任せます。これによって、フォーマット設定が不要なときの送信時間が短縮されます。
3. 装置のオペレーターにとって必要なすべての非リテラル・データが、アプリケーション・プログラム出力メッセージに入っていることを確認します。装置に以前から残っているデータをあてにしてはなりません。
4. PT 充てんオプションを使用して、プログラム出力データを受け取る装置上のフィールドには、メッセージからのデータのみが入っていることを確認します。
5. アプリケーション側で無保護フィールドの消去が必要とされている場合には、SCA または DSCA の全無保護域消去オプションを使用します。

形式設定操作の制御のために 2 つの MFS 機能が提供されています。この 2 つの機能は、メッセージ・フィールドのシステム制御域 (SCA) および DEV ステートメントのデフォルト SCA (DSCA) オプションです。どちらを使用しても、IMS は、出力の送信前に強制的に送信を再設定したり、すべての無保護フィールドまたは区画を消去したりすることができます。形式書き込み強制オプションを使用すると、装置バッファーの内容の消去、全フィールドの設定、さらに全リテラルの送信を行うことができます。全無保護域消去オプションを使用すると、すべての無保護フィールドまたはすべての区画が消去されてヌルになり、その後でデータが書き込まれます。

関連概念:

573 ページの『システム制御域 (SCA) とデフォルト SCA (DSCA)』

3290 画面のフォーマット設定

3290 画面は、論理装置 (LU) と呼ばれるいくつかの独立した区域に分割することができます。各 LU には、基本状態と形式設定状態とがあります。LU が形式設定状態であれば、その LU を標準形式モードあるいは区画 形式モードにすることができます。

以下に、3290 画面のフォーマット設定について説明していきます。

画面の分割

3290 の大型画面には、小さい文字セル (6 × 12 ピクセル) ならば最大 62 行 x 160 桁の表示、大きい文字セル (9 × 15 ピクセル) ならば最大 50 行 x 106 桁の表示を行うことができます。

3290 画面はいくつかの区域に分割することができ、各区域は独立してオペレーターと対話することができます。2 通りの分割方法があります。

- 画面を個別の LU に分割する。
- 論理装置を個別の区画に分割する。

初めの方法では、3290 端末装置およびその画面を、最大 4 つの個別の LU として定義することができます。各 LU は互いに独立しており、IMS に対して固有のアドレスを持つ別個の端末として定義されます。このサポートは、IMS には透過的です。3290 端末と IMS の間で複数の入力メッセージまたは出力メッセージ (あるいは、この両方) を同時にアクティブにする要求を IMS アプリケーションで出す場合、複数の LU を定義しておくと役に立ちます。ただし各論理装置では、1 つの入力メッセージまたは出力メッセージだけしかアクティブにすることができません。

さらに、ソフトウェアで区画を設定すれば、各論理装置を最大 16 の区画に分けることができます。この場合、各アプリケーション・メッセージは区画セットを指定することができます。したがって、そのメッセージの各論理ページは、その区画内の特定の区画と関連付けられます。次の場合は、ソフトウェアによる区画設定が便利です。

- オペレーターが一度に複数の論理ページを見る必要がある場合。
- 出力ページを表示する区画とデータを入力する区画が別に必要な場合。

- 論理装置が形式設定モードになっているときに、IMS システム・エラー・メッセージを受信する区画を定義しておきたい場合。この機能は、現行の MFS SYSMSG フィールド・サポートの代わりに使用することができます。
- スクロールが必要な場合。スクロールは、区画ビューポート内のデータを上下に移動させるために使用します。この機能は、区画モードの 3290 でのみ定義することができます。明示区画スクロールを使用すれば、物理画面上のビューポートよりも大きな表示スペースに、MFS ページを定義することができます。このため、IMS と端末との間で、メッセージを表示するために必要な対話の回数を減らすことができます。

3290 では、物理デバイス 1 台につき最大 16 区画まで指定することができます。さらに、LU に区画を定義した場合は、実際に定義した区画の数に関係なく、1 つの LU で使用できる区画が最低でも 8 区画は存在していなければなりません。つまり、ある LU を 9 区画と定義した場合、その物理デバイスにはあと 7 区画しか残っていないので、他の LU で区画を定義することはできません。必然的に、(最大 4 つまで可能なうちの) 2 つの LU にしか区画を定義できません。

区画を定義する際は、次のことも考慮しなければなりません。

- 区画は長方形でなければなりません。
- 複数物理ページ入力を使用している場合を除いて、1 つの入力メッセージは 1 つの区画の 1 物理ページから構成されます。複数物理ページ入力を使用している場合は、1 つの入力メッセージを構成する複数物理ページは、1 つの区画から得なければなりません。
- 現行 PDB でシステム・メッセージ用の区画を定義せずに、DOF でもシステム・メッセージ・フィールドを定義しなかった場合、システム・メッセージが出されると現行の区画形式モードは無効になり、3290 (または該当する特定の LU) は標準形式モードに戻されます。

端末の状態およびモード

3290 は、1 台の LU として、または 3290 を分割して定義した LU のうちの 1 台として、端末基本状態または端末形式設定状態に置かれます。

端末基本状態の 3290 は、初めて IMS に接続されるとき、または CLEAR キーが押されたときには、現在サポートされている他の SLU 2 ノードと同じように作動します。この状態のときは、IMS への入力メッセージは基本編集で編集され、MOD と関連付けられていない出力メッセージは、デフォルトの MFS MOD を使用して形式設定されます。

端末形式設定状態の 3290 は、次のいずれかのモードなります。

- 標準形式モード
- 区画形式モード

この形式モードの選択は、メッセージ出力時に動的に行われます。出力メッセージは MOD と関連づけられており、その MOD では DOF の名前が指定されています。DOF に何を指定したかによって、3290 の形式モードが決定されます。

- DOF に区画記述子ブロック (PDB) の名前が指定されないかぎり、3290 は標準形式モードになります。端末は普通の SLU 2 ノードとして形式設定され、作動します。

- DOF に区画記述子ブロック (PDB) の名前が指定されていれば、3290 は区画形式モードになります。

区画セットの初期設定、ページング、および活動化

3290 (または分割されている LU のうちのいずれか) が区画形式モードになっていれば、次のことを実行する際にさまざまな方法を取ることができます。

- 出力メッセージの 1 つ以上の論理ページで、複数の区画を初期設定する。
- その後でオペレーターが区画への論理ページのフローを制御する。
- ある特定の区画をアクティブ区画 にする。

初期設定およびオペレーター制御によるページングは、3 つのオプションのうちの 1 つを選択して決めます。このオプションは、PDB の PAGINGOP オペランドで指定します。どのオプションを選択したかによって、次のいずれかの初期設定が行われます。

1. メッセージの最初の論理ページが、該当区画へ送られます。
2. いずれかの区画で第 2 論理ページに達するまで、メッセージの第 1 論理ページが、それぞれ該当区画へ送られます。
3. 各区画が、その区画あての最初の論理ページを受け取ります。

このオプションは、どの区画がアクティブであるかによって、オペレーター制御ページングが影響を受けるかどうかを決定します。

3290 が区画形式モードになると、ある特定の区画が アクティブ 区画になります。活動区画の決定方法は 2 通りあります。

- 論理ページは、DPAGE ステートメントを使用して、それぞれの区画へ送られます。 DPAGE ステートメントの 1 つに ACTVPID オペランドを指定して、初期設定区画を指示しておくことができます。アプリケーション・プログラムは、ACTVPID を使用してどの区画がアクティブ区画であるかを宣言することができます。
- ACTVPID キーワードが見つからない場合は、PDB 内の最初の PD ステートメントで定義された区画がアクティブ区画になります。

アクティブ区画が、初めにデータを受け取った区画とはかぎりません。

関連概念:

611 ページの『区画形式モードの 3290』

3180 画面のフォーマット設定

3290 と同様に、3180 端末は SLU 2 装置として IMS でサポートされています。3180 での区画化およびスクロールのサポートは、3290 で提供されるサポートとほぼ同じです。

例外**:** 3180 の場合

- 3180 では定義できるのは 1 つの区画だけで、サイズも制限されています。 (一方、3290 では、さまざまなサイズの複数の区画を定義することができます。)
- 3180 では、論理装置の表示画面サイズおよびビューポートの位置を画素 (ピクセル) で指定することができません。(3290 はピクセルをサポートします。)

- 3180 では、ユーザーがアクティブ区画を指定することができません。(3290 では、アクティブ区画を指定することができます。)

これらの制限は、3180 を IMS に接続するときと、他のサブシステムに接続するときでは、使用する画面サイズが異なる場合にのみ当てはまります。画面サイズが同じであれば、3180 カスタマー・セットアップ・インストールの指示に従ってください。特別な IMS コードは不要です。

前バージョンの MFS との装置の互換性

IMS システム定義時に、装置タイプ記号名 (オプション 1) を使用して 3270 装置を定義していなければ、装置形式定義の変更は不要です。

IMS システム定義時に、装置タイプ記号名 (3270-An) (オプション 2、3、および 4) を使用して 3270 装置を定義するのであれば、必要に応じて 3270 装置形式定義に変更を加えます。

以下の表の例には、装置タイプ記号名を画面サイズに関連付ける場合の標準が示してあります。標準の使用をお勧めします。

表 *114. 3270* 装置に関する *MFS* 装置定義の互換性：

装置および画面サイズ	装置および画面サイズ[1]	新規の **IMS** システム定義[1]
3275 または 3277 (12X40)	MFS: DEV TYPE= (3270,1) モデル 1	MFS: DEV TYPE= 3270-A5 [2、4]
3275、3276、3277、3278 (24X80)	MFS: DEV TYPE= (3270,2) モデル 2	MFS: DEV TYPE= 3270-A2 [2、4]
3276、3278 (12X80)	MFS: DEV TYPE= (3270,1) モデル 1	MFS: DEV TYPE= 3270-A1 [2、3]
3276、3278 (32X80)	MFS: DEV TYPE= (3270,2) モデル 2	MFS: DEV TYPE= 3270-A3 [2、3]
3276、3278 (43X80)	MFS: DEV TYPE= (3270,2) モデル 2	MFS: DEV TYPE= 3270-A4 [2、3]
3278 (27X132)	MFS: DEV TYPE= (3270,2) モデル 2	MFS: DEV TYPE= 3270-A7 [2、3]

注:

1. TYPE または TERMINAL マクロで画面サイズを指定した場合。
2. 新しい装置でこの形式を使用するが、元の装置では使用しない場合は、TYPE= (3270,1) または (3270,2) を対応する画面サイズの 3270-An に変更して、再コンパイルします。
3. 以下の表のオプション 3 を参照。
4. 以下の表のオプション 4 を参照。

以下の表は、より大きい画面の装置で特定のオプションを選択したときの利点と欠点、および既存の装置形式を選択する場合に必要なアクションを示しています。

表 *115.* 大型画面装置の利点と欠点

Option	利点	欠点	変換アクションの要/不要
1	既存の MFS 形式を変更せずに、そのまま使用できる。	フルスクリーンを使用することができない。	不要 (前掲の表に示されている現行形式を使用する。)
2	フルスクリーンを使用することができる。	新規の装置形式を設計する必要がある。	不要 (新規の形式を定義する。)
3	新規の表示画面装置で、移行過程の一環として既存の形式を使用することができ、段階を追って新規の装置形式で置き換えることができる。	装置記号名を使用するために、既存の装置形式を変更する必要がある。	必要 (前掲の表を参照。)
4	現行画面サイズと新規画面サイズの定義の整合性がある。	形式をすべて変更する必要がある。	必要 (前掲の表を参照。)

IBM 3278-52/3283-52 および IBM 5550 ファミリー (3270 として) に関する互換性

IBM 3278-52/3283-52 のメッセージ形式定義は、上位互換性があります。ただし、5550 ファミリーの装置用に漢字機能を使用して作成したメッセージ形式は、IBM 3278-52/3283-52 で使用することはできません。

既存の 3270 および IBM 5550 ファミリー (3270 として) に関する互換性

既存の 3270 および 3278-52/3283-52 メッセージ形式に、フィールドの枠取りや入力制御の指定を追加するときは、次のことに注意してください。

- フィールドの枠取り
  - 3270 表示装置では、左線、右線、上線、下線は、ユーザー・フィールドの位置を占めません。拡張属性の動的変更を行わない場合は、アプリケーション・プログラムを修正する必要はありません。
  - SCS1 プリンターでは、MFS は左線と右線用の印刷位置を確保します。フィールドが左端桁から始まっていたり、右端桁で終わっていたりすると、左線または右線のための位置を確保できないので、左右の線は正しく印刷されません。これを訂正するには、1 バイトを切り捨てるようにアプリケーション・プログラムを修正します。2 つの隣接するフィールドが論理的には 1 つのものであり、上線と下線がつなげられるものであれば、アプリケーション・プログラムを修正する必要はありません。

    いずれの場合も、動的変更を行うのであれば、アプリケーション・プログラムを修正しなければなりません。
- DBCS/EBCDIC 混合フィールド
  - 3270 表示装置では、SO/SI 制御文字は画面上の 1 バイトを占有します。つまり、画面上の実際の長さはメッセージ形式の長さと同じです。したがって、既存のアプリケーション・プログラムを変更する必要はありません。

DBCS/EBCDIC 混合データを既存の EBCDIC フィールドに割り当てるときには、SO と SI が対になっていること、EGCS データが偶数長であること、MFLD が DFLD へマップされても SO および SI が切り捨てられないことを、アプリケーション・プログラムで確かめておく必要があります。

- SCS1 プリンターでは、DBCS/EBCDIC 混合データを使用するときには、MIX/MIXS を指定しなければなりません。この場合、メッセージ長と出力の長さは異なるので、各フィールドの特性に合わせて、アプリケーション・プログラムの MFLD を修正してください。

STACK/UNSTACK を使用した MFS 3270 装置形式から記号名形式への変換

IMS MFS 言語ユーティリティーのコンパイル・ステートメントである STACK および UNSTACK を使用すれば、既存の MFS 3270 装置形式をユーザー定義の装置タイプ記号名形式に変換することができます。STACK ステートメントを使用して、1 つ以上の SYSIN または SYSLIB レコードを記述すると同時に、いったん処理されたこれらのレコードを、後で使用するためにストレージに残しておくよう要求します。UNSTACK ステートメントでは、以前に処理した SYSIN/SYSLIB レコード・スタックの検索を要求します。

例えば、次の既存の 3270 形式定義を使用します。

```
label    FMT
         DEV       TYPE=(3270,2), ...
         DIV       TYPE=INOUT
         DPAGE     CURSOR=((2,3))
label    DFLD
label    DFLD
label    DFLD
         FMTEND
```

DEV ステートメントと、コンパイル・ステートメントの STACK および UNSTACK を次のように使用すれば、大型画面の表示装置用にユーザー定義の装置タイプ記号名 (TYPE=3270-An を使用) の形式を作成することができます。

```
label    FMT
         DEV       TYPE=3270,2,...
         STACK     ON
         DIV       TYPE=INOUT
         DPAGE     CURSOR=((1,2))
label    DFLD
label    DFLD
label    DFLD
         STACK     OFF
         DEV       TYPE=3270-A2,...
         UNSTACK
         FMTEND
```

UNSTACK ステートメントは、STACK ON と STACK OFF の間にあるステートメントを繰り返す機能があります。3270 モデル 2 の装置形式のほかに、3270-A2 用の装置形式が作成され、その装置レイアウトは 3270 モデル 2 のレイアウトと同じです。

3270 装置形式の変換例

次の例で、3270 装置に関する MFS 装置定義の互換性をさらにわかりやすく説明します。

システムには 3270 モデル 1 とモデル 2 の表示装置がインストールされており、さらに、画面サイズが 12×80、24×80、32×80、および 43×80 の各表示装置を追加インストールしたと仮定します。追加装置については新たに IMS システム定義が行われ、3270 モデル 1 とモデル 2 の装置については、装置記号名を指定するために再定義が行われました。

これらの 3270 表示装置に関する、IMS システム定義の指定は次のとおりです。

- TYPE=3270-A1, SIZE= (12x80) は、画面サイズが 12x80 である追加装置用。
- TYPE=3270-A2, SIZE= (24x80) は、3270 モデル 2 および画面サイズが 24x80 である追加装置用。
- TYPE=3270-A3, SIZE= (32x80) は、画面サイズが 32x80 である追加装置用。
- TYPE=3270-A4, SIZE= (43x80) は、画面サイズが 43x80 である追加装置用。
- TYPE=3270-A5, SIZE= (12x40) は、3270 モデル 1 装置用。

3270 モデル 1 とモデル 2 およびその他の装置で使用するために既存の 3270 モデル 1 およびモデル 2 の装置形式定義を変換するには、その MFS 定義を次のように変更する必要があります。

既存の定義 :

```
label    FMT
         DEV       TYPE=(3270,1)
         DIV       TYPE=INOUT
         DPAGE                  ...
label    DFLD                   ...
label    DFLD                   ...
label    DFLD                   ...
         DEV       TYPE=(3270,2)
         DIV       TYPE=INOUT
         DPAGE                  ...
label    DFLD                   ...
label    DFLD                   ...
label    DFLD                   ...
         FMTEND                 ...
```

変更が加えられ再コンパイルされたもの :

```
label    FMT
         DEV       TYPE=3270-A5 (changed from (3270,1) to 3270
                                 display with 12x40 screen size)
         STACK     ON
         DIV       TYPE=INOUT
         DPAGE                  ...
label    DFLD                   ...
label    DFLD                   ...
label    DFLD                   ...
         STACK     OFF
         DEV       TYPE=3270-A1 (3270 display with 12x80 screen
                                     size)
         UNSTACK
         DEV       TYPE-3270-A2 (changed from (3270,2) to 3270
                                     display with 24x80 screen size)
         STACK     ON
         DIV       TYPE=INOUT
         DPAGE                  ...
label    DFLD                   ...
label    DFLD                   ...
label    DFLD                   ...
         STACK     OFF
         DEV       TYPE=3270-A3(3270 display with 32x80 screen
```

```
                                size)
        UNSTACK  ,KEEP
        DEV      TYPE=3270-A4(3270 display with 43x80 screen
                                size)
        UNSTACK
        FMTEND
```

新規の装置形式は、変更を適用して再コンパイルした後、各画面サイズが有効になるように設計されているため、前の形式定義は次のように再コンパイルできます。

```
label   FMT
        DEV      TYPE=3270-A5
        DIV      TYPE=INOUT
        DPAGE                  ...
label   DFLD                   ...
label   DFLD                   ...
label   DFLD                   ...(existing device fields
                               using 12x40 screen size)
        DEV      TYPE=3270-A1
        DPAGE                  ...
label   DFLD                   ...(new device fields using
                               12x80 screen size)
        .
        .
label   DFLD                   ...
        DEV      TYPE=3270-A2
        DIV      TYPE=INOUT
        DPAGE                  ...
label   DFLD                   ...(existing device fields
                               using 24x80 screen size)
label   DFLD                   ...
label   DFLD                   ...
        DEV      TYPE=3270-A3
        DIV      TYPE=INOUT
        DPAGE                  ...
label   DFLD                   ...(new device fields using
                               32x80 screen size)
        .
        .
label   DFLD                   ...
        DEV      TYPE=3270-A4
        DIV      TYPE=INOUT
        DPAGE                  ...
label   DFLD                   ...(new device fields using
                               43x80 screen size)
        .
        .
label   DFLD                   ...
        FMTEND
```

3270 印刷装置および SLU 1 に関する互換性

拡張属性カラー、強調表示、およびプログラム式シンボルを使用する場合、または垂直形式設定あるいは行密度設定データ・ストリームを使用する場合は、アプリケーション・プログラムを調べて、必要に応じてプログラムを修正してください。

現在インストールされている 3270 印刷装置と互換性がある 3270 印刷装置を SNA 以外の制御装置に追加した場合、その追加印刷装置は、既存の 3270P モデル 1 またはモデル 2 の装置形式を使用します。印刷バッファーについては、480 文字 (現在のモデル 1) または 1920 文字 (現在のモデル 2) の既存の IMS システム定義を使用します。

装置接続機構を SLU 1 に変更する MFS ユーザーは、MFS 装置形式定義を以下の表に示すように変更してください。以下の表は、現行装置、MFS 装置タイプ、新規の制御装置、システム定義、MFS 装置タイプ、および z/OS に必要な変更をリストしています。

表 *116. 3270* 印刷装置および *SLU 1* 装置に関する *MFS* 装置定義の互換性：

現行装置	現行 **MFS DEV TYPE**	新規の装置または制御装置	新規の **IMS** システム定義	新規の **MFS DEV TYPE**	**z/OS** に必要な変更
3284/3286	3270P	3274 または 3276 SNA 制御装置に接続される 3827/3289	SLUTYPE1	SCS1	注を参照

注:

DEV TYPE=(3270P,n) を DEV TYPE=SCS1 に変更して、再コンパイルします。または、すべてのプリンターを新規の制御装置に変更しない場合は、DEV TYPE=3270P のあとに次のステートメントを追加して、再コンパイルします。

```
STACK ON
(3270P DPAGE, DFLD ステートメント)
STACK OFF
DEV TYPE=SCS1
UNSTACK
```

SLU P に関する互換性

MFS がサポートする別の装置用に作成されたアプリケーション・プログラムは、DPM 装置に合うように DIF および DOF を定義しておけば、プログラムを変更しなくても SLU P (DPM-An) 装置で実行することができます。

3270 表示装置での事前変更フィールドのように、装置依存の固有の機能をプログラムが使用している場合は、変更が必要になることがあります。この場合、リモート・プログラムに渡した事前変更フィールドが入力メッセージに入れられて返されてくる場合にかぎり、プログラムを変更しなくてもそのまま実行することができます。このためには、リモート・プログラムが、出力メッセージ・フィールドの属性バイトを正しく解釈し、指示された装置機能を IMS アプリケーション・プログラムの要求どおりに扱うことが必要になります。

MFS を使用していない既存の IMS アプリケーション・プログラムは、3600 または 3790 の適切なバッファー・サイズに合うように、またメッセージ・テキストが SCS 文字ストリングの互換性のあるサブセットになるように、調整しなければなりません。

MFS メッセージおよび装置形式のシステム性能の拡張

メッセージと装置形式をどのように設計しても、通常はメッセージの編集に必要な時間やリソースに大きな影響はありません。ただし、送信時間と応答時間にはかなりの影響を与えます。

MFS がサポートする装置のシステム性能の拡張

MFS がサポートする装置を使用する際にシステム性能を拡張するには、次のようにしてください。

メッセージ形式バッファー・プールの操作の評価

IMS /DISPLAY POOL コマンドを使用して、メッセージ形式バッファー・プールの操作を評価することができます。

次の値を減らすことが目標です。

```
I/O+DI    (The sum of the numbers of fetch
 REQ1      I/O operations and directory I/O operations
           divided by the number of block requests from
           the pool.)
```

性能向上のためのヒント

この値を減らすためには、次の項目から 1 つ以上選んで実行してください。

- 形式ブロック入出力を減らします。形式ブロックの読み取りに費やされる CPU サイクルおよびチャネル/装置時間は、MFS 処理コストの中で最も大きな比重を占め、しかも調整がきく部分です。形式ブロック入出力を減らすためには、次の手法を使用することができます。
  - $$IMSDIR (オプションの MFS 索引ディレクトリー) を評価しインプリメントする。この場合は、MFS 制御ブロックをいくつか選択し、その使用頻度に基づいて索引を作成する必要があります。ほとんどの場合、$$IMSDIR を使用すると、オンライン操作中に、1 つの形式ブロックにつき 1 回ずつ読み取りを減らすことができます。
  - MFBP (メッセージ形式バッファー・プール) のサイズを増やす。
  - 取り出し要求エレメント (FRE) の数を増やす。
- セグメント数を最小限に抑えます。アプリケーション・プログラムにとって都合がいいように、メッセージをセグメント化するか、あるいはセグメント・サイズの制約に従ってセグメント化すべきです。 MFS および DL/I でのセグメント化処理には、相当数の CPU サイクルが必要になるので、不必要に多数のセグメント化を行ってはなりません。
- オプション 2 の入力を使用します。アプリケーション入力をセグメント化させることで、特定の条件下では、セグメントの装置入力が渡されないようにすることができます。このような場合、オプション 2 の入力メッセージを使用すると、処理時間が多少短縮され、IMS のメッセージ・キューに必要なスペースを減らすことができます。
- オプション 3 入力を使用します。たくさんのフィールドが定義されていても、そのうちの数個のフィールドのみを入力として受け取る場合は、オプション 1 や 2 よりもオプション 3 を使用するほうがパフォーマンスが向上します。編集時のバッファー・プール・スペースが余計に必要になりますが、メッセージ・キュー・スペースの必要量は減少します。定義済みのフィールドの大部分が入力として受け取られる場合は、オプション 3 のパフォーマンスは、処理時間とメッセージ・キューのスペースのどちらの点でもオプション 1 や 2 に劣ります。
- 複数個の DFLD リテラルを結合します。隣接している、あるいはほぼ隣接している複数の装置フィールド位置に、DFLD リテラルを入れる場合は、それらのリテラルをひとまとめにして、DFLD リテラル定義の数を減らすようにします。リテラルをまとめるときは、その数が DFLD リテラルの最大長を超えないようにします。いくつかの DFLD リテラルを 1 つにまとめることで、ブロック・サイ

ズが小さくなり、それによって MFS のプロセス時間が短縮されます。3270 または SLU 2 表示装置ついては、送信時間が短縮されます。

- どの MSG 記述子からも参照されない DFLD は定義しないようにします。このような DFLD はブロック・スペースを占有するため、数多く使用すると逆に MFS 処理時間に悪影響を及ぼします。
- 出力メッセージ・フィールドを結合します (可能な場合)。1 つのセグメントの連続する複数の出力メッセージ・フィールドを、相対的に同じ長さを持つ連続する複数の装置フィールドにマップする場合は、メッセージ・フィールド同士、装置フィールド同士を 1 つにまとめて、1 つのメッセージ・フィールドを 1 つの装置フィールドにマップさせるようにします。表示される各フィールドが 1 個のブランクだけで区切られており、装置フィールドの長さと等しいメッセージ・データが常に存在するような場合は、このような結合を行うことでパフォーマンスがかなり向上します。

  ただし、メッセージ・フィールドの結合が、フォーマット設定に関するアプリケーション・プログラムの負担を増やすことになるのであれば、フィールドの結合はお勧めできません。
- 同一形式を重複して定義しないようにします。連結したライブラリーに重複ライブラリーがある場合、最初のライブラリーのコピーが取り出されるとは限りません。
- 単純な入力には個別に形式を定義しないようにします。ほとんどの MFS 装置形式には、オペレーターが単純なトランザクションやコマンド (その形式が設計されるアプリケーションに関連するものまたは非関連のもの) を入力できるように、いくつかのユーザー入力フィールドが必要です。どのような形式でも 4 個の制御ブロックが必要になるので、特に単純な入力の設計のためだけの形式を、必要以上に定義してはなりません。

関連資料:

544 ページの『入力メッセージ・フォーマット設定オプション』

/DISPLAY POOL コマンド (コマンド)

3270 または SLU 2 表示装置のシステム性能の拡張

3270 または SLU 2 表示装置を使用しているときに、システム性能を拡張する場合は、次のことを行ってください。

- 事前に形式設定された画面を使用します。これは、3270 または SLU 2 表示装置を使用しているときに、MFS のパフォーマンスを向上させるうえで最も有効な手段です。画面上にフィールドを定義して、リテラルを設定するためには、通常十分な量のデータが必要です。ただし、これらのフィールド定義およびリテラルは、常に送信する必要があるわけではありません。その装置に表示されている形式を利用することができれば、リモート端末への送信時間を最大 50% 減らすことができます。
- メッセージ出力にヌルを埋め込みます。DPAGE ステートメントで FILL=NULL または PT オプションを指定しておくと、装置に送信されるデータ量、および出力を形式設定するための処理量が減少します。
- 混合モードの操作を減らします。選択ライト・ペンが即時検出可能フィールドで使用され、装置の他のフィールドの内容を変更すると、混合モード操作が行われます。混合モードの操作では、複数の入出力操作が実行されるので、応答時間、

回線使用率、および処理時間がいずれも増大します。さらに、その結果得られるメッセージの内容は、選択ペンが使用されたという標識以外は、Enter キーを押したときに作成されるデータとまったく同じです。

- ページング要求を使用します。アプリケーションの設計上可能であれば、オペレーターが論理ページ要求を入力する形を取らずに、PA1 (プログラム・アクセス・キー 1) のページ先送り機能を使用します。PA1 機能を使用すれば、オペレーターに要求される動作が減るとともに、通信回線の使用時間も短縮され、さらにページ要求処理の前に入力を編集する必要がなくなります。
- あとに非リテラル DFLD が続くリテラル DFLD の長さを決めて、最後の有効リテラル文字と、非リテラル・フィールドの属性位置の直前の位置との間にスペースを入れます。このアクションによって、ブロック・サイズおよび文字伝送が減少します。ただし、開けるスペースが 2 から 5 文字までの場合にのみ、このアクションを行うようにしてください。
- 保護 (PROTECT) 属性を付ける DFLD の長さを大きくします。非リテラル DFLD を PROTECT 属性を付けて定義し、次の装置フィールドとの間を複数個のブランクで区切っている場合、この非リテラル DFLD が出力データを受け取るよう要求されるときは、長さを増やすことを考えてください。 この出力データの発信元は、アプリケーション・プログラム、/FORMAT コマンド、または MFLD リテラルのいずれかです。メッセージ・データをその DFLD にマップするときに、複数個の MOD を使用することができます。DFLD の長さを大きく取っておけば、文字充てん (FILL=C' ') を指定しないかぎり、文字伝送は減少するはずです。FILL=C' ' は、なるべく指定しないようにしてください。
- CLEAR キーの使用を最小限に抑えます。端末オペレーターには、必要以上に CLEAR キーを使用しないよう指示してください。さらに、ERASE INPUT および ERASE EOF などの、その他のファンクション・キーについても、端末オペレーターが正しく使用できるように説明しておいてください。

  画面フォーマットを設計するときは、CLEAR キーの使用を減らすことを目標においてください。単純な入力は、フォーマット設定された画面から行うようにします。そのためには、あらゆる形式設定済み画面の装置フィールドの位置を、単純入力用の標準装置フィールドと同じ位置に設定してください。この標準をすべての形式定義に強制します。

  CLEAR キーの使用回数を減らすことで、応答時間が短縮され、通信回線の使用効率が増大します。

大型画面の 3270 または SLU 2 装置

複数のページを組み合わせて大型画面に表示する場合は、ページ数を減らすとそれに比例してオペレーター・ページングが減少します。 IMS システム定義 TERMINAL マクロの OUTBUF キーワード、または ETO ログオン記述子で、1 ページ全体のデータ量を指定できない場合は、そのページを送信するのに複数の VTAM SEND が必要になります。

関連概念:

480 ページの『3270 または SLU 2 画面のフォーマット設定』

関連タスク:

拡張端末オプション (ETO) (コミュニケーションおよびコネクション)

DPM を使用する SLU P および ISC サブシステム

OPTIONS=PPAGE が DIV ステートメントに指定されている場合、1 つの PPAGE (表示ページ) 内のフィールドのセットは、1 つ以上のレコードに一緒に入れられて送信されます。リモート・プログラムまたは ISC サブシステムから要求時ページングが要求されるたびに、追加の表示ページが送信されます。このレベルでページングを行うと、リモート・プログラムまたは ISC サブシステム側の処理は最も単純になりますが、IMS にとっては最も処理時間がかかるようになります。

OPTIONS=DPAGE が指定されていると、論理ページの全フィールドは、1 つ以上のレコードに入れられて一緒に送信されます。リモート・プログラムまたは ISC サブシステムから要求時ページングが要求されるたびに、追加の論理ページが送信されます。このレベルでページングを行うと、論理ページに複数の表示ページが含まれている場合は、リモート・プログラムまたは IMS サブシステム側でのデータ処理がさらに複雑になりますが、IMS にとっては処理時間の負担が軽くなります。

OPTIONS=MSG を指定すると、メッセージ内のすべてのデータが一緒に送信され、ページングは行われません。この方法は、より多くの処理とロジックがリモート・プログラムまたは ISC サブシステムで必要になります。しかし、リモート・プログラムまたは ISC サブシステムによって、全ページが実際に使用されるのであれば、IMS のパフォーマンスにとっては最善の方法といえます。リモート・プログラムまたは ISC サブシステムが使用しないページが多数あるときに、このオプションを使用すると、不必要な回線トラフィックと IMS 処理が行われることになります。

自動ページ化出力を指定し (SCA バイト 1、ビット 5)、さらに DPM-Bn の場合にオプション PPAGE または DPAGE を指定すると、メッセージの全データが送信され、要求時ページングは実行されません。

RCD ステートメントを使用して、レコードのフィールド位置を自由に決めることができます。RCD ステートメントのあとに続く DFLD は、新規レコードの最初のユーザー・データ位置で始まります。フィールドをレコードに入れるときは、フィールドがレコード境界をまたがらないようにしたり、論理的に関連性のあるフィールドを同一レコード内にまとめて入れたりすることができます。

制約事項: ISC サブシステムでは、フィールドが複数のレコードにまたがることはできません。

最大長のレコードを作成する場合にのみ、RCD ステートメントでレコード境界を設定して、送信時間および IMS 処理時間を短縮することができます。RCDCTL の指定だけでレコードにフィールドを入れていく場合は、SPAN オプションを指定することにより、最小限必要なレコードだけをリモート・プログラムに送信することができます。ただし、SPAN を指定するときは、複数のレコードに分割されて入っているフィールドを、リモート・プログラムで 1 つにまとめなければなりません。

プログラム式シンボル・バッファーのロード

プログラム式シンボル (PS) バッファーが必要なときに、他のなんらかの手段 (例えば、VTAM アプリケーション) によってまだロードが行われていない場合は、バッファーにロードを行わなければなりません。

プログラム式シンボル・バッファーがロードされているかどうかをアプリケーション・プログラムを使用して判別する方法:

必要なプログラム式シンボルは、すでに前の使用者によって装置のバッファーにロードされていることがあります。そのことがわかっていれば、プログラム式データ・ストリーム全体を再送信する手間が省けます。あとの使用者のために、装置のプログラム式シンボル・バッファーの現在の状況を保存する 1 つの方法は、手書きで記録することです。

別の方法としては、ユーザー作成のアプリケーション・プログラムで所定のプログラム式シンボルを使用してみることがあります。必要なプログラム式シンボルがすでにロードされていれば、装置にアプリケーション・プログラムからの出力が正常に表示されます。必要なプログラム式シンボルがすでにロードされていれば、装置にアプリケーション・プログラムからの出力が正常に表示されます。プログラム式シンボルがロードされていないと、IMS メッセージ・キューに出力メッセージが返されて、端末が操作不能になり、マスター端末オペレーター (MTO) にメッセージが送られます。MTO では、この状況を訂正するための手順が必要です。例えば、MTO は次のいずれかの操作を行うことができます。

- LTERM の再割り当てを行い、正しい PS ロード・メッセージを持つ LTERM を割り当て、端末を再始動させて、その後で最初の LTERM を端末に割り当て直す。
- 端末に PS 機能が備わっていなければ、その機能を持つ端末に LTERM を再度割り当てる。
- 端末に PS 機能が備わっていなければ、拒否されたメッセージをデキューする。

例外: SLU 2 端末で拒否された出力が、応答モードでの応答はなかった場合です。この場合、MTO はエラー・メッセージを受け取り、バッファーをロードするためのトランザクションを入力することができます。

プログラム式シンボル・バッファーのロード方法:

オペレーターは、(装置をオンにしたばかりのため) プログラム式シンボル・バッファーをロードする必要があるとわかった時点で、プログラム式シンボルをロードする応答モードのトランザクションを入力しなければなりません。

インストール・システムのすべてのユーザーが使用できるように、プログラム式シンボルをロードするユーザー作成のアプリケーション・プログラムを用意しておく必要があります。このアプリケーション・プログラムによって送信されるメッセージの最初の部分は、プログラム式シンボル・データ・ストリームでなければならず、装置に表示されるユーザー・データ (例えば、THE PROGRAMMED SYMBOL LOAD FOR *programmed-symbol-name* COMPLETE) はそのあとに入っていなければなりません。プログラム式シンボルがロードされると、このユーザー・データが装置に表示され、端末オペレーターにロードの完了が通知されます。IMS は、プログラム式シンボル・メッセージの送信に使用される構造化フィールド書き込み (WSF) 3270 コマンドを MFS バイパス・オプションを使用した場合にサポートします。このため、このアプリケーション・プログラムでは、MFS バイパス・オプションを使用してください。

ロードが行われるプログラム式シンボル・バッファーに、プリンターまたは別のディスプレイのバッファーも含まれている場合は、他の方法を使用しなければなりません。 BSC 接続装置では、プログラム式シンボル・バッファーのロードには 3 KB という制約があります。

例えば、以下で、自動化操作プログラム・インターフェース (AOI) のアプリケーション・プログラムを使用するプログラム式シンボル・バッファーのロードを示します。

1. ディスプレイ A のオペレーターが、ディスプレイ A、プリンター B、およびディスプレイ C に、プログラム式シンボルをロードするよう要求するトランザクション (応答または会話型) を入力します。
2. 別の AOI トランザクションは、プリンター B とディスプレイ C の LTERM を、一時的に特定の PTERM に割り当てます。 AOI プログラムは、プリンター B とディスプレイ C にダミーの LTERM を割り当てます。
3. AOI プログラムは、プリンター B とディスプレイ C のダミーの LTERM にプログラム式シンボル・メッセージを挿入します。
4. AOI プログラムは、ディスプレイ A にプログラム式シンボル・メッセージを送ります。
5. オペレーターは、ディスプレイおよびプリンターの両方に出力されたメッセージを見て、トランザクションが正しく実行されたかどうかを確認します。
6. 別の AOI トランザクションで、LTERM を元の割り当て状態に戻します。

プログラム式シンボルのロード時における問題の解決方法:

プログラム式シンボル・バッファーのロード中にハードウェア・エラーが発生すると、次のアクションがとられます。

1. プログラム式シンボル・ロード・メッセージは、IMS メッセージ・キューに戻されます。
2. プログラム式シンボルの使用できない端末は、SLU 2 装置を除いて、サービス休止になります。
3. このエラーは、IMS ログに記録されます。
4. IMS マスター端末にメッセージが送られます。

ハードウェア・エラーが訂正され、端末がサービス中になれば、プログラム式シンボル・ロード・メッセージが再送信されます。

プログラム式シンボル・ロード・メッセージのエラーにより、プログラム式シンボルがロードできなかった場合、オペレーターは次のアクションを取る必要があります。

1. /DEQ によってメッセージをキューから取り出します (マスター端末オペレーターが /DEQ コマンドを出してください)。
2. エラーを訂正します。
3. 再度トランザクションを入力して、プログラム式シンボル・ロード・メッセージを再送信します。

プログラム式シンボル・バッファーの状況を次の使用者に知らせる手段がある場合は、プログラム式シンボル・バッファーがロードされている端末をオフにしてはなりません。電源障害が発生したり、端末がオフにされると、プログラム式シンボル・バッファーの内容は失われます。

端末がオンのときに、ディスプレイへの送信待ちの IMS メッセージがない場合は、IMS トランザクション (または IMS 以外の方式) によって必要なプログラム式シンボル・バッファーすべてをロードしてください。ただし、送信待ちの IMS メッセージが存在していて、しかもこれらのメッセージが 1 つ以上のプログラム式シンボル・バッファーを使用する必要があるときは、プログラム式シンボル・バッファーを再ロードするまで、キューに入れられたメッセージの送信を延期しなければなりません。これを行うには、応答モードのトランザクションを使用してプログラム式シンボル・バッファーをロードします。

プログラム式シンボル・バッファーがロードされていないにもかかわらず、プログラム式シンボル・バッファーを使用するメッセージが端末に送信されると、次のようなアクションが取られます。

- SLU 2 以外の装置の場合、IMS はその端末をサービス休止にし、マスター端末にメッセージが送信され、出力メッセージをメッセージ・キューに戻します。
- SLU 2 装置の場合、メッセージは拒否され、センス・コードが IMS に返されます。IMS は、次に以下の処理を行います。
  - 無効なメッセージを IMS キューに戻す。
  - IMS ログにエラーを記録する。
  - その出力が応答モードでの応答であった場合は、IMS マスター端末にエラー・メッセージを送って端末をサービス休止にする。応答モードでない場合は、エラー・メッセージを端末に送って、端末を保護モードのままにしておきます。

ユーザー作成のアプリケーション・プログラムが、ある LTERM に特定のプログラム式シンボル・ロード・バッファーを必要とする非送信請求メッセージをキューに入れるために設計されている場合、メッセージの最初の部分に、ロードのためのプログラム式シンボル・データ・ストリームを入れることができますが、このメッセージは MFS で処理することはできません。

ある端末の IMS キュー内にプログラム式シンボルを使用するメッセージが待機していて、そのプログラム式シンボルがまだ端末にロードされていない場合は、次のいずれかの処理を実行してください。

- VTAM 接続の端末であれば、VTAM アプリケーションを使用してプログラム式シンボル・バッファーをロードしてください。
- プログラム式シンボル・バッファーを必要とするキューに入れられたメッセージが、『通常の』ユーザー出力 (例えば、応答モードでも会話型でもない出力) ならば、プログラム式シンボル・バッファーをロードする応答モードのトランザクションを実行することでロードが行われ、待機中のメッセージは正しく表示されます。バッファーのロードに複数のメッセージが必要な場合は、複数の応答モード・トランザクションを使用することができます。
- プログラム式シンボル・バッファーを必要とするメッセージをデキューします (/DEQ) (または、マスター端末オペレーターにメッセージのデキューさせま

す)。次に、必要なプログラム式シンボル・バッファーをロードするトランザクションを入力してから、最初にキューに入れられたメッセージを生成したトランザクションを再入力します。

- メッセージがキューに入れられた LTERM を、一時的に別の物理端末に割り当てます。次に、プログラム式シンボル・バッファーをロードします。その後で、LTERM を元の物理端末に割り当て直します。一時的な割り当てではあっても、メッセージ送信の起こらない端末 (例えば、保護画面モードにある 3270 表示装置または SLUTYPE2) には、LTERM を割り当てる必要があります。

システム間連絡のための MFS 定義

IMS と CICS との間のシステム間連絡 (ISC) システムでは、以下のプロトタイプ MFS 定義を使用することができます。

この例では、以下のことを想定しています。

- CICS は、MFS 編集を要求するときに、8 バイトまたは 4 バイトの名前を使用できる。
- CICS からのメッセージは、複数ページ入力でも単一ページ入力でもかまわない。
- CICS への出力は、1 つ以上のセグメントで構成された、1 ページまたは複数ページからなる 1 つのメッセージにすることができる。
- 要求時ページ単位出力または自動ページ化出力を、IMS から要求することができる。

IMS メッセージ通信を使用して CICS へメッセージを送信する場合にも、3270 端末オペレーターはこれらの形式を使用することができます。また、例えば 3270 端末オペレーターが IMS メッセージ通信を呼び出してメッセージを CICS へ渡し、CICS から IMS へ、さらに 3270 端末へと応答を返すことができます。このときの経路指定は、MFS が行います。次の例は、MFS 定義形式を示します。

```
FMTDIS    FMT
          DEV    TYPE=3270-A2,FEAT=IGNORE
          DIV    TYPE=INOUT
DFLDIND1  DFLD   LTH=5,POS=(1,2)
DFLDIND2  DFLD   LTH=100,POS=(1,8)
          FMTEND
FMTDP2    FMT
          DEV    TYPE=DPM-B1,FEAT=IGNORE,
                 MODE=RECORD,DSCA=X'00A0'                        X
          DIV    TYPE=OUTPUT,OPTIONS=(MSG,NODNM)
PPAGE1    PPAGE
DFLDOUT1  DFLD   LTH=5
DFLDOUT2  DFLD   LTH=100
          FMTEND
FMTDPM    FMT
          DEV    TYPE=DPM-B1,FEAT=IGNORE,MODE=RECORD
          DIV    TYPE=INPUT,OPTIONS=(DPAGE,NODNM),                    X
                 PRN=DFLDINP3,                                        X
                 RDPN=DFLDINP4,                                       X
                 RPRN=DFLDINP5
PPAGE2    PPAGE
DFLDINP1  DFLD   LTH=5
DFLDINP2  DFLD   LTH=100
          DIV    TYPE=OUTPUT,OPTIONS=(DPAGE,NODNM)
DPAGE2    DPAGE
DPAGE3    PPAGE
```

```
DFLDOUT3  DFLD   LTH=5
DFLDOUT4  DFLD   LTH=100
          DFLD   SCA,LTH=2
          FMTEND
MFSMOD1   MSG    TYPE=OUTPUT,SOR=(FMTDP2,IGNORE),                    X
                 NXT=MFSMID1
          SEG
          MFLD   DFLDOUT1,LTH=5
          MFLD   DFLDOUT2,LTH=100
          MSGEND

MFSMODE2  MSG    TYPE=OUTPUT,SOR=(FMTDPM,IGNORE),                    X
                 NXT=MFSMID1
          SEG
          MFLD   DFLDOUT3,LTH=5
          MFLD   DFLDOUT4,LTH=100
          MFLD   (,SCA),LTH=2
          MSGEND
MFSMID1   MSG    TYPE=INPUT,SOR=(FMTDPM,IGNORE),                     X
                 NXT=MFSMODD
          SEG
          MFLD   DFLDINP1,LTH=5
          MFLD   DFLDINP3,LTH=8
          MFLD   DFLDINP4,LTH=8
          MFLD   DFLDINP5,LTH=8
          MFLD   DFLDINP2,LTH=100
          MSGEND
MFSMIDD   MSG    TYPE=INPUT,SOR=(FMTDIS,IGNORE),                     X
                 NXT=MFSMOD1
          SEG
          MFLD   DFLDIND1,LTH=5
          MFLD   DFLDIND2,LTH=100
          MSGEND
MFSMODD   MSG    TYPE=INPUT,SOR=(FMTDIS,IGNORE),
                 NXT=MFSMIDD
          SEG
          MFLD   DFLDIND1,LTH=5
          MFLD   DFLDIND2,LTH=100
          MSGEND
          END
```

MFS メッセージ形式

メッセージ形式サービス (MFS) を使用してアプリケーション・プログラムを装置と通信させる場合は、この章にあるトピックを参照してください。

入力メッセージ形式

MFS は、MFS アプリケーション設計者が MFS 言語ユーティリティー・プログラムへの入力として作成したメッセージ定義に基づいて、装置からの入力データを IMS アプリケーション・メッセージの形式に編集します。入力メッセージは、IMS アプリケーション・プログラムが DL/I GU 呼び出し、または GN 呼び出しを発行したときにプログラムに渡されるすべてのセグメントで構成されています。

入力メッセージの形式は、MFS 言語ユーティリティーに対して定義しておきます。以下のように、メッセージは 1 つ以上のセグメントから構成され、各セグメントは 1 つ以上のフィールドから構成されています。

```
MESSAGE
    SEGMENTS
        FIELDS
```

メッセージのフィールド形式は、データ・ソース、フィールド長、位置調整、切り捨て、充てん (埋め込み) 文字の使用について、ユーティリティーに明確に定義されます。ただし、MFS がそのフィールドを実際にどのようにフォーマット設定するかは、そのメッセージでどのフォーマット設定オプションが選択されるかによって決まります。選択されたオプションは、メッセージ・テキストに先行する 2 バイトの ZZ フィールド (Z2) の第 2 バイトで示されます。MFS に依存するアプリケーション・プログラムは、このフィールドを検査して、所定のオプションが使用されていることを確認しなければなりません。Z2 フィールドの X'00' は、MFS がメッセージを形式設定しなかったことを示します。

論理ページ

3270 または SLU 2 からの入力メッセージは、現在装置に表示されている DPAGE から作成されます。その他の装置では、装置入力形式が 2 個以上の DPAGE で定義されている場合、入力された装置データによって、どの入力 LPAGE を使用して入力メッセージを作成するかを決定します。ただし、ISC (DPM-Bn) サブシステムでは、OPTIONS=DNM または COND= を使用して、DPAGE を選択することができます。

LPAGE を定義するときは、LPAGE はそれぞれ 1 つ以上の DPAGE と関連付けられます。

関連資料:

544 ページの『入力メッセージ・フォーマット設定オプション』

装置に依存する入力情報 (3270 または SLU 2)

情報を入力する際にあるオプションを使用すると、アプリケーション・プログラムは装置依存型のプログラムになります。

カーソル位置

MFS 言語ユーティリティーのオプションとして、入力が IMS に送信された時点での装置上のカーソル位置を、メッセージのフィールドに入れておくことができます。このオプションを使用できるのは 3270 または SLU 2 表示装置のみです。また、このオプションを使用するということは、プログラムが装置依存型のプログラムになることを意味します。カーソル情報の形式は 2 個の 2 バイト 2 進数からなり、最初の 2 進数は行番号を、次の 2 進数は桁番号を表します。この行および桁の最小値は 1 です。3270-An 装置タイプの場合、SIZE= オペランドの最初のパラメーターが行の最大値です。桁の最大値は、SIZE= オペランドの 2 番目のパラメーターです。

以下の表は、MFS 装置タイプの行および桁の最大値をリストします。

表 *117.* 行および桁の最大値 *(3270* 装置タイプの場合*)*

	最大値	
MFS 装置タイプ	行	桁
3270,1	12	40
3270,2	24	80
3270-An		

表 *117.* 行および桁の最大値 *(3270* 装置タイプの場合*) (*続き*)*

	最大値	
MFS 装置タイプ	行	桁
SIZE=(12,40)	12	40
SIZE=(12,80)	12	80
SIZE=(24,80)	24	80
SIZE=(32,80)	32	80
SIZE=(43,80)	43	80
SIZE=(27,132)	27	132
SIZE=(62,160)	62	160

選択ペン

選択ライト・ペンを使用すると、以下のように、入力フィールドにさまざまな形で影響が現れます。

- ATTR 出力フィールド・オプションを動的に使用せずに、検出可能フィールドを作成した場合
  - DET 属性、STRIP 属性が定義されている装置フィールドを参照するメッセージ・フィールドは、装置独立フィールドとして渡されます。
  - そのメッセージ・フィールドに入る最初のデータ・バイトは、装置フィールドの指示文字の次のバイトです。
  - IDET 属性、STRIP 属性が定義されている装置フィールドを参照するメッセージ・フィールドにも、装置独立データが入ります。
  - 指示文字が除去されます。
  - このフィールドが選択されたときに装置上で変更されたフィールドがない場合は、このフィールドのデータは渡されません。この場合、メッセージに組み込まれる装置情報は、DFLD ステートメントの PEN= オペランドで リテラル に指定される値だけです。
- ATTR 出力フィールド・オプションを動的に使用して、検出可能フィールドを作成した場合
  - 据え置き検出可能または即時検出可能フィールドとして動的に設定されたフィールドでは、入力から指示文字が取り除かれません。
  - 即時検出可能に変更されたフィールドを選択しても、装置上で他のフィールドがまったく変更されていない場合は、装置入力データはメッセージに組み込まれません。
- 即時検出選択ペンのリテラル・データを受け取るメッセージ・フィールドを定義すると、次のいずれかが起こります。
  - 選択ペンによる即時検出の結果として装置入力が生じたのでなければ、そのメッセージ・フィールドには要求どおりの埋め込みが行われます。

- 選択ペンによる即時検出の結果として装置入力が生じ、しかも装置上の少なくとも 1 つのフィールドが変更されている場合、 1 個のデータ文字としての疑問符 (?) に要求どおりの埋め込みが行われて、メッセージ・フィールドに渡されます。
- 選択ペンによる即時検出の結果として装置入力が生じ、しかも他のどのフィールドも変更されていない場合は、検出されたフィールドに PEN= リテラルが定義してあれば、そのリテラルは要求どおりにメッセージに組み込まれます。検出されたフィールドに、PEN= リテラル が定義されていなければ、そのメッセージ・フィールドには 1 個のデータ・バイトとして疑問符 (?) が入ります。いずれの場合も、他の装置情報がメッセージに組み込まれることはありません。

- ライト・ペン検出可能フィールドに EGCS 属性を定義する場合は、DFLD ステートメントに ATTR=NOSTRIP を指定したうえで、入力データから 2 個の指示文字をバイパスまたは除去するようにアプリケーションを設計する必要があります。 ATTR=STRIP を指定するか、デフォルトとして使用する場合、MFS は最初の指示文字だけを除去するため、フィールド内の最後のデータ文字が切り捨てられます。

磁気ストライプ読取装置

磁気ストライプ読取装置の使用は、アプリケーション・プログラムには透過的です。オペレーター ID (OID) カード読取装置については、囲み文字 (SOR、EOR、EOI、LRC) が取り除かれ、パリティー検査が行われてから編集が行われます。

PF キー

PF キーは、使用されても、アプリケーション・プログラムには透過であり、認識されません。

PF キー

アプリケーション・プログラムでは、プログラム・アクセス・キー情報を使用することはできません。

出力メッセージ形式

MFS は、IMS アプリケーション・プログラムが作成した出力セグメントを、メッセージの宛先となっている装置またはリモート・プログラムに適した所定の装置形式に編集します。

通常、IMS アプリケーション・プログラムから渡される出力セグメントに、装置関連データは含まれていません。装置またはリモート・プログラムへ出力を送るのに必要なすべての情報は、MFS 言語ユーティリティー・プログラムに対してメッセージ形式を定義するときに指定しておきます。 DPM を使用するリモート・プログラムは、MFS では変換処理されない特定の装置依存情報を用意します。

1 つの出力メッセージは、入出力 PCB に対する GU 呼び出しが出されてから、PURG 呼び出しまたは入出力 PCB に対する別の GU 呼び出しが出されるまで、あるいは正常にプログラムが終了するまでの間に ISRT 呼び出しによって IMS に渡されたすべてのセグメントから構成されます。

出力メッセージの形式は、入力メッセージの形式と同様に MFS 言語ユーティリティーに対して定義しておきます。どのメッセージも 1 つ以上のセグメントから構成され、各セグメントは 1 つ以上のフィールドから構成されています。

```
MESSAGE
    SEGMENTs
        FIELDs
```

論理ページ

出力セグメントについては、論理ページ (LPAGE ステートメント) を定義することによって、フォーマット設定のためにグループ化しておくこともできます。

```
MESSAGE
    LPAGEs
        SEGMENTs
            FIELDs
```

LPAGE を定義するときは、各 LPAGE を、論理ページ用の装置形式を定義している特定の DPAGE と関連付けます。LPAGE が定義されていなければ、MFS は定義されたメッセージを 1 つの LPAGE と見なし、単一 LPAGE メッセージに適用される規則を適用します。その規則は以下のとおりです。

メッセージが、1 つのセグメントからなる 1 つの LPAGE で構成されている場合、アプリケーション・プログラムによって挿入されるどのセグメントも、同じ方法で編集されます。

メッセージが、複数個のセグメントからなる 1 つの LPAGE で構成されている場合、各メッセージ・セグメントは、それぞれ定義されている順番どおりに挿入しなければなりません。LPAGE のすべてのセグメントを IMS に渡す必要はありませんが、いずれかのセグメントを省略する場合には、十分に注意を払う必要があります。オプション 1 または 2 のセグメントは、LPAGE の終わりまでのすべてのセグメントを省略する場合にのみ省略できます。それ以外の場合は、セグメント位置を表すためにヌル・セグメントを挿入する必要があります。オプション 3 では、出力メッセージ・セグメントを省略することができますが、送信するセグメントにはセグメント番号 ID を入れなければなりません。

複数の連続セグメントを 1 つの出力メッセージとして IMS へ渡すことができます。LPAGE が N 個のセグメントからなると定義されている場合、セグメント N+1 はセグメント 1 であるとして編集されます。ただし、セグメント N+1 よりも前にあるセグメントの Z2 フィールドに、ページ・ビット (X'40') がセットされていれば、それがセグメント 1 と見なされます。いくつかのセグメントを省略したうえで、複数の連続出力セグメントを IMS に渡すときは、連続出力セグメントの先頭にくるセグメントの Z2 フィールドのビット 1 (X'40') をオンにしておかなければなりません。

メッセージに複数の LPAGE がある場合、連続するセグメントの中の先頭のセグメント内のデータによって連続するセグメントがどの LPAGE に属しているかが決定され、それによって、セグメントで実行される編集方法が決定されます。連続セグメントの第 1 セグメントから、使用する LPAGE を決定できないときは、最後に定義されている LPAGE が使用されます。セグメントの省略に関する規則も同じです。メッセージの Z2 フィールドのビット (X'80') は、アウトバウンド・データ・ストリームの中に構造化データがあることを表します。構造化データを使用した出

力メッセージは、MODNAME をブランクまたは 2 進ゼロとして定義するか、あるいは MFS バイパス機能を使用しなければなりません。

セグメント形式

どの出力メッセージにも 4 バイトの接頭部があります。この接頭部はセグメントの長さを定義し、また、必要であれば LPAGE を構成する一連のセグメントの最初のセグメントであるかどうかを指定します。

オプション 3 出力メッセージの場合は、さらに、LPAGE を構成する一連のセグメント内の相対セグメント番号を示す 2 バイトを加えなければなりません。以下の表は、出力セグメントの形式を示しています。

表 *118.* 出力セグメントの形式

LL	Z1	Z2	SN	FIELDS

ここで、

LL メッセージ・セグメントの全長を表す 2 バイトの 2 進数フィールドです。全長とは、LL、Z1、Z2、および SN (存在する場合) の長さも含めた、メッセージ・セグメント全体の長さです。LL の値は、テキスト内のバイト数 (セグメントのすべてのフィールド) に 4 (オプション 3 であれば 6) を加えた値です。アプリケーション・プログラムがこの値を入れなければなりません。処理中のトランザクションの出力セグメントのサイズに制限がある場合、LL はその制限値を超えてはなりません。

制約事項**:** セグメント長は、IMS システム定義時に定めたメッセージ・キュー・バッファーのデータ部のサイズ (バッファー・サイズ 接頭部のサイズ) よりも短くなければなりません。また、MFS 言語ユーティリティーに対して定義したセグメント長より短くすることも可能です。MFS ユーティリティーに対して定義したセグメントより長いセグメントを挿入すると、セグメントは途中で切り捨てられてしまいます。エラー・メッセージは出されません。切り捨てまたは省略されたフィールドは、MFS 言語ユーティリティーで定義した形式定義どおりに埋め込みが行われます。

PL/I を使用するときは、LL フィールドを 2 進フルワードとして定義する必要があります。 PL/I アプリケーション・プログラムでは、実際のセグメント長から 2 バイトを差し引いた値をそのフィールドに入れなければなりません。 例えば、出力メッセージ・セグメントが 16 バイトの場合は LL=14 です。これは LL の長さ (4 バイト - 2 バイト) + Z1 (1 バイト) + Z2 (1 バイト) + TEXT (10 バイト) の合計になります。

Z1 2 進数のゼロが入る 1 バイトのフィールドで、IMS が使用するために確保されています。

Z2 1 バイトのフィールドで、出力装置のさまざまな機能を制御するために、アプリケーション・プログラムで使用することができます。

このフィールドの詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

SN オプション 3 でのみ使用することができます。2 バイトの 2 進数フィールドで、LPAGE 内のセグメントの相対セグメント番号を入れます。最初のセグメントの相対番号は 1 です。

LPAGE 内のオプション 1 または 2 の出力の一連のセグメントの位置を維持するため、ヌル・セグメントを使用することができます。LPAGE を構成する一連のセグメントの中間部分にあるセグメントを省略するときは、必ずヌル・セグメントを使用しなければなりません。LPAGE を構成する一連のセグメントの途中から最後までを省略するときは、ヌル・セグメントは不要です。ヌル・セグメントは長さが 5 バイトで、1 個のデータ・バイト (X'3F') だけが入っています。

次の例は、COBOL でのヌル文字のコーディング方法を示しています。

COBOL でのヌル文字のコーディング

```
ID DIVISION.
PROGRAM-ID. SAMPLPGM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  PART1 PIC 9(3) VALUE 123.
77  CUR-NAME PIC 99 COMP VALUE 0.
77  CUR-PART PIC 99 COMP VALUE 0.
01     NULLC.
    02 FILLER PIC 9 COMP-3 VALUE 3.
01  LINE-A.
    02 NAME-1.
       03 NAME-2 OCCURS 30 PIC X.
    02 PARTNUM.
       03 PARTNUM1 OCCURS 10 PIC 9.
PROCEDURE DIVISION.
    MOVE ''ONES' TO NAME-1.
    MOVE 6 TO CUR-NAME.
    MOVE NULLC TO NAME-2 (CUR-NAME).
    MOVE 4 TO CUR-PART.
    MOVE NULLC TO PARTNUM1 (CUR-PART).
```

フィールド形式 (オプション 1 および 2)

オプション 1 および 2 の出力セグメントでは、すべてのフィールドが固定長でしかも固定位置にあるものとして定義されています。

フィールドのデータは、次の 2 通りの方法で切り捨ておよび省略を行うことができます。

- 短セグメントを挿入する方法
- フィールド内にヌル文字 (X'3F') を挿入する方法

各フィールドは左から右に走査されてヌル文字の有無が調べられます。最初のヌル文字が検出されると、そこでそのフィールドは終わりになります。フィールドの第 1 文字をヌル文字にすると、そのフィールドは省略されることになります (これは、使用する充てん文字次第です)。ヌル文字を使用するかしないかに関係なく、セグメント内のすべてのフィールドの配置は変りません。切り捨てまたは省略されたフィールドは、MFS 言語ユーティリティーで定義した形式定義どおりに埋め込みが行われます。

MFLD 定義に ATTR=YES が指定されており、さらにフィールドの属性部分の第 1 または第 2 バイトが X'3F' であれば、そのフィールドは省略され、DFLD ステートメントで指定した属性が使用されます。

関連概念:

568 ページの『出力メッセージ・フォーマット設定オプション』

フィールド形式 (オプション 3)

オプション 3 の出力では、セグメントの大きさに収まりさえすれば、各フィールドをその所属セグメントに任意の順番および長さで配列することができます。

短フィールドまたは省略フィールドへの埋め込みは、MFS 言語ユーティリティーに対して定義されたとおりに行われます。各フィールドの前には、4 バイトのフィールド接頭部を付けなければなりません。このフィールド接頭部の形式は、以下の表に示すように、オプション 3 入力フィールドに対して MFS が用意する接頭部の形式と同じでなければなりません。

表 *119.* オプション *3* 出力セグメントの形式

FL	FO	DATA

ここで、

FL フィールドの長さ。4 バイトのフィールド接頭部を含めた長さです。 FL は 2 バイトの 2 進数からなり、位置合わせは必要ありません。

FO セグメントのフィールドの相対オフセット。MFS 言語ユーティリティーに対するメッセージ定義に基づいています。FO は 2 バイトの 2 進数からなり、位置合わせは必要ありません。セグメント内に定義した最初のフィールドの相対オフセットは 4 です。また、2 番目のフィールドの相対オフセットは、最初のフィールドの長さ (MFS 言語ユーティリティーに対して定義した長さ) に 4 を加えた値です。

FL および FO の内容にエラーがあった場合には、予測できない結果になります。

オプション 3 を使用する場合、出力セグメントでフィールドを特定の順序に並べる必要はありませんが、セグメントのすべてのフィールドは連続していなければなりません。つまり、2 番目のフィールドのフィールド接頭部は、最初のフィールドのデータの終わりの次のバイトから始まっていなければなりません。オプション 3 のフィールドにヌル文字が入っていても、装置に送信されるデータには何の影響もありません。ヌル文字は他の非図形文字と同様に扱われ、ブランクに置き換えられます。

出力メッセージ・フィールドに装置制御文字を入れても無効です。 3270 表示装置および SLU 2 端末では、制御文字 HT、CR、LF、NL、および BS はヌル文字に変更されます。その他のすべての装置では、上記の制御文字はブランクに変更されます。他のすべての非図形文字 (X'00' から X'3F'、 および X'FF') はブランクに変更されてから送信されます。DPM 装置では、GRAPHIC=NO が指定してあれば、制御文字を入れることができます。

関連概念:

568 ページの『出力メッセージ・フォーマット設定オプション』

装置依存の出力情報

情報を出力する際にあるオプションを使用すると、そのアプリケーション・プログラムは装置依存型のプログラムになります。出力を受け取る装置の特定機構を、アプリケーション・プログラムで制御できるオプションもあります。次に、さまざまな出力オプションの効果について説明します。

システム制御域 (SCA)

MFS 言語ユーティリティーのオプションの 1 つとして、メッセージの先頭のセグメント (LPAGE が定義されている場合は、いずれかまたはすべての LPAGE の先頭セグメント) に SCA フィールドを作成する機能があります。また、アプリケーション・プログラムは、このフィールドを使用することによって、特定の装置機構を制御することができます (ただし、そのメッセージの宛先の装置でその特定機構が使用されている場合)。 SCA フィールドの先頭の 2 バイトは、以下の表に示すように定義されます。

使用上の注意は、以下の両方の表にあるとおりです。

表 *120. TYPE=3270*、*SLU 2*、*DPM-An*、または *DPM-Bn* に有効なバイトおよびビット

バイト	ビット	説明
0	0 から 7	0 でなければなりません。
1	0	1 でなければなりません。
	1	強制フォーマット書き込み (装置バッファーを消去し、すべての必要データを書く)
	2	書き込み前に無保護フィールドを消去
	3	装置アラームを作動
	4	出力を候補プリンターにコピー
	5	B'0' の場合、3270 ならば、出力の送信時に画面を保護します。DPM では、要求時ページングを実行できます。
		B'1' の場合、3270 ならば、出力の送信時に画面を保護しません。DPM-B では、自動ページングを実行できます。
	6	区画形式の 3290 の場合、B'0' ならば、既存の区画を消去しません。B'1' ならば、すべての区画を消去してからメッセージを送信します。 その他の装置については、0 でなければなりません。
	7	0 でなければなりません。

表 *120. TYPE=3270*、*SLU 2*、*DPM-An*、または *DPM-Bn* に有効なバイトおよびビット *(*続き*)*

バイト	ビット	説明

注:

1. 区画形式モードの 3290 では、現行メッセージの DOF が最後に実行された DOF と同じかどうかが検査されます。同じであれば、出力メッセージに送信前に全区画を消去するかしないかが、SCA および DSCA オペランドのビット 6 の値で判別されます。
2. ビット 6 のデフォルトは B'0' で、『区画は消去されません』。 このビットの指定がなければ、出力は区画ページング・オプションの指定に従って送信されるので、出力を受信しない区画はそのままです。
3. ビット 6 が B'1' に設定されていると、既存の区画を消去してから、区画ページング・オプションの指定に従って出力が送信されます。
4. SCA のビット設定値と DSCA のビット設定値とで『論理和』が形成されます。例えば、DPM-B の DSCA でバイト 1 のビット 5 が B'0' に設定されていても、アプリケーション・プログラムは SCA 値を B'1' に設定することで、自動ページ化出力を要求することができます。(この要求が受け入れられるのは、出力メッセージの最初の LPAGE の最初のセグメントに、この要求を渡したときだけです。)
5. SCA 情報は、パラメーター SCA で指名した DFLD に入れられて、リモート・プログラムまたは ISC サブシステムへ送られます。装置タイプにとって無効なビットはすべてリセットされます。有効なビットは使用されます。

表 *121. TYPE=FIDS*、*FIDS3*、*FIDS4*、*FIDS7*、*FIJP*、*FIPB*、*FIFP* に有効なバイトおよびビット

バイト	ビット	説明
0	0 から 7	0 でなければなりません。
1	0	1 でなければなりません。
	1-2	FIN 出力装置には適用されません。
	3	出力メッセージ・ヘッダー内に「装置アラーム」をセットします。
	4	FIN 出力装置には適用されません。
	5 から 7	0 でなければなりません。

注:

1. ビット 1、2、および 4 は、3270 でのみ機能し、金融機関ワークステーションには適用されません。プログラムがこれらのビットをオンにセットしても、金融機関ワークステーション用にメッセージが編集される場合には、これらのビットは無視されます。
2. TYPE=SCS1、または SCS2 の場合、SCA パラメーターは無視されます。
3. TYPE=3270P の場合、「装置アラームのセット」以外のビットはすべて無視されます。

カーソル位置

アプリケーション・プログラムは、フィールドにカーソル属性をセットするか、または出力メッセージに特別のカーソル位置付けフィールドを入れることによって、画面上のカーソル位置を設定することができます。

推奨事項: カーソル属性を使用してください。これを使用すると、アプリケーション・プログラムが装置上のフィールドの位置を知る必要がないためです。

MFS 言語ユーティリティーのオプションを使用すると、出力セグメントの中にフィールドを定義することができます。このフィールドによって、アプリケーション・プログラムからカーソルを装置上の特定の行と桁に置くよう要求することができます。使用する装置出力形式によって、LPAGE あたり 1 つ以上のカーソル位置付けフィールドを設けることができます。フィールドに無効が数字が入っていると、フィールド自体が無視され、カーソルの位置が装置出力形式で要求された位置になります。

カーソル・フィールドには、2 バイトの 2 進数が 2 個入っていなければならず (位置合わせは不要)、最初の 2 バイトは行番号、2 番目の 2 バイトは桁番号をそれぞれ表しています。この行および桁の最小値は 1 です。3270-An 装置タイプの場合、SIZE= オペランドの最初のパラメーターが行の最大値です。桁の最大値は、SIZE= オペランドの 2 番目のパラメーターです。以下の表は、行および桁の有効な値を示しています。

表 *122.* 行および桁の最大値 *(MFS* 装置タイプの場合*)*

	最大値	
MFS 装置タイプ	行	桁
FIDS (240 文字)	6	40
FIDS3 (480 文字)	12	40
FIDS4 (1024 文字)	16	64
FIDS7 (1920 文字)	24	80
3270,1 (480 文字)	12	40
3270,2 (1920 文字)	24	80
3270-An		
SIZE=(12,40) (480 文字)	12	40
SIZE=(12,80) (960 文字)	12	80
SIZE=(24,80) (1920 文字)	24	80
SIZE=(32,80) (2560 文字)	32	80
SIZE=(43,80) (3440 文字)	43	80
SIZE=(27,132) (3564 文字)	27	132
SIZE=(62,160) (9920 文字)	62	160

関連概念:

611 ページの『区画形式モードの 3290』

595 ページの『ISC (DPM-Bn) サブシステムの出力形式制御』

関連資料:

544 ページの『入力メッセージ・フォーマット設定オプション』

『動的属性変更』

動的属性変更

MFS 言語ユーティリティーには、IMS アプリケーション・プログラムに装置フィールドの属性を動的に変更、置換、またはシミュレートさせることができるオプションがあります。

この動的属性変更を使用するためには、出力メッセージ定義の MFLD ステートメントで ATTR=YES を指定しておかなければなりません。これにより MFS は、出力メッセージ・フィールドのデータ・バイトのうち、最初の 2 バイトを属性定義用として確保します。この 2 バイトのデータの指定にエラーが検出された場合、あるいは第 1 あるいは第 2 の属性バイトが X'3F' である場合、以下の図に示すような結果になります。

データが送信されない場合でも、属性は必ず送信されます。

事前定義属性付きの装置フィールドに動的属性変更を指定すると、出力操作のたびに、そのフィールドの属性が装置に送信されます。これは、装置フィールドのデータが出力メッセージ内に入っていない場合でも送信されます。以下の属性が使用されます。

- 出力メッセージ・フィールドに属性が備わっており、その属性が有効な場合は、動的属性変更が行われる。
- メッセージ・フィールドが使用中の LPAGE に含まれていない場合、またはそのフィールドの属性が有効でない場合は、その装置フィールドの事前定義属性が使用される。

属性シミュレーションを指定すると、装置フィールドの最初のバイトが属性データ用として確保されます。シミュレート可能な属性は、次のものです。

- カーソル位置 (3604 表示装置のみ)
- 表示不能
- 高輝度表示
- 変更済み属性

2 つの属性バイトは、以下の表で定義されているとおりです。

表 *123. 2* つの属性バイトの定義：

バイト	ビット	定義
0	0 から 1	この 2 つのビットを両方ともオンにすると、装置のカーソルは、このフィールドの最初の桁位置に置かれます。物理ページと対応している LPAGE の一連の LPAGE で最初に検出されたカーソル位置付け要求 (カーソル属性またはカーソル行/桁値を指定した最初の MFLD) が実行されます。これらのビットは 00 か 11 でなければなりません。
	2 から 7	オフにしておかなければなりません。
1	0	オンにしておかなければなりません。
	1	1. オンにすると、これらの属性指定は、このフィールドに対して定義されている属性バイトと置き換わります。 2. オフにすると、これらの属性指定は、このフィールドに対して定義されている属性バイトに追加されます (論理 『OR』 演算)。ビット位置が 0 であれば、すでに定義されている属性をそのまま使用することを表します (つまり、ビット 2 が 0 ならば、DFLD の定義に基づいてこのフィールドは保護または無保護のフィールドになります)。ビット位置が 1 であれば、それに対応する属性を使用することを表します (つまり、ビット 3 が 1 ならば、このフィールドは数字属性を持ちます)。

表 123. 2 つの属性バイトの定義 (続き) :

バイト	ビット	定義
	2	保護
	3	数値
	4	高輝度 (強制的に検出可能かつ表示可能にされます)。シミュレートされる場合は、装置フィールドの第 1 バイトに * が表示されます。
	5	表示不能 (強制的に検出不能にされます)。シミュレートされる場合は、他のどの属性がどのように指定されていても、データの送信は行われません。
	6	検出可能 (強制的に通常輝度にされます)。
	7	事前変更済み。シミュレートされる場合は、装置フィールドの第 1 バイトに下線 (_) が表示されます。

注:

1. ビット 4、5、および 6 は、そのうちのいずれか 1 つしか指定できません。これらのビットのうちの複数をセットしたときは、ビット 4 がビット 5 および 6 に優先し、ビット 5 はビット 6 に優先します。
2. ビット 4 と 7 の両方がシミュレートされる場合は、装置フィールドの第 1 バイトに ! が表示されます。

属性を検出可能へ動的に変更するためには、IMS アプリケーション・プログラムでさらに適切なアクションを行って装置を正しく機能させる必要があります。検出可能フィールドには、1 つの指示文字と特定の埋め込み文字を付ける必要があります。

DPM の場合、IMS アプリケーション・プログラムからリモート・プログラムにフィールド属性情報を渡すことはできますが、MFS DFLD 定義で ATTR=(YES,*nn*) を指定しておかないと、フィールド属性情報を指定することはできません。

特定の端末タイプでどの拡張属性を使用できるかについては、該当するコンポーネントの解説書を参照してください。

関連資料:

507 ページの『装置依存の出力情報』

拡張フィールド属性の動的変更

アプリケーション・プログラムで拡張属性データを変更するためには、MFLD ステートメントで ATTR=*nn* を指定しておかなければなりません。エラーが起こると、その拡張属性バイトについては、DFLD EATTR= の指定が使用されます。

拡張属性の変更を行うためには、1 つの属性につき 2 バイトを余分に確保しておく必要があります。これらの拡張属性変更バイトに指定できる値と、その結果として使用される値は次のとおりです。

指定　使用される値

X'00'　装置デフォルト

有効な値
　　ユーザーの指定

無効な値または省略
　　DFLD ステートメントの EATTR= で指定されている値

重複　最後の (右端) 指定

オンラインの実行時に、動的変更として ATTR=PROT を指定すると、DFLD ステートメントで定義されているフィールド妥当性検査属性、または動的変更として指定したすべてのフィールド妥当性検査属性はリセットされます。

制約事項**:** トリガー・フィールドは MFS ではサポートされません。

以下の表は、拡張属性変更バイトの形式を示しています。

表 *124.* 拡張属性変更バイトの形式

ATTR 1 のタイプ	**ATTR 1** の値	**ATTR 2** のタイプ	**ATTR 2** の値	**ATTR n** のタイプ	**ATTR n** の値
	1	2	3	2xn_2	2xn_1

タイプ

16 進数の指定値

01　妥当性検査置換

02　妥当性検査追加

03　フィールドの枠取り置換

04　フィールドの枠取り追加

05　入力制御置換

06　入力制御追加

フィールドの枠取りが表示されるのは、3270 表示装置、3270P または SCS1 と定義されたプリンターです。これらの装置は、3270 構造化フィールドおよび属性処理オプションをサポートすると同時に、拡張図形文字セット (EGCS) をサポートするものでなければなりません。

文字の指定値 (C は文字を表します。)

C1　強調表示

C2　カラー

C3　プログラム式シンボル

値

フィールド妥当性検査の 16 進数値は次のとおりです。

ビット　意味

0 から **4**
　　予約済み

5　全桁入力必須

6　必須フィールド

7 予約済み

フィールド強調表示の場合は次のとおりです。

文字 意味

X'00' 装置デフォルト

X'F1' 明滅

X'F2' 反転表示

X'F4' 下線

フィールド・カラー (7 色を使える型式のみ) の場合は次のとおりです。

文字 意味

X'00' 装置デフォルト

X'F1' 青色

X'F2' 赤色

X'F3' ピンク色

X'F4' 緑色

X'F5' 青緑色

X'F6' 黄色

X'F7' 無色

フィールドの枠取りの 16 進数値は次のとおりです。

ビット 意味

0 から **3**
予約済み

4 左線

5 上線

6 右線

7 下線

X'00' デフォルト (枠取りなし)

(DBCS/EBCDIC 混合フィールドの) 入力制御の 16 進数は次のとおりです。

ビット 意味

0 から **6**
予約済み

7 SO/SI の作成

X'00' デフォルト (SO/SI の作成なし)

プログラム式シンボルの場合、有効なローカル ID 値の範囲は、X'40' から X'FE' または装置のデフォルトである X'00' です。

COBOL 言語で 2 進妥当性検査属性のタイプと値を指定する方法は、以下のコード例に示されています。

```
VAL_REP_MFILL  PIC 9(3)  COMP VALUE 260   (replace-mandatory fill)
*
VAL_REP_MFLD   PIC 9(3)  COMP VALUE 258   (replace-mandatory field)
*
VAL_ADD_MFILL  PIC 9(3)  COMP VALUE 516   (add-mandatory fill)
*
VAL_ADD_MFLD   PIC 9(3)  COMP VALUE 514   (add-mandatory field)
*
```

COBOL 言語でフィールド枠取り属性、入力制御タイプ、および値を指定する方法は、以下のコード例に示されています。

```
       01  BINVALUE.
           02  VAL0000                 PIC S999 COMP   VALUE   +0.
           02  VAL0000X    REDEFINES   VAL0000.
               03  FILLER              PIC X.
               03  VAL00               PIC X.
      *                                      (NO FIELD OUTLINE)
           02  VAL0001                 PIC S999 COMP   VALUE   +1.
           02  VAL0001X    REDEFINES   VAL0001.
               03  FILLER              PIC X.
               03  VAL01               PIC X.
      *                                      (UNDERLINE)
           02  VAL0002                 PIC S999 COMP   VALUE   +2.
           02  VAL0002X    REDEFINES   VAL0002.
               03  FILLER              PIC X.
               03  VAL02               PIC X.
      *                                      (RIGHTLINE)
           02  VAL0003                 PIC S999 COMP   VALUE   +3.
           02  VAL0003X    REDEFINES   VAL0003.
               03  FILLER              PIC X.
               03  VAL03               PIC X.
      *                                      (RIGHTLINE & UNDERLINE)
           02  VAL0004                 PIC S999 COMP   VALUE   +4.
           02  VAL0004X    REDEFINES   VAL0004.
               03  FILLER              PIC X.
               03  VAL04               PIC X.
      *                                      (OVERLINE)
           02  VAL0005                 PIC S999 COMP   VALUE   +5.
           02  VAL0005X    REDEFINES   VAL0005.
               03  FILLER              PIC X.
               03  VAL05               PIC X.
      *                                      (OVERLINE & UNDERLINE)
           02  VAL0006                 PIC S999 COMP   VALUE   +6.
           02  VAL0006X    REDEFINES   VAL0006.
               03  FILLER              PIC X.
               03  VAL06               PIC X.
      *                                      (OVERLINE & RIGHTLINE)
           02  VAL0007                 PIC S999 COMP   VALUE   +7.
           02  VAL0007X    REDEFINES   VAL0007.
               03  FILLER              PIC X.
               03  VAL07               PIC X.
      *                                      (OVERLINE & RIGHTLINE
      *                                       & UNDERLINE)
           02  VAL0008                 PIC S999 COMP   VALUE   +8.
           02  VAL0008X    REDEFINES   VAL0008.
               03  FILLER              PIC X.
               03  VAL08               PIC X.
      *                                      (LEFTLINE)
```

```
           02  VAL0009                PIC S999 COMP   VALUE   +9.
           02  VAL0009X    REDEFINES  VAL0009.
               03  FILLER             PIC X.
               03  VAL09              PIC X.
      *                                      (LEFTLINE & UNDERLINE)
           02  VAL000A                PIC S999 COMP   VALUE  +10.
           02  VAL000AX    REDEFINES  VAL000A.
               03  FILLER             PIC X.
               03  VAL0A              PIC X.
      *                                      (LEFTLINE & RIGHTLINE)
           02  VAL000B                PIC S999 COMP   VALUE  +11.
           02  VAL000BX    REDEFINES  VAL000B.
               03  FILLER             PIC X.
               03  VAL0B              PIC X.
      *                                      (LEFTLINE & RIGHTLINE
      *                                       & UNDERLINE)
           02  VAL000C                PIC S999 COMP   VALUE  +12.
           02  VAL000CX    REDEFINES  VAL000C.
               03  FILLER             PIC X.
               03  VAL0C              PIC X.
      *                                      (LEFTLINE & OVERLINE)
           02  VAL000D                PIC S999 COMP   VALUE  +13.
           02  VAL000DX    REDEFINES  VAL000D.
               03  FILLER             PIC X.
               03  VAL0D              PIC X.
      *                                      (LEFTLINE & OVERLINE
      *                                       & UNDERLINE)
           02  VAL000E                PIC S999 COMP   VALUE  +14.
           02  VAL000EX    REDEFINES  VAL000E.
               03  FILLER             PIC X.
               03  VAL0E              PIC X.
      *                                      (LEFTLINE & OVERLINE
      *                                       & RIGHTLINE)
           02  VAL000F                PIC S999 COMP   VALUE  +15.
           02  VAL000FX    REDEFINES  VAL000F.
               03  FILLER             PIC X.
               03  VAL0F              PIC X.
      *                                      (BOX)
```

例

以下に、EATTR= オペランドおよび ATTR=(*,nn*) オペランドの使用例を示します。

```
AX     DFLD  EATTR=(VMFILL,HUL),ATTR=(NUM,HI)
AY     MFLD  AX,ATTR=(,2)
```

DFLD ステートメントの EATTR= オペランドは、指定したフィールドにデータを完全に埋め込むこと、高輝度、下線を引くことを要求します。DFLD ステートメントの ATTR= オペランドは、指定したフィールドに数値のみを指定すること、および高輝度を要求します。

ATTR=(,2) オペランドを指定すると、アプリケーション・プログラムが EATTR= オペランドに指定した 2 つの拡張属性を動的に変更できることを示します。このオペランドを指定する場合は、変更された属性バイト用に MFLD ステートメントの LTH= の値を 4 バイト増やす必要があります。アプリケーション・プログラムは、妥当性検査属性および拡張強調表示属性を動的に変更することができます。カラーとプログラム式シンボルの拡張属性は、動的に変更はできません。これらの属性は

EATTR= オペランドには指定されていないためです。既存の 3270 の属性の場合も、MFLD ステートメントで ATTR=YES が指定されていないため、動的に変更できません。

データをフィールドに入力する際に、動的に拡張強調表示を明滅に変更し、必須フィールドの妥当性検査を追加する場合、前の例で MFLD「AY」によって参照されているフィールドには、以下の表に示す拡張属性タイプおよび値を入れる必要があります。

表 *125. COBOL* 言語での拡張属性タイプおよび値

ATTR 1 のタイプ	**ATTR 1** の値	**ATTR 2** のタイプ	**ATTR 2** の値	フィールド・データ
C1	F1	02	02	データ
0	1	2	3	4–n

カラーおよびプログラム式シンボルの指定があったとしても、無視されます。属性変更バイトを何バイト指定したかに関係なく、MFS は、DFLD の EATTR= オペランドで指定されている拡張属性の個数を送ります。

妥当性検査置換タイプ (X'01') ではなく妥当性検査追加タイプ (X'02') が指定されているため、妥当性検査属性バイトを置換ではなく追加に変更します。

```
BX    DFLD    EATTR=(CD,HD,PC'Z'),ATTR=(PROT)
BY    MFLD    BX,ATTR=(YES,3)
```

DFLD ステートメントの EATTR= オペランドは、プログラム式シンボル・バッファー・ローカル ID が 『Z』 で、保護属性付きのフィールドを要求します。IMS アプリケーション・プログラムで動的変更を行わなければ、装置のデフォルトのカラーおよび強調表示が使用されます。この例では、ATTR=(YES,3) が指定されているため、カラー、拡張強調表示、プログラム式シンボル・バッファーのローカル ID、既存の 3270 属性を動的に変更することができます。

プログラム式シンボル・ローカル ID を、DFLD ステートメントで指定した値 (PC'Z') のままにしておく間は、カラー、拡張強調表示、および 3270 属性バイトを動的に変更することができます。例えば、カラーをピンク色に、拡張強調表示を反転表示に、そして 3270 属性バイトを数値および無保護に、それぞれ動的に変更するためには、MFLD 「BY」で参照されるフィールドに必要な属性変更バイトを使用します。以下の表に、それらの属性変更バイトを示します。

表 *126.* 動的変更属性バイトの例

既存 **3270 ATTR** 変更		**ATTR 1** のタイプ	**ATTR 1** の値	**ATTR 2** のタイプ	**ATTR 2** の値	**ATTR 3** のタイプ	**ATTR 3** の値	フィールド・データ
00	D0	C2	F3	C1	F2	40	40	データ
0	1	2	3	4	5	6	7	8–n

既存の 3270 属性変更バイトのバイト 1, ビット 1 がオンであれば、IMS は既存属性バイトを置き換えます。追加は行われません。これによって、フィールドが無保護に変わり、数値属性が指定されます。3 番目の属性のタイプとして X'40' (無効な

タイプ) が指定されているため、プログラム式シンボルについては、IMS は DFLD の指定を使用します。

関連資料:

518 ページの『DBCS/EBCDIC 混合データの動的変更』

EGCS データの動的変更

EGCS データを動的に変更することによって、EBCDIC または EGCS データを 3270 表示装置の特定のフィールドにマップすることができます。

この機能を使用すれば、以下のことを行えます。

- EBCDIC または EGCS データを入力することができます。
- アプリケーション・プログラムは、EBCDIC または EGCS データを受け取ることができます。
- EBCDIC または EGCS データを、SLU P リモート・プログラムまたは ISC サブシステムに渡すことができます。

MFLD ステートメントで ATTR=(*,nn*) を指定し、それと対応する DFLD ステートメントでプログラム式シンボル属性を指定しておけば、アプリケーション・プログラムは、そのフィールドのプログラム式シンボル属性を変更することができます。EGCS のプログラム式シンボル属性を動的に変更するためには、さらに 2 バイトを追加する必要があります。この追加バイトは、MFLD データの前にくるので、MFLD LTH= の指定の中にその長さも含めておかなければなりません。

IMS アプリケーション・プログラムは、次のすべての条件を満たしていれば、DFLD のプログラム式シンボル属性を変更することができます。

- DFLD は、EATTR=PX'hh'、PC'c'、EGCS'hh'、または EGCS を指定します。
- 対応する MFLD ステートメントで、ATTR=(*,nn*) が指定されている (ここで、*nn* は 1 から 4 の値)。
- アプリケーション・プログラムでは、データ・フィールドの前に 2 × *nn* の追加バイトが入っている。
- 2 つの属性バイトからなる 1 つのセットで、最初のバイトには X'C3' が、2 番目のバイトには有効な値 (X'00' またはX'40' から X'FE') が入っている。

以下の表は、DFLD ステートメントでプログラム式シンボル属性を指定した場合としない場合、また、IMS アプリケーション・プログラムがその属性を変更した場合としない場合のそれぞれについて、MFS がプログラム式シンボル属性タイプの値バイトに何を送信するかを示したものです。

表 *127.* 属性タイプ値のバイトの内容

アプリケーション・プログラムのプログラム式シンボル属性バイトの **X** および	**C3 EATTR=**	**ATTR=**	**EATTR=**
	プログラム式シンボルが指定されている	プログラム式シンボルはデフォルト	指定なし
X'40_FE'[1]	X'40_FE' を送信する	X'40_FE' を送信する	属性を送信しない
デフォルトの X'00'[1]	X'00' を送信する	X'00' を送信する	属性を送信しない

表 127. 属性タイプ値のバイトの内容 (続き)

アプリケーション・プログラムのプログラム式シンボル属性バイトの X および	C3 EATTR=	ATTR=	EATTR=
指定なし [2]	DFLD 指定のプログラム式シンボルを送信する	属性を送信しない	N/A
省略または無効 [3]	DFLD 指定のプログラム式シンボルを送信する	X'00' を送信する	属性を送信しない

注:

1. この DFLD ステートメントにマップする MFLD ステートメントのうち、少なくとも 1 つに ATTR=*nn* が指定されています。IMS アプリケーション・プログラムが指定するプログラム式シンボル属性は、X'40' から X'FE' のいずれかです。
2. この DFLD ステートメントにマップするどの MFLD ステートメントにも、ATTR=*nn* は指定されていません。
3. この DFLD ステートメントにマップする MFLD ステートメントのうち、少なくとも 1 つに ATTR=*nn* が指定されています。アプリケーション・プログラムがこの属性を指定しなかったか、指定した属性が X'00' または X'40' から X'FE' のいずれの値でもありません。

DBCS/EBCDIC 混合データの動的変更

プログラム式シンボルおよび入力制御属性バイトを動的に変更することによって、3270 表示装置の特定のフィールドに EBCDIC または EGCS データをマップすることができます。DBCS/EBCDIC 混合データも、動的に変更することができます。DBCS は EGCS のサブセットです。したがって、以下の図のように EGCS フィールドに DBCS データを入れることができます。

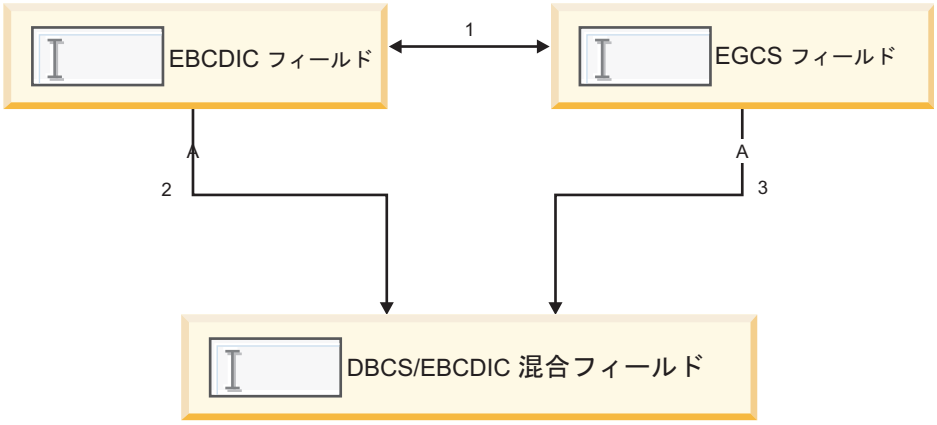


図 29. *DBCS/EBCDIC* 混合フィールドの動的変更

IMS アプリケーション・プログラムは、次の条件をすべて満たしていれば、フィールドを EBCDIC、EGCS、または DBCS/EBCDIC 混合フィールドにすることができます。

- DFLD ステートメントには、次のうちのいずれかが指定されている。

  ```
  EATTR=(EGCS,MIXD)
  EATTR=(EGCS'00',MIX)
  EATTR=(EGCS'00',MIXD)
  ```

  DBCS キーワードがないことに注意してください。DBCS フィールドは、EGCS キーワードを使用して指定されています。初期属性では、EGCS フィールド、DBCS/EBCDIC 混合フィールド、または EBCDIC フィールドを指定しなければなりません。
- 対応する MFLD ステートメントで、ATTR=(,*nn*) が指定されている (ここで、*nn* は 2 以上)。
- アプリケーション・プログラムでは、データ・フィールドの前に 2 × *nn* の追加バイトが入っている。

  nn=2 の場合、データ・フィールドの前の 2 つの属性バイト・セット (4 バイト) にアプリケーション・プログラムが指定した値に基づいて、初期属性は以下の表のように変更されます。

表 *128. DBCS/EBCDIC* 混合フィールドの動的変更

属性バイト	**EBCDIC**	**EGCS**	混合
40404040	EBCDIC	EGCS	混合
05014040	混合	混合	混合
0501C3F8	EGCS	EGCS	EGCS
C3F84040	EGCS	EGCS	EGCS
C3F80501	混合	混合	混合
0500C3F8	EGCS	EGCS	EGCS
C3000501	混合	混合	混合
C3000500	EBCDIC	EBCDIC	EBCDIC

初期属性が EGCS フィールドを指定し、さらにアプリケーション・プログラムが入力制御属性の動的変更を指定して DBCS/EBCDIC 混合フィールドに変えた場合、MFS は、その EGCS フィールドのプログラム式シンボルの初期属性値を、装置のデフォルトと置き換えます。

関連資料:

511 ページの『拡張フィールド属性の動的変更』

メッセージ出力記述子名の指定

MFS 端末あての出力メッセージは、メッセージ出力記述子 (MOD) に基づいて形式設定されます。

IMS が使用する MOD は、出力呼び出し、挿入 (ISRT) かパージ (PURG) のいずれかの中で指定することができます。`ISRT` と `PURG` のどちらでも、出力メッセージのセグメントを提供する呼び出しで出力 MOD 名パラメーターを指定することができます。

出力 MOD 名パラメーターを指定すると、IMS はその名前を持つメッセージ出力記述子を選択します。その呼び出しが TP PCB または代替応答 PCB に送信される

場合、IMS は TP PCB の「メッセージ出力記述子名」フィールドを更新し、出力呼び出しに指定されている名前をそのフィールドに入れます。 MOD 名を明示的に指定せずに、代替 PCB を使用して出力メッセージを挿入すると、その場合の MOD 名は前の MOD 名に設定されます。

IMS がメッセージを形式設定する際に使用する MOD は、プログラムで指定した名前によって決まります。

指定した名前
: 使用される記述子

有効な出力 **MOD** 名
: 出力 MOD 名で指定したメッセージ出力記述子

8 個のブランク
: IMS デフォルト・メッセージ出力記述子 (3270 または SLU 2 のみ。その他の装置では、出力用の IMS 基本編集が使用されます。)

無効な出力 **MOD** 名
: IMS で決まっているエラー時のデフォルト・メッセージ出力記述子

出力 MOD 名パラメーターを指定しなかった場合には、IMS は入出力 PCB の「メッセージ出力記述子名」フィールドに入っている名前の MOD を使用して、メッセージを形式設定します。

3270 または SLU 2 のための MFS バイパス

IMS MFS では、IMS アプリケーション・プログラムが入出力メッセージの MFS フォーマット設定をバイパスできるようになっています。

このオプションを使用することで、IMS アプリケーション・プログラムは、プログラム式シンボル・バッファーのロード、3270 表示装置の形式設定および更新のための装置依存データ・ストリームの送信、または 3270 印刷装置へのメッセージの書き込みなどを行うことができます。バイパスを使用できるのは、SLU 2 装置と 3270 装置だけです。IMS アプリケーション・プログラムは、ディスプレイから入力メッセージを受信したときに、アテンション識別 (AID) バイト、カーソル・アドレス、SBA 副指令、およびバッファー・アドレスを検査することができます (オプション機能)。非 SNA VTAM で送信されるデータは、システム定義の OUTBUF パラメーターで指定されている値に等しいか、それよりも小さくなければなりません。MFS をバイパスしてプリンターに送信されるデータ量は、4 KB までに制限されています。

MFS は、DFS.EDT および DFS.EDTN という 2 つの特別なメッセージ出力記述子 (MOD) 名を識別します。

出力メッセージで MFS のフォーマット設定がバイパスされるのは、アプリケーション・プログラムの CALL ステートメントの MOD 名パラメーターとして DFS.EDT または DFS.EDTN が指定されている場合だけです。 IMS システム・メッセージ、IMS エラー・メッセージ、MOD 名が指定されていないアプリケーション・プログラム・メッセージ、およびメッセージ通信は、常に MFS によって (IMS 提供の形式で) フォーマット設定されます。

出力で MFS をバイパスするときは、コマンド・コードから始まって最後のデータ・バイトに至るまでの 3270 データ・ストリーム全体を、アプリケーション・プログラムが責任をもって作成しなければなりません。ただし、プリンターあての 3270 出力を、MFS をバイパスして送信する場合は例外です。3271/3274 制御装置で使用される 16 進数の EBCDIC コマンド・コードを次に示します。

コマンド
: **3271/3274**

全無保護域消去
: 6F

消去/書き込み
: F5

消去/書き込み代替
: 7E

バッファー読み取り
: F2

変更読み取り
: F6

全変更読み取り
: 6E

書き込み
: F1

構造化フィールド書き込み
: F3

ユーザー作成のアプリケーション・プログラムがプリンターに出力を送信する方法は 2 通りあります。

- アプリケーション・プログラムではコマンド・コードおよび WCC 文字を用意し、適切なコマンドを用意したことを示すために、メッセージ・セグメントの Z2 フィールドのビット 0 を 1 (X'80') に設定する方法。
- IMS がコマンド・コードおよび他の文字を用意する方法。ただし、最大長よりも短い行長で印刷するためには、データ・ストリームの適切な個所に、NL (改行) 文字を挿入しておく必要があります。この方法がデフォルトとして採用されています。

MFS バイパスのための入力形式の指定**:**

MFS をバイパスした後、IMS アプリケーション・プログラムは出力メッセージに指定した MOD 名に応じて次のいずれかの形で入力を受け入れなければなりません。

2 つの入力形式は以下のとおりです。

- MODNAME=DFS.EDT を指定すると、入力データは編集されます。
- MODNAME=DFS.EDTN を指定すると、入力データの編集は行われません。

MODNAME=DFS.EDT

AID およびカーソル・アドレスはデータ・ストリームから削除され、どの SBA またはフィールド開始順序もブランクで置き換えることができます。さらに、基本入力編集ルーチンが編集を行います。受信した AID コードが、
CLEAR、PA2、PA3、PFK12、または選択ペン・アテンションのいずれかであれば、既存の IMS 機能が実行されます。PA1 を受信した場合でも、IMS は PA2 を受信したときと同じ機能を実行します (つまり、次の出力メッセージがあれば、そのメッセージが送信されます)。

MODNAME=DFS.EDTN

会話型モードのトランザクションならば、端末から受信したとおりの入力がそのまま全部アプリケーションに渡されます。会話型モードでないトランザクションならば、端末から受信したデータ・ストリームの AID 文字の前に、トランザクション・コードが付いていなければなりません。

パスワードを IMS アプリケーション・プログラムに渡してはなりません。基本編集が実行されるのは、宛先フィールドおよびパスワード・フィールドのみです。トランザクション・コードの直後に、括弧に入ったパスワードが続いている場合、そのパスワードは基本編集によって除去されます。残りのデータの編集は行われません。また、CLEAR、PA1、PA2、PA3、または選択ペン・アテンションの各キーが押されると、AID コードについては既存の IMS 機能はバイパスされます。PFK12 でコピーが実行されます (許可されている場合)。

物理端末入力編集出口ルーチンを使用してトランザクション・コードを所定の位置に置くか、会話モードか事前設定宛先モードで IMS からトランザクション・コードを受け取ります。

端末が会話型モードになっていれば、メッセージは会話の形でアプリケーション・プログラムに送信されます。端末が事前設定モードになっていれば、トランザクション・コードがメッセージの先頭に追加され、メッセージは /SET コマンドで設定されている宛先に送信されます。したがって、事前設定モードのときに、入力データの最初の文字がスラッシュ (/) であっても、入力データは IMS コマンドとは見なされません。コマンドとして認識させるには、入力データ・ストリームでカーソル・アドレスの直後に /RESET を指定しなければなりません。このためには、形式設定されていない画面 (画面にフィールドが定義されていない状態) から /RESET コマンドを入力します。画面が形式設定済み (画面にフィールドが定義されている状態) であれば、CLEAR キーを押して画面の形式設定を解除します。ただし、この場合、アプリケーション・プログラムは CLEAR (消去) の AID バイトを受信することになり、画面を形式設定しないデータ・ストリームを作成しなければなりません。

例:

```
Data stream = F5C3, erases the 3270 buffer.
Data stream = F5C3114040, erases the 3275 buffer.
              Entering: The /RESET command
              resets preset mode.
```

MFS および基本編集 (MOD 名が DFS.EDTN) をバイパス中で、かつ事前設定モードになっているときに、不定様式画面から /RESET を受信すると、入力はコマンド

と見なされて、端末は事前設定モードからはずされます。ユーザーは、画面を不定様式のままにしておくデータ・ストリームを送信しなければなりません。

端末が会話型モードでも事前設定モードでもないときに、トランザクション・コードおよびパスワード (必要な場合) を入力メッセージと一緒に入力する場合は、IMS システム定義に物理端末入力編集出口ルーチンを入れておかなければなりません。物理端末入力編集ルーチンは、制御権を受け取ってから IMS での宛先およびセキュリティー検査を行い、トランザクション・コードおよびパスワード (必要な場合) を AID コードの前に置くように入力を変更する必要があります。

IMS システム定義の TERMINAL または TYPE マクロの OPTIONS キーワードで、キーボードをロック状態にしておくことが指定されており、MOD 名が DFS.EDTN であるために MFS がバイパスされている場合は、アプリケーション・プログラムが、3270 キーボードのアンロックおよび MDT フラグのリセットを行わなければなりません。

MFS をバイパスした後で、次の出力メッセージに MOD 名が入っていない場合、あるいは MOD 名が DFS.EDT と DFS.EDTN のどちらでもない場合は、出力メッセージは MFS によって形式設定されます。

MFS バイパスは、IMS のもとでサブシステムを実行するために使用するのが主な目的であるため、通常のアプリケーションでは使用しないでください。IMS アプリケーション・プログラムは、3270 データ・ストリームを処理する場合は装置依存になってしまうため、アプリケーションの開発作業が複雑になります。

MFS バイパスで読み取りコマンドが実行されると、読み取りコマンドが入っている出力メッセージは、入力を受信した時点でデキューされるか、再度キューに入れられます。このどちらの処理が実行されるかは、システム定義時に TERMINAL マクロに指定されたオプション (PAGDEL/NPGDEL) によって決まります。

区分化される **SLU 2 (3290)** 用の **MFS** バイパス**:**

アプリケーションで MOD を DFS.EDT または DFS.EDTN と指定して出力メッセージを作成すれば、SLU 2 端末を区画モードで作動させることができます。DFS.EDTN を使用すると、会話型アプリケーションは照会を送信し、照会の応答を受信することができます。

出力の場合、アプリケーション・プログラムは、出力メッセージに、それぞれのデータと一緒に区画作成データ・ストリームを用意しなければなりません。また、SLU 2 装置依存モジュールは、最終以外の会話型出力メッセージに方向変換 (CD) をセットします。これによって、読み取りおよび照会を、構造化フィールド書き込みデータ・ストリームに入れて送信することができます。

照会応答入力が処理されるのは、その前の照会の MOD が DFS.EDTN の場合だけです。照会応答入力を受け取っても、そのデータ・ストリームにはトランザクション・コードが入っていません。

01 から 0F の区画では、X'88' バイトのあとに未使用の 2 バイト長のフィールドが続きます。このフィールドに X'80' バイトが続いていれば、その次のバイトは PID バイト (X'01' から X'0F') です。区画 00 の入力は、区画化されていない SLU 2 からの入力データと同じ形式になっています。

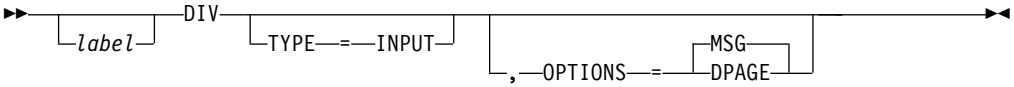
DFS.EDT または DFS.EDTN を使用する入力内に、最初の AID バイト X'88' が入っていると、2 番目の AID バイトのデコードが適切に行われます。2 番目の AID バイトによって、次のうちのいずれかが起こります。

- デコード化された 2 番目の AID バイトが X'80' であれば、3 番目の AID バイトがデコードされます。その AID バイトに続くデータ・ストリームは、次のようにしてアプリケーションに渡されます。
  - DFS.EDT が指定されている場合は、基本編集を使用する。
  - DFS.EDTN が指定されている場合は、完全なデータ・ストリームとしてアプリケーション・プログラムに渡される。
- 2 番目の AID バイトが X'80' でないときは、アプリケーションが MOD に DFS.EDTN を指定したときにだけ、入力がアプリケーションに渡されます。DFS.EDTN を指定したときは、X'88' AID バイトで始まる完全なデータ・ストリームがアプリケーション・プログラムに渡されます。

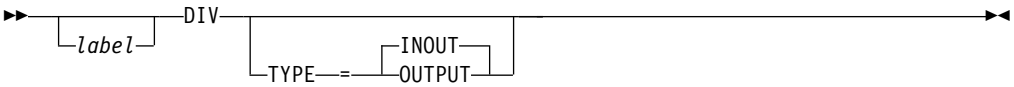
DIV ステートメント**:**

DIV ステートメントは、DIF または DOF 内の装置形式を定義します。フォーマットは入力用、出力用、入出力用があり、複数の物理ページから構成できます。 DEV TYPE=SCS1、SCS2、または DPM-AN の場合は、2 つの DIV ステートメント (DIV TYPE=OUTPUT および DIV TYPE=INPUT) を定義することができます。その他の装置タイプでは、DEV ごとに 1 つの DIV ステートメントしか認められません。

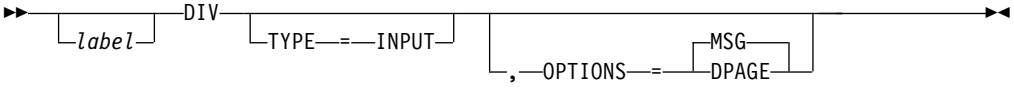
DEV TYPE=SCS1、**SCS2** のいずれかで、**DIV TYPE=INPUT** のときのフォーマット



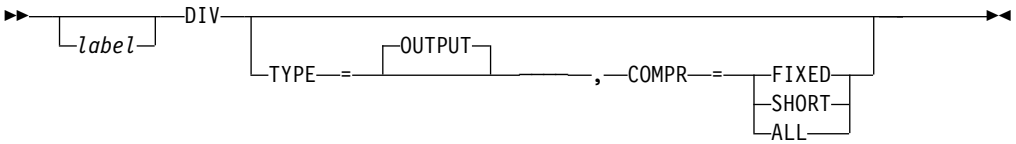
DEV TYPE=3270 または **3270-An** のときのフォーマット



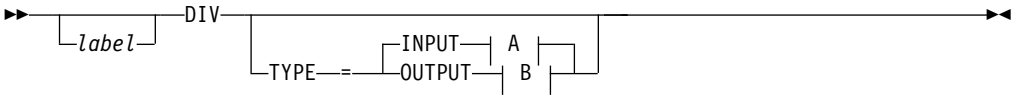
DEV TYPE=FIN のときのフォーマット



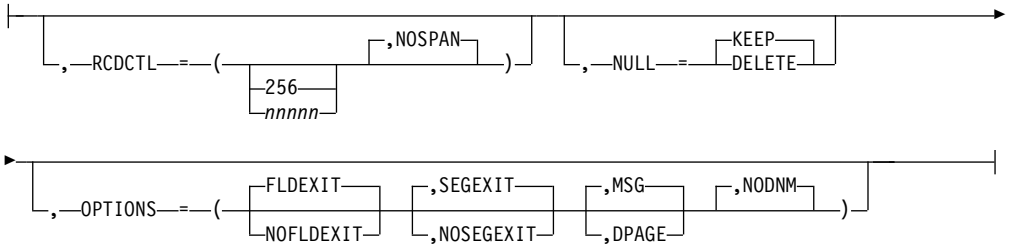
DEV TYPE=SCS1、SCS2、3270P、FIDS、FIDS3、FIDS4、FIDS7、FIJP、FIPB、FIFP のいずれかで、**DIV TYPE=OUTPUT** のときのフォーマット



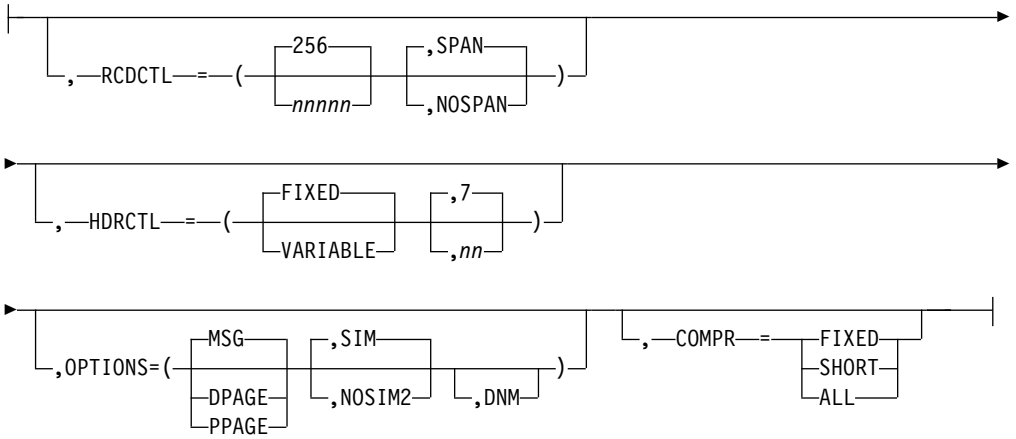
DEV TYPE=DPM-An のときのフォーマット



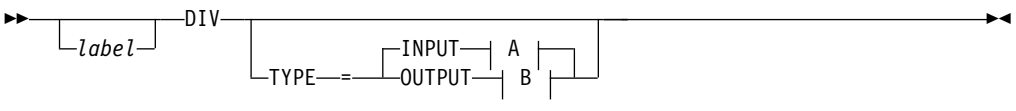
A:



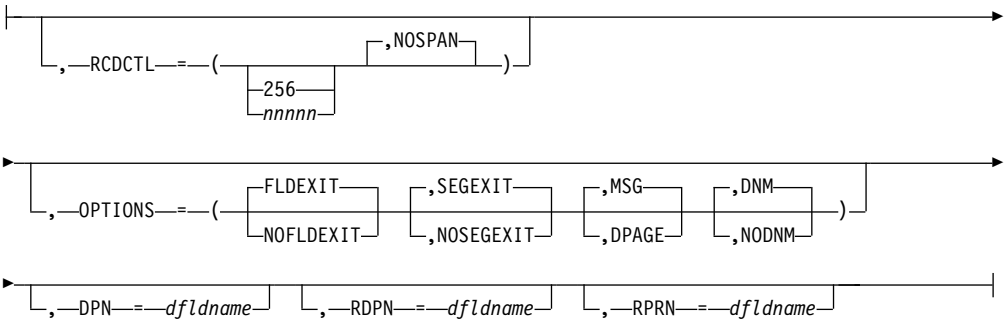
B:



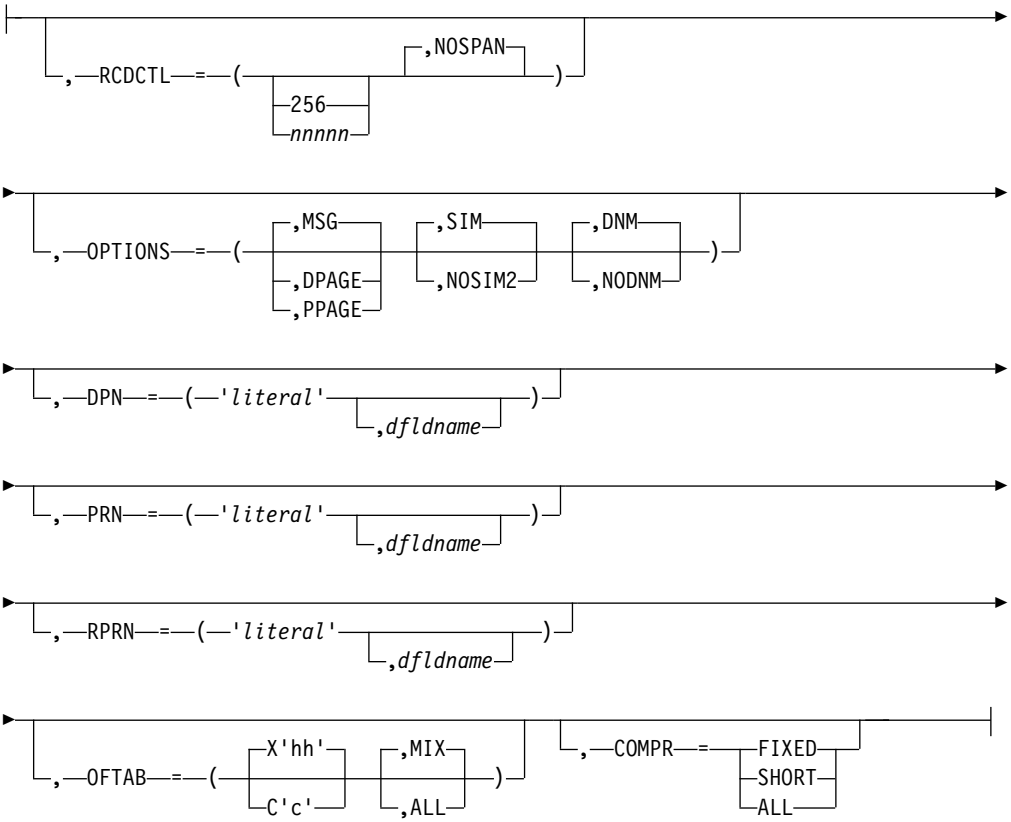
DEV TYPE=DPM-Bn のときのフォーマット



A:



<B>:



パラメーター

label
1 文字から 8 文字の英数字で、このステートメントを識別できるように固有の名前を指定します。

TYPE=
入力、出力、またはその両方としての形式を表します。

INOUT
入出力の形式を表します。

INPUT|OUTPUT
入力のみの形式 (INPUT) または出力のみの形式 (OUTPUT) を表します。所定の DEV ステートメントのキーワードが使用されます。例えば、次のようになります。

- DEV TYPE=SCS1 で WIDTH=80 を指定すると、フィールドは、出力時に 1 桁から 80 桁の範囲で印刷され、入力も 1 桁から 80 桁の範囲で受信されることになります。
- DEV TYPE=SCS2 で WIDTH=80 とすると、カード・リーダーとカード・パンチの両方で穿孔位置の数が同じであることを意味します。
- DEV TYPE=SCS1 で WIDTH=80 および HTAB=(SET,5) を指定すると、そのフィールドは、出力時に 5 から 80 桁までの範囲で印刷され、入力も 5 から 80 桁までの範囲で受信されることになります。この場合、入力時の DFLD POS=(1,5) または POS=5 は、桁を 1、左マージン位置を 1 に指定したときと同じことになります。

  現在左マージンが設定されている位置に関係なく、同じ方法でデータを入力してください。

RCDCTL=
レコード定義を作成します。RCD ステートメントが同じ形式定義で使用されていてもかまいません。RCDCTL は、DEV ステートメントに MODE=RECORD が指定されているときのみ有効です。

OPTIONS=MSG では、最初のデータ・フィールドがメッセージの最初のフィールドです。 OPTIONS=DPAGE および PPAGE の場合、最初のデータ・フィールドは、それぞれ DPAGE または PPAGE の最初のフィールドになります。最初のデータ・フィールドが出力メッセージ・ヘッダーと同じレコードに収まらない場合、OPTIONS=DPAGE または PPAGE が指定されていれば、最初のデータ・レコードは次の伝送で送信されます。出力メッセージ・ヘッダーは単独で送信されます (OPTIONS=MSG では常に単独)。

256
入力伝送または出力伝送の最大長。値 256 は、DEV TYPE=DPM-An または DPM-Bn の場合にのみ指定することができます。

nnnnn
入力伝送または出力伝送の最大長。値は、DEV TYPE=DPM-An または DPM-Bn の場合にのみ指定することができます。この長さは 32000 以下、メッセージ出力ヘッダーの長さ以上でなければなりません。

TYPE=OUTPUT を指定する場合、*nnnnn* は IMS システム定義の OUTBUF= マクロで指定されている出力バッファー・サイズ以下でなければなりません。*nnnnn* が指定されている OUTBUF= の値よりも大きいと、1 つのレコードが複数の出力伝送を要求し、リモート・プログラムで好ましくない結果が生じることがあります。定義されたレコードにフィールド群がぴったり収まらず、しかも NOSPAN が指定されていると、レコードが完全に埋まらないことがあります。

SPAN
フィールドが複数のレコードにまたがるようにする指定です。

TYPE=OUTPUT が指定されている場合、DEV TYPE=DPM-An でのみ SPAN を指定することができます。フィールドは、レコード境界をまたぐことはできますが、PPAGE 境界をまたぐことはできません。リモート・プログラムには、その部分フィールドを関連付ける論理を入れるか、あるいはそれらの部分フィールドを個別に取り扱う必要があります。

NOSPAN
フィールドが複数のレコードをまたがないようにする指定です。 1 つのレコードにはすべてのフィールドが入っていますが、指定した値よりも長いフィールドはありません。NOSPAN がデフォルトです。

NULL=
MFS が末尾のヌルを処理する方法を指定します。 NULL= は、DEV TYPE=DPM-An および TYPE=INPUT の場合にのみ指定することができます。

KEEP
末尾のヌルを無視するよう MFS に指示します。

DELETE
末尾のヌルを検索して置換するよう MFS に指示します。 MFS は、入力メッセージ・フィールドで、末尾または全体がヌルのフィールドを検索して、ヌルをメッセージ定義で定義されている充てん文字で置き換えます。

OPTIONS=
データのフォーマット設定およびマッピングを指定します。

DNM
データ名を指定します。

- TYPE=INPUT の場合

  DNM は DEV TYPE=DPM-Bn の場合にのみ指定することができます。メッセージ・ヘッダーで DPAGE データ名が DSN パラメーターとして提供され、さらにその DPAGE データ名が定義済みの DPAGE データ名と一致すると、特定の DPAGE が選択され、現行またはデータのみの伝送がマップされます。この条件が満たされないと、最後に定義された DPAGE 名がデータのマッピングに使用されます。ただし、その DPAGE が条件付きと定義されている場合を除きます。

- TYPE=OUTPUT の場合
  - DNM は DEV TYPE=DPM-An または DPM-Bn の場合にのみ指定することができます。

    DEV TYPE=DPM-An の場合、DEV ステートメントで、FORS キーワードとともに DNM を使用して、メッセージ・ヘッダーにリテラルを指定します。*IMS V14* アプリケーション・プログラミングの「メッセージ処理」のトピックを参照してください。このパラメーターはオプションです。

    DEV TYPE=DPM-Bn の場合、MFS の DD ヘッダーには次のものが入っています。

    - FMT 名 (OPTIONS=MSG の場合)
    - DPAGE 名 (OPTIONS=DPAGE の場合)
    - PPAGE 名 (OPTIONS=PPAGE の場合)

NODNM

データ名がないことを指定します。

- TYPE=INPUT の場合

  NODNM は DEV TYPE=DPM-An または DPM-Bn のいずれかの場合にのみ指定することができます。MFS は、受け取ったデータおよび COND= パラメーターで条件付きのテストを実行させて、特定の DPAGE を選択することができます。

- TYPE=OUTPUT の場合

  NODNM は DEV TYPE=DPM-Bn の場合にのみ指定することができます。NODNM を指定すると、DD ヘッダーにはデータ構造名 (DSN) が含まれません。

DPAGE

データのさまざまな受信方法および伝送方法を指定します。これらの方法は、装置タイプと、TYPE=INPUT または TYPE=OUTPUT のいずれであるかによって異なります。

- TYPE=INPUT の場合
  - SCS1、SCS2、または FIN の場合、あるいは DEV TYPE=DPM-An または DPM-Bn の場合、DPAGE は入力メッセージが複数の DPAGE から作成されていることを指定します。

    MFS 定義で複数の DPAGE 入力が要求されないかぎり、2 つ以上の DPAGE からメッセージを作成することはできません。

    1 つの DPAGE が送信され、その中に選択した DPAGE に定義したデータよりもたくさんのデータが入っている場合、あるいは複数のページを送信する場合、入力メッセージは拒否され、他のサブシステムにエラー・メッセージが送られます。

- TYPE=OUTPUT の場合

  DEV TYPE=DPM-An または DPM-Bn の場合、DPAGE は、IMS がすべての DFLD を送信することを指定します。これらの DFLD は、一緒に 1 つのページにまとめられます。論理ページは、1 つ以上のレコード内に送信されます。DPAGE で PPAGE ステートメントが定義されていると、その PPAGE ステートメントの 1 つ 1 つが新しいレコードを開始します。リモート・プログラムから要求時ページングが要求されると、追加の論理ページが送信されます。各論理ページの前には出力メッセージ・ヘッダーが置かれ、DPAGE のラベルがそのヘッダーに挿入されます。 DEV TYPE=DPM-Bn の場合、DD ヘッダーのデータ構造名はオプションで、DNM または NODNM の指定によって決まります。

FLDEXIT

DEV TYPE=DPM-An または DPM-Bn で、TYPE=INPUT の場合に、MSG 定義 MFLD の出口ルーチンが呼び出される指定です。

デフォルトは FLDEXIT です。

このパラメーターは、DEV TYPE=DPM-An または DPM-Bn で、TYPE=INPUT の場合にのみ指定することができます。

NOFLDEXIT
MSG 定義 MFLD の出口ルーチンがバイパスされる指定です。

MSG
メッセージのさまざまな作成方法および伝送方法を指定します。これらの方法は、装置と、TYPE=INPUT または TYPE=OUTPUT のいずれであるかによって異なります。

- TYPE=INPUT の場合

  DEV TYPE=SCS1、SCS2、または FIN の場合、あるいは DEV TYPE=DPM-An または DPM-Bn の場合、MSG は入力メッセージを 1 つの DPAGE から作成できることを指定します。

- TYPE=OUTPUT の場合

  DEV TYPE=DPM-An または DPM-Bn で TYPE=OUTPUT の場合、MSG がデフォルトになり、IMS がメッセージ内のすべての DFLD を 1 つのメッセージにまとめて送信することを指定します。メッセージの前に出力メッセージ・ヘッダーが付きます。すべての DFLD が送信されます。 DEV TYPE=DPM-Bn の場合、ヘッダーのデータ構造名はオプションです。

PPAGE
IMS が複数の DFLD を、1 つのチェーン内の 1 つの表示ページ (PPAGE) にまとめて送信する指定です。PPAGE は、DEV TYPE=DPM-An または DPM-Bn で、TYPE=OUTPUT の場合にのみ指定することができます。この表示ページは、1 つ以上のレコードからなるグループに送信されます。リモート・プログラムから IMS に要求時ページングが送信されると、追加の表示ページが送信されます。どのプレゼンテーション・ページの前にも出力メッセージ・ヘッダーが置かれ、PPAGE ステートメントのラベルがそのヘッダーに挿入されます。 DEV TYPE=DPM-Bn の場合、DD ヘッダーのデータ構造名はオプションで、DNM または NODNM の指定によって決まります。

SEGEXIT
DEV TYPE=DPM-An または DPM-Bn で、TYPE=INPUT の場合に、MSG 定義 SEG の出口ルーチンが呼び出される指定です。デフォルトは SEGEXIT です。

このパラメーターは、DEV TYPE=DPM-An または DPM-Bn で、TYPE=INPUT の場合にのみ指定することができます。

NOSEGEXIT
MSG 定義 SEG の出口ルーチンがバイパスされる指定です。

SIM
MFS が属性をシミュレートする指定です。これは、DEV TYPE=DPM-An または DPM-Bn で TYPE=OUTPUT の場合にのみ有効です。 SIM は、MFS が IMS アプリケーション・プログラムによって指定された属性をシミュレートし、シミュレートされた属性を対応する DFLD (ATTR=YES または YES,*nn* で定義されている) に置くことを示します。フィールドの最初のバイトが、シミュレートされた属性を入れるのに用いられます。

MFLD が、ATTR=YES または YES,*nn* オペランドを指定した対応する DFLD に対して、ATTR=YES または YES,*nn* オペランドによって 3270 属

性情報を提供しなければ、フィールドの最初のバイトにブランクが 1 つ送られます。アプリケーション設計者は、シミュレートされた属性をリモート・プログラムまたは ISC サブシステムの中で解釈しなければなりません。

SIM/NOSIM2 のデフォルトは SIM です。

NOSIM2

MFS がリモート・プログラムまたはサブシステムに、2 バイト長のビット・ストリングを送信することを指定します。このビット・ストリングは、IMS アプリケーション・プログラムから受信されたとおり正確に送信されます。3270 拡張バイトは、ある場合には (ATTR=YES,*nn*)、常に、アプリケーション・プログラムから受信したとおりに送信され、3270 属性の 2 バイトのストリングのあとに続きます。

MFLD が属性情報を提供しないと、2 進ゼロがその 2 バイトに送信され、そのフィールドのデータが処理されます。

MFLD ステートメントで使用する ATTR パラメーターについて詳しくは、MFS 言語ユーティリティー (DFSUPAA0) (システム・ユーティリティー) を参照してください。

HDRCTL=

DEV TYPE=DPM-An および DIV TYPE=OUTPUT の場合のみ、出力メッセージ・ヘッダーの特性を指定します。

FIXED

完全に埋め込まれた出力メッセージ・ヘッダーが、リモート・プログラムに送信される指定です。固定出力メッセージ・ヘッダーの構造は、この FMT 定義を使用している DPM 出力メッセージの構造と同じです。基本となる DPM 出力メッセージ・ヘッダーは、長さが 7 で、バージョン ID を含んでいます。

VARIABLE

MIDNAME および DATANAME が末尾のブランクを省略し、それに応じて長さフィールドが行末調整されることを指定します。 MIDNAME を使用しないときは、MIDNAME フィールド自体もその長さも存在しません。

nn ヘッダーの最小長を指定します。このヘッダーの最小長とは、MFS フィールドを除く基本ヘッダーのことです。デフォルトは 7 で、これは DPM の基本メッセージ・ヘッダーの長さです。7 以外の値を指定すると、リモート・プログラムに誤った結果が生じることになります。

パラメーター RDPN=、DPN=、PRN=、および RPRN= は、ISC ATTACH 機能管理ヘッダーおよび等価の ISC SCHEDULER 機能管理ヘッダーの両方を参照します。

RDPN=

DIV TYPE=INPUT の場合、*dfldname* を指定すると、この *dfldname* を参照している入力メッセージ MFLD に、推奨する戻り宛先プロセス名 (RDPN) を提供することができます。 *dfldname* が指定されていなければ、RDPN は入力メッセージに提供されません。

DPN=

DIV TYPE=OUTPUT の場合、'*literal*' を指定すると、このリテラルを出力 ATTACH メッセージ・ヘッダーの DPN として使用するように MFS に要求

することになります。 *literal* は 8 文字までで、必ず単一引用符で囲むようにします。*dfldname* も指定されていれば、この *dfldname* を参照している MFLD で提供されるデータが、出力 ATTACH メッセージ・ヘッダーで DPN として使用されます。*dfldname* を参照する出力メッセージ MFLD がないときは、*literal* が使用されます。*dfldname* を参照している MFLD の中のデータが 8 文字を超える場合は、初めの 8 文字が使用されます。

PRN=

DIV TYPE=INPUT の場合、*dfldname* を指定すると、この *dfldname* を参照している入力メッセージ MFLD に、推奨する 1 次リソース名 (PRN) を提供することができます。*dfldname* が指定されていなければ、アプリケーション・プログラムへの入力メッセージに PRN は提供されません。

DIV TYPE=OUTPUT の場合、'*literal*' を指定すると、 *literal* を出力 ATTACH メッセージ・ヘッダーの PRN として使用するように MFS に要求することになります。 *literal* は 8 文字までで、必ず単一引用符で囲むようにします。*dfldname* も指定されていれば、この *dfldname* を参照している MFLD で提供されるデータが、出力 ATTACH メッセージ・ヘッダーで PRN として使用されます。*dfldname* を参照する出力メッセージ MFLD がないときは、'*literal*' が使用されます。*dfldname* を参照している MFLD の中のデータが 8 文字を超える場合は、初めの 8 文字が使用されます。

RPRN=

DIV TYPE=INPUT の場合、*dfldname* を指定すると、この *dfldname* を参照している入力メッセージ MFLD に、推奨する戻り 1 次リソース名 (RPRN) を提供することができます。 *dfldname* が指定されていなければ、アプリケーション・プログラムへの入力メッセージに RPRN は提供されません。

DIV TYPE=OUTPUT の場合、'*literal*' を指定すると、*literal* を出力 ATTACH メッセージ・ヘッダーの戻り 1 次リソース名 (RPRN) として使用するよう MFS に要求します。*literal* は 8 文字までで、必ず単一引用符で囲むようにします。*dfldname* も指定されていれば、この *dfldname* を参照している MFLD で提供されるデータは、出力 ATTACH メッセージ・ヘッダーで RPRN として使用されます。 *dfldname* を参照する出力メッセージ MFLD がないときは、'*literal*' が使用されます。*dfldname* を参照している MFLD の中のデータが 8 文字を超える場合は、初めの 8 文字が使用されます。

OFTAB=

メッセージの出力データ・ストリームに、出力フィールドのタブ分離文字を挿入するよう MFS に指示します。OPTIONS=DNM および OFTAB の場合、OFTAB 文字は DD ヘッダー内に置かれ、標識は MIX または ALL にセットされます。OPTIONS=NODNM の場合、DD ヘッダーは送信されません。

X'*hh*'

出力フィールドのタブ分離文字として使用する 16 進文字 (*hh*) を指定します。X'3F' および X'40' は無効です。

C'*c*'

出力フィールドのタブ分離文字として使用する文字 (*c*) を指定します。この文字にブランク (**C' '**) を指定してはなりません。

指定された文字は、IMS アプリケーション・プログラムからのデータ・ストリーム内に入れることはできません。データ・ストリーム内に入れられると、その文字はブランク (X'40') に変更されます。

出力フィールド・タブ分離文字を定義するときは、MIX と ALL の一方も指定できます。デフォルトは MIX です。

MIX
データの入っていない個別フィールド、または定義されている DFLD の長さよりも短いデータの入った個別フィールドに、それぞれ出力フィールドのタブ分離文字が挿入される指定です。

ALL
データの長さに関係なく、あらゆるフィールドに、出力フィールドのタブ分離文字を挿入する指定です。

COMPR=
短フィールド、固定長フィールド、またはアプリケーション・プログラムによって提供される全フィールドから、末尾のブランクを取り除くよう MFS に指示します。

DPM-An 装置の場合、次の項目がすべて指定されていれば、セグメントの終わりで末尾のブランクが除去されます。

- FILL=NULL または FILL=PT
- マップされる現行セグメントに GRAPHIC=YES
- MSG セグメントに OPT=1 または OPT=2

これらの条件に合っていれば、末尾のブランクは次のように置き換えられます。

FIXED
固定長フィールドの末尾のブランクをヌルに置き換える指定です。

SHORT
アプリケーションによって短縮されたフィールドの末尾のブランクを、ヌルに置き換える指定です。

ALL
あらゆるフィールドの末尾のブランクを、ヌルに置き換える指定です。

末尾のヌルは、レコードの最後で圧縮されます。MFLD ステートメントの FILL= オペランドについて詳しくは、MFS 言語ユーティリティー (DFSUPAA0) (システム・ユーティリティー) を参照してください。

DPM-Bn 装置の場合は、次の項目のすべてが指定されていれば、末尾のブランクが除去されます。

- OFTAB (現行の DIV ステートメント上で)、FILL=NULL、または FILL=PT
- マップされる現行セグメントに GRAPHIC=YES
- MSG セグメントに OPT=1 または OPT=2

これらの条件を満たしていれば、末尾のブランクは次のように除去されます。

FIXED
固定長フィールドから末尾ブランクを取り除きます。

SHORT
アプリケーションによって短縮されたフィールドの末尾のブランクを取り除く指定です。

ALL
すべてのフィールドから末尾ブランクを取り除きます。

関連概念:

590 ページの『SLU P DPM-An の出力形式制御』

597 ページの『末尾ブランクの圧縮』

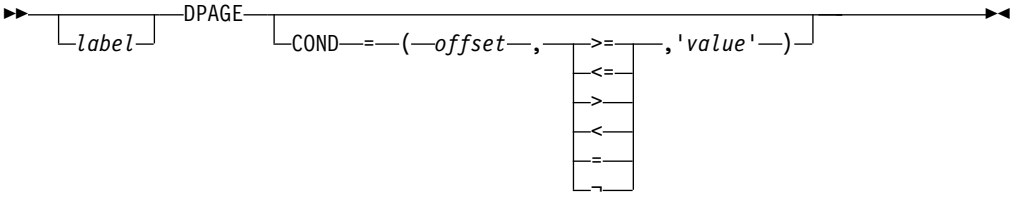
関連資料:

559 ページの『DPM-An 用ヌル文字の任意削除』

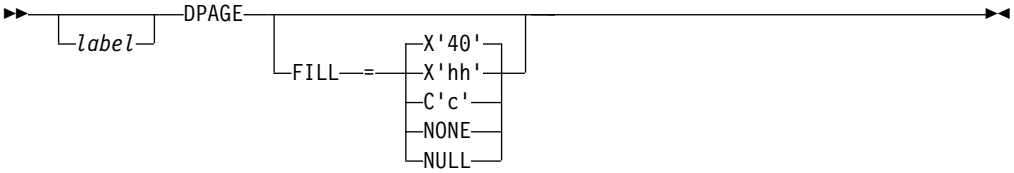
DPAGE ステートメント**:**

DPAGE ステートメントは、装置形式の論理ページを定義します。この装置形式 (FMT) を参照するメッセージ記述子に LPAGE ステートメントが含まれていない場合、さらに、特定の装置オプションが要求されていなければ、このステートメントは省略することができます。

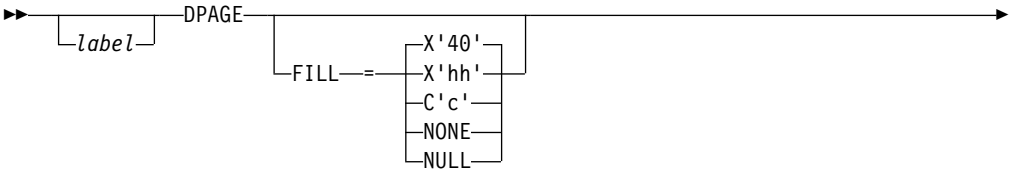
DEV TYPE=DPM-An または **DPM-Bn** で、**DIV TYPE=INPUT** のときのフォーマット

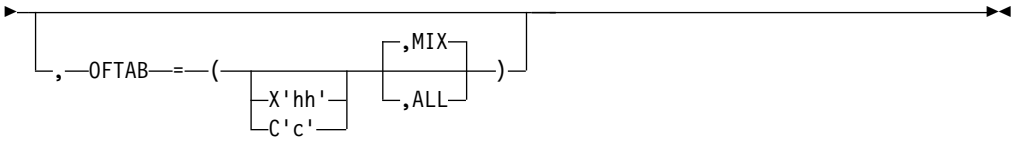


DEV TYPE=DPM-An で、**DIV TYPE=OUTPUT** のときのフォーマット

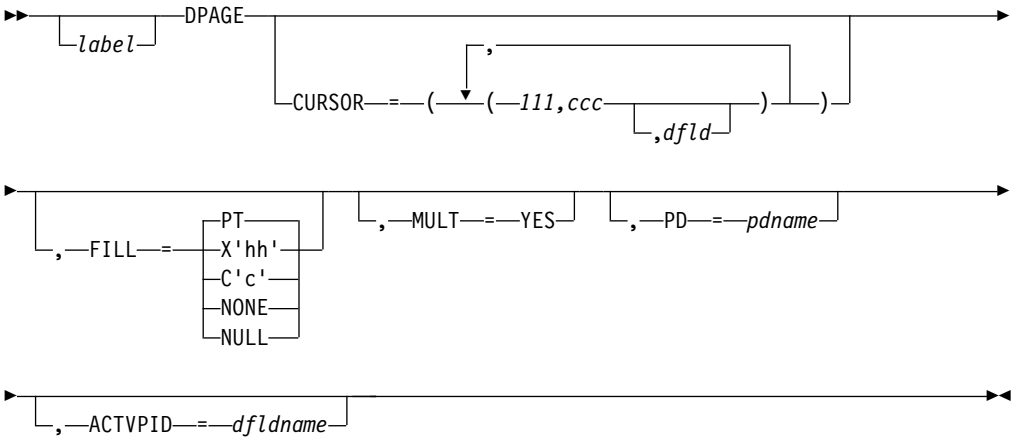


DEV TYPE=DPM-Bn で、**DIV TYPE=OUTPUT** のときのフォーマット

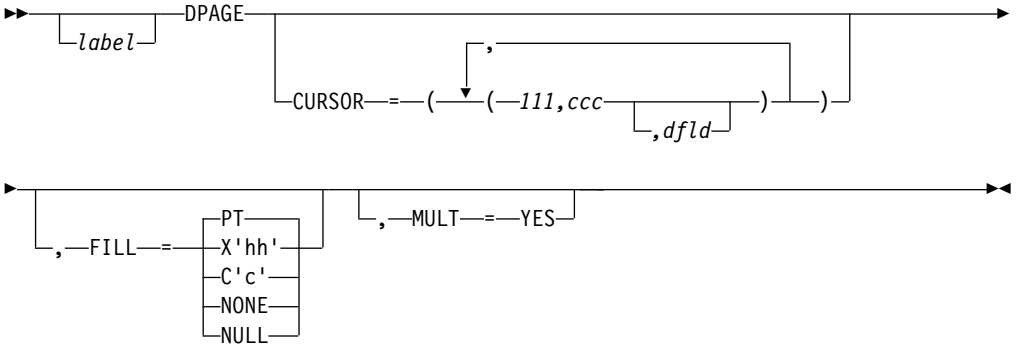




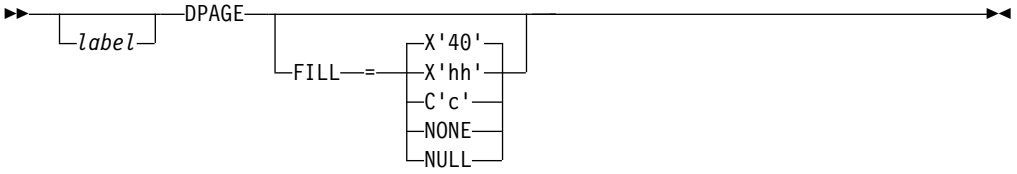
DEV TYPE=3270-An のときのフォーマット



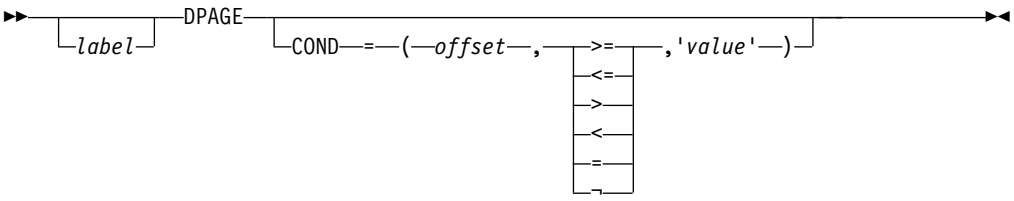
DEV TYPE=3270 のときのフォーマット



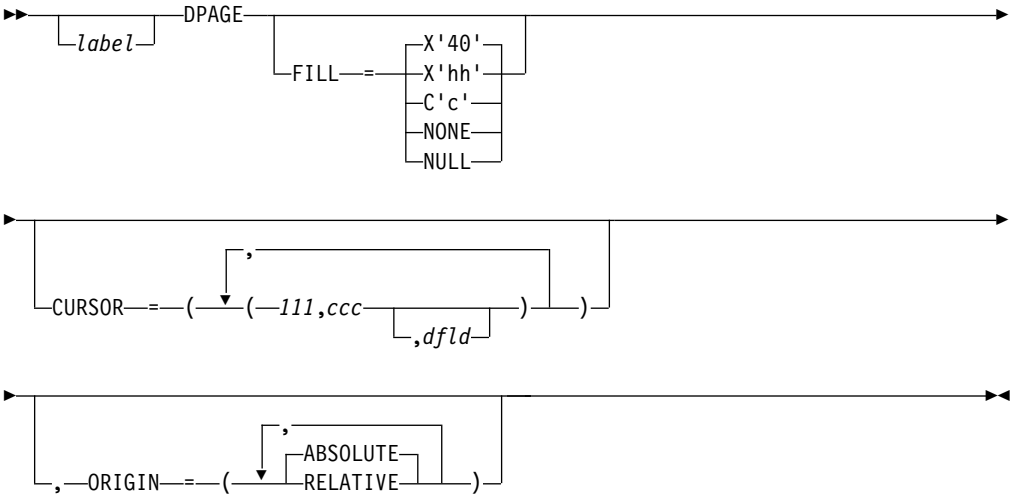
DEV TYPE=3270P のときのフォーマット



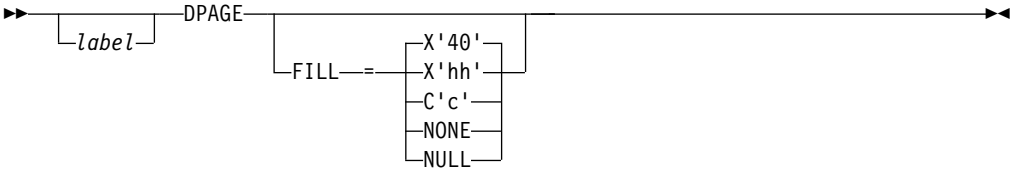
DEV TYPE=FIN のときのフォーマット



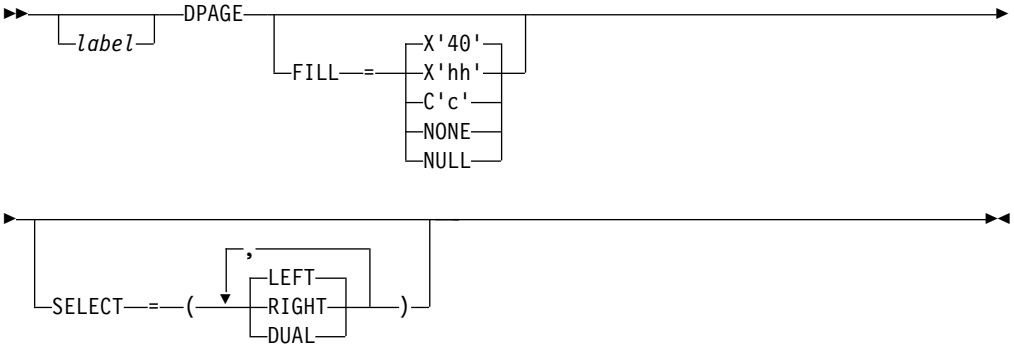
DEV TYPE=FIDS、**FIDS3**、**FIDS4**、**FIDS7** のときのフォーマット



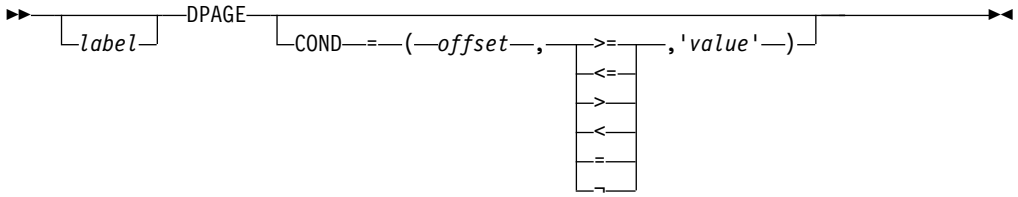
DEV TYPE=FIJP または **FIPB** のときのフォーマット



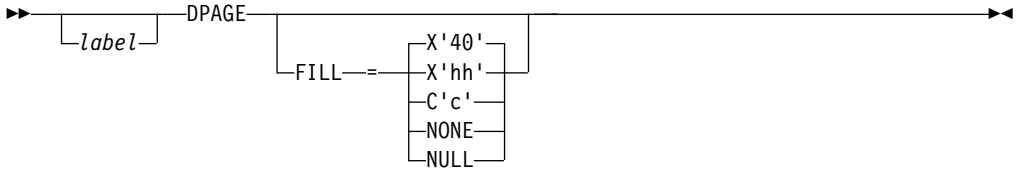
DEV TYPE=FIFP のときのフォーマット



DEV TYPE=SCS1 または **SCS2** で、**DIV TYPE=INPUT** のときのフォーマット



DEV TYPE=SCS1 または **SCS2** で、**DIV TYPE=OUTPUT** のときのフォーマット



パラメーター

label

この装置形式が LPAGE SOR= 参照を含んでいるとき、あるいはこの装置に対して 1 つの DPAGE ステートメントのみが定義されているとき、1 バイトから 8 バイトの英数字名を指定できます。 1 つの FMT 定義内で複数の DEV ステートメントが定義されているときは、そのそれぞれが同じラベルをもつ DPAGE ステートメントを含んでいなければなりません。

装置タイプが DPM-An で、DIV ステートメントで OPTIONS=DPAGE が指定されているときは、この名前がデータ名として、出力メッセージ・ヘッダーを通じてリモート・プログラムに送られます。 *label* が省略されると、MFS は診断名を生成し、ヘッダー内のリモート・プログラムに送信します。 DPAGE ステートメントが省略されると、FMT ステートメント上のラベルが出力メッセージ・ヘッダーに送信されます。OPTIONS=DNM であれば、FMT ステートメント上のラベルは、DD ヘッダー内の DSN として送信されます。

COND=

最初の入力レコードに対して行う条件付きテストを指定します。指定するオフセットは、ゼロを基準にして指定します。オフセットを指定するときは、必ず入力レコードの LLZZ フィールドを考慮に入れておく必要があります (例えば、最初のデータ・バイトは、オフセット 4 になります)。条件が満たされれば、この DPAGE のあとに続く定義済みの DFLD を使用して、入力を形式設定することができます。どの条件も満たされないときは、最後に定義されている DPAGE が使用されます。ただし、その最後に定義されている DPAGE に COND= が指定されている場合を除きます。最後の DPAGE に COND= パラメーターが指定されていて、しかもその条件が満たされないと、入力メッセージは拒否されます。メッセージ入力定義では、複数の LPAGE 定義が認められています。

このキーワードが指定され、DIV ステートメントに OPTIONS=NODNM が指定されていると、この指定が DPAGE 選択に使用されます。このキーワードが

指定され、DIV ステートメントに OPTIONS=DNM が指定されていると、COND= 指定は無視され、DD ヘッダーにあるデータ構造名が DPAGE 選択に使用されます。

COND= 比較の際、金融機関、SCS1、SCS2 のキーボードから入力された英小文字が、英大文字に変換されることはありません。したがって、リテラル・オペランドも小文字で指定しなければなりません。

FILL=
出力装置フィールドに使う充てん文字を指定します。デフォルト値は、3270 ディスプレイを除くすべての装置タイプで X'40'、3270 ディスプレイでは PT です。 3270 出力に EGCS フィールドがあるときは、FILL=PT または FILL=NULL だけを指定してください。 FILL=PT では、出力フィールドにデータが送られる場合のみ、そのフィールド (1 バイトまたは 2 バイト) が消去されます。したがって、アプリケーション・プログラム・メッセージで MFLD が省略されていると、DFLD は消去されません。 DPM-Bn では、OFTAB が指定されていると、FILL= は無視され、FILL=NULL と見なされます。

NONE
装置フィールドを埋めるときにメッセージ出力記述子にある充てん文字を使うときは、これを指定してください。

X'*hh*'
装置フィールドを埋める 16 進文字 (*hh*) を指定します。

C'*c*'
装置フィールドを埋める 1 つの文字 (*c*) を指定します。

NULL
フィールドを充てん文字で埋めないことを意味します。 3270 表示装置以外の装置では、メッセージ・データが装置フィールドを埋めない場合、短縮行が作成されます。

DPM-An 装置では、リモート・プログラムまたはサブシステムに送信されるあらゆるレコードから、末尾のヌル (X'3F') が除去されます。取り除かれる末尾ヌルは、最初の非ヌル文字までで、非ヌル文字に挟まれているヌル文字はそのまま送信されます。レコード全体がヌルながら、その後ろにさらにデータ・レコードが続くときは、ヌルを 1 個だけ含んだレコードがリモート・プログラムに送信されます。レコード全体がヌルで、その後ろにさらにレコードが続いていて、しかも OPTIONS=MSG または DPAGE であるか、PPAGE で OPTIONS=PPAGE であれば、その DPAGE または PPAGE の末尾に至るまでのすべてのヌル・レコードが削除されます。

PT 3270 ディスプレイ以外では、NULL と同じです。 3270 ディスプレイでは、装置フィールド (DFLD) をいっぱいに満たさない出力フィールドがあると、その後ろにプログラム・タブ文字が挿入され、フィールドの既存のデータを消去します。それ以外は、FILL=NULL と同じです。

3270 表示装置では、X'3F' より小さな値を指定しておくと、制御文字は X'00'、その他の非グラフィック文字は X'40' に変更されます。その他の装置では、FILL=X'hh' または FILL=C'c' に X'3F' より小さな値が指定されると、その指定は無視され、デフォルトで X'3F' と見なされます (これは FILL=NULL 指定と同等です)。

MULT=YES

この DPAGE で、複数の物理ページにわたる入力メッセージが許可される指定です。

CURSOR=

物理ページ上のカーソル位置を指定します。 1 つの論理ページまたはメッセージが複数の物理ページからなっているときは、複数のカーソル位置が必要になることもあります。 *lll* の値には行番号を、*ccc* の値には桁を指定します。*lll* および *ccc* の値は、両方とも 1 以上でなければなりません。カーソル位置は、定義されたフィールドに指定するか、またはデフォルトでなければなりません。3270 表示装置の *lll,ccc* のデフォルト値は 1,2 です。 金融機関表示コンポーネントでは、カーソル位置が指定されていないと、MFS はカーソルの位置付けを行いません。このカーソルは、通常、装置の出力データの終わりに位置付けされているものです。金融機関ディスプレイ・コンポーネントでは、ORIGIN= パラメーターがどう指定されていても、すべてのカーソル位置付けが絶対位置になります。

dfld パラメーターは、アプリケーション・プログラムに入力の際のカーソル情報を供給する方法、および、出力の際にアプリケーション・プログラムでカーソル位置を指定する方法を提供します。

推奨事項**:** 出力のカーソル位置付けには、カーソル属性機能 (MFLD ステートメントに ATTR=YES を指定) を使用してください。

dfld パラメーターは、カーソル位置の入ったフィールドの名前を指定します。この名前は MFLD ステートメントで参照することができ、この DEV 定義にある DFLD ステートメントのラベルとして使用してはなりません。このフィールドのフォーマットは 2 個の 2 進ハーフワードからなり、一方に行番号、他方に桁番号が入ります。メッセージ入力記述子によって参照されるフィールドには、メッセージ項目にカーソル位置が入っています。メッセージ出力記述子がこのフィールドを参照するときは、アプリケーション・プログラムがそこにカーソル位置を 2 個の 2 進ハーフワードとして格納します。1 つには行番号、もう 1 つには桁番号が含まれています。このフィールドで 2 進ゼロを指定すると、*lll,ccc* に指定した値は、出力中のカーソルの位置付めで使用されることになります。入力時にこのフィールドに 2 進ゼロが入っていると、それはカーソル位置が定義されていないことを意味します。この *dfld* を参照する入力 MFLD は、GRAPHIC=NO が指定されているセグメント内で定義するか、あるいは EXIT=(0,2) を使用して 2 進数を 10 進数に変換する必要があります。

ORIGIN=

定義された各物理ページについて、金融機関ディスプレイにおけるページの位置付け方法を指定します。デフォルト値は ABSOLUTE です。

ABSOLUTE

前の画面を消去し、ページを 1 行目の 1 桁目の位置に定めます。DFLD ステートメントで指定された行および桁は、画面上で、データの実際の行および桁になります。

RELATIVE

出力時にカーソルが位置付けられていた行の次の行の 1 桁目を、ページ起点とします。装置へのあらゆる出力に一貫性をもたせるように計画しておかないと、好ましくない結果になることがあります。

OFTAB=

いま記述中の DPAGE の出力データ・ストリームに、DPAGE ステートメントで指定された出力フィールド・タブ分離文字を挿入するよう MFS に指示します。

X'*hh*'

出力フィールドのタブ分離文字として使用する 16 進文字 (*hh*) を指定します。X'3F' および X'40' は無効です。

C'*c*'

出力フィールドのタブ分離文字として使用する文字 (*c*) を指定します。この文字にブランク (**C' '**) を指定してはなりません。

指定された文字は、IMS アプリケーション・プログラムからのデータ・ストリームに入れることはできません。データ・ストリーム内に入れられると、その文字はブランク (X'40') に変更されます。

出力フィールド・タブ分離文字を定義するときは、MIX と ALL の一方も指定できます。デフォルト値は MIX です。

MIX

データをまったく含まなかったり、定義された DFLD 長に満たないデータしか含まなかったりするフィールドがあれば、そこに出力フィールド・タブ分離文字を挿入します。

ALL

データ長に関係なく、すべてのフィールドに出力フィールド・タブ分離文字を挿入する指定です。

SELECT=

前の DEV ステートメントで FEAT=DUAL を指定された FIFP 装置に対して、キャリッジ選択を指定します。適切な用紙が取り付けられているか、そして、その左マージンが正確に設定されているかどうかを、必ず確認するようにしてください。デフォルト値は LEFT です。

LEFT

この DPAGE で定義される対応物理ページを左プラテンへ送ります。

RIGHT

この DPAGE で定義される対応物理ページを右プラテンへ送ります。

DUAL

この DPAGE で定義される対応物理ページを左右両方のプラテンへ送ります。

PD=

(区画フォーマット・モードの 3180 と 3290 では) この DPAGE ステートメントと関連付けるべき区画の区画記述子名を指定します。このパラメーターにより、メッセージの論理ページと区画がマッピングされます。 PD の名前は、DEV ステートメントで指定された PDB ステートメントに含まれていなければなりません。

ACTVPID=

(区画フォーマット・モードの 3290 では) アクティブにしたい区画識別番号 (PID) を含んでいる、メッセージ中の出力フィールドの名前を指定します。この *dfldname* は、必ず MFLD ステートメントで参照しなければならず、DEV 定義

内の DFLD ステートメントのラベルとしては使用しないでください。アプリケーション・プログラムは、活動化する区画の PID をこのフィールドに格納します。 PID のフォーマットは 2 バイトの 2 進数で、その値は X'0000' から X'000F' でなければなりません。

制約事項: 3180 ではこのオペランドを指定しないでください。この装置では 1 つの区画しか使用できないため、アクティブ区画を指定する必要はありません。

MFS メッセージ・フォーマット設定機能

IMS は、MFS のメッセージ・フォーマット設定機能を提供します。制御ブロックは、各種の装置タイプ用のメッセージをフォーマット設定します。

入力メッセージ・フォーマット設定

以下の情報を使用して、MFS 入力メッセージをフォーマット設定します。

関連概念:

568 ページの『出力メッセージ・フォーマット設定オプション』

565 ページの『ISC (DPM-Bn) サブシステムの入力形式制御』

MFS が受け入れる入力メッセージ**:**

MFS が処理できるのは、MFS と動作するように IMS TM に対して定義されている装置から入力されたデータだけです。ただし、特定の入力メッセージについての MFS の使い方は、メッセージの内容によって異なり、多くの場合、その前の出力メッセージによって決まります。

3770、**SLU 1**、および **NTO**

3770、SLU 1、または NTO からのデータを MFS で処理するには、これらの装置で MFS が動作するように、IMS TM システム定義か、拡張端末オプション(ETO) が使用可能な場合はユーザー記述子で、これらの装置を定義する必要があります。

MFS が動作するように装置を定義した後も、次のいずれかの状態が発生するまで端末は不定様式モード (MFS ではなく、基本編集を使用) で動作します。

- *//midname* が入力されて IMS に送信された。
- 端末への出力メッセージがメッセージ出力記述子 (MOD) を用いて処理され、しかもその MOD には、以後の入力データを処理するために使われるメッセージ入力記述子 (MID) が指定してある。

//midname を受信すると、MFS に制御権が渡されるため、MFS は指定された MID を用いてデータを編集します。*//midname* の後にデータが続いていると (データも入力する場合は *//midname* のあとに 1 個のブランクを続ける)、MFS は *//midname* とブランクを廃棄し、指定された MID に基づいて残りのデータをフォーマット設定します。 *//midname* のあとにデータが続いていなければ、MFS は、端末から送られてくる次の行をメッセージの第 1 行と見なします。

出力メッセージが MOD を用いて処理され、その MOD に MID 名が指定してあると、その端末からの次の入力は、その MID を用いてフォーマット設定されま

す。この出力メッセージは、アプリケーション・プログラム、IMS TM の /FORMAT コマンド、メッセージ通信、または別の IMS TM 機能で作成することができます。

『フォーマット設定モード』 (IMS TM 基本編集でなく MFS を使用) になると、以下のいずれかが起こるまで、その装置はフォーマット設定モードで作動し続けます。

- // または //b (// に 1 個のブランクが続く) を受信する。端末は不定様式モードに戻され、// (およびブランク) は廃棄されます。この 2 本のスラッシュはエスケープ文字です。
- //bH およびデータを受信する。端末は不定様式モードに戻され、// とブランクは廃棄され、データは IMS TM 基本編集によってフォーマット設定されます。
- MID を指定していない MOD を用いて、出力メッセージが端末に送信される。

3270 および **SLU 2**

すべての 3270 装置および SLU 2 装置は、MFS と動作するように自動的に定義されます。

制約事項**:** 以下の場合が、3270 装置および SLU 2 装置がフォーマット設定モードで作動しない状態です。

- 最初に電源オンにしたとき
- CLEAR キーを押した後
- 出力メッセージを処理するときに用いた MOD に、次に受信する入力データに使用する MID が指定されていない場合
- アプリケーション・プログラムが DFS.EDT または DFS.EDTN の MOD 名を使用することにより MFS をバイパスした場合

不定様式モードで作動しているときにこれらの端末から入力できるのは、IMS TM コマンド、VTAM に関する端末テスト要求、ページング要求、および MFS を必要としないトランザクション・コードまたはメッセージ通信データだけに限られます。

金融機関ワークステーションおよび **SLU P** ワークステーション

金融機関用ワークステーションまたは SLU P ワークステーションからのデータを MFS で処理するには、端末で MFS が動作するように、IMS TM システム定義か、ETO が使用可能な場合はユーザー記述子で、端末を定義する必要があります。MFS が動作するように装置を定義した後も、次のいずれかの状態が発生するまで、ワークステーションは不定様式モード (MFS ではなく IMS TM 基本編集を使用) で動作します。

- 金融機関または SLU P ワークステーションのリモート・アプリケーション・プログラムが、入力メッセージ・ヘッダーに MID の名前を指定することにより、MFS フォーマット設定を要求する。
- ワークステーション・オペレーターが *//midname* を入力し、それが入力メッセージそのものの最初の部分または唯一の部分として、リモート・アプリケーション・プログラムにより IMS TM に送信される。

SLU P を正しくフォーマット設定するには、入力メッセージ・ヘッダーにバージョン識別子 (バージョン ID) を入れます。このバージョン ID によって、入力メッセージのマッピングの際に正しいレベルの MFS 記述子 (装置入力形式または DIF) が使われることが保証されます。この検査が必要ない場合は、16 進数のゼロ (X'0000') のバージョン ID を送信するか、メッセージ・ヘッダーからバージョン ID を削除することができます。

SLU P または金融機関ワークステーションに送信される出力メッセージが、MID が指定されている MOD を用いてフォーマット設定される場合、IMS TM は MID の名前を、出力メッセージ・ヘッダーの一部としてワークステーションに送信します。IMS TM はこれらのシステムの端末装置を直接制御しません。したがって、IMS TM では、次の入力が正しい MID を用いて処理される保証はありません。リモート・プログラムは、MID 名を保管しておき、IMS TM に送信する次の入力メッセージに、その MID 名を DPN として組み込まなければなりません。

金融機関および SLU P ワークステーションは、現行メッセージに関連する MID または MOD が存在している場合だけフォーマット設定モードで作業を続けます。

システム間連絡 **(ISC)** サブシステム

ISC サブシステムからのデータを MFS で処理するには、IMS TM システム定義時に TYPE マクロの UNITYPE=LUTYPE6 として、または ETO ユーザー記述子で ISC サブシステムを定義しておかなければなりません。そのように定義した場合でも、ISC アプリケーション・プログラムが入力メッセージ・ヘッダーの DPN フィールドに MID の名前を指定して MFS フォーマット設定を要求するまでは、ISC サブシステムは不定様式モードで動作します (MFS ではなく IMS TM 基本編集または ISC 編集を使用)。

ISC サブシステムに送信される出力メッセージが、MID が指定されている MOD を用いてフォーマット設定される場合、IMS TM は MID の名前を出力メッセージ・ヘッダーの RDPN フィールド内の ISC サブシステムに送信します。 IMS TM は ISC サブシステムを直接制御しません。したがって、IMS TM では、次の入力が正しい MID を用いて処理される保証はありません。ISC アプリケーション・プログラムは、MID 名を保管しておき、IMS に送信する次の入力メッセージにその MID 名を組み込まなければなりません。

ISC サブシステムは、現行メッセージに関係する MID または MOD が存在している場合にのみ、フォーマット設定モードで作動し続けます。

事前設定宛先モードの端末から入力されるメッセージのフォーマット設定

事前設定宛先モードを使用すると、ある端末から入力されるすべてのメッセージの宛先を、1 つの宛先に固定することができます。事前設定宛先モードに入るには、/SET コマンドを使用します。端末が事前設定モードであると、すべての入力メッセージは (MFS または基本編集により処理されてから)、/SET コマンドで設定された宛先に送られます。入力メッセージにメッセージ宛先を入れる必要はありません。

IMS TM 基本編集で事前設定モードの端末からの入力を処理する場合、事前設定の宛先名は最初のセグメントの先頭に追加されます。しかし、MFS が事前設定モードの端末からの入力を処理する場合には、事前設定宛先名は最初のセグメントの先頭

に追加されず、入力メッセージの形式は、ユーザーによるメッセージ定義と入力とによって決まります。MFS には、オペレーターがメッセージの宛先を入力しなくても、入力セグメントの中にスペースを予約しておいたり、トランザクション・コードを挿入したりする方法が数多く用意されています。

高速機能を用いる場合のメッセージのフォーマット設定

高速機能を使用する場合は、すべてのメッセージが単一セグメント・メッセージでなければならないという制限はありますが、MFS の使用法は他の IMS TM アプリケーションの場合と同じです。

関連タスク:

- 拡張端末オプション (ETO) (コミュニケーションおよびコネクション)

関連資料:

- /SET コマンド (コマンド)
- /FORMAT コマンド (コマンド)
- システム間連絡編集の使用 (アプリケーション・プログラミング)

MFS による入力メッセージのフォーマット設定方法**:**

フォーマット設定モードで作動している MFS サポートの装置からの入力データは、メッセージ入力記述子 (MID) と装置入力形式 (DIF) の 2 つの MFS 制御ブロックの内容に基づいてフォーマット設定されます。 MID では、IMS TM アプリケーション・プログラムに表示するためにデータをどのようにフォーマット設定するかを定義すると同時に、入出力装置と関連付けられている DIF を指し示します。DIF では、その装置から受け取るデータを記述します。

MID により作成されたメッセージがコマンドである場合は、そのコマンドは、「*IMS V14* コミュニケーションおよびコネクション」に記載されているコマンド形式と構文規則に従っていなければなりません。

入力メッセージ・フォーマット設定オプション*:*

MFS は、3 種類のメッセージ・フォーマット設定オプションをサポートします。ユーザーがどのオプションを選択したかにより、MFS が MID 定義をどのように解釈するかが決まります。この解釈に基づいて、データをフォーマット設定してアプリケーション・プログラムに渡すためにメッセージ・フィールドに入れます。

MID では、1 つ以上の MFLD ステートメントを使用して、以下についてメッセージ・フィールドを記述します。

- 長さ
- 入力データを取り出す装置フィールド
- 装置データを受け取らないメッセージ・フィールドのリテラル・データ
- 入力データでメッセージ・フィールドがいっぱいにならない場合に使用する充てん文字
- フィールドの位置調整 (右寄せまたは左寄せ) または切り捨て (右端または左端) の指定

- フィールドの最初の 2 バイトを属性データ用として確保するかどうか

どのフォーマット設定オプションを使用するかは、MID の MSG ステートメント (OPT=) で指定します。あるアプリケーションについてどのオプションを選択するとよいかは、設計上の重要な意思決定であり、装置データ・ストリームの複雑さと可変性、使用されるプログラミング言語、特定のオプションのもとでアプリケーションを処理する場合のプログラムの複雑さなどを考慮して決定する必要があります。オプションの説明の中の NULL (ヌル文字) とは、X'3F' で表されるものです。

MFS オプション **1**

オプション 1 を使うことの効果は、ヌル充てん文字が定義されているかどうかで決まります。MFS 言語ユーティリティーで、オプション 1 メッセージのフィールドがどれもヌル充てん文字を使わないと定義されていると、以下のようになります。

- メッセージは、常に定められた個数のセグメントで構成される。
- 各セグメントは、常に定められた長さで、また定義されたすべてのフィールドが入る。
- どのフィールドにも、データだけ、データと充てん文字、または充てん文字だけが入る。

オプション 1 メッセージのフィールドがヌル充てん文字を使うものと定義されていると、以下のようになります。

- ヌル充てん文字が定義されていて、装置からの入力データがないフィールドは、メッセージ・セグメントから除かれる。このようにして 1 つのセグメントからすべてのフィールドが除去され、かつリテラルが定義されていない (明示的にもデフォルトでも) 場合は、セグメントが除去されます。これ以外の場合には、セグメントの長さが短くされ、そのセグメント内の後続フィールドの相対位置が変化します。
- ヌル充てんが定義されていて、受信する装置データだけではフィールドがいっぱいにならないフィールドには、埋め込み (充てん) は行われない。装置フィールドの受信文字数が入力データの文字数となります。これによって、セグメントの長さと、そのセグメントのすべての後続フィールドの相対位置が変化します。

MFS オプション **2**

オプション 2 のフォーマット設定はオプション 1 とほぼ同じですが、編集後にセグメントに装置からの入力データがまったく含まれていない場合の処理だけが異なります。そのようなセグメントがあり、しかも装置からの入力データが入っているセグメントがそのあとに続いていない場合には、メッセージは打ち切られます。メッセージの最後のセグメントが、装置からの入力データが入っている最後のセグメントになります。装置からの入力データが入っていないセグメントが作成され、そのあとに装置からの入力データが入っているセグメントが続いている場合には、ただ 1 バイトのデータ (X'3F') が入っているセグメントが作成され、それが埋め込みセグメント、つまりヌル・セグメントであることを表します。リテラルを入れるように定義されている最初のセグメントでこの状態が発生した場合、無効なトランザクション・コードが生成されます。これは MFS が装置入力データを受け取らないセグメントに明示的なリテラルまたはデフォルト・リテラルを挿入しないために起きます。

MFS オプション 3

オプション 3 によるフォーマット設定では、入力装置から受信したフィールドだけがプログラムに渡されます。セグメントがプログラムに渡されるのは、セグメントに装置から受信したフィールドが入っている場合だけです。セグメントは相対セグメント番号によって識別され、セグメントのフィールドはセグメント・オフセットにより識別されます。ヌル充てん文字を使用すると定義されているセグメントおよびフィールドは、いずれも可変長です。空のフィールド (データが入っていないフィールド) には、充てん文字が埋め込まれません。セグメントは、相対セグメント番号順にアプリケーション・プログラムに渡されます。各セグメントのフィールドは、セグメント・オフセット順に配列されます。

オプション 3 のメッセージには、MID で指定したリテラル (明示またはデフォルトの値) は組み込まれません。

会話型トランザクションでオプション 3 を使用する場合は、メッセージからトランザクション・コードが取り除かれません。これは、フィールドおよびフィールドのオフセットがテキストの中で維持されているためです。トランザクション・コードは、SPA にも入っています。

制約事項**:** オプション 3 の入力メッセージ形式を使用して、IMS TM コマンドを入力することはできません。ただし、IMS TM コマンドは、IMS 提供のデフォルト形式を用いて、消去後の画面から、あるいはユーザーが定義したオプション 1 およびオプション 2 の入力メッセージ形式から入力することができます。

関連概念:

499 ページの『入力メッセージ形式』

571 ページの『出力装置フィールド用の充てん文字』

関連資料:

507 ページの『装置依存の出力情報』

メッセージ・セグメント定義の例*:*

例では、まずメッセージ・セグメントの定義を示し、ついでオプション 1、2、および 3 の場合の内容、長さ (バイト数)、および各フィールドのタイプのコードを示します。

フィールド・タイプには、以下の表に示すようにラベルが付けられています。

表 *129.* 入力メッセージ・フィールド・タイプ :

タイプ・コード	説明
A	セグメントの全長。フィールド A、B、C も含みます。2 バイトの 2 進数。
B	Z1 フィールド IMS TM が使用します。
C	Z2 フィールド - フォーマット設定オプションを表します。1 バイトの 2 進数。
D	相対セグメント番号。2 バイトの 2 進数。
E	フィールド長。フィールド E、F の長さも含みます。2 バイトの 2 進数。

表 *129.* 入力メッセージ・フィールド・タイプ *(*続き*)*:

タイプ・コード	説明
F	定義されたセグメント内の相対フィールド・オフセット。2 バイトの 2 進数。
G	フィールド

注:

1. フィールド A、D、E、F の境界合わせは行われません。
2. フィールド A、B、D は、ハーフワード境界上になければなりません。これを行うには、GU または GN 呼び出しを IMS TM に対して出すときに、必ず入出力域を境界上に配置します。
3. PLITDLI インターフェースの場合、長さ (LL) フィールドは 2 進数フルワードとして宣言しなければなりません。LL フィールドの値は、セグメントの長さから 2 バイトを引いた値です。例えば、入力メッセージ・セグメントが 16 バイトであれば、LL は 14 バイトで、これは LL の長さ (4 バイト引く 2 バイト)、ZZ の長さ (2 バイト)、およびテキストの長さ (10 バイト) の合計を表します。

入力メッセージ形式の例**: 1**

以下の表は、入力メッセージの定義を記述しています。

表 *130.* 入力メッセージ定義の例*: 1*

セグメント番号	フィールド名	フィールド長	フィールド値
1	LL	2	0072
	ZZ	2	XXXX
	TRANCODE	8	YYYY
	テキスト	10	MAN NO.
	テキスト	50	NAME
2	LL	2	0059
	ZZ	2	XXXX
	テキスト	5	DEPT
	テキスト	50	LOCATION
3	LL	2	0064
	ZZ	2	XXXX
	テキスト	10	PART NO.
	テキスト	50	DESCRIPTION
4	LL	2	0019
	ZZ	2	XXXX
	テキスト	10	QUANTITY
	テキスト	5	ORDER PRIORITY

どのフィールドも、左寄せ、ブランク充てん文字が定義されています。

入力項目 :

フィールド名
　　入力

NAME
ABJONES

PART NO.
23696

DESCRIPTION
WIDGET

トランザクション・コードは、メッセージ入力記述子 (MID) からリテラルとして得られます。入力メッセージがアプリケーション・プログラムに渡されるときは、以下のいずれかの表で示されるようになります。

表 *131. 例 1:* オプション *1* のアプリケーション・プログラム・ビュー

セグメント番号	フィールド・タイプ	フィールド長	フィールド値
1	A	2	0072
	B	1	XX
	C	1	01
	TRANCODE	8	YYYY
	テキスト	10	ブランク
	テキスト	50	ABJONES
2	A	2	0059
	B	1	XX
	C	1	01
	テキスト	5	ブランク
	テキスト	50	ブランク
3	A	2	0064
	B	1	XX
	C	1	01
	テキスト	10	23696
	テキスト	50	WIDGET
4	A	2	0019
	B	1	XX
	C	1	01
	テキスト	10	ブランク
	テキスト	5	ブランク

表 *132. 例 1:* オプション *2* のアプリケーション・プログラム・ビュー

セグメント番号	フィールド・タイプ	フィールド長	フィールド値
1	A	2	0072
	B	1	XX
	C	1	02
	TRANCODE	8	YYYY
	テキスト	10	ブランク
	テキスト	50	ABJONES

表 132. 例 1: オプション 2 のアプリケーション・プログラム・ビュー (続き)

セグメント番号	フィールド・タイプ	フィールド長	フィールド値
2	A	2	0005
	B	1	XX
	C	1	02
	テキスト	1	X'3F'
3	A	2	0064
	B	1	XX
	C	1	02
	テキスト	10	23696
	テキスト	50	WIDGET

表 133. 例 1: オプション 3 のアプリケーション・プログラム・ビュー

セグメント番号	フィールド・タイプ	フィールド長	フィールド値
1	A	2	0060
	B	1	XX
	C	1	03
	D	2	0001
	E	2	0054
	F	2	0022
	G	50	ABJONES
2	A	2	0074
	B	1	XX
	C	1	03
	テキスト	2	0003
	D	2	0014
	E	2	0004
	F	2	23696
	G	2	0054
	F	2	0014
	G	50	WIDGET

オプション 3 の例では、第 1 セグメントにトランザクション・コードが入っていません。これはオプション 3 のセグメントにはリテラルが挿入されないからです。トランザクション・コードの妥当性検査はメッセージのフォーマット設定後に行われるため、会話モードまたは事前設定宛先モードの端末からこのメッセージを受信した場合以外はこのメッセージは拒否されます。

入力メッセージ形式の例**: 2**

各セグメントの定義は例 1 と同じです。フィールドの定義も例 1 とほぼ同じですが、以下のフィールドが例 1 と異なります。

フィールド名
　　内容

NAME
　　ヌル・パッド

DEPT　ヌル・パッド

LOCATION
　　ヌル・パッド

PART NO.
　　右寄せ、EBCDIC ゼロのパッド

QUANTITY
　　ヌル・パッド

入力項目 :

フィールド名
　　入力

NAME
　　ABJONES

PART NO.
　　23696

DESCRIPTION
　　WIDGET

PRIORITY
　　HI

トランザクション・コードは、3270 PF キーによるリテラルとして、または金融機関ワークステーションからの特殊データ・フィールドとして得られます。入力メッセージは、以下のいずれかの表で示す形式になります。

表 *134. 例 2:* オプション *1* のアプリケーション・プログラム・ビュー

セグメント番号	フィールド・タイプ	フィールド長	フィールド値
1	A	2	0029
	B	1	XX
	C	1	01
	TRANCODE	8	YYYY
	テキスト	10	ブランク
	テキスト	50	ABJONES
2	2 番目のセグメントは渡されません。セグメントのすべてのフィールドはヌル充てん文字で埋められ、このフィールドについては装置から入力データを受信しなかったためです。		

表 *134.* 例 *2:* オプション *1* のアプリケーション・プログラム・ビュー *(*続き*)*

セグメント番号	フィールド・タイプ	フィールド長	フィールド値
3	A	2	0064
	B	1	XX
	C	1	01
	テキスト	10	0000023696
	テキスト	50	WIDGET
4	A	2	0009
	B	1	XX
	C	1	01
	テキスト	5	HI

表 *135.* 例 *2:* オプション *2* のアプリケーション・プログラム・ビュー

セグメント番号	フィールド・タイプ	フィールド長	フィールド値
1	A	2	0029
	B	1	XX
	C	1	02
	TRANCODE	8	YYYY
	テキスト	10	ブランク
	テキスト	7	ABJONES
2	A	2	0009
	B	1	XX
	C	1	02
	テキスト	1	X'3F'
3	A	2	0064
	B	1	XX
	C	1	02
	テキスト	10	0000023696
	テキスト	50	WIDGET
4	A	2	0009
	B	1	XX
	C	1	02
	テキスト	5	HI

表 *136. 例 2:* オプション *3* のアプリケーション・プログラム・ビュー

セグメント番号	フィールド・タイプ	フィールド長	フィールド値
1	A	2	0029
	B	1	XX
	C	1	03
	D	2	0001
	E	2	0012
	F	2	0004
	G	8	TRANCODE
	E	2	0011
	F	2	0022
	G	7	ABJONES
2	A	2	0074
	B	1	XX
	C	1	03
	D	2	0003
	E	2	0014
	F	2	0004
	G	10	0000023696
	E	2	0054
	F	2	0014
	G	50	WIDGET
3	A	2	0015
	B	1	XX
	C	1	03
	D	2	0004
	E	2	0009
	F	2	0014
	G	5	HI

カーソル位置入力と ***FILL=NULL:***

MFS では、入力の際にアプリケーション・プログラムにカーソル位置を知らせる場合に、問題が発生することがあります。

問題は以下の場合に発生します。

- 入力メッセージがフォーマット設定オプション 1 または 2 を使用する場合
- カーソル位置データ用の MFLD を定義したセグメントにヌル充てん文字 (FILL=NULL) を使用する MFLD を少なくとも 1 つ定義した場合

上記のことが起こった場合は、カーソル位置 63 (X'3F') は、通常の 4 バイト・フィールドではなく、カーソル・データが圧縮されて入っている 3 バイト・フィールドになります。問題が生じる可能性のある MFLD には、メッセージ 『DFS1150』 のフラグが付けられます。

この問題を避けるには、カーソル・データ・フィールドの MFLD ステートメントを、EXIT=(0,2) の指定に変更します。こうしておけば、IMS TM 提供のフィールド編集ルーチンが、カーソル・データ・フィールドの内容を 2 進数から EBCDIC に変換します。アプリケーション・プログラムも、EBCDIC 形式を処理できるように変更しなければなりません。

入力論理ページの選択*:*

入力論理ページ (LPAGE) により、アプリケーション・プログラムに送られる入力メッセージの内容が決まります。入力論理ページは、関連するメッセージ・セグメント定義およびフィールド定義からなる、ユーザー定義グループです。入力 LPAGE は、LPAGE ステートメントで識別します。LPAGE ステートメントがないと、MSG のすべてのメッセージ・フィールド定義が 1 つの LPAGE として扱われます。LPAGE ステートメントによって識別される 1 つの入力 LPAGE は、1 つ以上の入力装置ページ (DPAGE) を参照することができます。

入力 DPAGE は、入力 LPAGE で使用することができる装置形式を定義します。これは、ユーザーが定義した装置フィールド定義グループから構成されます。入力 DPAGE は、DPAGE ステートメントで識別します。DPAGE ステートメントがないと、DIV ステートメント以後のすべての装置フィールド定義は 1 つの DPAGE と見なされます。複数の DPAGE を定義する場合は、DPAGE ステートメントのそれぞれにラベルを付ける必要があります。ラベルの付いた DPAGE ステートメントにより識別される DPAGE は、LPAGE ステートメントで参照しなければなりません。

3270 と SLU 2 の装置入力データは、常に、現在表示されている DPAGE により処理されます。他の装置の場合、その形式に複数の DPAGE が定義されている場合は、装置から受信する最初の入力レコードに対して条件テストが行われます。このテストの結果、入力データを処理する DPAGE が選択されます。入力メッセージのフォーマット設定には、選択された DPAGE を参照する LPAGE が使用されます。

入力 LPAGE が定義されていない場合、メッセージ・フィールドは、どの DPAGE の装置フィールドでも参照することができます。ただし、どの入力メッセージにおいても、そこで使用できる装置入力データは、1 つの DPAGE で定義されているフィールドのものに限られます。

入力メッセージのフィールド編集ルーチンとセグメント編集ルーチン*:*

プログラミングを簡略化するために、MFS アプリケーションの設計者は、数値の妥当性検査やブランクから数値のゼロへの変換などの一般的な編集機能を実行する際に、SLU P 装置を除くすべての装置で、入力メッセージのフィールド編集ルーチンとセグメント編集ルーチンを使用することを考えてください。

フィールド編集ルーチンやセグメント編集ルーチンを既存のアプリケーションで使用することはありませんが、従来、個々のアプリケーション・プログラムごとにコーディングしなければならなかった機能を、標準のフィールド編集ルーチンで代行できるので、新しいアプリケーションについてはプログラミングを単純化することができます。「*IMS V14* 出口ルーチン」に、IMS が提供するフィールド編集ルーチンとセグメント編集ルーチンがリストされています。通常、編集はリモート・プ

ログラムが行うため、SLU P (DPM-An および ISC) 装置では、入力メッセージのフィールドまたはセグメント出口ルーチンを使用不可にすることができます。

フィールド編集ルーチンおよびセグメント編集ルーチンを使用すると、IMS TM 制御領域で余分な処理が行われることになるので、使用頻度が高い場合にはパフォーマンスが大きく低下します。ただし、これらの編集ルーチンは、メッセージ処理領域での処理時間を短縮し、ロギング時間とキューイング時間を短縮し、さらにアプリケーション・プログラムのスケジューリングを行わないでフィールドの検査と訂正を行えるようにすることによって、パフォーマンスを向上させることができます。効率のよいユーザー作成ルーチンを作成することが、最も重要です。

これらのルーチンは IMS TM 制御領域で実行されるので、編集ルーチンが異常終了すると、IMS TM 制御領域も異常終了します。

IMS 提供のフィールド編集ルーチンとセグメント編集ルーチン

IMS TM は、MFS アプリケーションの設計者に必要なフィールド編集ルーチンとセグメント編集ルーチンの両方を提供します。「*IMS V14* 出口ルーチン」は、IMS 提供のルーチンをリストしています。

z/OS のもとでは、これらの IMS 提供ルーチンの代わりとして使用するプログラム・コードは、RMODE 24、AMODE 31 で実行可能であって、さらに、31 ビット・アドレッシングが可能なリソースを参照しない場合であっても 31 ビット・アドレッシングを行えなければなりません。AMODE とはアドレッシング・モードのことで、AMODE 31 でモジュールを実行すると、拡張アーキテクチャー処理装置は、命令アドレスとデータ・アドレスがともに 31 ビット長であると見なします。

フィールド編集ルーチン **(DFSME000)**

フィールド編集ルーチンの機能は、エントリー・ベクトルに基づいて決定されます。3 つのフォーマット設定オプションをすべて扱うことができます。ただし、オプション 1 と 2 の場合には、MFLD ステートメントで FILL=NULL が指定されているときにエントリー・ベクトル 1 を与えると、好ましくない結果を生じることがあります。

入力メッセージ・リテラル・フィールド*:*

入力メッセージ・フィールドは、MID の定義時に指定するリテラル・データを入れるように定義することができます。 MFS が常に入力メッセージの一部として挿入するデフォルト・リテラルを定義することができる。装置から受け取るフィールドのデータがない場合、MFS が入力メッセージの一部として挿入するリテラルを定義することもできる。

デフォルト・リテラルを使用すれば、アプリケーションのプログラミングをさらに簡略化することができます。すなわち、デフォルト・リテラルを使用すれば、アプリケーション・プログラムで、『データなし』状態を調べたり、例外処理を行ったりする必要がなくなります。また、デフォルト・リテラルにより、オペレーターが入力したデータが 0 の値なのか、あるいは『データが入力されなかった』状態であるのかを、アプリケーション・プログラムで見分けることが可能になります。

例えば、以下の MFLD 定義について考えてみましょう。

```
MFLD (DFLD1,'NO DATA'),LTH=7,JUST=R,FILL=C'0'
```

例えば、アプリケーション・プログラムは、オペレーターによる入力を以下のように表示します。

オペレーターによる入力
: プログラムによる表示データ

296 0000296

0 0000000

データ入力なし
: NO DATA

デフォルト・リテラルを使用しないと、値 0 を入力した結果とデータを入力しない場合の結果はどちらも 0000000 になります。

アプリケーション・プログラムを変更しなくてもデフォルト・リテラルを変更することができます。また、複数のメッセージ記述子または入力論理ページを使用すれば、複数のデフォルト・リテラルを使うことができます。

入力メッセージ・フィールドに装置からデータが入力されなかった場合、デフォルト・リテラルを使用することにより、装置独立の方法でそのフィールドにデータを挿入できるため、装置の独立性は高まります。デフォルト・リテラルのこの機能は、 3270 装置または SLU 2 装置でよく使われます。これらの装置の入力の装置形式は出力の形式と同じです。これらの装置では、入力時にデフォルト・リテラルを指定する場合 (MID)、デフォルト (トランザクション・コード、データ、あるいはその両方) を与えることができます。

入力メッセージ・フィールド属性データ*:*

入力メッセージの非リテラル・フィールドは、属性データ、拡張属性データ、あるいはその両方を入れるように定義することができます。

そのように定義されたメッセージ・フィールドについては、MFS が最初のいくつかのバイトをブランクに初期設定し、そこを属性データまたは拡張属性データ (またはその両方) を入れる場所として確保します。これらの先頭の数バイトは、フィールド編集ルーチンによって、または出力メッセージを作成するときに充てんされます。属性バイトまたは拡張属性バイト (またはその両方) のスペースを指定する場合は、そのスペースをフィールド長に含めておかなければなりません。

アプリケーション・プログラムが入力メッセージを更新して、装置に送り返すこともあります。出力メッセージで属性データを使用する場合、入力メッセージにも属性データ・バイトを定義しておけば、アプリケーション・プログラムでのメッセージ定義が簡単になります。

フィールド編集ルーチンを使用する場合は、エラーを起こしたフィールドに属性バイトをセットするように設計しておくことができます (IMS に用意されているフィールド編集ルーチンは、このように設計されています)。このように、エラーのあるフィールドを、セグメント編集ルーチンが装置にメッセージを返す前に強調表示することができます。この場合、アプリケーション・プログラムは属性バイトの処理に関係しません。

IMS TM パスワード*:*

入力メッセージ定義では、入力メッセージの一部としてIMS TM パスワードを定義することができます。入力メッセージに 1 つ以上のフィールドを定義しておいて、その全体が IMS TM パスワードを構成するようにします。

このパスワード定義方法を使用すると、ユーザーが入力したフィールド・データ、3270、SLU 2、 3770 のオペレーター識別カード読取機構で読み取ったデータ、または 3270 磁気ストライプ読取装置で読み取ったデータからパスワードを作成することができます。

推奨事項**:** SLU 2 または 3270 を使用する場合は、IMS TM パスワードの位置として特定の装置フィールドを定義することもできますが、入力メッセージ・フィールドと装置フィールドの両方が定義されている場合は、このセクションで述べている方法が優先されます。

入力メッセージ・フィールド用充てん文字*:*

装置から受け取ったデータの長さが指定のフィールド長より短い場合、フィールドのデータを受け取っておらず、しかもデフォルト・リテラルが定義されていない場合、あるいは SLU P から受け取ったデータにヌル文字が入っていて NULL=DELETE が指定されている場合は、MFS は充てん文字を使用してメッセージ・フィールドを埋め込みます。

使用できる充てん文字は、ブランク (X'40')、任意の EBCDIC 16 進文字 (X'hh')、または EBCDIC 図形文字 (C'c') のいずれかです。ヌル圧縮を選択することもできます。ヌル圧縮では、データが欠落している分だけ、メッセージが左方向に詰められます。 MFS が実際にどのようにメッセージ・フィールドに埋め込みを行うかは、どの充てん文字が選択され、どのメッセージ・フォーマット設定オプションが使用されるかによって異なります。

入力モード *(3270*、*SLU 2*、*ISC* サブシステム以外の装置*):*

MFS は、MFS 言語ユーティリティー・プログラムに対して定義されたとおりの順序で入力メッセージ・フィールドが入力されるものと想定しています。 SLU 2 と 3270 以外の装置については、MFS アプリケーション設計者は、フィールドの定義方法および MFS によるそれらのフィールドのスキャン方法を選択することができます。レコード・モードとストリーム・モードのいずれか一方を選択できます。デフォルトのモードはレコード・モードです。

レコード・モードの場合 :

- 入力装置から送られてくる特定レコード (3770、または SLU 1 からの 1 行または 1 枚のカード、金融機関または SLU P ワークステーションからの 1 回の送信) 内に入力フィールドが存在するように定義される。
- 各フィールドを複数のレコードにまたがらせることはできません。
- あるレコードに対して定義されているフィールドは、必ずそのレコード内に存在していなければならない。そうしないと MFS での処理対象にされません。
- MFS がレコードの終わりを認識すると、現フィールドはそこで打ち切られ、そのレコードに対して定義されているその他のフィールドは、装置データなし (メッセージ充てん) として処理される。

- IMS TM が受信したレコードに規定のフィールド数より多くのデータ・フィールドが入っている場合、余分なデータ・フィールドは MFS により無視される。

金融機関または SLU P ワークステーションのリモート・プログラムからの入力データの場合、最初のレコードのデータ・フィールドが、同じレコードに入らないときは、入力メッセージ・ヘッダーまた *//midname* を別個に送信することができます。入力メッセージ・ヘッダーまたは *//midname* のあとにデータが続いていなければ、MFS は次に受信する伝送内容を入力メッセージの最初のレコードと見なします。

ストリーム・モードの場合 :

- 各フィールドはデータの連続した流れ (ストリーム) として定義されるため、レコード境界には左右されない。
- 各フィールドを複数の入力レコードにまたがらせることもでき、定義された順番どおりに入力するかぎり、どの入力レコードからでもフィールドを入力することができる。

入力フィールド・タブ ***(3270*** と ***SLU 2*** 以外の装置***):***

入力フィールド・タブ (FTAB) は、DEV ステートメントに定義される文字で、入力したデータの長さが定義したフィールド長より短い場合、またはフィールドにデータが指定されていない場合に入力フィールドを区切るのに使用します。 FTAB を検出すると、MFS の入力スキャンは、定義されている次のフィールドの最初の位置に移動します。FTAB は、3270 または SLU 2 以外の装置からの入力だけで定義することができます。FTAB が設定されていない場合、各装置入力フィールドは定義されているとおりの長さであると見なされます。

入力フィールドの分離文字として選択する文字は、データ・ストリームでユーザー・データとして使われない文字でなければなりません。FTAB が固有でないと、MFS によってデータが誤って解釈される場合があります。

例えば、以下の図は、DFLD フィールド定義と、その定義の結果である装置形式を示しています。

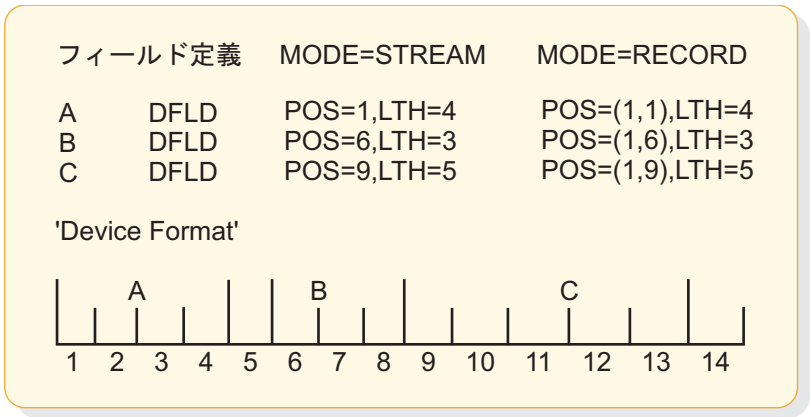


図 *30. FTAB* 条件の記述

FTAB を定義するときには、FORCE、MIX、または ALL のいずれかを指定して、FTAB の用途を限定します。上記の図は、FTAB 制限によって、オペレーターが 3 つのフィールドからなる可変のメッセージを入力した後に、MFS 入力スキャンの結果がどのように影響を受けるかを示しています。

以下の図に、FTAB 指定によって得られた正しい結果および正しくない結果の例をいくつか示します。両方向を指す矢印は、FTAB 制限が入力スキャンに影響しないことを表します。入力例 2、3、6 はいずれも、FTAB 制限の指定値にかかわらず常に正しい結果をもたらします。しかし、例 8 は常に好ましくない結果をもたらします。以下のセクション (FORCE、MIX、および ALL) では、好ましくない結果を表しているのはどの例であるか、さらに、これらの結果がなぜ好ましくないのかについて検討します。

FORCE

FORCE はデフォルト値です。FTAB が見つかるまでは、各装置入力フィールドは定義されたとおりの長さであると見なされます。最初の FTAB が見つかると、現行フィールドのデータの終わりになります。FTAB に続くデータ・バイトは、次のフィールドの第 1 バイトであると見なされます。レコード・モードでは、現行レコードのすべての後続フィールドに FTAB が 1 つ必要です。ストリーム・モードでは、すべての後続フィールドに FTAB が 1 つ必要です。後続フィールドで FTAB を使用すると、FTAB に続く文字が、定義された次のフィールドの第 1 文字であることを表します。 (ALL 条件を指定した場合と同じです。)

以下の図の例 1、2、3、5、6、7 からは、望むとおりの結果が得られます。例 4 では、フィールド B のあとに FTAB が入力されていないので、正しい結果が得られません (例 5 と対比)。例 8 では、FTAB が入力されておらず、0 がブランク (未定義) の位置にあり、したがって後続フィールドの位置がずれてしまい、正しい結果が得られません (例 1 と対比)。

MIX

FTAB が見つかるまでは、各装置入力フィールドは定義されたとおりの長さであると見なされます。最初の FTAB が見つかると、現行フィールドのデータの終わりになります。FTAB に続くデータ・バイトは、次のフィールドの第 1 バイトであると見なされます。後続フィールドのうち、定義されたとおりの長さを占めるフィールドには、FTAB は必要ありません。FTAB を入力してから次のフィールドを続けた場合 (例えば、例の中のフィールド B と C)、好ましくない結果になります (例 5 を参照)。MIX 条件の FTAB は、タイプライターのタブ停止位置を各定義フィールドの先頭桁 (例の 1、6、および 9 桁目) にセットしたのと同じ働きをします。

以下の図の例 1、2、3、4、6、7 からは、望むとおりの結果が得られます。例 5 のフィールド B は、定義されたとおりのフィールドの長さを占めており、FTAB を入力する必要はないので、正しい結果が得られません。この FTAB は、フィールド C のデータが存在しないことを意味するものと解釈されます (例 4 と対比)。例 8 では、FTAB が入力されておらず、0 がブランク (未定義) の位置にあり、したがって後続フィールドの位置がずれてしまい、正しい結果が得られません (例 1 と対比)。

ALL

ALL を指定する場合、それぞれの装置入力フィールドが定義した長さ以下であるか、定義した長さ以上であるかに関係なく、FTAB で終了していなければなりません。FTAB が見つかると、現行フィールドのデータの終わりになります。FTAB に続くデータ・バイトは、次のフィールドの第 1 バイトであると見なされます。

以下の図の例 2、3、5、6 からは、望むとおりの結果が得られます。例 1、4、7、8 では、必要な FTAB が入力されていないため、正しい結果は得られません。

	OPERATOR INPUT EXAMPLE	FIELD	MFS DFLD DATA					
			FTAB=(',',FORCE)		FTAB=(',',MIX)		FTAB=(',',ALL)	
			LTH	DATA	LTH	DATA	LTH	DATA
1	PART␢02315231	A	4	PART	4	PART	4	PART
		B	3	023	3	023	0	
		C	5	15231	5	15231	0	
2	P1,23,15231	A	2	P1	2	P1	2	P1
		B	2	23	2	23	2	23
		C	5	15231	5	15231	5	15231
3	PART,23,15	A	4	PART	4	PART	4	PART
		B	2	23	2	23	2	23
		C	2	15	2	15	2	15
4	PART,02315231	A	4	PART	4	PART	4	PART
		B	3	023	3	023	3	023
		C	0		5	15231	0	
5	PART,023,15231	A	4	PART	4	PART	4	PART
		B	3	023	3	023	3	023
		C	5	15231	0		5	15231
6	PART,,15231	A	4	PART	4	PART	4	PART
		B	0	15231	0	15231	0	15231
		C	5		5		5	
7	PART␢,15231	A	4	PART	4	PART	4	PART
		B	0	15231	0	15231	3	152
		C	5		5		0	
8	PART02315231	A	4	PART	4	PART	4	PART
		B	3	231	3	231	0	
		C	4	5231	4	5231	0	

注: この例では、コンマは FTAB 文字として使用されます。

図 *31. FTAB* が *FORCE*、*MIX*、および *ALL* で定義されているときの *MFS* の入力スキャン

DPM-An 用ヌル文字の任意削除*:*

MFS では、SLU P (DPM-An) リモート・プログラムからの伝送レコードおよび入力データ・フィールドの末尾のヌル文字の削除を選択することができます。(ヌル文字とは 16 進数のゼロの X'00' です。) この装置入力形式は、DIV ステートメントで NULL=KEEP または NULL=DELETE を指定することができます。
NULL=DELETE を指定すると、MFS がデータ・フィールドと伝送レコードの末尾のヌル文字をスキャンして削除します。KEEP はデフォルトであり、データに末尾のヌル文字があっても MFS はそれらをそのまま残し、有効なデータ文字として扱うことを意味します。

リモート・プログラムが、末尾のヌル文字を充てん文字に置き換えてしまっていても、MFS はその充てん文字を有効なデータ文字として処理します。

NULL=DELETE を指定すると、レコードの終わりにあるヌル文字は、データ・フィールドがスキャンされる前に削除されます。レコード・モードでは、レコードの終わりは FTAB またはこれ以外の最初に見つかった非ヌル文字 (レコードの終わりから逆方向に検索) によって判別されます。ストリーム・モードの場合は、FTAB でレコードの終わりが指示されている場合にかぎり、レコードの終わりにある末尾のヌル文字が削除されます。指示されていなければ、リモート・プログラムから受け取ったレコードはそのまま処理されます。

データ・フィールドのスキャン中は、フィールドで最初に末尾のヌル文字が検出された位置が、現行フィールドのデータの終わりです。データは編集されてメッセージ・フィールドに移されますが、必要に応じてフィールドにメッセージ充てん文字が埋め込まれます。フィールド全体がヌル文字 (例えば、レコードの終わりにあるヌル文字) の場合は、指定の充てん文字でメッセージ・フィールド全体が埋められます。

伝送されてくる各レコードについて、フィールドの末尾のヌル文字がスキャンされます。処理中の現行レコードに FTAB 文字が検出されると、そのレコードのフィールドの末尾のヌル文字スキャンは中止され、次のレコードのスキャンを再開します。

IMS TM へのヌル文字の送信と削除操作は、いずれも実行時間が増えることになります。NULL=DELETE オプションを使用するか、リモート・プログラムを使ってヌル文字を削除するかを決めるに際には、その結果生じる相対的コストを比較検討する必要があります。また、FTAB オプションの FORCE、MIX、および ALL の影響も考慮しなければなりません。これらの要因は、以下のことによって左右されます。

- NULL=DELETE を使用して FTAB=ALL を指定すると、MFS は、レコードの終わりにある末尾のヌル文字だけを削除する。
- ストリーム・モードで NULL=DELETE を用いるときは、レコードの終わりの省略されたフィールド (1 つ以上) を FTAB で示す必要がある。そうしない場合は、フィールドに対して定義されている文字数と同数のヌル文字を送信しなければなりません。
- FTAB と NULL=DELETE を指定すると、ヌル文字と FTAB を併用することができる。あるレコードに FTAB を用い、次のレコードにはヌル文字を用いることができます。FTAB が入っていないレコードからは、ヌル文字が取り除かれます。FTAB が入っているレコード内のヌル文字は、データとして処理されます。
- NULL=DELETE のときは、末尾に有効な (ヌル文字を意味しない) 16 進数のゼロが入っている 2 進データの前には、前のフィールド用の FTAB 文字が必要である。これは、末尾の X'00' が削除されるのを防ぐためです。

関連資料:

524 ページの『DIV ステートメント』

DPM-An* 用ヌル文字の任意削除の例*:

以下は、DPM-An 用のオプショナル・ヌル文字削除の例です。

以下に示す 3 つの例では、コンマが指定された FTAB であり、X'5F' は入力 16 進データで、文字は以下のように定義されています。

X'6B'=C","

X'C1'=C"A"

X'C2'=C"B"

X'C3'=C"C"

C"b"=ブランク

X'40'=C"b"

例 **1: 2** 進データとヌル文字の入力

```
  Device Input Format             Message Input Definition
  INFMT FMT                       INMSG MSG  TYPE=INPUT,SOR=INFMT
        DEV TYPE=DPM-A1, FTAB=(;;MIX)   SEG
        DIV TYPE=INPUT, NULL=DELETE
        PPAGE
A       DFLD  LTH=3                     MFLD A, LTH=3
B       DFLD  LTH=2                     MFLD B, LTH=2
        FMTEND                          MSGEND
```

入力メッセージ	レコード	フィールド	**DFLD** データ	**MFLD** データ
(1) X'C1C2C3005F'	1	A	C"ABC"	C"ABC"
		B	X'005F'	X'005F'
(2) X'C1C26B005F'	1	A	C"AB"	C"ABb"
		B	X'005F'	X'005F'
(3) X'C1C200005F'	1	A	C"AB"	C"ABb"
		B	X'005F'	X'005F'
(4) X'C1C2C35F00'	1	A	C"ABC"	C"ABC"
		B	X'5F'	X'5F40'
(5) X'C1C26B5F00'	1	A	C"AB"	C"ABb"
		B	X'5F'	X'5F40'

注: 入力メッセージ (4) と (5) のレコードの終わりにある X'00' (ヌル) は、データ・フィールド (A と B) がスキャンされる前に削除されます。したがって、FTAB (この例ではコンマ) がフィールド A のあとにあっても、フィールド B の結果は同じになります。X'00' をフィールド B のデータとして処理する場合は、X'5F00' に続けて FTAB (この例ではコンマ) を入力する必要があります。

例 **2:** レコード・モード入力

```
  Device Input Format             Message Input Definition
  INFMT FMT                       INMSG MSG  TYPE=INPUT,SOR=INFMT
        DEV TYPE=DPM-A1, FTAB=(;;MIX),  SEG
            MODE=RECORD
        DIV TYPE=INPUT, RCDCTL=12,      MFLD A,LTH=3,FILL=C'*'
            NULL=DELETE
        PPAGE                           MFLD B,LTH=3,FILL=C'*'
A       DFLD  LTH=3                     MFLD C,LTH=3,FILL=C'*'
B       DFLD  LTH=3                   MFLD D,LTH=3,FILL=C'*'
C       DFLD  LTH=3                   SEG
D       DFLD  LTH=3                   MFLD E,LTH=5,FILL=C'*'
E       DFLD  LTH=5                   MFLD F,LTH=7,FILL=C'*'
F       DFLD  LTH=7                   SEG
G       DFLD  LTH=5                     MFLD G,LTH=5,FILL=C'*'
        FMTEND                          MSGEND
```

入力メッセージ	レコード	フィールド	**DFLD** データ	セグメント	**MFLD** データ
(1) X'C10000C20000C3C3C3000000'	1	A	C'A'	1	C'A**'
		B	C'B'		C'B**'
		C	C'CCC'		C'CCC'
		D	データなし		C'***'
X'C5C56BC6C66B000000000000'	2	E	C'EE'	2	C'EE***'
		F	C'FF'		C'FF*****'
X'0000000000'	3	G	データなし	3	C'*****'
(2) X'C10000C20000C3C3C3'	1	A	C'A'	1	C'A**'
		B	C'B'		C'B**'
		C	C'CCC'		C'CCC'
		D	データなし		C'***'
X'C5C56BC6C6'	2	E	C'EE'	2	C'EE***'
		F	C'FF'		C'FF*****'
入力レコードなし	3	G	データなし	3	C'*****'

注: この例では、フィールド D と G にはデータが入力されていません。入力メッセージ 1 では、データが入力されなかったフィールドの位置にヌル文字が入っています。入力メッセージ 2 では、データが入力されなかったフィールドの位置にヌル文字が入っていませんが、両方の入力メッセージの結果は同じになります。

例 **3:** ストリーム・モード入力

```
Device Input Format             Message Input Definition
  INFMT FMT                       INMSG MSG  TYPE=INPUT,SOR=INFMT
        DEV TYPE=DPM-A1, FTAB=(;;MIX),  SEG
            MODE=STREAM
        DIV TYPE=INPUT, NULL=DELETE     MFLD A,LTH=3,FILL=C'*'
        PPAGE                           MFLD B,LTH=3,FILL=C'*'
A       DFLD  LTH=3                     MFLD C,LTH=3,FILL=C'*'
B       DFLD  LTH=3                     MFLD D,LTH=3,FILL=C'*'
C       DFLD  LTH=3                     SEG
D       DFLD  LTH=3                     MFLD E,LTH=5,FILL=C'*'
E       DFLD  LTH=5                     MFLD F,LTH=7,FILL=C'*'
F       DFLD  LTH=7                     SEG
G       DFLD  LTH=5                     MFLD G,LTH=5,FILL=C'*'
        FMTEND                          MSGEND
```

入力メッセージ	レコード	フィールド	**DFLD** データ	セグメント	**MFLD** データ
(1) X'C10000C20000C3C3C3000000'	1	A	C'A'	1	C'A**'
		B	C'B'		C'B**'
		C	C'CCC'		C'CCC'
		D	データなし		C'***'
X'C5C56BC6C66B000000000000'	2	E	C'EE'	2	C'EE***'
		F	C'FF'		C'FF*****'
X'00000000000000'	3	G	データなし	3	C'*****'

入力メッセージ	レコード	フィールド	**DFLD** データ	セグメント	**MFLD** データ
(2) X'C10000C20000C3C3C3'	1	A	C'A'	1	C'A**'
		B	C'B'		C'B**'
		C	C'CCC'		C'CCC'
	2	D	C'EE'		C'EE*'
X'C5C56BC6C6'		E	C'FF'	2	C'FF***'
		F	データなし		C'*******'
入力レコードなし	3	G	データなし	3	C'*****'

注: この例では、フィールド D と G にはデータが入力されていません。入力メッセージ 1 では、データが入力されなかったフィールドの位置にヌル文字が入っています。入力メッセージ 2 では、データが入力されなかったフィールドの位置にヌル文字が入っていないので、フィールド D、E、F では好ましくない結果になります。

複数物理ページ入力メッセージ ***(3270*** 表示装置と ***SLU 2*** 表示装置***):***

3270 表示装置と SLU 2 表示装置に複数の物理ページの入力を指定すると、使用中の装置の物理的な容量に関係なく、1 つのトランザクションに同じ入力メッセージを複数作成することができます。

この機能を使用すれば、1 つの出力論理ページに含まれる複数の物理ページを利用して、複数の物理ページからなる入力メッセージを入力することができます。出力に複数の物理ページが定義されている場合には、DPAGE ステートメントで MULT=YES を指定するだけで、複数物理ページ入力を得ることができます。

区画モードの 3290 情報表示パネルで、1 つの区画から複数物理ページ入力を行えるのは、現行区画用の DPAGE ステートメントに MULT=YES が指定してある場合だけです。1 つの入力メッセージを構成する複数の物理ページは、同一の区画から入力されたものでなければなりません。

現在の区画の DPAGE ステートメントで MULT=YES が指定されていない場合は、1 つの区画の 1 つの物理ページで 1 つの入力メッセージが構成されるため、入力メッセージの論理ページが 1 つに制限されます。

入力メッセージは、複数の DPAGE から作成することができます。この機能は、3270 および SLU 2 以外の装置で使用することができます。

関連概念:

570 ページの『出力メッセージの物理ページング』

複数 DPAGE 入力に関する一般規則

複数 DPAGE 入力の場合は、次の規則に従ってください。

1. マップされたいずれかの入力 LPAGE にデータ・セグメントが含まれていない場合 (例えば、セグメント・ルーチンがすべてのセグメントを取り消したため)、入力メッセージは拒否され、エラー・メッセージが別のサブシステムに送信される。

2. 入力端末への MFS エコーは無視される。
3. DPAGE からでも MFS パスワードを作成できるが、いったん作成されると他のパスワードは無視される。付加 FM ヘッダーにパスワードが入っている場合は、DPM-Bn ではこのパスワードが使用されます。
4. 入力メッセージ・オプション 1、2、3 は、LPAGE に対して使用される。オプション 2 を要求すると、LPAGE の終わりにあるヌル・セグメントは排除されます。この結果、入力メッセージに次の LPAGE があれば、その中のセグメントの相対位置が変更されます。オプション 1 または 2 を要求すると、2 番目 (およびそれ以降すべて) の LPAGE の最初のセグメントは、その前の LPAGE の終わりにヌル・セグメントがあるかどうかに関係なく、Z2 フィールドのページ・ビット (X'40') がオンにされます。オプション 3 を要求すると、新しい LPAGE の最初のセグメントは、どれもセグメント ID が 1 になります。
5. MFS 定義で複数 DPAGE 入力を要求しても、単一 DPAGE のメッセージ作成が制限されるわけではない。
6. 制御要求が最初の入力 DPAGE で入力された場合、要求は処理されるが、入力メッセージは拒否される。制御要求が、最初の入力 DPAGE 以外の入力 DPAGE で入力された場合、要求は無視され、入力メッセージは受理されます。
7. オペレーターの論理ページ要求が、最初の入力 DPAGE で入力された (つまり、入力セグメントの最初の桁位置が等号 (=) である) 場合、要求は処理されるが、入力メッセージは拒否される。

MFS 定義の中で複数 DPAGE 入力を要求していなければ、2 つ以上の DPAGE からメッセージを作成することはできないので、代わりに以下の規則が適用されます。

1. 1 つの伝送に、使用する DPAGE で定義されている以上のデータが入っていると、入力メッセージは拒否され、エラー・メッセージが別のサブシステムに送信される。
2. メッセージが複数の伝送で構成されていると、その入力メッセージは拒否され、エラー・メッセージが別のサブシステムに送信される。

3270 および SLU 2 の入力置換文字

IMS TM は、端末からの入力として (例えば、ERROR キーを使用して) X'3F' を受け取ることができます。

置換文字 (X'3F') は、フィールドにエラーがあることをホスト・アプリケーションに知らせる手段になります。MFS は、X'3F' を入力データ・ストリームで IMS TM 機能として使用します。この 2 つの方法で X'3F' 文字を使用しても混乱が起きないように、3270 および SLU 2 表示装置で使うために、DEV ステートメントでパラメーター (SUB=) を与えておきます。

3270 および SLU 2 データ・ストリームに入っている X'3F' 文字を MFS が受信したときに、それと置き換えるユーザー指定文字を、このパラメーターで定義しておくことができます。以下のいずれかが当てはまる場合、変換は行われません。

SUB= パラメーターを指定しなかった。

SUB= パラメーターを X'3F' と指定した。

受信した入力が MFS をバイパスする。

ここで指定する SUB 文字は、データ・ストリームの他の場所には現れない文字でなければなりません。したがって、非図形文字を使用します。

ISC (DPM-Bn) サブシステムの入力形式制御

ISC ノードを使用した MFS の主要な入力メッセージ・フォーマット設定機能を使用します。

以下の DPAGE 選択オプションを使用してメッセージをフォーマット設定し、複数 DPAGE のメッセージを作成できます。

入力 DPAGE 選択

DIV ステートメントに OPTIONS=(DNM) パラメーターを指定すると、データ構造名 (DSN) を用いて DPAGE を選択することが可能になります。

複数の DPAGE を定義する場合は、すべての DPAGE に DPAGE ラベルを指定しなければなりません。どの DPAGE も選択されなければ、メッセージは拒否され、エラー・メッセージが別のサブシステムに送信されます。

OPTIONS=NODNM と複数の DPAGE を定義した場合は、最初の入力レコードで条件付きテストが実施されます。テスト (COND= の指定がデータと一致するか) の結果によって、入力データのフォーマット設定に使用する DPAGE が決定されます。条件が満たされず、しかも定義されたすべての DPAGE が条件付きであれば、入力メッセージは拒否され、エラー・メッセージが別のサブシステムに送信されます。

単一伝送チェーン

単一伝送チェーンの場合、条件付きデータを使用して DPAGE を選択することができます。

条件付きデータを使用した ***DPAGE*** 選択

単一伝送チェーンを使用した複数 DPAGE 入力では、OPTIONS=NODNM パラメーターを使用します。最初の入力レコードのデータに基づいて、フォーマット設定のための最初の (または唯一の) DPAGE が選択されます。データが COND= で定義されているどの条件とも一致しない場合は、最後に定義されている DPAGE が選択されますが、それは、最後の DPAGE に COND= が指定されていない場合に限ります。条件が満たされず、しかも定義されたすべての DPAGE が条件付きであれば、入力メッセージは拒否され、エラー・メッセージが別のサブシステムに送信されます。 DD ヘッダーで与えられた DSN は無視されます。追加の DPAGE がある場合 (選択された DPAGE で定義されているよりも多くのデータが与えられた場合) は、後続レコードのデータに基づいて、フォーマット設定のために次の DPAGE が選択されます。

複数伝送チェーン

複数伝送チェーンの場合、DSN を使用して、あるいは条件付きテストを使用して、DPAGE を選択することができます。

DSN を使用した ***DPAGE*** 選択

複数伝送チェーンを使用した複数 DPAGE 入力では、OPTIONS=DNM パラメーターを使用します。メッセージ・チェーンごとに DD ヘッダーに入っている DSN を使用して、フォーマット設定のための DPAGE が選択されます。一致する DPAGE がない場合は、そのメッセージは拒否され、エラー・メッセージ (DFS2113) が別のサブシステムに送信されます。

データの条件付きテストを使用した ***DPAGE*** 選択

メッセージの各チェーン (または任意のチェーン) で DD ヘッダーに DSN が入っていても、DIV ステートメントで OPTIONS=NODNM が指定してあると、DSN は無視されます。各チェーンの最初のレコードのデータに基づいて、フォーマット設定のための DPAGE が選択されます。そのデータが条件と一致しておらず、しかも最後に定義された DPAGE が条件付きでない場合 (つまり、COND= パラメーターが指定されていない場合) は、この DPAGE がフォーマット設定のために選択されます。条件が満たされず、しかも定義されたすべての DPAGE が条件付きであれば、入力メッセージは拒否され、エラー・メッセージが別のサブシステムに送信されます。

条件付き DPAGE および無条件 DPAGE をどのように指定するかは、OPTIONS=DNM と OPTIONS=NODNM のどちらを指定するかによって決まります。

- OPTIONS=DNM の場合は、条件付き DPAGE は、DPAGE ステートメントでラベル付きで指定される。
- OPTIONS=NODNM の場合 :
  - 条件付き DPAGE を指定するには、DPAGE ステートメントで COND= キーワードを指定する。
  - 無条件 DPAGE を指定するには、COND= キーワードを省略する。

MFS は、レコード・モードとストリーム・モードの 2 つの入力モードをサポートしています。

レコード・モード

レコード・モードでは、ATTACH マネージャーが MFS に渡すレコードは MFS に定義されたレコードと 1 対 1 で対応しています。レコードおよび各レコードに定義されたフィールドは、順次処理されます。各フィールドを複数のレコードにまたがらせることはできません。レコードに定義されたフィールドのデータは、そのレコード内に入っていないと、MFS で処理されません。レコードの終わりで定義したフィールドにデータがない場合は、MFS に短いレコードを渡すことができます。レコードの最後のフィールド以外のフィールドのデータが、そのフィールドに対応する DFLD で定義されている長さに満たない場合、またはそのフィールドのデータがない場合は、フィールド・タブ分離文字を挿入して、データの省略または切り捨てを示す必要があります。レコード全体のデータがない場合は、そのあとに他のデータ・レコードが続くのであれば、ヌルまたは 1 バイトのレコード (FTAB 文字 1 つを含む) が与えられていなければなりません。以下のような場合には、レコードを省略することができます。

- 単一 DPAGE 入力の DPAGE の終わり。
- 複数伝送チェーンを使用した複数 DPAGE 入力の DPAGE の終わり。

- 単一伝送チェーンを使用した複数 DPAGE 入力の最後の DPAGE の終わり。別の DPAGE のデータが続いている場合は、DPAGE からこのレコードを排除することはできません。

ストリーム・モード

ストリーム・モードでは、レコード境界は無視されるので、各フィールドがレコード境界にまたがっていてもかまいません。 DPAGE のどのフィールドでも、データを省略するときには FTAB でそのことを指示する必要があります。

ただし、以下の場合には、DPAGE の終わりまでのデータを省略するのであれば、FTAB がなくてもかまいません。

- 単一 DPAGE 入力の DPAGE の終わり。
- 複数伝送チェーンを使用した複数 DPAGE 入力の DPAGE の終わり。
- 単一伝送チェーンを使用した複数 DPAGE 入力の最後の DPAGE の終わり。別の DPAGE のデータが続いている場合は、この DPAGE から FTAB を排除することはできません。

ATTACH マネージャーは、IMS へ入力を渡す際に、UNDEFINED、RU、VLVB、および CHAINED ASSEMBLY の 4 つの非ブロック化アルゴリズムを提供します。以下のように指定します。

- UNDEFINED または RU を指定すると、1 つの RU は、処理される 1 つの MFS レコードに相当する。ATTACH FM ヘッダーに UNDEFINED が指定されていると、IMS TM は RU アルゴリズムをデフォルトとして使用します。
- VLVB を指定すると、1 つの VLVB レコードは、処理される 1 つの MFS レコードに相当する。
- CHAINED ASSEMBLY を指定すると、1 つの入力チェーンが、DPAGE 全体として処理される 1 つの MFS レコードに相当する。

MFS レコード・モードでは、VLVB 非ブロック化アルゴリズムを使用してください。MFS レコード・モードでは、以下を使用してはなりません。

- CHAINED ASSEMBLY 非ブロック化アルゴリズム。使用すると、入力チェーン全体が 1 つの MFS レコードとして処理されます。
- UNDEFINED または RU 非ブロック化アルゴリズム。使用すると、MFS レコード定義が RU の大きさに左右されることになります。

MFS ストリーム・モードでは、すべての非ブロック化オプションを使用することができます。ほとんどの場合、UNDEFINED または RU アルゴリズムを使用する方がバッファー・スペースが少なくて済みます。

ページング要求

ISC を使用しているときにページング要求を入力するには、FM ヘッダーを使用します。

関連概念:

541 ページの『入力メッセージ・フォーマット設定』

出力メッセージ・フォーマット設定

IMS は以下の MFS 出力メッセージのフォーマット設定、物理ページングと論理ページング、および出力装置の要件をサポートします。

MFS が受け入れる出力メッセージ**:**

出力メッセージが IMS TM 基本編集と MFS のどちらで処理されるかは、装置タイプ、装置定義、および現在処理中のメッセージによって決定されます。

SLU 2 および 3270 装置あての出力メッセージは、アプリケーション・プログラムによってバイパスされないかぎり、すべて MFS によって処理されます。

3770、金融機関ワークステーション、SLU 1、NTO、SLU P、または ISC サブシステムあての出力メッセージを MFS で処理するためには、IMS TM システム定義時に、MFS が作動するようにこれらの装置を定義しておかなければなりません。

MFS が作動するように装置が定義されている場合でも、アプリケーション・プログラム、前の入力メッセージに対応する MID、または `/FORMAT` コマンドで MOD 名を指定していないかぎり、MFS は出力メッセージを処理しません。また、別の MFS 装置からのメッセージ通信は、メッセージが MOD と関連付けられているときにかぎり、MFS によって処理されます。

オンライン変更ユーティリティーの実行時に、変更または削除するトランザクションにアクセスする場合、これをオンライン変更コマンド `/MODIFY PREPARE` が発行された後で、しかも `/MODIFY COMMIT` が発行される前に実行すると、エラー・メッセージが表示されます。

関連資料:

/MODIFY コマンド (コマンド)

MFS メッセージ・フォーマット設定機能**:**

MFS によって処理される出力メッセージは、メッセージ出力記述子 (MOD) と装置出力形式 (DOF) という 2 つの MFS 制御ブロックの内容に基づいてフォーマット設定されます。

MOD では、出力メッセージの内容と、必要に応じて出力メッセージの一部と見なされるリテラル・データを定義しておくこともできます。メッセージ・フィールド (MFLD) では、DOF の装置フィールド (DFLD) 定義を介して装置フィールド位置を指します。装置出力形式 (DOF) には、ハードウェア機構の使用、装置フィールドの位置と属性、および形式の一部と見なす定数データを指定しておきます。

出力メッセージ・フォーマット設定オプション*:*

出力データについても、MFS では 3 つのメッセージ・フォーマット設定オプションが用意されています。どのオプションを選択するかによって、データをフォーマット設定する方法と、アプリケーション・プログラムが出力メッセージを組み立てる方法が決まります。

MOD の MSG ステートメントの OPT= オペランドで、オプションが 1 か 2 か 3 かを指定します。

アプリケーション・プログラムで挿入するセグメントは、MFS 言語ユーティリティー・プログラムで定義したとおりの順序で並んでいなければなりません。1 つの論理ページにすべてのセグメントを挿入する必要はありませんが、セグメントを省略する場合は、十分に注意を払う必要があります。オプション 1 または 2 を使用する場合にセグメントを省略できるのは、そのセグメント以降から論理ページの最後のセグメントまでをすべて省略する場合に限られます。それ以外の場合には、ヌル・セグメント (X'3F') を必ず挿入して、セグメントの位置を明示しておかなければなりません。オプション 3 を使用する出力メッセージ・セグメントには、必ず 2 バイトの相対セグメント番号を入れておかなければなりません。

オプション 1 または 2 を使用する出力セグメントでは、メッセージ・フィールドを固定長かつ固定位置であると定義します。フィールドの切り捨てや省略は、以下の方法で行うことができます。

- 短いセグメントを挿入する方法
- フィールドにヌル文字 (X'3F') を挿入する方法。フィールドは左から右にスキャンされてヌル文字が調べられ、ヌル文字を最初に検出したところでそのフィールドが打ち切られます。フィールドの最初の文字がヌル文字ならば、そのフィールドは効果的に省略されることになります (使用する充てん文字によって決まります)。ヌル文字を使用するかしないかに関係なく、セグメント内のすべてのフィールドの配置は変りません。切り捨てまたは省略されたフィールドは、MFS 言語ユーティリティーで定義した形式定義どおりに埋め込みが行われます。

オプション 3 を用いるセグメントでは、メッセージ・フィールドをどのような順序でも配列でき、また、セグメント・サイズの範囲内であれば、メッセージ・フィールドの長さは任意に決めることができます。短フィールドまたは省略フィールドへの埋め込みは、MFS 言語ユーティリティーに対して定義されたとおりに行われます。各フィールドの前には、4 バイトのフィールド接頭部を付けなければなりません。このフィールド接頭部の形式は、オプション 3 入力フィールドに対して MFS が用意する接頭部の形式と同じでなければなりません。

オプション 3 を使用する場合、出力セグメントでフィールドを特定の順序に並べる必要はありませんが、セグメントのすべてのフィールドは連続していなければなりません。つまり、2 番目のフィールドのフィールド接頭部は、最初のフィールドのデータの終わりの次のバイトから始まっていなければなりません。オプション 3 のフィールドにヌル文字が入っていても、装置に送信されるデータには何の影響もありません。他の非図形文字と同じように、ヌル文字はブランクで置き換えられます。

制約事項**:** MFS では、出力メッセージ・フィールドに装置制御文字を入れても無効です。 3270 表示装置と SLU 2 端末では、制御文字 HT、CR、LF、NL、および BS は、ヌル文字 (X'00') に変更されます。その他の装置では、これらの制御文字はブランク (X'40') に変更されます。その他のすべての非図形文字 (X'00' から X'3F' および X'FF') は、送信前にブランクに変更されます。ただし、EGCS 使用可能な装置のシフトアウト/シフトイン (SO/SI) 文字 (X'0E' および X'0F') は例外です。(SO/SI 文字がブランクに変換されるのは、明らかに DBCS フィールドである場合だけです。) SLU P (DPM-An) リモート・プログラムと ISC (DPM-Bn) サブシステムの場合は、GRAPHIC=NO を出力に指定できるという例外が許されています。これを指定して非図形データを使用可能にした場合は、オプション 1 と 2 でセグメントを切り捨てる目的にヌル (X'3F') を使えなくなります。

オプション **1** または **2** を用いた出力セグメントの例

```
Definition              Output data length
Segment
Field, length=10        4
Field, length=20        field omitted
Field, length=5         5
Field, length=15        15
```

上記のセグメントにより、以下のような結果になります。

```
CONTENTS |54|0|0| DATA 1|*|    |*       | DATA 3 | DATA 4|
-----------------------------------------------------------
LENGTH    2  1 1  4      1  5   20      5        15
```

オプション **3** を用いた出力セグメントの例

オプション 3 を用いるセグメントで同じ結果を得ようとすると、セグメントは以下のようになります (* はヌル文字 (X'3F') を表します)。

```
CONTENTS |42|0|0|04|08|04| DATA 1|09|34| DATA 3 |19|39| DATA 4|
-------------------------------------------------------------
LENGTH    2  1 1 2  2  2   4      2  2    5      2  2   15
```

関連概念:

541 ページの『入力メッセージ・フォーマット設定』

関連資料:

505 ページの『フィールド形式 (オプション 1 および 2)』

506 ページの『フィールド形式 (オプション 3)』

出力メッセージのオペレーター論理ページング*:*

出力メッセージでは、オペレーター論理ページングを許可することを定義しておくことができます (MOD の MSG ステートメントの PAGE= オペランドで指定)。オペレーター論理ページングを使用して、出力メッセージの特定論理ページを要求します。

SLU P (DPM-An) や ISC サブシステム (DPM-Bn) 用にユーザーが作成したリモート・プログラムでも、オペレーター論理ページングを使用することができます。リモート・プログラムは、出力メッセージの特定の論理ページを送るよう IMS に要求することができます。

関連概念:

605 ページの『装置におけるページング操作』

602 ページの『オペレーターによる MFS の制御』

出力メッセージの物理ページング*:*

1 つの論理ページは、1 つ以上の物理ページで構成するように定義できます。物理ページングを使用すれば、ある論理ページのデータを、複数の物理ページにして装置で表示することができます。物理ページの割り当ては、形式定義の中で行います。表示装置では、物理ページのサイズは画面の容量 (表示できる行数と桁数) によって定義されます。ほとんどのプリンターの場合、物理ページは、ユーザーが指定したページの長さ (行数) とプリンターの 1 行の長さによって決まります。

SLU P (DPM-An) や ISC サブシステム (DPM-Bn) の物理ページは、ユーザーが指定したページング・オプションと、装置ページまたは表示ページを指定する DPAGE または PPAGE ステートメントとを用いて定義します。物理ページングを使用すると、メッセージからのデータを複数の表示ページまたは論理ページの形でリモート・プログラムやサブシステムに送信することができます。

通常、1 つの論理ページはただ 1 つの物理ページから構成されます。1 つの論理ページに対して複数の物理ページを使用するのは、通常、大型画面用に設計されている論理ページを小型画面の装置でも使用する場合だけです。このときの物理ページは、大型画面の装置用に定義されているページとは全く別の形式に設定することができます。以下の図は、画面サイズが 24×80 の 3277-2 型または 3276/3278 で、1 つの物理ページを作成するメッセージを用いた物理ページングの使用を示したものです。

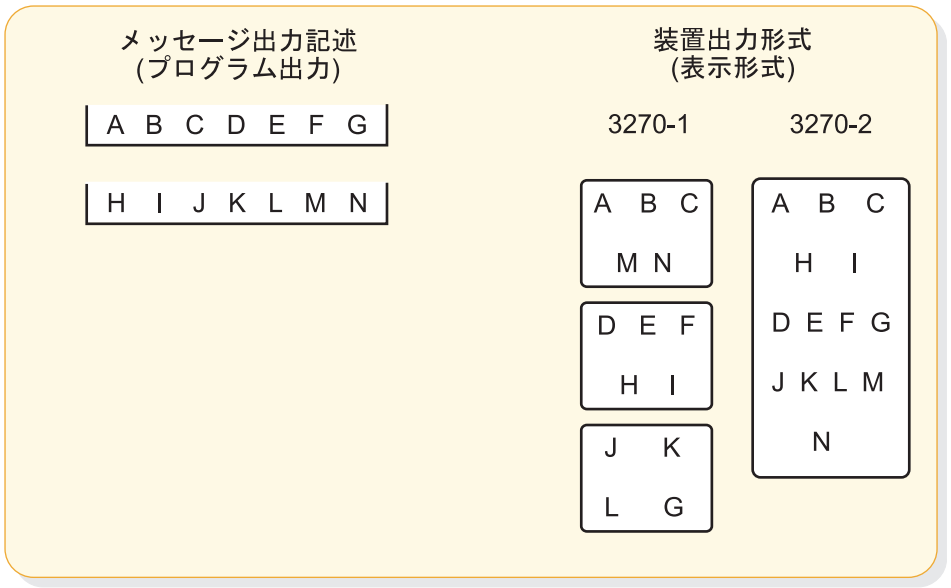


図 *32. 3270* または *SLU 2* の物理ページング

関連概念:

563 ページの『複数物理ページ入力メッセージ (3270 表示装置と SLU 2 表示装置)』

出力装置フィールド用の充てん文字*:*

指定されているフィールドの長さよりもアプリケーション・プログラムから受け取ったデータの長さが短いか、またはあるフィールドに対するデータをまったく受け取らなかった場合、MFS はそのような出力装置フィールドに充てん文字を埋め込みます。

充てん文字は、メッセージ定義 (MSG ステートメント) または形式定義 (DPAGE ステートメント)、あるいは両方を用いて定義します。充てん文字を両方で指定した場合、DPAGE に指定した充てん文字が使用されます。DPAGE ステートメントに FILL=NONE を指定すると、MSG ステートメントで指定した充てん文字が使用されます。 MSG ステートメントで指定した充てん文字は、MOD の MFLD で定義したフィールドばかりでなく、DOF で定義したすべての非リテラル・フィールドで

も使用されます。一般に、装置タイプに合わせた充てん文字を使用すれば、メッセージの表示と装置の性能が向上します。 DPAGE ステートメントで指定できる充てん文字は、以下のとおりです。

- ブランク (X'40')
- ブランク (C' ')
- 任意の 16 進 EBCDIC 図形文字 (*hh*')
- EBCDIC 図形文字 (C'*c*')

MSG ステートメントで指定できる充てん文字は、以下のとおりです。

- ブランク (C' ')
- EBCDIC 図形文字 (C'*c*')

3270 または SLU 2 表示装置の画面をフォーマット設定する際に、データがまったく送られてこないフィールド、または一部分のデータしか送られてこないフィールドには、指定されている EBCDIC の図形充てん文字が埋め込まれます。画面上に残っている前のメッセージの情報は消去されます。ただし、充てん文字を使用すると、伝送時間が長くなります。

ヌル充てん文字を指定することもでき、その場合には、3270 または SLU 2 のフォーマット設定された画面では、フィールドには充てん文字の埋め込みが行われません (したがって、前のメッセージのデータのうち、現行メッセージによって更新されていないデータはそのまま表示)。3270 と SLU 2 表示装置以外の装置では、装置フィールドがメッセージ・データで埋まらないときには、短縮された行が作成されます。3270 または SLU 2 表示装置でヌル充てん文字を使用すると伝送時間を短縮させることができますが、フィールドの一部分にしかデータが出力されないので、前の表示内容がフィールドに残るようなことがあると、混乱が起こる原因になります。その他の装置にヌル充てんを指定すると、IMS 制御領域での処理が増えますが、伝送時間と印刷時間は短縮されます。

3270 または SLU 2 のフォーマット設定された画面では、装置フィールドに新しいデータを表示した後で、そのフィールドに前のデータがまだ残っていればそれを消去してしまうプログラム・タブ機能を要求することができます。この機能は充てん文字を生成することはありません。プログラムのタブ充てんを用いると、フォーマット設定された画面上の表示フィールドに新しいデータが伝送されないかぎり、そのフィールドは消去されません。

プログラムが出力データ・フィールドのうちのわずかしか送信しないときには、システム制御域 (SCA) で「全無保護域消去」機能を指定して、無保護フィールドに残っている表示不要のデータを表示しないようにできます。

EGCS フィールドが存在する 3270 出力では、DPAGE または MSG ステートメントで FILL=PT または FILL=NULL だけを指定しなければなりません。その他の指定を行うと、装置はメッセージを拒否してしまいます。

関連概念:

573 ページの『システム制御域 (SCA) とデフォルト SCA (DSCA)』

関連資料:

544 ページの『入力メッセージ・フォーマット設定オプション』

システム制御域 *(SCA)* とデフォルト *SCA (DSCA):*

システム制御域 (SCA) は、出力メッセージが装置あてに送信されたときに、特定の装置動作を要求するための手段になります。

装置へのこのような要求は、(SCA を介して) メッセージ・フィールドで定義するか、(デフォルト SCA, つまり DSCA を介して) 装置形式定義の中で定義することができます。SCA は、1 つのメッセージ・フィールドとして定義されます。IMS アプリケーション・プログラムは、出力が端末装置に送信されたときに装置で実行される動作を、SCA を用いて指定しておくことができます。

要求できる 3270 および SLU 2 の機能は以下のとおりです。

- 強制的形式書き出し
- 書き込み前に無保護フィールドを消去
- メッセージ送信前の全区画の消去
- 装置アラームを作動
- このメッセージの無保護画面
- 出力を候補プリンターにコピー

3270 および SLU 2 装置については、MFS は IMS アプリケーション・プログラムからの情報を解釈し、指定されている操作を行います。

FIN ワークステーションへの出力の場合、SCA で「装置アラームの作動」を要求することができます。この場合、MFS は、FIN ワークステーションへ送信する出力メッセージのヘッダーに「装置アラーム」を指定します。

SLU P (DPM-An) または ISC サブシステム (DPM-Bn) の場合についても、3270 および FIN で使用できるすべての機能を、SCA として定義されたメッセージ・フィールドに、IMS アプリケーション・プログラムで指定することができます。DOF では、装置フィールド (DFLD ステートメント) を SCA として定義します。SLU P リモート・プログラムまたは ISC サブシステムに関しては、MFS は IMS からの指定を解釈しません。MFS は、その指定をユーザー定義の装置フィールド SCA で中継して、リモート・プログラムまたは ISC サブシステムに送るだけです。

3270、SLU 2、FIN、SLU P、ISC 以外の装置では、SCA は無視されます。

SCA の指定が可能なすべての装置には、DOF の DEV ステートメントでデフォルト・システム制御域 (DSCA) も定義することができます。DSCA でも同種の機能を指定することができます。DOF DSCA が使用されたときにはいつでも、そこで指定されている機能が、宛先装置で実行可能なものであれば必ず実行されます。DSCA で指定した機能は、SCA フィールドの有無にかかわらず実行されます。DSCA と SCA で指定したそれぞれの要求が矛盾する場合は、DSCA で指定した機能だけが実行されます。DSCA の指定の中のフラグの設定値が正しくないものはリセットされ、正しい設定値だけが使用されます。

SLU P リモート・プログラムの場合も同様に、DSCA 情報が SCA の指定を無効にします。MFS は、SCA または DSCA の情報を解釈することはなく、SCA と定義された装置フィールドに入れてリモート・プログラムに送信するだけです。

SCA の指定を使って出力を制御する IMS アプリケーション・プログラムは、装置に依存することになります。

関連概念:

571 ページの『出力装置フィールド用の充てん文字』

480 ページの『3270 または SLU 2 画面のフォーマット設定』

出力メッセージ・リテラル・フィールド*:*

指定したリテラル・データを出力メッセージ・フィールドに入れるには、MOD の定義の段階で指定することができます。MFS は、指定されているリテラルを出力メッセージに組み入れてから、そのメッセージを装置あてに送信します。

ユーザーは、独自のリテラル・フィールドを定義することも、MFS で提供される多数のリテラルの中から適切なリテラルを選択することもできます。あるいは、この両方を同時に行うこともできます。MFS 提供のリテラルは、システム・リテラルと呼ばれ、以下のものがあります。

- 各種の日付形式
- タイム・スタンプ
- 出力メッセージ・シーケンス番号
- 論理端末名
- 論理ページ番号
- メッセージ・キュー番号

関連概念:

577 ページの『拡張図形文字セット (EGCS)』

関連資料:

MFLD ステートメント (システム・ユーティリティー)

出力装置フィールドの属性*:*

装置フィールドの属性は、DOF の DFLD ステートメントで定義します。3270 表示装置については、DFLD ステートメントの ATTR= キーワードまたは EATTR= キーワードで特定の属性を定義することができます。特定の属性を指定しなければ、デフォルトの属性が使用されます。

3270 プリンター、3770 端末、および 3601 ワークステーションについては、DFLD ステートメントで ATTR=YES または ATTR=*nn* を指定することによって、属性シミュレーションを定義することができます。装置フィールドに対応するメッセージ・フィールド定義では、アプリケーション・プログラムが装置フィールド属性を動的に変更、置換、またはシミュレートできることを指定しておくことができます。

出力装置の拡張フィールド属性*:*

拡張フィールド属性は、3270 表示装置、および 3270P または SCS1 と定義されているプリンターで使えるもので、3270 構造化フィールドおよび属性処理オプションをサポートします。

この属性は、フィールドの枠取りまたは DBCS 操作が必要な場合に、拡張図形文字セット (EGCS) をサポートする 3270P プリンターまたは SCS1 プリンターでも適用されます。これらの拡張フィールド属性により、既存の 3270 フィールド属性にはない追加のフィールド属性を定義することができます。既存の 3270 フィールド属性と同様に、拡張フィールド属性も文字のフィールドと関連していますが、文字バッファーの表示位置を占めることはありません。以下のようなフィールド特性を定義することができます。

- カラー (7 色を使える型式のみ)
- 強調表示
- プログラム式シンボル (PS)
- 妥当性検査
- フィールドの枠取り
- DBCS/EBCDIC 混合データの入力制御

拡張フィールド属性は、DFLD ステートメントの EATTR= キーワードで定義します。対応する MFLD ステートメントで、ATTR=YES または ATTR=*nn* に ATTR=*nn* を指定しておけば、拡張フィールド属性を動的に変更することができます。

既存のフィールド属性と拡張フィールド属性 (保護および妥当性検査を除く) を任意に組み合わせて、1 回の表示出力ストリームで送信することができます。

事前に属性を定義した装置フィールドに動的属性変更 (ATTR=YES) を指定すると、その装置フィールドのデータが出力メッセージに入っていなくても、出力操作のたびにそのフィールドの属性が装置に送信されます。

以下の属性が使用されます。

- 出力メッセージ・フィールドに属性が備わっており、その属性が有効な場合は、動的属性変更が行われる。
- メッセージ・フィールドが使用中の LPAGE に含まれていない場合、またはそのフィールドの属性が有効でない場合は、その装置フィールドの事前定義属性が使用される。

非リテラル 3270 表示装置フィールド用の主なデフォルトの属性は以下のとおりです。

- 英字
- 無保護
- 通常表示輝度
- 非変更

リテラル表示装置フィールド用の主なデフォルトの属性は以下のとおりです。

- 数値
- 通常表示輝度

リテラル表示装置フィールドには、以下の属性が強制的に付けられます。

- 保護
- 非変更

3270 以外の表示装置には属性シミュレーションを定義することができますが、これらの属性が適用されるのは、アプリケーション・プログラムが要求したときだけです。装置フィールド定義では、フィールドの最初の 1 バイトを属性データ用に確保しておきます。したがって、アプリケーション・プログラムが属性要求を行うと、その要求は装置フィールドの第 1 バイトに表されます。

シミュレートされるフィールド属性は以下のとおりです。

属性　　取られるアクション

高輝度表示
　　第 1 バイトにアスタリスク (*) が置かれます。

変更フィールド
　　第 1 バイトに下線文字 (_) が置かれます。

高輝度表示の変更フィールド
　　第 1 バイトに感嘆符 (!) が置かれます。

表示なし
　　DPM の場合以外は、他の属性にかかわらずデータは送信されない。

3604 のカーソル位置も、シミュレートされる属性として指定することができます。

シミュレートされる属性データを受け取ることが定義されているフィールドに、アプリケーション・プログラムが属性データを送らなかった場合は、その第 1 バイトはブランクになります。

アプリケーション・プログラムに属性データの変更、置換、またはシミュレーションを行わせるためには、メッセージ・フィールド定義で ATTR=YES または ATTR=*nn*、あるいはこの両方を指定しておかなければなりません。属性をこのように定義すると、出力メッセージ・フィールドの先頭バイトが属性データ用に予約されます。指定にエラーがあると、その属性バイトの DFLD ATTR= 指定または EATTR= 指定が使用されます。ただし、その他の属性または拡張属性の指定は処理されます。

DPM 装置の場合は、ATTR=YES または ATTR=*nn* を指定した MFLD 定義に対応する DFLD ステートメントに、ATTR=YES または ATTR=*nn* を指定することにより、IMS アプリケーション・プログラムから属性データまたは拡張属性データ (あるいはその両方) を受け取るフィールドを定義することができます。IMS アプリケーション・プログラムからの 3270 属性は、シミュレートした属性に変換して、装置フィールドの最初のバイトに格納するか、変更せずに (IMS アプリケーション・プログラムから受け取った 2 進数の 2 バイトのまま)、装置フィールドの最初の 2 バイトに格納することができます。属性、拡張属性、またはシミュレートした属性を送信するかどうかは、装置形式を定義するときに決めておきます。属性データを受け取ると定義されているフィールドに、IMS アプリケーション・プログラムから属性データが送られてこない場合は、属性シミュレーションが要求されていると最初のバイトはブランクにされ、2 進属性が要求されていると最初の 2 バイトに 2 進数のゼロが入ります。

拡張図形文字セット ***(EGCS):***

拡張図形文字セット (EGCS) により、EBCDIC で使用できる数以上の図形文字を使うことができます。これは、プログラム式シンボル機能の拡張です。プログラム式シンボルは、IBM 3270 情報表示装置と SCS1 プリンターのオプション機構です。これらの装置では、追加の文字セットを保管し使用することができます。

3270 表示装置や SCS1 プリンターと関連して、DBCS または DBCS/EBCDIC 混合フィールドを説明している箇所では、これらの装置では DBCS データを扱えるものと仮定しています。このような装置としては、例えば、5550 は 3270 表示装置としてサポートされており、5553 と 5557 は SCS1 プリンターとしてサポートされています。

定義**:** 2 バイト文字セット (DBCS) は EGCS のサブセットです。DBCS の図形文字は、それぞれ 2 バイトを使って表されます。有効なコードの範囲は、X'4040'、またはバイト 1 が X'41' から X'FE' で、バイト 2 が X'41' から X'FE' です。

3270 表示装置または SCS1 装置タイプでは、DFLD ステートメントの EATTR= パラメーターで EGCS フィールドを定義します。

EGCS リテラルはすべて、G'SO XX .... XX SI' の形式をとります。ここで SO (シフトアウト)は X'0E'、SI (シフトイン) は X'0F' です。

SCS1 装置タイプの EGCS は、G'SO XX XX XX SI' の形式をとり、データを囲む一対の制御文字として指定します。データを囲む文字の SO (シフトアウト) と SI (シフトイン) は実際の文字ではなく、X'0E' と X'0F' という 1 バイトのコードです。

EGCS リテラルには偶数個の文字を指定しなければなりません。奇数個を指定すると、警告メッセージが出されます。すべての文字 (X'00' から X'FF') が EGCS リテラルとして有効です。ただし、図形文字の定義範囲 X'40' から X'FE' 以外の文字を指定すると、警告メッセージが出されます。

制約事項**:** EGCS リテラルの 16 進数値が X'7D'7D の場合、この値は引用符と等しいため、EQU ステートメントを使用してそのリテラルを等価にすることはできません。

MFS 言語ユーティリティーに EGCS リテラルを認識させるためには、EGCS リテラルを定義するときに以下の制約を守ってください。

- ALPHA ステートメントを使用して、SO 文字および SI 文字を英字で定義してはならない。
- G'SO の 3 文字 (SO は 1 文字) を MFS 言語ユーティリティーへ入力するときは、継続行にまたがっていてはならず、同じ行に入っていなければならない。SI' の 2 文字についても同じことが言えます。

EGCS リテラルは次の行に続けることができます。SI 文字を 70、71、または 72 桁目にコーディングして EGCS データを終了させることができます。このときこの SI 文字はリテラルには含まれません。SI が 70 桁目にあると、71 桁目にあるデータは無視されます (データが単一引用符である場合を除く)。リテラルの継続行には

SO 文字を入れる必要はありませんが、15 桁目に入れるのであれば使用してもかまいません (これは、EGCS データの始まりを示すもので、リテラルには含まれません)。

制約事項: IMS は、インバウンドでもアウトバウンドでも 2 バイト充てん機能をサポートしません。アウトバウンド・データの場合、MFS 充てん機能はメッセージ・レベルで実行されます。出力メッセージの EGCS フィールドにデータが入っていないか、いくつかの EGCS が省略されている場合、MFS が RA (アドレス反復) 副指令を挿入するのを防ぐためには、FILL=PT (デフォルト) または FILL=NULL を指定しなければなりません。

MFS 言語ユーティリティーは、初期入力ステートメントとエラー・メッセージで入力レコードの EGCS リテラルを表示する場合に限って、出力のリストの中で SO 文字および SI 文字を使用します。装置イメージ・マップの一部になっている EGCS リテラルは、一連の G で表示されます。EGCS リテラルを含んだ EXEC PARM= オペランドの DIAGNOSTIC、COMPOSITE、および SUBSTITUTE を用いて作成されたこのほかのユーティリティー出力には、G、SO、および SI の文字は入りません。出力されるのは、SO 文字と SI 文字の間にあるデータだけです。

ユーザーはフィールドを表示する画面上の位置 (行と桁) を定義する必要があります。特定のプロダクトを導入したことによって画面の配置が制約を受けるときは、そのことも考慮してください。以下のような場合には警告メッセージが出されます。

- DFLD 属性が EGCS であるのにもかかわらず、フィールド位置パラメーターに奇数桁番号が指定されていない (3270 のみ)。
- 指定された EGCS リテラルの文字数が偶数でない。
- DFLD の長さが偶数でない。

3283-52 型に EGCS フィールドを定義するときは、指定する長さを必ず偶数にし、1 つの EGCS フィールドが複数の装置行にまたがる場合は、WIDTH= と POS= を指定して、各装置行で印刷位置が偶数になるようにすることが必要です。

関連概念:

574 ページの『出力メッセージ・リテラル・フィールド』

***DBCS/EBCDIC* 混合フィールド:**

2 バイト文字セット (DBCS) は、それぞれの文字が 2 バイトで表現される図形文字セットの 1 つであり、拡張図形文字セット (EGCS) のサブセットです。DBCS は、中国語、日本語、および韓国語などのいくつかのアジア言語を現すために使用されます。なぜなら、これらの書記言語は 1 バイトで現すことができる 256 を超える文字からなるためです。EGCS と同様に、この表現方法もプログラム式シンボル機能の拡張によって実現されます。

DBCS は EGCS のサブセットであるため、DBCS フィールドは、EGCS キーワードとパラメーターを使って指定します。MFS は、このフィールドを EGCS データとほぼ同じように扱います。ただし、DBCS データは、DBCS フィールドと DBCS/EBCDIC 混合フィールドの 2 つのフィールド・タイプで使用できます。DBCS フィールドは DBCS データだけを受け入れるため、このタイプのフィールドでは特別な制御文字は必要ありません。(DBCS データの有効なコード範囲は、

X'4040'、または 2 バイトとも X'41' から X'FE' です。) 一方、混合フィールドでは、DBCS データと EBCDIC データとが混在 しているので、DBCS データを SO (シフトアウト) と SI (シフトイン) の制御文字で囲まなければなりません。

DBCS を使用するには、DBCS データを扱うことができるディスプレイおよびプリンターが必要です。5550 ファミリー (3270 として) は、そのような装置グループの 1 つですが、他の 3270 DBCS 装置も使用できます。

DBCS および **EBCDIC** 混合フィールド

DBCS データが SO/SI 文字で囲まれていれば、3270 DBCS 装置の混合フィールドには、EBCDIC と DBCS の両方のデータが入ります。このような混合フィールドには、SO/SI 制御文字で囲んだ DBCS データ項目を複数入れることができます。

3270 表示装置からのインバウンド・データやアウトバウンド・データは、いずれも必ず SO/SI 制御文字で囲まれていなければなりません。ただし、インバウンド・データの場合は、端末が自動的に制御文字を作成します。 DBCS/EBCDIC 混合フィールドを明示的に指定するには、DFLD ステートメントの EATTR= パラメーターで MIX および MIXS キーワードを使用します。

例えば、以下の図は DBCS/EBCDIC 混合フィールドの例です。

以下の図に示した DBCS/EBCDIC 混合データは、以下の 16 文字で構成されています。

- EBCDIC データの 'ABCD' と 'EF' (6 バイト)
- DBCS データの 'GGGG' と 'GG' (6 バイト)
- 2 組の SO/SI 制御文字 (4 バイト)

SO 制御文字は X'0E' で表され、SI 制御文字は X'0F' で表されます。

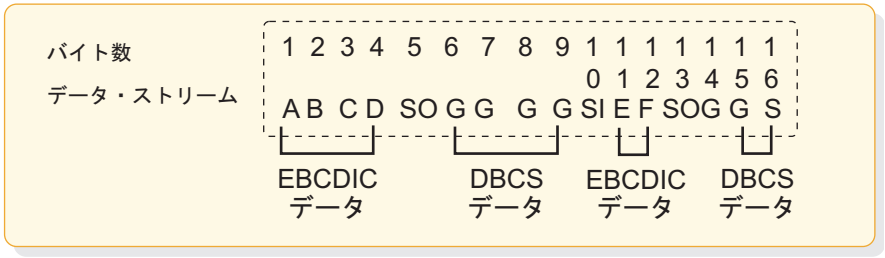


図 *33. DBCS/EBCDIC* 混合データ

DBCS を使用すると、MFS はデータを直接 3270 表示装置に送りますが、SCS1 プリンターには SO/SI のブランクの印刷処理を行ってからデータを送信します。3270 表示装置と SCS1 プリンターでは SO/SI 制御文字が以下のように処理されます。

- 3270 表示装置では、SO または SI 制御文字は画面上でそれぞれ 1 桁を占め、ブランクとして表示される。
- SCS1 プリンターの場合 :

- EATTR=MIXS を指定すると、SO または SI 制御文字はリスト上で桁位置を占めません。ブランクを挿入させないようにするには、EATTR=MIXS (SO/SI ブランク印刷抑止オプション) を指定します。
- EATTR=MIX を指定すると、SO/SI ブランク印刷オプションにより、混合データ・フィールドの SO 制御文字の前と SI 制御文字のあとにブランクが挿入されます。MIX を指定すると、3270 表示装置と SCS1 プリンターとで同じ出力が得られます。

アプリケーション・プログラムでは、SO/SI が入っている混合データの長さは、出力が印刷された場合、データの長さが異なります。

上記の図に示した DBCS/EBCDIC 混合データの長さは、アプリケーション・プログラムでは 16 バイトです。この文字ストリングを DFLD EATTR=MIX と指定したフィールドに送ると、16 バイトの文字ストリングが印刷されます。しかし、DFLD EATTR=MIXS と指定したフィールドに送れば、12 バイトの文字ストリングが印刷されます (SO/SI 制御文字の 4 バイトが抑止されます)。DFLD の長さ属性は、それぞれ LTH=16 と LTH=12 になります。

SO/SI 制御文字の処理

3270 表示装置では、SO/SI 制御文字で囲まれた DBCS データを、既存の EBCDIC フィールドの一部分として受け入れることができます。既存の EBCDIC フィールドに DBCS データを混ぜて入れるときは、IMS アプリケーション・プログラムで、3270 表示フィールドに入っている DBCS データが正しいかどうかを調べなければなりません。EBCDIC フィールドの DBCS データは、次の条件を満たすときに正しいといえます。

- DBCS 文字の長さが偶数バイトである。
- 対になっていない SO または SI 制御文字がない。

DFLD ステートメントで MIX または MIXS を指定しておくと、MFS はこれらの条件を検査し、SO/SI 制御文字で囲まれた DBCS データの位置合わせを行い、正しくない SO/SI 制御文字を訂正します。

DBCS/EBCDIC 混合リテラル

以下のコード例に示すように、DBCS/EBCDIC 混合リテラルは DFLD/MFLD リテラルとして指定することができます。

```
   literal format:' .......SO____SI..SO__SI'

DFLD
 'literal'

MFLD
 ,'literal'
 ,(dlfdname,'literal')
```

DBCS/EBCDIC 混合リテラル内の DBCS データは、MFS リストの装置イメージ・マップでは一連の G で表示されます。

MFS 言語ユーティリティーで EATTR= を指定せずに EBCDIC フィールド内に DBCS/EBCDIC 混合データを含む DFLD/MFLD リテラルを指定すると、3270 表示装置と SCS1 プリンターの両方の出力で混合フィールドの検査が行われます。

EATTR=MIX での DBCS/EBCDIC 混合フィールド属性は、SCS1 だけに割り当てられます。LTH パラメーターを指定しても無視されます。この結果、フィールド長はリテラルの長さと等しくなります。

以下の表は、DBCS/EBCDIC 混合フィールドの SO/SI 制御文字に関して IMS MFS 言語ユーティリティーが行う処理を示しています。 DFLD/MFLD 出力リテラルと MFLD 入力リテラルに続いて装置およびフィールドがリストされています。

表 *137. IMS MFS* 言語ユーティリティーが行う *SO/SI* 処理

装置、フィールド	**DFLD/MFLD** 出力リテラル	**MFLD** 入力リテラル
3270 表示装置、DBCS/EBCDIC 混合フィールド	• SO/SI が対になっているか検査する。 • 偶数長であるか検査する。 • 境界合わせを行う (警告メッセージを出す)。	SO/SI の検査は行わない。
SCS1 プリンター、DBCS/EBCDIC 混合フィールド	• SO/SI が対になっているか検査する。 • 偶数長であるか検査する。 • SO/SI ブランク印刷オプションに基づいて SO/SI の訂正と境界合わせを行う。	適用できない。

以下の表は、DBCS/EBCDIC フィールド内の SO/SI 制御文字に関して MFS メッセージ・エディターが行う処理を示しています。アウトバウンド・データ・フィールドとインバウンド・データ・フィールドに続いて装置およびフィールドがリストされています。

表 *138. MFS* メッセージ・エディターが行う *SO/SI* 処理

装置、フィールド	アウトバウンド・データ・フィールド	インバウンド・データ・フィールド
3270 表示装置、DBCS/EBCDIC 混合フィールド	• SO/SI が対になっているか検査する。 • 偶数長であるか検査する。 • 境界合わせを行う。	SO/SI の検査は行わない。
SCS1 プリンター、DBCS/EBCDIC 混合フィールド	• SO/SI が対になっているか検査する。 • 偶数長であるか検査する。 • SO/SI ブランク印刷オプションに基づいて SO/SI の訂正と境界合わせを行う。	適用できない。

DBCS/EBCDIC 混合リテラルの継続規則

混合リテラルの継続規則は、EGCS リテラルの継続規則と同じです。継続規則は以下のとおりです。

- EGCS リテラルは次の行に続けることができます。

- SI 文字を 70、71、または 72 桁目にコーディングして EGCS データを終了させることができます。このときこの SI 文字はリテラルには含まれません。SI が 70 桁目にあると、71 桁目にあるデータ (文字が単一引用符である場合を除く) は無視されます。
- リテラルの継続行には SO 文字を入れる必要はないが、15 桁目に使用することができる。 (これは、EGCS データの始まりを示すもので、リテラルには含まれません。)

混合リテラルには DBCS 文字ストリングが含まれているので、継続の場合、以下の点を考慮する必要があります。

- データに EBCDIC と DBCS が混合しているときは、DBCS データを必ず SO および SI 制御文字で囲む。SI 文字は、EGCS リテラルの 70 桁目から 72 桁目のどこに入れてもかまいません。混合リテラルでは、SO と SI はユーザー・データの一部です。したがって、71 桁目までデータを入れ、72 桁目には非ブランク文字を入れ、次の行は 15 桁目 (SO の場合) または 16 桁目から始めなければなりません。以下のコード例は、混合リテラルでの継続の例を示しています。
- DBCS 文字の最初のバイトが 71 桁目にきた場合は、72 桁目に非ブランク文字を入れ、次の行の 16 桁目に DBCS 文字の 2 番目のバイトを入れて、リテラルを継続することができる。別の解決策として、最初の行を 17 桁目から始めることもできます。

混合リテラルでの継続

混合リテラル

```
'abc{K1K2K3}'def where
                  abc & def  = EBCDIC characters
                  K1K2K3     = DBCS characters
                  {          = shift out X'0E'
                  }          = shift in X'0F'
```

混合リテラルでの継続の例

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                'zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzabc{K1}
              {K2K3}def'


|...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
                'zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzabc{K1K2K3}
              {}def'
```

SO/SI の対の検査と **SO/SI** の訂正

MFS は、SO/SI の対の検査で見つかった対になっていない SO 文字と SI 文字を、以下のようにして訂正します。

- EATRR=MIX が指定された 3270 の表示フィールドまたは SCS1 の印刷フィールド内では、すべての SO 制御文字 (フィールド内の対になっていない最後の SO 制御文字を除く) とすべての重複 SI 制御文字をブランクに置き換える。

  フィールドの最後の非対 SO 制御文字については、DBCS フィールドの長さが偶数になるように、最終バイトまたはその前のバイトに SI 制御文字を入れます。

SI 制御文字を最終バイトの前のバイトに入れた場合は、最終バイトを充てん文字に置き換えます。 SO 制御文字をフィールドの最終バイトに入れた場合は、それをブランクに置き換えます。

- EATRR=MIXS が指定された SCS1 印刷フィールドでは、フィールド内の対になっていない最後の SO 制御文字を除くすべての SO 制御文字とすべての重複 SI 制御文字を除去する。

  フィールドの最後の対になっていない SO 制御文字については、DBCS フィールドの長さが偶数になるように、最終バイトまたはその前のバイトに SI 制御文字を入れます。SI 制御文字を最終バイトの前のバイトに入れた場合は、最終バイトを充てん文字に置き換えます。SO 制御文字をフィールドの最終バイトに入れた場合は、それを充てん文字に置き換えます。

SCS1 プリンターの場合、そのフィールドに定義されている SO/SI の対の数を超える SO/SI 制御文字があると、対になっているかどうかに関係なく次のように処理されます。

- EATTR=MIX が指定されていればブランクに置き換えられる。
- EATTR=MIXS が指定されていれば除去される。

DBCS/EBCDIC フィールドの DBCS データの長さが奇数の場合は、奇数の SI の位置を 1 バイト左に移し、フィールドの残りの部分にブランクを埋め込みます。

入力制御と **DBCS/EBCDIC** 混合フィールド **(3270** 表示装置**)**

DBCS/EBCDIC データを DBCS/EBCDIC フィールドに送信するとき、MFS は、SO/SI が対になっていて偶数長であるかを検査して、必要であれば SO/SI を訂正し、境界に位置合わせをします。これにより、DBCS/EBCDIC フィールドは、3270 表示画面上または SCS1 印刷出力上に正しく現れます。

混合フィールドから DBCS/EBCDIC データを受け取ると、MFS はそのデータをそのまま受け渡します。これは、3270 表示装置を使用する場合は、SO/SI が対になっていることと偶数長であることが常に保証されているからです。

ただし、DBCS/EBCDIC データを DBCS/EBCDIC フィールドに送信し、このフィールドからユーザーが入力した DBCS/EBCDIC データを受け取る場合は、アプリケーション・プログラムではデータの変更に対応しなければなりません。3270 表示装置は、オペレーターが入力した DBCS データを受け取ると、データと SO/SI 制御文字を組み立て、必ず SO/SI が対になっていて偶数長になるように、データの切り捨てや再度の位置合わせを行います。IMS アプリケーション・プログラムは、送信データの一部を受信データとして使う場合、このことを考慮しなければなりません。

DBCS/EBCDIC 混合フィールドと水平タブ **(SCS1** プリンター**)**

オンライン水平タブ設定を使用する場合、DBCS/EBCDIC フィールドにはタブを設定することができません。これは、混合フィールドの DBCS データの実際の位置が、奇数の境界にくるのか偶数の境界にくるのかを、前もって判断することができないためです。

フィールドの枠取り

この機能は、ユーザーが定義する 3270 表示フィールドと SCS1 印刷フィールドで使用します。

フィールドを囲む枠は、それぞれ上線 (OVER)、下線 (UNDER)、左線 (LEFT)、右線 (RIGHT) と呼ばれ、個別に指定することも、任意に組み合わせて指定することもできます。

以下の図に表示されたフィールドの左端と右端のエリアは、以下のようになります。

- 3270 表示装置では、3270 基本属性バイト。左端の属性バイトは最初のフィールドの、右端の属性バイトは次のフィールドの属性をそれぞれ示します。
- SCS1 プリンターでは、MFS がユーザー定義フィールドのために確保しておく左右のブランク。

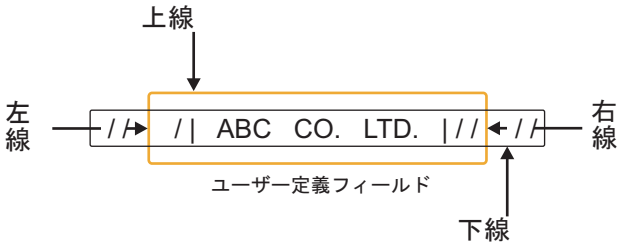


図 *34.* ユーザー・フィールドとフィールドの枠取り

フィールド枠の接続とフィールドの結合

以下の図に示すように、複数のフィールドをつなげて枠取りを行うことができます。

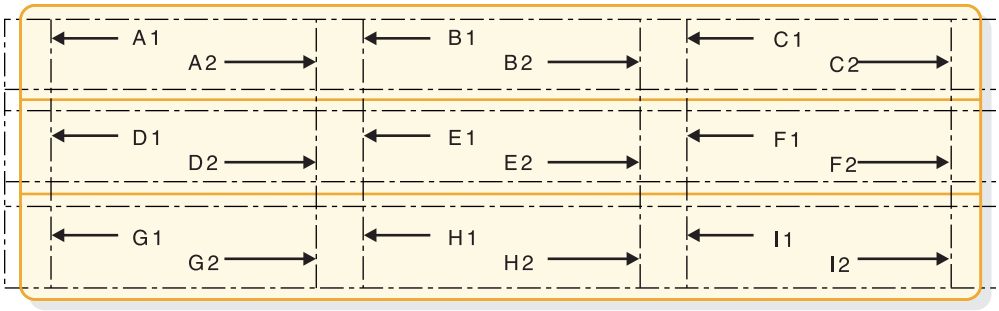


図 *35.* ユーザー・フィールド連結時のフィールドの枠取り

上記の図は、9 個の論理フィールドから構成されています。A1、B1、... I1 は 3270 表示装置用に定義されたフィールドで、A2、B2、... I2 は SCS1 プリンター用に定義されたフィールドです。3270 表示装置の場合は、3270 基本属性バイトがフィールドとフィールドとの間に存在することに注意してください。SCS1 プリンターの場合は、どの印刷位置も失われずにフィールドがつながり、フィールドの枠がつながります。上記の図にある各フィールドの枠の指定については、以下の表に示します。

表 *139.* 各フィールドの枠の指定

フィールド	**LEFT**	**RIGHT**	**OVER**	**UNDER**
A1、A2	X		X	
B1、B2			X	
C1、C2		X	X	
D1、D2	X		X	
E1、E2			X	
F1、F2		X	X	
G1、G2	X		X	X
H1、H2			X	X
I1、I2		X	X	X

IMS アプリケーション・プログラムでディスプレイとプリンターに同じ出力を作成したい場合は、プログラムで 3270 表示装置用のメッセージ・フィールドを定義すれば済みます。

SCS1 プリンターでフィールドの枠取りを指定すると、MFS 言語ユーティリティーは左線と右線のための 1 バイトを確保しようとしますが、隣接するフィールドを確保できないときは、警告メッセージを出します。

カーソル位置付け*:*

3270、3604、または SLU 2 表示装置のカーソル位置は、行と桁位置によって物理ページ単位で行われます。カーソルを常にある特定位置に置く必要があるときは、装置依存性に問題なければ、DPAGE ステートメントでカーソル位置を定義しておくことができます。

また、入力時にカーソル位置がアプリケーション・プログラムに通知され、出力時にアプリケーション・プログラムでカーソル位置を動的に指定できるように、DPAGE ステートメントで定義しておくこともできます。出力時にカーソル位置を動的に定義するには、装置フィールド名を、行や桁位置とともに指定します。さらに、このフィールドを MID MFLD ステートメントで参照すれば、メッセージ入力時にはそのメッセージ・フィールドにカーソル位置が示されます。このメッセージ・フィールドを MOD MFLD ステートメントで参照すれば、アプリケーション・プログラムでメッセージ・フィールドを使用して出力時のカーソル位置を指定することができます。

アプリケーション・プログラムのカーソル位置要求が使用されるのは、装置タイプが 3270-An の場合は指定するサイズが TERMINAL マクロの SIZE= オペランドの行と桁指定の範囲内である場合、つまり、3270-1 型または 3270-2 型の行と桁の境界内にある場合です。それ以外の場合は、DPAGE ステートメントで指定した行と桁位置か、デフォルトの位置 (1 行目、2 桁目) になります。

関連資料**:** TERMINAL マクロについては、「*IMS V14* システム定義」を参照してください。

入力時にカーソル位置を知らせるオプションは、3270 または SLU 2 装置でのみ使えるものです。このカーソル位置付け方式では、アプリケーション・プログラムが

特定のフィールド位置を使用することが必要であり、したがって装置依存型プログラムとなるため、出力時にはお勧めできません。MFS は、カーソル位置を 1 つの装置フィールド属性と見なすため、フィールド属性機能を使用してカーソル位置を設定することができます。

装置フィールドに対応するメッセージ・フィールドで ATTR=YES または ATTR=nn を指定しておけば、アプリケーション・プログラムは装置フィールドの属性を、動的に置換、変更、あるいはシミュレートすることができます。このように定義したメッセージ・フィールドでは、少なくとも先頭の 2 バイトがアプリケーション・プログラムが与える属性データまたは拡張属性データ用に予約されます。

区画形式モードの 3290 では、区画記述子ブロック (PDB) に定義されている最初の区画記述子 (PD) ステートメントで最初の区画が作成されます。そして、区画移動キーや、その後の出力メッセージと関連付けられている DPAGE ステートメントの ACTVPID= キーワードで指定変更されないかぎり、カーソルが入っているこの区画がアクティブ区画になります。

区画移動キーを押すと、カーソルはアプリケーション・プログラムで定義された次の順番の区画に移動し、その区画が今度はアクティブ区画になります。ACTVPID= キーワードを指定しておけば、アプリケーション・プログラムは特定の区画にカーソルを置き、その区画を活動化することができます。

プロンプト機能*:*

プロンプト機能は、現在の出力ページが最終ページであるかどうかを自動的にオペレーターに通知するための手段として用いることができます。

通知テキストをリテラルで定義しておくと、MFS がメッセージの最終論理ページをフォーマット設定するときに、MFS はそのリテラルを指定された装置フィールドに挿入します。次になにを入力しなければならないかをプロンプト・テキストでオペレーターに伝えるようにすれば、オペレーターの負担が軽減されます。

推奨事項**:** 3270 または SLU 2 装置の場合、PROMPT と FILL=NULL を組み合わせて使用するときは注意が必要です。いったんプロンプト・リテラルが表示されると、オペレーターの入力によって画面の形式が再設定されないかぎり、そのプロンプト・リテラルは画面に残されたままになるためです。

システム・メッセージ・フィールド ***(3270*** または ***SLU 2*** 表示装置***):***

3270 または SLU 2 表示装置の出力形式には、システム・メッセージ・フィールドの組み込みを定義しておくことができます。このように定義すると、装置がフォーマット設定モードのときには必ず「REQUESTED FORMAT BLOCK NOT AVAILABLE」以外のすべての IMS メッセージがシステム・メッセージ・フィールドに送信されます。システム・メッセージ・フィールドを使用するか、または DSCA 指定のバイト 1 のビット 5 を B'0' に設定しておくと、画面形式が IMS メッセージによって破壊されるのを防ぐことができます。

MFS は、システム・メッセージ・フィールドにメッセージを送るときに装置アラーム (ある場合) を活動化しますが、変更データ・タグ (MDT) のリセット、カーソルの移動、および表示の保護/無保護状況の変更は行いません。ただし、複数セグメントのメッセージのイベントの場合を除きます。この場合、状況は保護に変わり、

Enter キーを押してメッセージの次のセグメント (複数可) を調べなければなりません。IMS エラー・メッセージは、入力での MDT に対する即時応答なので、MDT はオペレーターが入力したときのままの状態になっています。したがってオペレーターは、入力のうちのエラー部分だけを訂正すれば済みます。

オペレーター識別 (OID) カード読取機構からの入力が終わった後は、装置はフォーマット設定モードではなくなっています。したがって、IMS メッセージは SYSMSG フィールドへは送られず、デフォルトのシステム・メッセージ形式で送られます。XRF (拡張回復機能) テークオーバーが行われた後も、装置はフォーマット設定モードではなくなっているので同じようになります。

印刷ページのフォーマット制御*:*

DEV ステートメントの PAGE= キーワードは、プリンターに送信される出力メッセージの形式制御の大部分を行います。

WIDTH= キーワードで、フォーマット設定をさらに細かく制御します。FEAT=(1...10) キーワードとともに WIDTH= を指定すると、3270P として指定されたプリンターの形式をさらに細かく指示することができます。(追加情報は、DEV ステートメントの下の WIDTH= を参照。) WIDTH= キーワードを HTAB=、VTAB=、 VT=、SLDI= および SLDP= キーワードと一緒に指定すると、3770、または SLU 1 プリンターのフォーマット設定制御をさらに細かく指示することができます。

PAGE= オペランド (DEFN、SPACE、FLOAT、または EJECT) を、ページ行数 (1 ページあたりの行数) と一緒に使用すると、出力メッセージの印刷を MFS がどのように制御するかが決まります。PAGE= オペランドには、以下のものがあります。

DEFN

MFS は DFLD ステートメントで定義されているとおりに各行を印刷します。このモードでは、最初の DFLD によって定義されている行が 1 より大きいと、印刷位置は定義されている最初の行まで移動します。定義された DFLD と DFLD との間にブランク行がある場合、印刷位置はそれらのブランクを飛び越して移動します。しかし、定義されている最後の行がページ行数に達しなくても、MFS は出力ページの末尾にブランク行をつけ足しません。出力の現在行の次の行から次ページの出力が開始されます。PAGE= キーワードで指定した値は、DFLD POS= キーワードの行数指定値の妥当性を確かめるために用いられます。

SPACE

このオペランドを指定すると、DEFN の場合と同じ印刷モードが得られますが、定義された最終行がページ行数に達していないときにはページの末尾にブランク行が追加される点が異なります。プリンターは、一連の改行を行ってから次の印刷位置に置かれます。このオプションは、ページ・イジェクト機構が備わっていない装置の印刷出力で、ページが切れ目なくつながってしまうことを防ぐために使用します。

FLOAT

このオペランドは、DFLD ステートメントによって定義されている行や、

フォーマット設定後にデータがまったく入っていない行 (すべてがブランクかヌルの行) を印刷しないように要求するために使用します。

EJECT

このオペランドは FIN、3770、または SLU 1、プリンターに対して指定します。EJECT では、以下のオプションを (必要に応じ組み合わせることも可) 指定することができます。

BGNPP または **ENDPP**

MFS は、出力メッセージの各物理ページの前 (BGNPP)、または後 (ENDPP) でページ替えを行います。

BGNMSG

MFS は、出力メッセージのデータの印刷が開始される前にページ替えを行います。

ENDMSG

MFS は、出力メッセージのすべてのデータが印刷されてしまった後にページ替えを行います。

MFS がページに行を追加したり、ページの行を削除したりすることはありません。EJECT は、 FIN、3770、または SLU 1、プリンターに対して指定します。

関連概念:

590 ページの『3270P プリンターの出力形式制御』

『3770 および SLU 1 プリンターの形式制御』

3770 および ***SLU 1*** プリンターの形式制御**:**

MFS では、3770 プリンターおよび SLU 1 (印刷データ・セット) (DEV TYPE=SCS1) あての出力メッセージの形式を制御するための指定が何通りかあります。

行幅

DEV ステートメントの WIDTH= キーワードで、印刷行の最大幅を 1 桁目を基準にして指定します。これを指定すると、物理デバイスの行幅の代わりにこの幅が使用されます。行幅を指定すると、同時に、印刷ページの右マージンの位置も決定します (1 桁目を基準)。指定できる値は、物理デバイスの行幅に等しいか、またはそれより小さい値です。例えば、WIDTH=80 と指定したとすると、1 桁目から 80 桁目にデータを印刷することができます。

左マージン位置

左マージンの位置は、DEV ステートメントの HTAB= キーワードにある左マージン・オペランドで指定できます。MFS は、出力メッセージをプリンターあてに送信する前に、この指定に基づいて装置の左マージンをセットします。出力メッセージが常に 1 桁目 (デフォルトの左マージン位置) 以外の位置から始まる場合は、左マージンを指定しておかなければなりません。例えば、常に 5 桁目から 80 桁目に出力フィールドを定義する場合は、DEV ステートメントで HTAB=(5) と WIDTH=80 を指定しておきます。

水平タブ

水平タブ停止位置は、DEV ステートメントの HTAB= キーワードで指定することができます。MFS は、出力メッセージの送信に先立って、その位置に水平タブ停止をセットします。

MFS で、タブ制御文字をメッセージに挿入することもできます。そうすることにより、送信される文字数を減らすことができます。タブ制御文字の挿入は、HTAB= キーワードに ONLINE または OFFLINE オペランドを指定することによって制御します。OFFLINE を指定すると、オフライン MFS 言語ユーティリティー・プログラムが制御ブロックをコンパイルする時点で、MFS はタブ制御文字を挿入します。ONLINE を指定すると、MFS は、メッセージをオンラインで処理しているときにタブ制御文字を挿入します。タブ制御文字は、有効な充てん文字を指定してある (DPAGE ステートメントで FILL=X'hh' または FILL=C'c' を指定した) か、またはデフォルトの充てん文字 (X'40') を使用するメッセージにしか挿入することができないため、それ以外のメッセージへの挿入を MFS に指示してはなりません。

定義したほとんどの装置フィールドに対しメッセージ定義から常にデータを与える場合、あるいは充てん文字がブランクでない場合は、OFFLINE を指定します。データが送られてこない装置フィールドがある場合や、データにブランクが入っている場合は、ONLINE を指定します。 ONLINE を指定すると MFS オンライン・プロセス量は増えますが、装置への文字の伝送量は減ります。

垂直タブ

出力メッセージのページのどこに垂直タブ制御文字を挿入すべきかを MFS に指示するには、DEV ステートメントの VT= キーワードを使用します。 MFS は、1 行目を基準にして垂直タブ停止位置を決めるものと見なし、また、宛先の装置では、メッセージの送信前に VTAB= キーワードの指定または他の方法によってその垂直タブ停止位置がすでにセットされているものと見なします。垂直タブが必要な場合は、必ず VT= を指定しなければなりません。デフォルト値はありません。フォーマット設定後にデータがまったく入っていない行を MFS に削除させるようにページ制御を指定した場合は、VT= を指定してはなりません。 VT= キーワードを指定する場合は、EJECT BGNMSG または EJECT BGNPP も同時に指定することにより、ページの先頭で正しい位置合わせが行われるようにしておく必要があります。適切な EJECT 操作を指定せずに VT= を単独で指定すると、装置のフォーマット設定が正しく行われないことがあります。

上部マージンおよび下部マージン

DEV TYPE=SCS1 として指定したプリンターの上部および下部のマージンは、DEV ステートメントの VTAB= キーワードで指定することができます。フォーマット設定後にデータがまったく入っていない行を MFS に削除させるためにページ制御を指定した (PAGE=n,FLOAT を指定した) 場合は、VTAB= を指定してはなりません。

ページ行数 (PAGE=)、垂直タブ (VT=)、および上下マージン (VTAB=) を同時に指定すると、『垂直フォーマットの設定』データ・ストリームの指定になります。

行密度

DEV TYPE=SCS1 として指定したプリンターの場合、DEV ステートメント、DFLD ステートメント、あるいはこの両方のステートメントに SLDx= キーワードを使用して、出力ページの行密度を指定することができます。行密度は、1 インチ当りの行数または 1 インチ当たりのポイント数で設定します。DEV ステートメントと DFLD ステートメントの両方に SLDx= を指定すると、2 つの SLD データ・ストリーム (メッセージの始まりに 1 つとメッセージの中で 1 つ) が、SLDx が指定されているフィールドの直前で、しかも垂直タブと改行文字のあとに送信されます。メッセージに SLDx の指定が入っていると、そのメッセージの初めで設定されていた行密度からメッセージで指定されている行密度に、行密度が変更されます。メッセージ内で指定した行密度は、明示的にリセットされるまで有効です。

関連概念:

587 ページの『印刷ページのフォーマット制御』

3270P プリンターの出力形式制御**:**

MFS では、3270P プリンターあてメッセージの形式を制御するための指定がいくつかあります。

行幅

DEV ステートメントの WIDTH= キーワードで、印刷行の最大幅を 1 桁目を基準にして指定します。これを指定すると、物理デバイスの行幅の代わりにこの幅が使用されます。 3270P プリンターでのデフォルトは 120 です。WIDTH= を指定するときは、DEV ステートメントの FEAT= キーワードを使用して機構コード (1 から 10) も指定しなければなりません。

関連概念:

587 ページの『印刷ページのフォーマット制御』

SLU P DPM-An の出力形式制御**:**

DPM-An オプションを指定した SLU P 装置では、MFS でいくつかの指定を行って出力メッセージの形式を制御することができます。

DIV ステートメントと RCD ステートメントの RCDCTL= オペランドは、1 レコード内の装置フィールド (DFLD) 定義の関連グループを示します。通常、このレコードが 1 回の送信単位としてリモート・プログラムに送信されます (RCDCTL= の値がシステム定義の TERMINAL マクロの OUTBUF= パラメーターの値以下の場合)。

レコードの装置フィールドの数は、RCDCTL で指定した長さ (数値) によって決まります。レコードの装置フィールドの配置は、RCD ステートメントで決めることができます。RCDCTL で指定したサイズより短いレコードを作成することができます。SPAN/NOSPAN パラメーターにより、各フィールドを 2 つのレコードにまたがって入れることができるかどうかを決めます。出力メッセージはすべて、レコード・モードで送信されます。

PPAGE ステートメントでは、装置形式の表示ページを指定します。表示ページには 1 つ以上のレコードを入れることができます。

DPAGE ステートメントでは、装置形式の論理ページを定義します。論理ページには 1 つ以上のレコードを入れることができます。

ページング

DIV ステートメントの OPTIONS= 指定の MSG、DPAGE、PPAGE の各オペランドは、リモート・プログラムに出力メッセージを送信する方法を決定するときに使用します。

MSG これは、出力メッセージ内の全データを 1 つのチェーンにまとめて、リモート・プログラムに送信する指定です。これはデフォルトです。

TERMINAL というシステム定義マクロの COMPTn オペランドに、メディア・パラメーターとして PROGRAM2 が指定されていると、IMS はメッセージをリモート・プログラムに送信した後で別の出力メッセージを送信しません。つまり、次のメッセージを送信するには、その前に、リモート・プログラムから入力要求を受け取る必要があります。ところが、PROGRAM1 が指定されていると、IMS は、送信可能な別の出力メッセージがあれば、入力要求を待たずにそれをリモート・プログラムに送信します。

DPAGE

これは、論理ページの全データを 1 つのチェーンにまとめて、リモート・プログラムに送信する指定です。出力メッセージの次の論理ページを取り出すためには、リモート・プログラムがページング要求を出す必要があります。

PPAGE

これは、表示ページの全データを 1 つのチェーンにまとめて、リモート・プログラムに送信する指定です。出力メッセージの次の表示ページを取り出すためには、リモート・プログラムがページング要求を出す必要があります。

ページング要求は、入力メッセージ・ヘッダーまたはオペレーター制御テーブルを用いて指定することができます。OPTIONS=DPAGE または PPAGE を指定した場合、最終論理/表示ページをリモート・プログラムに送信してしまった後で、IMS MFS が行うことは、PROGRAM1 または PROGRAM2 の指定に関係なく、3270 および 3604 装置の場合と同じです。

どのチェーンにも出力メッセージ・ヘッダーが 1 つ含まれています。出力メッセージ・ヘッダーの DATANAME は、OPTIONS=MSG を指定した場合は形式名、OPTIONS=DPAGE を指定した場合は現行の装置論理ページ (DPAGE) の名前、OPTIONS=PPAGE を指定した場合は現行の表示ページ名です。

チェーンの最初の伝送レコードには、常に、出力メッセージ・ヘッダーが入っています。OPTIONS=MSG の場合は、最初の伝送レコードには出力メッセージ・ヘッダーしか入っておらず、メッセージ・データの伝送は次の伝送レコードから開始されます。

OPTIONS=DPAGE または PPAGE の場合、次のいずれかの状態のとき、最初の伝送レコードの出力メッセージ・ヘッダーのあとにデータが続きます。

- RCDCTL=(,SPAN) を指定してあり、RCDCTL の長さが出力メッセージ・ヘッダーの長さより長い。

- RCDCTL=(,NOSPAN) を指定してあり、RCDCTL の長さが RCDCTL の出力メッセージ・ヘッダーの長さより長く、少なくとも現行の DPAGE または PPAGE で定義されている最初のデータ・フィールドが最初の伝送レコードに完全に収まる。

出力メッセージ・ヘッダー

基本出力メッセージ・ヘッダー内には、以下の MFS フィールドがこの順で入っています。

VERSION ID (バージョン ID)

MIDNAME (MID 名)

DATANAME (データ名)

DATANAME は、OPTIONS=MSG の場合は FMT ラベル、OPTIONS=DPAGE の場合は DPAGE ラベル、OPTIONS=PPAGE の場合は PPAGE ラベルです。

DEV ステートメントで用紙リテラルを指定すると、出力メッセージ・ヘッダーに FORMSNAME フィールドが入ります。 OPTIONS=MSG の場合、FORMSNAME は基本ヘッダーの DATANAME のあとにあります。 OPTIONS=DPAGE または PPAGE の場合、オプションの用紙出力メッセージ・ヘッダーは基本出力メッセージ・ヘッダーの前にあります。この中には、以下のフィールドが入ります。

MIDNAME (MID 名)

FORMSNAME (用紙名)

この用紙ヘッダーは、チェーンの単独エレメントとしてリモート・プログラムに送信されます。リモート・プログラムは、このヘッダーを処理し、出力メッセージの最初の論理ページまたは表示ページを処理する準備が整ったならば、ページング要求を出す必要があります。

出力メッセージ・ヘッダーの長さは、DIV ステートメントの HDRCTL= オペランドで固定長または可変長として定義することができます。

固定長の基本出力メッセージ・ヘッダー (FORMSNAME がない場合) の長さは、OPTIONS=MSG の場合は 23 バイト、OPTIONS=DPAGE または PPAGE の場合は 25 バイトです。 FORMSNAME が存在しているときの基本出力メッセージ・ヘッダーの最大長は、OPTIONS=MSG の場合は 40 バイト、OPTIONS=DPAGE または PPAGE の場合は 33 バイトです。

- HDRCTL=FIXED を指定すると、MIDNAME と DATANAME の両フィールドは、定義可能な最大長になるまで常にブランクで埋められます。つまり、MIDNAME は 8 バイト (MIDNAME を指定しなかったときは 8 個のブランクが入る) まで、FMT 名は 6 バイトまで、DPAGE 名または PPAGE 名は 8 バイトまでです。したがって、基本出力メッセージ・ヘッダー内の DATANAME の位置は常に同じです。また、FORMSNAME が存在している場合も、その変位は常に同じであり、OPTIONS=MSG の場合は FMT 名の次に、OPTIONS=DPAGE または PPAGE の場合は MIDNAME の次にきます。
- HDRCTL=VARIABLE を指定すると、MIDNAME にも DATANAME にも埋め込みは行われません。 MIDNAME が 8 バイト未満であるか、省略されている場合、出力メッセージ・ヘッダーの DATANAME、FORMSNAME、またはその両方の位置は変わってきます。

以下の表は、OPTIONS=MSG の場合の固定長出力メッセージ・ヘッダーの形式を示しています。

表 *140. OPTIONS=MSG* の場合の固定長出力メッセージ・ヘッダーの形式

フィールドのバイト数	BASE 7	LI 1	MIDNAME 8	L2 1	DATANAME 6	L3 1	FORMSNAME (ユーザーがコーディングしたリテラル)

BASE 長さ 7 バイトの基本 DPM-An 出力ヘッダーで、バージョン ID が入ります。

L1 MIDNAME の全長に 1 を加えた値です。9 の値が入っています。

MIDNAME (MID 名)
入力で使用される MIDNAME が入ります。この名前が 8 文字未満のときは、8 バイトになるまでブランクで埋められます。MIDNAME を指定しないときは、このフィールドには 8 個のブランクが入ります。

L2 形式名 (DATANAME) の全長に 1 を加えた値です。7 の値が入っています。

DATANAME (データ名)
データ・フィールドのフォーマット設定をするために用いた形式名。長さ 6 文字未満の形式名を指定すると、6 バイトになるまで埋め込みが行われます。

L3 用紙リテラルの長さに 1 を加えた値が入ります。最大値は 17 です。

FORMSNAME (用紙名)
DEV ステートメントの FORS= パラメーターで指定したリテラルが入ります。長さは 1 文字から 16 バイトです。DEV ステートメントの FORS= を指定しなかったときは、L3 フィールドも FORMSNAME フィールドも出力メッセージ・ヘッダーには作られません。

DIV ステートメントの HDRCTL= オペランドで可変長の出力メッセージ・ヘッダーを指定すると、OPTIONS=MSG の場合の出力メッセージ・ヘッダーの形式は固定長のときと同じですが、MIDNAME と DATANAME の末尾のブランクは省かれ、それに応じて長さフィールドも調整されます。MIDNAME を使用しないときは、MIDNAME フィールド自体もその長さも存在しません。

以下の表は、OPTIONS=DPAGE または PPAGE の場合の固定長の基本出力メッセージ・ヘッダー (FORMSNAME がない場合) の形式を示しています。

表 *141. OPTIONS=DPAGE* または *PPAGE* の場合の固定長の基本出力メッセージ・ヘッダー *(FORMSNAME* がない場合*)*

フィールドのバイト数	BASE 7	L1 1	MIDNAME 8	L2 1	DATANAME 8

BASE 内容は OPTIONS=MSG の場合と同じです。

L1 内容は OPTIONS=MSG の場合と同じです。

MIDNAME (MID 名)
内容は OPTIONS=MSG の場合と同じです。

L2 DPAGE 名または PPAGE 名 (DATANAME) の全長に 1 を加えた値です。 9 の値が入っています。

DATANAME (データ名**)**
現行の論理ページまたは表示ページのデータ・フィールドをフォーマット設定するときに用いた DPAGE または PPAGE の名前が入ります。長さ 8 未満の DPAGE 名または PPAGE 名が指定してあると、8 バイトになるまでブランクが埋め込まれます。

以下の表は、OPTIONS=DPAGE または PPAGE の場合のオプションの用紙出力メッセージ・ヘッダーの形式を示しています。

表 *142. OPTIONS=DPAGE* または *PPAGE* の場合のオプションの用紙出力メッセージ・ヘッダー

フィールドのバイト数	BASE 5	L1 1	MIDNAME 8	L2 1	FORMSNAME (ユーザーがコーディングしたリテラル)

BASE オプションの用紙出力メッセージ・ヘッダー内の BASE フィールドには、バージョン ID が入りません。

L1 9 の値が入っています。

MIDNAME (MID 名**)**
内容は OPTIONS=MSG の場合と同じです。

L3 コーディングされている用紙リテラルの長さに 1 を加えた値が入ります。

FORMSNAME (用紙名**)**
ユーザーがコーディングしたリテラルが、OPTIONS=MSG の場合の固定長出力メッセージ・ヘッダーとして入ります。

命名規則

形式、装置論理ページ、表示ページ (つまり、FMT、DPAGE、PPAGE の各ステートメントのラベル) の命名規則を設定してください。例えば、3790 のパネルまたは機能のプログラム・サブルーチンに関して、リモート・プログラムが FMT 名、DPAGE 名、PPAGE 名を解釈できるようにするための命名規則を設定することができます。DPM-An 出力メッセージ・ヘッダーも標準化してください。

PPAGE ステートメントのユーザー作成ラベルには、形式定義内で固有の名前を付けなければなりません。また、IMS システム内でも固有のラベルにすることをお勧めします。

形式定義で OPTIONS=PPAGE が選択してあると、出力メッセージ・ヘッダーの DATANAME としてその PPAGE ラベルが送信されます。ラベルは、データの処理方法を決定できるだけの情報をリモート・プログラムに与えるものでなければなりません。PPAGE のラベルをユーザーがコーディングしておかなかった場合は、MFS が PPAGE のラベルを生成し、名前を出力メッセージ・ヘッダーに入れて送信します。MFS 生成の名前は、リモート・プログラムで使用することはできますが、特定の PPAGE 用に生成された名前は MFS 定義の再コンパイルのたびに変更される可能性があるため、MFS が生成したラベルをそのまま使用することはお勧めできません。

DPM 出力レコード内のヌル文字の削除

伝送レコード内のヌル文字の削除については、「*IMS V14* データベース・ユーティリティー」の DPAGE ステートメントの FILL=NULL の項を参照してください。

関連資料:

524 ページの『DIV ステートメント』

ISC (DPM-Bn) サブシステムの出力形式制御

IMS は、ISC ノードを使用して MFS の主要な出力メッセージ・フォーマット設定機能をサポートします。

形式制御

MFS には、ISC ノードへの出力メッセージの形式を制御するための指定がいくつかあります。DIV ステートメントで OPTIONS=DPAGE または OPTIONS=PPAGE を指定すると、MFS は、出力メッセージを複数の論理ページまたは表示ページの形にして送信します。システム制御域 (SCA) のバイト 1 ビット 5 をセットしておくと、メッセージのこれらのページは、要求に応じて、または自動的に送信されます。

機能管理 (FM) ヘッダー

FM ヘッダーは、ページングなどの機能を制御する出力メッセージのヘッダーです。

ページングされた出力メッセージ

DPM-Bn のページング・サポートの場合に、DIV ステートメントで OPTIONS=DPAGE または OPTIONS=PPAGE が指定してあると、MFS は出力メッセージを複数の論理ページまたは表示ページの形にして送信します。

要求時ページング

要求時ページングを使用すると、相手のサブシステムからページング要求を受け取ったときにだけ、論理ページまたは表示ページが送信されます。メッセージの最初の出力には、ATTACH FM ヘッダーだけしか入っていません。 DIV OPTIONS=DNM を指定すると、データ構造名 (DSN) も送信されます。

自動ページ化出力

このオプションは、SCA 値に基づいてメッセージ単位で使用することができます。この機能を使用すると、論理ページまたは表示ページが複数伝送チェーン (ページ当たり 1 つの伝送チェーン) でただちに送信されます。このオプションを使用すると、受信側は、複数伝送チェーンの形で出力メッセージ全体を入手することになります。必要であれば、それぞれの送信チェーンの中に DSN が入ります。

制約事項**:** ページング要求を入力してメッセージの受信を制御することはできません。

メッセージ内のページの可変長フィールドのデータがない場合、ヌルのデータ・チェーンが発生することがあります。

DEV ステートメントで DSCA= オペランドまたは MFLD ステートメントの SCA オプションのバイト 1 ビット 5 で、自動ページ化出力を指定します。

対応する MSG 定義で PAGE=YES を指定し、同時に自動ページ化出力も要求すると、PAGE=YES 指定 (オペレーター論理ページング) 機能はリセットされ、出力メッセージはメッセージの最後でデキューされます。オペレーター論理ページングは、MFS 要求時ページング出力にのみ適用されます。

出力モード

IMS からの出力を処理するために、ATTACH マネージャーには 2 種類のブロック化アルゴリズムが用意されています。すなわち、可変長可変ブロック化 (VLVB) レコードとチェーン式要求応答単位 (RU、MFS ストリーム・モード) です。 MFS が相手のサブシステムに送信するにあたり ATTACH マネージャーに渡す各レコードの先頭には、長さフィールドが付いています。各長さフィールドには、MFS が渡すレコードのサイズが入っています。レコードそのものは、必要なだけ RU を用いて送信されます。フィールドは複数の RU 境界にまたがっていてもかまいませんが、複数のレコード境界にまたがっていてはなりません。伝送チェーンでの VLVB レコードの数と MFS レコードの最大サイズは、選択した出力モードと指定したページング・オプションによって決まります。

ストリーム・モードの場合、DFLD を定義する方法は、どの OPTIONS= キーワードを使用するかよって決まります。

- OPTIONS=MSG (ページングが定義されていない) の場合、DFLD は DPAGE 内に定義される。
- OPTIONS=DPAGE (ページングが定義されている) の場合、DFLD は DPAGE 内に定義される。
- OPTIONS=PPAGE (ページングが定義されている) の場合、DFLD は PPAGE 内に定義される。

3 つの OPTIONS= キーワードのどれを設定しても、1 つの DPAGE (または PPAGE) 内で定義されているすべての DFLD がグループ化され、1 つの MFS レコードとして送信され、1 つの DPAGE (または PPAGE) 内のすべてのデータが 1 つの MFS レコードに相当し、しかも 1 つの出力 RU チェーンに相当します。1 つ以上の RU チェーンが出力メッセージの単一伝送チェーンの形で送信されます。

DIV または DPAGE ステートメントの OFTAB パラメーターを定義すると、以下の場合には、連続出力フィールド・タブ分離文字は取り除かれ、サブシステムには送信されません。

- OPTIONS=MSG の場合はメッセージの終わり
- OPTIONS=DPAGE の場合は DPAGE の終わり
- OPTIONS=PPAGE の場合は PPAGE の終わり

レコード・モードでは、DPAGE または PPAGE に定義された DFLD は送信時にさらに小さいレコードに分けられます。そのときに作成される MFS レコードの最大長は、DIV ステートメントの RCDCTL パラメーターで定義します。RCDCTL= パラメーターを指定しなかった場合、デフォルトで最大 256 バイトの長さのレコードが使用できます。RCD ステートメントは、新しいレコード境界から DFLD を開始するために使用します。

OFTAB パラメーターを定義すると、レコードの終わりに連続出力フィールド・タブ分離文字 (フィールドを省略した場合と、最終データ・フィールドが短い場合) は、送信前に除去されます。このようにしてレコード全体が排除され、しかもそのあとにデータ・レコードがまだ続く場合は、1 つの出力フィールド・タブ分離文字が入った 1 バイト・レコードが送信されます。以下のような場合にレコードが排除されます。

- OPTIONS=MSG の場合はメッセージの終わり
- OPTIONS=DPAGE の場合は DPAGE の終わり
- OPTIONS=PPAGE の場合は PPAGE の終わり

出力メッセージ (OPTIONS=MSG) の単一伝送チェーンまたは 1 ページ (OPTIONS=DPAGE または PPAGE) で、1 つ以上の VLVB レコードが送信されます。

関連資料:

507 ページの『装置依存の出力情報』

FILL=NULL 指定:

DPAGE または MSG ステートメントに FILL=NULL を指定し、DIV または DPAGE ステートメントに OFTAB= パラメーターを指定し、フィールドの区切りを保ちます。DPAGE または MSG ステートメントに FILL=NULL を指定し、しかも DIV ステートメントにも DPAGE ステートメントにも OFTAB= パラメーターを指定しなければ、出力データ・ストリームは圧縮されるので、フィールドの区切りが明白でなくなります。

図形データには FILL=NULL を使用します。SEG ステートメントに GRAPHIC=NO および FILL=NULL を指定すると、非図形データ・ストリームの X'3F' は圧縮によってセグメントから除かれるため、意図したものと違う結果になることがあります。非図形データの出力は、固定長出力フィールドとして送信しなければなりません。その際、FILL=NULL は使用しないでください。

MSG ステートメントの OPT= オペランドでオプション 1 または 2 を指定すると、出力メッセージ・セグメント、および各セグメントで定義したメッセージ・フィールドは、MFS により順次処理されます。オプション 1 または 2 のセグメントのメッセージ・フィールドは、固定長フィールドとして定義され、その位置は固定されます。これらのフィールドのデータは、固定長フィールドとして渡すことも、アプリケーション・プログラムで短くすることもできます。データを短くするには、以下の 2 通りの方法があります。

- セグメントの終わりのいくつかのフィールドにデータがない場合に短セグメントを挿入する。
- フィールドにヌル文字 (X'3F') を入れる。MFS は、セグメント・データを左から右にスキャンしてヌル文字を探します。最初にヌル文字を検出したところで、対応する MFLD のデータが終わりになります。セグメント内のすべてのフィールドは、ヌル文字に関係なく、フィールドを定義したときの位置のままです。

末尾ブランクの圧縮:

セグメントの末尾にあるブランクは、次の条件のすべてが当てはまる場合に圧縮されます。

- DIV または DPAGE ステートメントに OFTAB= を指定したか、あるいは FILL=NULL または FILL=PT を指定した。
- セグメントに GRAPHIC=YES を指定した。
- MSG ステートメントで OPT=1 または OPT=2 を指定した。

COMPR の指定

末尾のブランクの圧縮 (COMPR=) には、FIXED、SHORT、または ALL を指定することができます。

FIXED

COMPR=FIXED を指定すると、MFS は固定長データ・フィールドから末尾のブランクを除去します。この結果、DFLD へのマッピングは、以下の場合と同じになります。つまり、アプリケーション・プログラムが短データ・フィールドを挿入した場合 (有効データのあとに X'3F' を挿入するか、または短セグメントを挿入することによって)、またはフィールド全体がブランクであるためそのフィールドを省略した場合 (フィールドの最初の位置に X'3F' を挿入するか、短セグメントを挿入することによって) です。

アプリケーション・プログラムによって短縮されたフィールドでは、COMPR=FIXED を指定した場合のような圧縮は行われません。このオプションは、アプリケーション・プログラムを単純化するために常に最大長のフィールド (例えば NAME フィールド) を渡すために用意されており、これらのブランクは受信装置側にとって意味がありません。受信装置側では、この圧縮オプションまたはアプリケーション・プログラムによって短縮または省略されたフィールドは、同じ意味を持つものと見なします。

SHORT

COMPR=SHORT を指定すると、アプリケーション・プログラムが短縮したデータ・フィールドの末尾のブランクは、MFS によって除去されます。この結果、DFLD へのマッピングは、アプリケーション・プログラムが末尾にブランクのない短フィールドを挿入したか、またはフィールドを省略したかのようになります。固定長フィールドでは、この圧縮が行われません。

このオプションは、アプリケーション・プログラムで変更を行わないような 3270 用のアプリケーション・プログラムのために用意されています。

ALL

COMPR=ALL を指定した場合、固定長フィールドと短フィールドの末尾のブランクは除去されます。

短フィールドの末尾のブランクまたはブランクが 1 つの短フィールドがあると、3270 で特定の操作が実行されます (つまり、画面上のブランクが 1 つのフィールド全体を消去し、プログラム・タブ文字 (FILL=PT) を挿入するか、または更新済みのフィールドの残りの部分を消去して、1 つ以上のヌル文字 (FILL=NULL) を挿入します)。

回線伝送時間の節減

以下の方法のいずれかを使用すると、回線伝送時間が節約されます。

- COMPR=ALL を指定し、固定長フィールドと短フィールド内の末尾のブランクを除去する。
- レコード・モードを定義し、レコードの終わりに発生するようにフィールドを定義する。

可変長出力時のブランクの圧縮

以下のコード例は、IMS アプリケーションから入力されるデータを示しています。

セグメント 1 :

```
DLZZ  FIELD A1   | FIELD A2 |FIELD A3 |FIELD A4 |FIELDC1|FIELD C2
0200  AAAAA44444|1234563...|43.......|A4A4A4
0800       00000|      F   |0F
```

セグメント 2 :

```
DLZZ  FIELD B1   | FIELD B2 |FIELD D1 |FIELD D2 |FIELD D3|FIELD E1
0300  BBBBBBBBBB|4444444444|DDDDDD43.|3........|D3D3D3D3
0400             |0000000000|      0F |F        |
```

注**:** 入力される両セグメントは、プログラムによって短縮が行われます。

以下の表は、上記のコード例で使用した MFS 定義を示しています。

表 *143. IMS* アプリケーションから入力されるデータの *MFS* 定義

MSGOUT	**MSG**	**TYPE=OUTPUT、SOR=FMTOUT**
	SEG	
	MFLD	A1,LTH=10
	MFLD	A2,LTH=10
	MFLD	A3,LTH=10
	MFLD	A4,LTH=10
	MFLD	C1,LTH=10
	MFLD	C2,LTH=10
	SEG	
	MFLD	B1,LTH=10
	MFLD	B2,LTH=10
	MFLD	D1,LTH=10
	MFLD	D2,LTH=10
	MFLD	D3,LTH=10
	MFLD	E1,LTH=10
	MSGEND	
FMTOUT	FMT	

以下のコード例は、ブランクが圧縮される可変長出力の例を示しています。

以下のコード例は、レコード・モードでブランクが圧縮される可変長出力を示しています。

```
VLVB    FIELD A1  THRU A4: (First record)
01      AAAAA,123456,,A4A4A4
06
VLVB    FIELD B1:          (Second record)
00      BBBBBBBBBB
0C
VLVB    NO DATA:           (Third record)
00
03
VLVB    FIELDS D1 and D3:  (Fourth record)
01        DDDDDD,,D3D3D3D3
02
```

注:

1. フィールド A2 は短縮されている。
2. フィールド A3 にはデータがない。
3. フィールド A4 は短縮されている。レコード内の後書き分離文字は送信されません。
4. フィールド B2 にはデータがない。
5. フィールド C1 と C2 にはデータがない。ただし、そのあとにデータが続いているので、1 バイト・レコードが送信されます。
6. フィールド D1 は短縮されている。
7. フィールド D2 にはデータがない。
8. フィールド E1 にはデータがない。そのあとにもうデータが続いていないので、レコードは送信されません。

以下の表は、上記のコード例で示されているとおり、レコード・モード出力で使用する MFS 定義を示しています。

表 *144.* レコード・モードの場合の *MFS* 定義

フィールド	タイプ	定義
	DEV	TYPE=DPM-B1, FEAT=5, MODE=RECORD
	DIV	TYPE=OUTPUT, X OFTAB=(c',',MIX), COMPR=ALL
A1	DFLD	LTH=10
A2	DFLD	LTH=10
A3	DFLD	LTH=10
A4	DFLD	LTH=10
	RCD	
B1	DFLD	LTH=10
B2	DFLD	LTH=10
	RCD	
C1	DFLD	LTH=10
C2	DFLD	LTH=10
	RCD	
D1	DFLD	LTH=10
D2	DFLD	LTH=10
D3	DFLD	LTH=10
	RCD	
E1	DFLD	LTH=10

表 *144.* レコード・モードの場合の *MFS* 定義 (続き)

フィールド	タイプ	定義

以下のコード例は、ストリーム・モードでブランクが圧縮される可変長出力を示しています。

```
VLVB    FIELDS A1 THROUGH D3:  (Single record)
03      AAAAA,123456,,A4A4A4,BBBBBBBBBB,,,DDDDDD,,D3D3D3D3
```

注**:** ストリーム・モードでは、短縮されたフィールド D3 および省略されたフィールド E1 には、分離文字は送信されません。

以下の表は、上記のコード例で示されているとおり、ストリーム・モード出力で使用する MFS 定義を示しています。

表 *145.* ストリーム・モードの場合の *MFS* 定義

フィールド	タイプ	定義
	DEV	TYPE=DPM-B1, FEAT=6, MODE=STREAM
	DIV	TYPE=OUTPUT, X OFTAB=(c',',MIX), COMPR=ALL
A1	DFLD	LTH=10
A2	DFLD	LTH=10
A3	DFLD	LTH=10
A4	DFLD	LTH=10
B1	DFLD	LTH=10
B2	DFLD	LTH=10
C1	DFLD	LTH=10
C2	DFLD	LTH=10
D1	DFLD	LTH=10
D2	DFLD	LTH=10
D3	DFLD	LTH=10
E1	DFLD	LTH=10
	FMTEND	

関連資料:

524 ページの『DIV ステートメント』

データ構造名**:**

DIV ステートメントで OPTIONS=NODNM をコーディングしないかぎり、データ構造名は別の DD ヘッダーに入れられて送信されます。OPTIONS=DNM をコーディングするか、デフォルトが使用されると、出力メッセージのそれぞれの伝送チェーン内、または要求時ページ出力メッセージのそれぞれの伝送チェーン内に、DD ヘッダーが入ります。

出力メッセージの唯一の伝送チェーン内、またはページ出力メッセージの最初の伝送チェーン内には、DD ヘッダーのデータ構造名パラメーターのほかに、バージョン ID パラメーターも入っています。

バージョン **ID**

DEV ステートメントには、DOF または DIF 制御ブロックにバージョン ID として保管される 2 バイトの値を指定するオプションがあります。このパラメーターがコーディングされていない場合は、MFS は、日付と時刻に基づくハッシュ・アルゴリズム手法を用いてバージョン ID を作成します。この値は、リモート・プログラムの形式定義の際にユーザーが参照できるように、MFS 言語ユーティリティーの出力としても印刷されます。

オペレーターによる MFS の制御

IMS は、ユーザーを支援し、リモート・プログラムで出力メッセージの表示や送信を制御できる MFS 機能を提供します。

関連概念:

570 ページの『出力メッセージのオペレーター論理ページング』

オペレーター論理ページング**:**

オペレーター論理ページングによって、ユーザー (あるいは、SLU P ならばリモート・プログラム、または ISC サブシステム) は、出力メッセージの特定の論理ページを要求することができます。この機能を使用するには、MOD の MSG ステートメントで、個々のメッセージごとに PAGE= オペランドを定義します。

提供される機能

MOD でオペレーター論理ページングが定義されていると、いったん出力メッセージの最初の物理ページを表示してから、次の機能を使用することができます。

- = を入力すると、現行メッセージにおける次の論理ページが表示される。
- *=n*、*=nn*、*=nnn*、または *=nnnn* (*n* は論理ページ番号) を入力すると、現行メッセージの特定の論理ページを表示できる。 *nnnn* の最大値は 4095 です。
- *=+n*、*=+nn*、または *=+nnn* を入力すると、現行論理ページから *n* ページ後の論理ページを表示できる。*nnn* の最大値は 999 です。
- *=-n*、*=-nn*、*=-nnn*、または *=>nnn* を入力すると、現行論理ページから *n* ページ前の論理ページを表示できる。*nnn* の最大値は 999 です。
- =L を入力すると、現行メッセージの最後の論理ページにある最初の物理ページが表示される。

形式設計の考慮事項

オペレーター論理ページングが許可されている場合は、現在表示されているページにページ要求を入力し、最初の入力セグメントの最初のフィールドで編集できるように、メッセージおよび装置の形式を設計する必要があります。このような設計が行われていない場合、あるいは PAGEREQ 機能を使用していない場合、ページング要求を入力できるのは、装置が消去された後だけです。

装置形式のインストール基準には、論理ページ要求、トランザクション・コード、および IMS コマンドを入力するための、特定の装置フィールドを含める必要があります。このトランザクション・コードがメッセージや PF キー・リテラルを介して正常に提供されると、PAGEREQ 機能を利用したり、最初のセグメントの先頭に、ヌル・パッド文字を使用してフィールドを定義することができます。装置上のページ要求フィールドを、このフィールドにマップすることもできます。ページ要求を入力しないと、フィールドにはヌル文字が埋め込まれ、セグメントから除去されるため、2 番目のフィールド (リテラル・トランザクション・コード) がセグメントの先頭に表示されます。

トランザクション・コードおよび論理ページ要求

PAGEREQ 機能を使用せずにページ要求を指定すると、MFS はまず定義されている MID に応じて入力データをフォーマット設定します。その後で、オペレーター論理ページングが指定されているかどうか、そして入力にページ要求が入っているかどうかを調べます。オペレーター論理ページングが指定されていない場合、このメッセージは標準 IMS 宛先決定処理を受けることになります。

オペレーター論理ページングが指定されている場合、MFS は最初のメッセージ・セグメントの最初のデータ (このメッセージのフォーマット設定がオプション 3 の場合は最初のフィールド) を検査して、等号 (=) を探します。MFS は、等号が見つからない場合、メッセージをその宛先へと経路指定します。等号が見つかった場合、最大 4 文字までの後続文字、または最初のブランクに出会うまでの後続文字は、すべてページ要求と見なされます。

単一セグメント・コマンドまたはトランザクションを宛先にしたメッセージは、高速機能アプリケーションで要求されるときと同様に、MID の単一セグメントとして定義する必要があります。 MID で複数のセグメントを定義する場合、その宛先が単一セグメントのコマンドまたはトランザクションであるときは、必ずセグメントを 1 つだけ作成するようにします。そのためには、十分注意して入力し、オプション 2 かヌルの圧縮 (FILL=NULL)、あるいはこの両方を使用してください。

オペレーター制御テーブル**:**

データあるいはそのデータ長のいずれかが、前もって定義されていた条件を満たしていれば、入力装置フィールドを定義して、MFS 制御機能を呼び出すことができます。これを行うには、1 つ以上のオペレーター制御テーブルを定義し、装置フィールド定義に関連テーブル名を指定します。

装置フィールドがオペレーター制御テーブルと関連付けて定義されている場合、入力データがオペレーター制御テーブルの条件を満たしていれば、MFS は装置入力フィールドを処理し、要求された制御機能を実行します。

オペレーター制御テーブルを使用すると、次の制御機能を利用することができます。

NEXTPP
現行メッセージの次の物理ページを提供します。

NEXTLP
現行メッセージの次の論理ページを提供します。

PAGEREQ

このフィールドの 2 番目の文字から最後の文字までが要求した論理ページを提供します。PAGEREQ 機能の指定方法は、オペレーター論理ページングの場合と同じです。先頭の文字は、ユーザーが定義したページ要求を表す『トリガー』文字です。 残りの文字は、*n*[*nnn*]、+*n*[*nn*]、-*n*[*nn*]、 または L (等号 (=) は使用できない) でなければなりません。

NEXTMSG

現行出力メッセージをデキューし、次のメッセージがあれば、そのメッセージの最初の物理ページを提供します。

NEXTMSGP

現行出力メッセージをデキューし、次のメッセージがあれば、そのメッセージの最初の物理ページを提供します。あるいは、キューにその他のメッセージが入っていないことをユーザーに通知します。

ENDMPPI

複数の物理ページからなる入力メッセージを終了します。これは、3270 でのみ使用可能です。

オペレーター論理ページング要求とは異なり、編集プロセス中でも MFS は常にこれらの機能を見つけます。

3270 または **SLU 2** 専用機能の定義**:**

SLU 2 または 3270 を使用する場合、次のようないくつかの方法で MFS 制御機能を呼び出すことができます。

- PAGEREQ を除くあらゆる MFS 制御機能に対して、PF キー、および選択ライト・ペンで検出可能な表示装置フィールドを定義することができます。
- PA1 キーは NEXTPP 機能に相当し、この機能のために予約されています。
- PA2 キーは NEXTMSG 機能に相当します。
- コピー機能として使用しない場合、PA3 キーは NEXTMSGP 機能に相当します。
- PF12 キー、またはデータ入力キーボード装置上の PA3 キーを押すと、コピー機能を呼び出すことができます。 IMS がサポートするこのコピー機能を使用すると、現在表示されている物理ページのコピーを、使用可能な候補プリンターで印刷することができます。このプリンターは、コピーする情報の入ったディスプレイ装置と同じ制御装置 (3271 または 3274 など) に接続されていなければなりません。

  制約事項**:** 装置でコピー機能が使用できるように定義されていない場合、または装置がコピー機能をサポートしていない場合は、コピー機能を要求しても無視されます。

  コピー機能の詳細については、ALPHA/NUM および NOPROT/PROT の DFLD ステートメントのフィールド定義を参照してください。

装置におけるページング操作:

MFS 装置のページング操作は、MFS 制御ブロック定義の内容、出力メッセージの内容、およびユーザーの入力によって異なります。装置がプリンターであれば、各論理ページの物理ページはそれぞれ順序どおりに装置に送信され、メッセージがデキューされます。

出力のページング中に、アクセスした出力メッセージの形式を変更するようなオンライン変更処理が発生した場合、エラー・メッセージが表示されたり、メッセージが予想外の形で表示されることがあります。

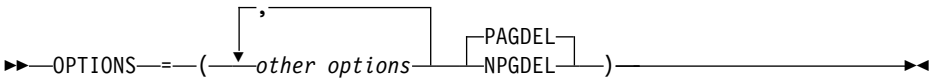
3604、3270、SLU 2 表示装置、または DPM ページング・オプションを使用する SLU P でオペレーター論理ページングを指定しない場合、NEXTPP 機能を使用して論理ページごとの物理ページを順に表示することができます。オペレーター論理ページングを指定していないので、最終論理ページの最後の物理ページが表示された後で NEXTPP を入力すると、次のメッセージが送信されます (キューの中にメッセージが 1 つしかない場合)。キューにメッセージがない場合は、なにも実行されません。

3604、3270、SLU 2 表示装置、または DPM ページング・オプションを使用する SLU P に対してオペレーター論理ページングが指定されている場合、NEXTPP 機能を使用して継続的にページを表示させることができます。ただし、最終論理ページの最後の物理ページのあとで NEXTPP を入力すると、MFS がエラー・メッセージを返し、ページ位置が最初のページに戻ります。順不同でページを表示したい場合は、PAGEREQ 機能を使用できるような形式、あるいはページ要求が最初の入力セグメントの最初のフィールドに編集されるような形式を設計する必要があります。このような設計でない場合は、画面を消去してから、ページ要求を形式が設定されていない入力として入力する必要があります。パフォーマンス上の理由から、この方法はなるべく避けてください。

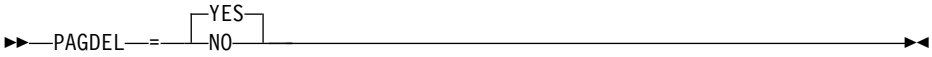
以下の表は、IMS が行うアクションと、正常なメッセージ送信完了後のユーザーによる入力やリモート・プログラムによるアクションの結果生じるメッセージおよび装置状況について説明しています。

図には次の考慮すべき要素が含まれています。

- マクロ/ステートメントの指定 :
  1. TERMINAL (または TYPE) マクロ (IMS システム定義)



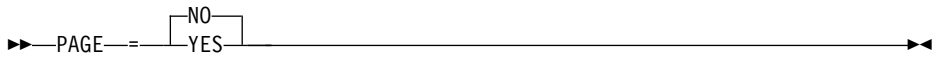
または



デフォルト (PAGEDEL=YES) を使用した場合、新規トランザクションの処理を呼び出す入力を行うと、現行トランザクションの出力メッセージがデキ

ューされます。現行の出力がデキューされるのを防ぐためには、OPTIONS=(...,NPGDEL,...) または PAGDEL=NO を非交換 3270 装置に指定する必要があります。

2. MSG ステートメント (MOD 定義)



PAGE=YES は、オペレーター論理ページングが許可されていることを表します。 PAGE=NO は、ページングが許可されていないことを表します。

- 現行メッセージにおける最終論理ページの最終物理ページが送信されたかどうか。
- メッセージが正常に伝送されてからユーザーが入力を行うまでに、自動的に実行される IMS のアクション。
- メッセージを受け取った後のユーザーの入力、またはリモート・プログラムが取るアクション。
  - PAGE ADVANCE : NEXTPP 要求を入力します (あるいは 3270 または SLU 2 で PA1 キーを押します)。
  - LOGICAL PAGE ADVANCE : NEXTLP 要求を入力します。
  - =PAGE : 特定の論理ページを要求します。
  - PAGEREQ : 特定の論理ページを要求します。
  - MESSAGE ADVANCE : NEXTMSG 要求を入力します (あるいは 3270 または SLU 2 で PA2 キーを押します)。
  - MESSAGE ADVANCE PROTECT : NEXTMSGP 要求を入力します (あるいは PA3 にコピー機能が定義されていない場合は、3270 または SLU 2 で PA3 キーを押します)。
  - ユーザーまたはリモート・プログラムがオペレーター制御機能の呼び出しを行わないデータを入力して Enter キー (あるいは 3270 または SLU 2 の PFK、CARD、IMMEDIATE DETECT) を押します。

    3270 または SLU 2 のオペレーターは、CLEAR キーを押すこともできます。この CLEAR キーを使用すると画面は無保護になり、これ以降の入力は IMS 基本編集によって編集されます。CLEAR が現行の出力メッセージの状況に対して、なんらかの影響を与えることはありません。 CLEAR を使用してからオペレーターがなんらかのアクションを行っても、その結果は CLEAR を使用しなかった場合と同じです。
- 以下の表では、IMS が行うアクションを次の略語を使用して説明しています。

MSG DEQ
: メッセージのデキュー。IMS は、現行の出力メッセージをメッセージ・キューから除去します。このアクションが実行されるまでは、そのメッセージを使用することができます。

MSG ENQ
: メッセージのエンキュー。IMS は、入力メッセージをメッセージ・キューに入れます。

PROTECT
IMS は、装置が IMS からの出力を受け取らないようにします。

UNPROTECT
IMS は、装置が IMS からの出力を受け取れるようにします。この装置あてのメッセージがキュー内に待機中である場合、IMS はそのメッセージを送信します (応答モード、会話型、または排他的装置状況によって確立される制御手順に従います)。

ページングされたメッセージが、システム定義時または DSCA か SCA 指定を使用して、無保護画面オプションを「無保護」に設定した端末に送られる場合、ページ間では画面が保護されず、以下の表に示す IMS のアクションは無視されます。メッセージが、無保護画面オプションを「保護」に設定した端末に送られる場合は、以下の表に示す IMS のアクションが適用されます。

以下の表では、以下の 4 つのケースで適用される現行メッセージのシステム定義値とメッセージ定義値、およびページ位置を仮定しています。

ケース **1**
PAGE=NO であり、現行メッセージの最終論理ページの最終物理ページが送信された。

ケース **2**
PAGE=NO であり、現行メッセージの最終論理ページの最終物理ページは送信されなかった。

ケース **3**
PAGE=YES であり、現行メッセージの最終論理ページの最終物理ページが送信された。

ケース **4**
PAGE=YES であり、現行メッセージの最終論理ページの最終物理ページは送信されなかった。

注**:** 最後のページにエラー・メッセージが送信された場合は、以下の表は適用されません。元のメッセージはまだキュー内に残っています。メッセージに対する正しい応答については、「*IMS V14* システム・ユーティリティー」を参照してください。

以下の表では、これらの 4 つのケースのそれぞれで、IMS がメッセージを正常に伝送し端末がそのメッセージを受信すると、IMS アクションが PROTECT になります。つまり、IMS は、装置が、IMS からの出力を受け取らないようにします。ケース 1 の場合、IMS は、さらに、IMS メッセージ・キューからそのメッセージをデキューします。

表 *146. PAGDEL* を指定して *MFS* を使用する装置でのページング操作： IMS-MFS が行うアクションおよびその結果として生じる端末状況とメッセージ状況

オペレーター・アクション	ケース **1** の **IMS** アクション	ケース **2** の **IMS** アクション	ケース **3** の **IMS** アクション	ケース **4** の **IMS** アクション
要求 PAGE ADVANCE (NEXTPP)	無保護	次の物理ページを送信、無保護	エラー・メッセージを送信、保護 [1]	次の物理ページを送信、保護

表 *146. PAGDEL* を指定して *MFS* を使用する装置でのページング操作 *(*続き*)*: IMS-MFS が行うアクションおよびその結果として生じる端末状況とメッセージ状況

オペレーター・アクション	ケース **1** の **IMS** アクション	ケース **2** の **IMS** アクション	ケース **3** の **IMS** アクション	ケース **4** の **IMS** アクション
要求 LOGICAL PAGE ADVANCE (NEXTLP)	無保護	現行メッセージにおける次の論理ページの最初の物理ページを送信 [2]	エラー・メッセージを送信、保護 [1]	現行メッセージにおける次の論理ページの最初の物理ページを送信 [2]
=PAGE を使用して特定の論理ページを要求	エラー・メッセージを送信、保護 [3]	MSG DEQ、エラー・メッセージを送信、保護 [3]	有効であれば、要求された論理ページの最初の物理ページを送信、保護 [1] 無効であれば、エラー・メッセージを送信、保護 [1]	
PAGEREQ を使用して特定の論理ページを要求	エラー・メッセージを送信、保護	エラー・メッセージを送信、保護 [1]	有効であれば、要求された論理ページの最初の物理ページを送信、保護 [1] 無効であれば、エラー・メッセージを送信、保護 [1]	
要求 MESSAGE ADVANCE (NEXTMSG)	無保護	MSG DEQ、無保護	MSG DEQ、無保護	MSG DEQ、無保護
要求 MESSAGE ADVANCE PROTECT (NEXTMSGP)	保護 [4]	MSG DEQ、保護 [4]	MSG DEQ、保護 [4]	MSG DEQ、保護 [4]
データを入力	MSG ENQ、無保護	MSG DEQ、MSG ENQ、無保護	MSG DEQ、MSG ENQ、無保護	MSG DEQ、MSG ENQ、無保護

注:

1. 元のメッセージはまだキュー内に残っています。メッセージに対する正しい応答については、「*IMS V14* システム・ユーティリティー」を参照してください。
2. 現行ページが最後の論理ページである場合、新しいページは送信されず、装置状況も無保護になります。
3. 装置が事前に設定されているか、あるいは会話中である場合、入力はキューに入れられます。エラー・メッセージは送信されず、装置状況も無保護になります。
4. メッセージがキュー内に残っている場合、排他状況もしくは会話中でメッセージを送信できない状態でないかぎり、そのメッセージは送信されます。送信するメッセージがない場合は、出力のないことを知らせるシステム・メッセージが送信されます。

以下の表では、ケース 2、3、および 4 で、IMS がメッセージを正常に伝送し端末がそのメッセージを受信すると、IMS アクションが PROTECT になります。つまり、IMS は、装置が、IMS からの出力を受け取らないようにします。ケース 1 の場合は、IMS は、IMS メッセージ・キューからそのメッセージをデキューします。

表 *147. NPAGDEL* を指定して *MFS* を使用する装置でのページング操作： IMS-MFS が行うアクションおよびその結果として生じる端末状況とメッセージ状況

オペレーター・アクション	ケース **1** の **IMS** アクション	ケース **2** の **IMS** アクション	ケース **3** の **IMS** アクション	ケース **4** の **IMS** アクション
要求 PAGE ADVANCE (NEXTPP)	無保護	次の物理ページを送信、保護	エラー・メッセージを送信、保護 [1]	次の物理ページを送信、保護
要求 LOGICAL PAGE ADVANCE (NEXTLP)	無保護	現行メッセージにおける次の論理ページの最初の物理ページを送信 [2]	エラー・メッセージを送信、保護 [1]	現行メッセージにおける次の論理ページの最初の物理ページを送信 [2]
=PAGE を使用して特定の論理ページを要求	エラー・メッセージを送信、保護 [3]	エラー・メッセージを送信、保護 [1、3]	有効であれば、要求された論理ページの最初の物理ページを送信、保護 無効であれば、エラー・メッセージを送信、保護 [1]	
PAGEREQ を使用して特定の論理ページを要求	エラー・メッセージを送信、保護	エラー・メッセージを送信、保護 [1]	有効であれば、要求された論理ページの最初の物理ページを送信、保護 無効であれば、エラー・メッセージを送信、保護 [1]	
要求 MESSAGE ADVANCE (NEXTMSG)	無保護	MSG DEQ、無保護	MSG DEQ、無保護	MSG DEQ、無保護
要求 MESSAGE ADVANCE PROTECT (NEXTMSGP)	保護 [4]	MSG DEQ、無保護	MSG DEQ、保護 [4]	MSG DEQ、保護 [4]
データを入力	MSG ENQ、無保護	MSG ENQ [5]	MSG ENQ [5]	MSG ENQ [5]

注:

1. 元のメッセージはまだキュー内に残っています。メッセージに対する正しい応答については、「*IMS V14* システム・ユーティリティー」を参照してください。
2. 現行ページが最後の論理ページである場合、新しいページは送信されず、装置状況も無保護になります。
3. 装置が事前に設定されているか、あるいは会話中である場合、入力はキューに入れられます。エラー・メッセージは送信されず、装置状況も無保護になります。
4. メッセージがキュー内に残っている場合、排他状況もしくは会話中でメッセージを送信できない状態でないかぎり、そのメッセージは送信されます。送信するメッセージがない場合は、出力のないことを知らせるシステム・メッセージが送信されます。
5. 元のメッセージはまだキュー内に残っています。装置で現在会話が進行中でないかぎり、最初の論理ページ内の最初の物理ページが送信されます。会話中であれば、エラー・メッセージが送信されます。会話の応答の後で引き続き処理を行うときは、NEXTMSG を入力してその応答をデキューする必要があります。

関連概念:

570 ページの『出力メッセージのオペレーター論理ページング』

『無保護画面オプション』

無保護画面オプション**:**

MFS によってフォーマット設定されたメッセージが、出力メッセージとして 3270 表示装置に送信されるときには、IMS によって画面を無保護状態のままにしておくことができます。このオプションは、端末単位で指定することも、MFS をバイパスするメッセージを除いてメッセージ単位で指定することも可能です。

無保護状況の端末オプションは、次のものに適用されます。

- MFS をバイパスするすべてのユーザー出力メッセージ
- IMS が生成したすべてのメッセージ (例えば、エラー、/BROADCAST, および /DISPLAY コマンド出力)
- MFS によって IMS 提供のデフォルト形式の 1 つ、あるいはユーザー指定の形式にフォーマット設定されたすべてのメッセージ

無保護画面オプションを選択しない場合、MFS がユーザー指定の形式または IMS 提供のデフォルト形式を使用してフォーマット設定したメッセージ、および IMS が生成したメッセージの画面表示の保護または無保護の状態は、メッセージ単位で異なります。

メッセージがページ化される場合、ページングする前の画面は保護されません。したがって、ページ化メッセージには、このオプションを使用しないでください。

このオプションは、次の項目のいずれかで指定します。

- MFLD ステートメントの SCA 出力メッセージ・オプション
- システム定義の TERMINAL マクロ指定
- DEV ステートメントでの DSCA 指定

DEV ステートメントの DSCA= オペランドおよび MFLD ステートメントの SCA 出力メッセージ・オプションのバイト 1、ビット 5 を使用して、3270 表示装置へのメッセージ送信時に画面の保護または無保護を定義します。

B'0' 出力の送信時に画面を保護します。デフォルトは B'0' (保護) です。このビットは ISC での自動ページ化出力に使用します。

B'1' 出力の送信時に画面が保護されません。

DSCA 値が B'0' に設定されており、PROT (保護) が TERMINAL または TYPE マクロでデフォルトとして指定または使用されている場合、アプリケーション・プログラムは、この出力の送信時に (SCA 値を B'1' に設定することで) 画面を無保護にするよう要求することができます。メッセージでオペレーター論理ページング (OLP) が使用されている (PAGE=YES が対応する MSG 定義内に指定されている) ときに、画面を無保護状態にするよう要求すると、OLP はリセットされます。IMS 提供のデフォルト形式を変更して、DSCA 値に B'1' を設定することもできます。

MFS をバイパスするユーザー・メッセージが画面を保護するかしないかは、システム定義の際の TERMINAL または TYPE マクロ上の OPTIONS 指定によって決まります。デフォルトは保護です。

MFS が、ユーザー定義形式の SYSMSG フィールドに送信される IMS メッセージをフォーマット設定する場合、画面が保護されるかしないかは、その装置に関する形式の中の DSCA または SCA オプションによって決まります。

ディスプレイが無保護状態であれば、IMS はいつでも端末装置に出力を送信することができます。IMS 出力の直前に Enter キー、PA キー、または PF キーを押すと、入力や要求が失われることがあります。これを避けるためには、入出力には MFS を使用し、NEXTMSGP 機能を入力するか、あるいは (PA3 キーがコピー機能として使用されていなければ) PA3 キーを押して、入力データの入力前に装置を保護状態にしておきます。

MFS を使用していないか、あるいは出力でのみ使用している場合でも、MOD 名に DFS.EDT を指定すれば、PA3 キーを押すと入力データが保護されます。この場合、PA3 キーは、コピー機能として使用しないでください。

以下の表は、システム定義時の TERMINAL または TYPE マクロの OPTIONS 指定、および送信される出力メッセージのタイプに基づいて IMS が行うアクション (保護または無保護) を示したものです。

表 *148. OPTIONS* 指定に基づいた *IMS* の保護アクションまたは無保護アクション：

出力メッセージ	**IMS** システム定義 **(PRO)**	**IMS** システム定義 **(UNPRO)**
IMS 生成メッセージ： DSCA|SCA=PROTECT	PROTECT	UNPROTECT
IMS 生成メッセージ： DSCA|SCA=UNPROTECT	UNPROTECT	UNPROTECT
MFS バイパス使用メッセージ	PROTECT	UNPROTECT
MFS およびユーザー指定形式や IMS 提供のデフォルト形式を使用するユーザー・メッセージ： DSCA|SCA=PROTECT	PROTECT	UNPROTECT
MFS およびユーザー指定形式や IMS 提供のデフォルト形式を使用するユーザー・メッセージ： DSCA|SCA=UNPROTECT	UNPROTECT	UNPROTECT

注:

1. PROTECT: 追加の出力を送信せずに入力を待ちます。
2. UNPROTECT : 出力メッセージが使用可能で、送信に適格であれば、出力を送信します。

関連概念:

605 ページの『装置におけるページング操作』

区画形式モードの **3290:**

IMS に SLU 2 端末として定義された装置では、3290 の区画設定およびスクロール機能がサポートされます。区分化およびスクロールの機能は、非 SNA VTAM を使用する装置には提供されません。

区画セット初期設定およびページング

区画セット初期設定およびページングを行う際に、3 種類のオプションの中から 1 つを選択することができます。このオプションを選択することで、メッセージを区画形式画面に初めて伝送するときに、出力メッセージの論理ページを該当する区画にいくつ振り分けるのかを決めることができます。(出力メッセージは 1 つ以上の論理ページから構成されており、それぞれの論理ページが DPAGE によって決められた特定の区画に振り分けられます。) ページング要求が出されたときに、それ以降の論理ページを該当する区画にどのように振り分けるのかも、このオプションによって決まります。 このオプションは、区画記述子ブロック (PDB) ステートメントの PAGINGOP= オペランドで指定してください。

次に 3 種類のオプションを示します。

オプション **1**

3290 LU に表示される初期データ・ストリームは、出力メッセージ内の最初の論理ページで構成されており、それは、DPAGE を介して該当区画にマップされます。それ以降は、オペレーターがページング要求を入力することによって、すべてのページングを制御します。 PA1 キーおよび PA2 キーは、標準の非区画モードのときと同じように使用します。端末では、基本ページング・サポートや OLP (オペレーター論理ページング) を使用することができます。

次の論理ページを要求すると、MFS は次の順番の論理ページを取り出し、それを関連区画に送信します。どの区画がアクティブであるかは関係ありません。次のページを要求すると、そのメッセージの次の順番のページが、入力の (アクティブ) 区画または別の区画へ送られる結果となります。

例えば、=+1 と入力すると、メッセージ内の論理ページは該当区画に送信されます。どのような区画でもかまいません。 =+3 を入力すると、最後に送信された論理ページから順に数えて 3 番目のページが次に送信されます。

オプション **2**

3290 LU に渡された初期データ・ストリームは、メッセージの最初の論理ページとそのあとに順に続く追加の論理ページとで構成されます。メッセージは、いずれかの区画で 2 番目の論理ページに達するまで、またはそのメッセージが終了するまで追加されます。それ以降は、オプション 1 の場合と同様に、ページング要求を入力してすべてのページングを制御します。

オプション **3**

3290 LU に渡された初期データ・ストリームは、区画セットの各区画の最初の論理ページで構成されています。それ以降は、ページング要求を入力することですべてのページングを制御しますが、オプション 1 と 2 とでは 1 つ決定的な違いがあります。それは、どのアクティブ区画から要求を入力したかによって、それ以降の論理ページが区画に送られてくる際の順番が決まるということです。論理ページに対するすべての要求は、そのアクティブ区画と関連付けられている論理ページだけに適用されます。

例えば、=+1 を入力した場合、そのアクティブ区画あてのメッセージの中から次の論理ページが渡されます。つまり、メッセージ内でたまたま次の順番にある論理ページが必ずしも渡されるというわけではありません。このオプ

ションによって、3290 オペレーターは、非区画環境でのページング・サポートとまったく同様に、アクティブ区画でも論理ページングの管理を行えます。

どのオプションを選択しても、初期データ・ストリームが送信されると、1 つの区画がアクティブになります。アクティブ区画とは、カーソルの置かれている区画のことです。

DPAGE ステートメントの 1 つに ACTVPID オペランドを指定して、初期設定区画を指示しておくことができます。アプリケーション・プログラムは、ACTVPID を使用してどの区画がアクティブ区画であるかを宣言することができます。オプション 2 または 3 の使用中にデータをいくつかの区画に送信した場合、複数の区画に ACTVPID キーワードが指定されている可能性があります。その場合、最後にアクティブにされた区画がアクティブ区画になります。ACTVPID キーワードが指定されていなければ、PDB の最初の区画記述子 (PD) ステートメントで定義された区画が、アクティブ区画になります。

表示の消去

画面およびバッファーを消去する方法として、次の 2 つのレベルがあります。

- CLEAR キー (X'6D') は、3290 をリセットして基本状況 (区画未設定モード) にし、バッファーをヌルに設定して、カーソルを画面の左上隅に置きます。また、アクティブ・メッセージをキューに戻して、区画化のために作成された制御ブロックの構造を削除します。
- CLEAR PARTITION キー (X'6A') は、アクティブ区画のバッファーのみをヌルにリセットし、アクティブ区画のビューポートを消去します。さらに、カーソルを区画の左上隅に置きます。その区画は、フォーマット設定されていないものと見なされます。つまり、この区画からの入力は MFS によって不定様式と見なされ、基本編集で処理されます。

区画移動キー

区画移動キーを使用すると、PD ステートメントで定義されている PDB 内の区画の順番どおりに、ある区画から次の区画へと移動することができます。

区画間の移動は、そこに指定されている区画 ID (PID) の値の順番で決まるのではなく、PD ステートメントの順序によって決まることに注意してください。

カーソルの移動先の区画がアクティブ区画になります。このキーを使用してもホストとの対話は行われません。

スクロール操作

縦方向スクロール・キーを使用すると、ビューポートのデータが上下に移動して、表示スペースのさまざまな部分がスクロール・ウィンドウに表示されます。このスクロール・ウィンドウは、その時点でビューポートにマップされた表示スペースの一部です。ビューポートの縦の長さが表示スペースと同じである場合は、このビューポートをスクロールさせることはできません。ビューポートの縦の長さが表示スペースよりも短い場合は、スクロールさせることができます。

1 回にスクロールする量は、PD ステートメントの SCROLLI キーワードで指定した行数によって決まります。デフォルトのスクロール移動量は 1 行です。スクロールを行っても、ホストとの対話は行われません。

関連概念:

482 ページの『3290 画面のフォーマット設定』

『区画形式モードの 3180』

関連資料:

507 ページの『装置依存の出力情報』

区画形式モードの **3180:**

IMS は、3290 の区画設定およびスクロールのサポート機能を通して、区画形式モードの 3180 をサポートします。

3180 との対話と区画形式モードの 3290 とはよく似ていますが、次のような相違点があります。

- 3180 では、特定のサイズに制限された 1 つの区画しか使用できません。 3290 では、さまざまなサイズの複数の区画を使用することができます。
- 3180 では、論理装置の表示画面サイズおよびビューポートの場所を画素 (ピクセル) で指定することができません。3290 では、行、桁、ピクセルでの指定もサポートされています。
- 3180 では、初期設定される区画は 1 つのみです。3290 では、アプリケーション・プログラムで ACTVPID キーワードを使用することによって、さまざまな区画の中からどの区画を初期設定するかを決めることができます。

3180 でアクティブ区画にできる区画は 1 つしかないため、PDB ステートメントの PAGINGOP= オペランドでオプション 1 を指定するか、デフォルトの 1 を使用するようにしてください。このオプションを使用する場合、3180 LU への初期データ・ストリームは、出力メッセージの最初の論理ページのみで構成されており、DPAGE の指定によって 1 つの区画にマップされます。次の論理ページを要求すると、MFS はメッセージ内で次の順番の論理ページを取り出し、それを区画に送信します。

3180 での画面の消去およびスクロールは、区画形式モードの 3290 と同じ方法で処理されます。

関連概念:

611 ページの『区画形式モードの 3290』

IMS 提供の MFS 形式セット

IMS には、システムが使用するためのいくつかの形式セットが用意されており、端末オペレーターが正しい MOD 名を入力しなかった場合には、そのデフォルトとして使用されます。IMS 提供の制御ブロックは、IMS.FORMAT ライブラリーに入っています。MFSTEST 機能を使用している場合、これらの制御ブロックは IMS.TFORMAT ライブラリーにも入っています。これらの制御ブロックは、/FORMAT コマンドのあとで該当する MOD 名を指定すると、MFS がインスト

ールされている IMS システムで使用することができます。さらに、この形式定義を使用したいときは、ユーザーが作成したメッセージ定義の SOR= オペランドで、その形式名を指定します。

IMS 提供の形式定義をさまざまなメッセージ定義と組み合わせて、いくつかの別個のメッセージ形式を作成します。MFS 3270 および SLU 2 マスター端末形式以外のあらゆる形式セットは、DFSDF1、DFSDF2 または DFSDF4 形式定義を使用します。これらの形式定義には、3270 または SLU タイプ 2 PF キーである PFK1 および PFK11 の 2 つのリテラルが含まれています。PFK1 を押すと、最初のメッセージ・セグメントの入力データの前に、/FORMAT コマンドが挿入されます。 PFK11 を押すと、NEXTMSGP 要求が出されます。

システム・メッセージ形式

システム・メッセージ形式は、IMS からの単一セグメントの出力メッセージと単一セグメントのブロードキャスト・メッセージで使用されます。入力セグメント (トランザクション、コマンド、またはメッセージ通信) は 2 つまで可能です。形式名は DFSDF1 です。MOD 名と MID 名は、それぞれ DFSMO1 と DFSMI1 です。この形式を取るメッセージは、3270 または SLU 2 装置の SYSMSG フィールドにも入れることができます。

複数セグメント・システム・メッセージ形式

複数セグメントのシステム・メッセージ形式は、IMS からの複数セグメント・メッセージと複数セグメント・ブロードキャスト・メッセージで使用されます。この形式では、最高 22 個のセグメントからなる出力メッセージを作成することができます。形式名は DFSDF2 です。MOD 名と MID 名は、それぞれ DFSMO5 と DFSMI2 です。この形式を取るメッセージは、3270 または SLU 2 装置の SYSMSG フィールドにも入れることができます。PA1 キーを使用すると、後続セグメントを取り出すことができます。

デフォルトの出力メッセージ形式

3270 装置または SLU 2 装置では、他の端末からのメッセージ通信と MOD 名が指定されていないアプリケーション・プログラムの出力メッセージで、デフォルトの出力メッセージ形式が使用されます。入力セグメント (トランザクション、コマンド、またはメッセージ通信) は 2 つまで可能です。形式名は DFSDF2 です。MOD 名と MID 名は、それぞれ DFSMO2 と DFSMI2 です。

ブロック・エラー・メッセージ形式

ブロック・エラー・メッセージ形式は、出力形式ブロックの選択時にエラーが発生した場合、MFS が送信する「DFS057I REQUESTED BLOCK NOT AVAILABLE」(DFS057I 要求ブロックが使用不能) というメッセージで使用されます。このメッセージには、(エラーの重大度を示す) 戻りコード、およびブロック名 (エラーを起こした MOD または DOF の名前) も入っています。このメッセージには、論理ページあたり最高 21 の出力セグメントを入れることができます。この形式の場合、入力セグメント (トランザクション、コマンド、またはメッセージ通信) は 2 つまで可能です。形式名は DFSDF2 です。MOD 名と MID 名は、それぞれ DFSMO3 と DFSMI2 です。

/DISPLAY コマンド形式

/DISPLAY コマンドの形式は、/DISPLAY コマンドの出力で使用されます。1 論理ページあたり最高 22 までのセグメントを入れることができます。この形式の場合、入力セグメント (トランザクション、コマンド、またはメッセージ通信) は 2 つまで可能です。形式名は DFSDF2 で、MOD 名と MID 名はそれぞれ DFSDSP01 と DFSMI2 です。

複数セグメント形式

複数セグメント形式は、複数セグメント・トランザクションとコマンドを入力するときに使用します。 /FORMAT コマンドで MOD 名として DFSMO4 を指定すれば、この形式を取り出すことができます。この形式は、4 セグメント以下の複数セグメント出力メッセージでも使用することができます。入力セグメントは 4 つまで可能です。形式名は DFSDF4 です。MOD 名と MID 名は、それぞれ DFSMO4 と DFSMI4 です。

MFS 3270 または SLU 2 マスター端末形式

MFS の 3270 または SLU 2 マスター端末形式は、3270 または SLU 2 マスター端末での MFS サポート (IMS 提供のオプション機能) を選択したときに使用されます。

MFS サインオン装置形式

MFS サインオン装置形式は、ユーザー・サインオンを必要とする端末 (拡張端末オプション (ETO) が定義されている端末など) で使用します。(ETO の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。) この形式は、3270 および SLU 2 装置にのみ適用されます。フォーマット設定済みの /SIGN ON コマンド・パネルを受信できる装置 (少なくとも 12 行、40 桁の装置) では、MOD と MID はそれぞれ DFSIGNP と DFSIGNI です。小さな画面の装置では、MOD と MID はそれぞれ DFSIGNN と DFSIGNJ になります。

関連概念:

『3270 または SLU 2 マスター端末の MFS フォーマット設定』

3270 または SLU 2 マスター端末の MFS フォーマット設定

3275、3277-2 型、または SIZE=24×80 の 3270-An と定義されている 3270 または SLU 2 表示装置が、IMS マスター端末になっている場合は、MFS を使用する IMS 提供の形式を選択することができます。IMS 提供の形式を使用するためには、OPTIONS=(...,FMTMAST,...) を、IMS システム定義時に COMM マクロに指定する必要があります。

この形式を使用するときは、表示画面が 4 つの区域に分けられ、そのためにいくつかの PF キーが確保されます。

画面の 4 つの区域とは次のものです。

メッセージ域
: この区域には、IMS コマンド出力 (/DISPLAY および /RDISPLAY の出力は除

く)、メッセージ通信の出力、DFSMO で始まる MOD 名を使用するアプリケーション・プログラム出力、および IMS システム・メッセージが表示されます。

表示域
: この区域には、/DISPLAY および /RDISPLAY コマンド出力が表示されます。

警告メッセージ域
: この区域には、次のような警告メッセージが表示されます。

  MASTER LINES WAITING

  MASTER MESSAGE WAITING

  DISPLAY LINES WAITING

  USER MESSAGE WAITING

  この区域に IMS パスワードを入力することもできます。

ユーザー作業域
: この区域には、オペレーターが入力を行います。

関連資料: これらの画面区域の形式および使用法については、「*IMS V14* システム管理」を参照してください。

IMS 提供のマスター端末形式定義には、3270 または SLU 2 プログラム・ファンクション (PF) キーのうちの 9 個についてリテラルが定義されています。1 から 7 までの PF キーを使用して、IMS コマンドを入力することができます。PF キーを押すと、そのキーに対応するコマンドが、入力したデータの前にある最初のメッセージ・セグメントに挿入されます。7 個のキーと対応するコマンドは、次のようになっています。

PF キー	コマンド
1	/DISPLAY
2	/DISPLAY ACTIVE
3	/DISPLAY STATUS
4	/START LINE
5	/STOP LINE
6	/DISPLAY POOL
7	/BROADCAST LTERM ALL

PF11 キーを押すと NEXTMSGP 要求が出されます。また、PF12 キーを押すとコピー機能要求が出されます。

PFK リテラル以外のマスター端末形式の定義は、変更しないでください。

マスター端末形式を使用するメッセージのうち、MOD 名が DFSMO で始まるメッセージ (DFSMO3 を除く) はすべてメッセージ域に表示されます。 MOD 名が DFSDSPO1 であるすべてのメッセージは表示域に表示されます。他の MOD 名を使用するメッセージがあると、警告メッセージ「USER MESSAGE WAITING」が表示されます。

関連概念:

614 ページの『IMS 提供の MFS 形式セット』

MFS 装置特性テーブル

MFS 装置特性テーブル (DFSUDT0x) は、TYPE または TERMINAL マクロ・ステートメントで 3270 または SLU 2 装置を TYPE=3270-An と定義すると、システム定義時に生成されます。

DFSUDT0 x の '*x*' は、 IMSGEN マクロの SUFFIX= キーワードに指定されたパラメーターと対応します。

MFS 装置特性テーブルは、 MFS 装置特性テーブル・ユーティリティー (DFSUTB00) で更新することができます。このユーティリティーを使用すると、システムを再生成せずにテーブルを更新することができます。このテーブルの各項目には、ユーザーが定義した装置タイプの記号名 (3270-An)、(SIZE= パラメーターからの) 関連画面サイズ、および (FEAT= パラメーターからの) 物理端末機構が入っています。同じ装置タイプの記号名に異なる物理端末機構 (FEAT= パラメーター) を指定すると、MFS 装置特性テーブルに別の項目が生成されます。

MFS のソース定義では、DEV ステートメントのオペランドとして、TYPE=3270-An および FEAT を指定します。指定された装置タイプについて、MFS が IMS.SDFSRESL ライブラリー内の指定の装置特性テーブル DFSUDT0*x* から画面サイズを取り出します。

MFS 言語ユーティリティー (DFSUPAA0) は、画面サイズ、機構、および装置タイプの指定を使用して、IMS システム定義の仕様に合わせて IMS.FORMAT ライブラリーの DIF/DOF メンバーを作成します。画面サイズを指定できるのは IMS システム定義の時だけであるため、IMS システム定義を実行してから、DEV TYPE=3270-An を指定したユーザー定義形式を処理する MFS 言語ユーティリティーを実行しなければなりません。

MFS 装置特性テーブルは、IMS システム定義のステージ 2 で作成され、その接尾部は、IMS 複合制御ブロック、中核、および安全ディレクトリー・ブロック・モジュールの接尾部と同じで、IMSGEN マクロの SUFFIX= キーワードで指定された値が入ります。ETO で定義された端末をシステムに追加すると、システム定義を再生成せずに、MFS の装置特性テーブル・ユーティリティーを使用して、テーブルの追加または更新を行うことができます。

テーブル名 (DFSUDT0*x*) の英数字の接尾部 (*x*) は、読み取られるテーブルのバージョンを示すレベル ID です。 MFSUTL、MFSBTCH1、MFSTEST、および MFSRVC プロシージャーについては、EXEC ステートメントの DEVCHAR=パラメーターを使用して、この *x* 接尾部を指定することもできます。MFS 言語ユーティリティーで同じ接尾部を繰り返し使用すると、IMS.SDFSRESL ライブラリーから、MFS 装置特性テーブルの同じバージョンが読み取られてしまいます。

MFS 装置特性テーブルが要求されているにもかかわらず、接尾部が渡されていなかったり、接尾部付きテーブルが IMS.SDFSRESL ライブラリー内に存在していない場合、MFS 言語ユーティリティーは、デフォルトの名前 (DFSUDT00) を持つ IMS 装置特性テーブルをロードしようとします。

注**:** デフォルト・テーブル (DFSUDT00) がシステム定義で作成されなかった場合は、これは失敗します。

ETO 端末装置のためのログオン・プロセスの間、その端末装置用の MFS 装置タイプを決めるために MFS 装置特性テーブルを使用します。BIND 固有データからの画面サイズ、および ETO ログオン記述子からの装置機構を検索引数として使用します。

特定の画面サイズには、記号名を 1 つだけ対応させます。装置タイプの記号名と画面サイズを対応づけるための標準を設定しておいてください。

推奨事項**:** ユーザー定義のシンボル名のそれぞれについて、リストされている画面サイズを使用してください。

ユーザー定義の記号名
: 画面サイズ

3270-A1
: 12×80

3270-A2
: 24×80

3270-A3
: 32×80

3270-A4
: 43×80

3270-A5
: 12×40

3270-A6
: 6×40

3270-A7
: 27×132

3270-A8
: 62×160

関連資料:

- MFS 言語ユーティリティー (DFSUPAA0) (システム・ユーティリティー)
- MFS 装置特性テーブル・ユーティリティー (DFSUTB00) (システム・ユーティリティー)
- TYPE マクロ (システム定義)
- TERMINAL マクロ (システム定義)

DPM 形式のバージョン識別機能

MFS の装置出力形式 (DOF) とは、リモート・プログラムが効率よくデータを探し出して処理できるように、そのリモート・プログラムへ渡すデータの形式を定義しておくものです。 MFS の DIF は、リモート・プログラムから IMS へデータを送る方法を定義します。

適切なフォーマット設定、および正確なデータの受け渡しと解釈を行うためには、MFS の DOF および DIF と、そのデータ形式を取るリモート・プログラム連絡ブロックとを同じレベルで作成する必要があります。MFS 制御ブロックの現行レベルは、バージョン識別 (バージョン ID) と呼ばれる固有の 2 バイトのフィールドで表します。バージョン ID は、DEV ステートメントでユーザーが指定した値です。ユーザーが指定していない場合は、ITB 形式でソース定義を IMS.REFERRAL ライブラリーに保管する際に、MFS 言語ユーティリティーによって作成されます。このバージョン ID は、MFS 言語ユーティリティーが DOF または DIF を処理するときに出す通知メッセージ DFS1048I および DFS1011I で印刷されます。また、バージョン ID の確認検査を行う場合は、リモート・プログラムにもバージョン ID を組み入れておく必要があります。

出力メッセージのマッピングで使用される DOF のバージョン ID は、出力メッセージ・ヘッダーに入っています。このヘッダーのバージョン ID を使用して、リモート・プログラムでは、その制御ブロックのレベルが、DOF のバージョン ID と同じであるかどうかを確認します。

一方、リモート・プログラムは、入力メッセージを IMS にマップする際に使用する制御ブロックのバージョン ID を、入力メッセージ・ヘッダー中に入れておく必要があります。そうすることで、データをマッピングして IMS アプリケーション・プログラムに渡すときに、正しいレベルの DIF が存在しているかどうかを検査することができます。入力時に送られてきたバージョン ID が DIF のバージョン ID と一致していない場合、その入力データは受理されず、リモート・プログラムにエラー・メッセージが送信されます。検査の必要がなければ、16 進数の 0 (X'0000') のバージョン ID を送信するか、あるいは入力メッセージ・ヘッダーからバージョン ID を削除します。この場合、リモート・プログラムも MFS も、その DIF でデータを正しくマップできるものと見なします。

第 6 章 OTMA 呼び出し可能インターフェース API 参照

IMS は、OTMA アクセスに OTMA 呼び出し可能インターフェース (C/I) API を提供します。

OTMA C/I で使用されるコードとメッセージについては、「*IMS V14* メッセージおよびコード 第 *2* 巻*: DFS* 以外メッセージ」を参照してください。

OTMA 呼び出し可能インターフェースの API 呼び出し

OTMA 呼び出し可能インターフェースのアプリケーション・プログラミング・インターフェース (API) を以下にリストします。

ヘッダー・ファイル DFSYC0.H の使用

API 呼び出し側プログラムにインクルードされているヘッダー・ファイルでは、各 API 呼び出しと、その呼び出しで使用される変数を宣言します。

OTMA 呼び出し可能インターフェースを使用する C/C++ プログラムの場合は、C/C++ ヘッダー・ファイル DFSYC0.H を C/C++ プログラムにインクルードする必要があります。

ロード・モジュール DFSYCRET

オブジェクト・スタブ DFSYCRET は、すべての API 呼び出しを受け取って、要求された関数を実行するための SVC 呼び出しを出します。API 呼び出し側プログラムがバインドされている間は、オブジェクト・スタブが使用可能であることが必要です。DFSYCRET は SDFSRESL または ADFSLOAD データ・セットにあります。

OTMA C/I に関するヒント

OTMA C/I を使用してプログラミングする際には、以下のヒントを利用してください。

- 一部の OTMA C/I API 呼び出しでは、関数または関数が実行する SRB ルーチンによってポストされる ECB パラメーターが使用されます。呼び出し元は、戻りコードと出力データを検査する前に、ECB をチェックし、ECB がポストされるのを待つ必要があります。ECB は、必ず 0 で初期設定してから OTMA C/I 呼び出しに渡してください。ECB パラメーターを使用する呼び出しは以下のとおりです。
  - otma_open
  - otma_openx
  - otma_send_receive
  - otma_send_receivex
  - otma_send_async
  - otma_receive_async

- otma_alloc を呼び出すたびに、後続の otma_send_receive 呼び出しのための独立セッションが作成されます。otma_alloc 呼び出しの 1 つを使用して、IMS に送信する IMS トランザクションまたは IMS コマンドの名前を指定することができます。トランザクション名の最大長は 8 文字です。otma_alloc 呼び出しでトランザクション名またはコマンドを指定しなかった場合は、otma_send_receive 呼び出しの送信バッファーの先頭で、トランザクション名とそれに続く 1 つ以上のブランク、またはコマンドを指定する必要があります。otma_send_receive 呼び出しの後には、IMS 会話トランザクションの場合を除いて、otma_free を呼び出す必要があります。会話トランザクションの送信に関する呼び出しサンプル C を参照してください。
- OTMA C/I は、IMS アプリケーション・データ・フォーマットの標準的な LLZZ 接頭部を作成します。ユーザーが LLZZ 接頭部を作成する必要はありません。
- 複数セグメントのメッセージを IMS に送信するには、otma_send_receive 呼び出しの送信セグメント・リストで各入力セグメントの長さを指定する必要があります。セグメント・リストの最初のエレメントでは、セグメントの数を指定します。最初のエレメントの後には、セグメント 1 の長さ、セグメント 2 の長さなどが続きます。
- 複数セグメントの出力メッセージを受信すると、出力セグメント・リストが otma_send_receive 呼び出しに提供されます。出力セグメント・リストの最初のエレメントには、出力セグメントの数が入っています。最初のエレメントの後には、出力セグメント 1 の長さ、出力セグメント 2 の長さなどが続きます。
- IMS にはサンプル・プログラム (DFSYCSMP) が用意されています。
- OTMA C/I では、otma_send_receive 呼び出しにコンテキスト・トークンを渡すことで、IMS に保護トランザクションを送信することもできます。
- 一部の OTMA C/I 呼び出しでは呼び出し側プログラムが待機する必要があるため、IMS での長期実行トランザクションや内部での OTMA C/I のハングを回避するために、呼び出し側プログラムにタイムアウト・ルーチンを実装することを強くお勧めします。
- OTMA C/I アプリケーションを効率よく実行するために、アプリケーションでの otma_open および otma_close 呼び出しの回数を制限してください。また、otma_open および otma_create のすべての呼び出しについて、呼び出しごとに異なるメンバー名を生成するのではなく、同じメンバー名を使用するようにしてください。
- otma_send_receive 呼び出しで指定した出力受信バッファーのサイズが小さすぎると、実際に返されるデータがその受信バッファーのサイズによって制限されます。otma_alloc 呼び出しで特別オプション SyncLevel1 が指定されていると、出力が拒否される可能性があります。ただし、otma_receive_async 呼び出しで出力受信バッファーのサイズが小さすぎる場合は、OTMA C/I によって常に出力が拒否されます。
- OTMA C/I は、IMS でのさまざまなプログラム間通信をサポートすることができます。詳しくは、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。
- 呼び出し元に異常条件を通知するための戻りコードが OTMA C/I から返される場合があります。この戻りコードは、デバッグのためにログに記録するか保管することをお勧めします。

- otma_send_receive 呼び出しでは、synclevel=none が設定された OTMA 送信後コミット・メッセージが IMS に送信されます。呼び出し元は、otma_send_receive に synclevel=confirm を設定することができます。
- otma_send_receive 呼び出しで入力 z/OS リソース・リカバリー・サービス (RRS) コンテキスト・トークンを指定すると、synclevel が SYNCPT に変更されて、保護トランザクションがサポートされるようになります。
- IMS で複雑なプログラム間通信を行うと、送信後コミット入力メッセージの結果が、予期された送信後コミット出力メッセージではなくコミット後送信出力メッセージになる可能性があります。OTMA C/I はこのシナリオでも正常に機能します。プログラム間通信の詳細については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。
- otma_send_async 呼び出しは、OTMA コミット後送信メッセージを IMS に送信します。
- otma_receive_async 呼び出しは、OTMA コミット後送信出力メッセージを IMS から受信します。
- OTMA C/I は、OTMA 再同期プロトコルおよび OTMA セキュリティーの PROFILE オプションをサポートしません。

関連資料:

640 ページの『同期処理用の OTMA C/I サンプル・プログラム』

651 ページの『非同期処理用の OTMA C/I サンプル・プログラム』

otma_create API

otma_create API を使用して、z/OS システム間カップリング・ファシリティー (XCF) と IMS の通信に使用するストレージを割り振ります。

説明

この呼び出しの後に、アンカーが返されます。アンカーは、後続の呼び出しで使用する必要があります。otma_create を呼び出す必要はありません。OTMA C/I は、otma_open 実行時に otma_create が呼び出されていないことを検出すると、通信用のストレージを割り振ります。otma_create を最初に呼び出す場合は、同じ入力パラメーターを後続の otma_open 呼び出しで再び使用する必要があります。

呼び出し

TCB モードのクライアントによって呼び出されます。

入力

***ecb** 次のイベント制御ブロックへのポインター。

***group_name**
XCF グループ名が入っているストリングへのポインター。(char[8])

***member_name**
このメンバーの XCF メンバー名が入っているストリングへのポインター。(char[16])

***partner_name**
IMS の XCF メンバー名が入っているストリングへのポインター。(char[16])

***sessions**
IMS でサポートされるようになっている並列セッションの数。 001 から 999 までの長整数。

***tpipe_prefix**
T パイプ名の先頭の 1 文字から 4 文字。(char[4])

OTMA T パイプの命名規則についての詳細は、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

重要: 入力フィールド **group_name**、**member_name**、および **partner_name** については、指し示される XCF 名がいずれも左揃えされ、ブランクで埋められ、正しい大文字の EBCDIC 文字で構成されている必要があります。これらの命名規則のいずれかに違反している場合は、基礎となる XCF のエラーが報告されます。

出力

***anchor**
アンカー・ワードへのポインター。

***retrsn**
戻りコード構造へのポインター。

C 言語の関数プロトタイプ

```
otma_create(
    otma_anchor_t           *anchor,        [out]
    otma_retrsn_t           *retrsn,        [out]
    ecb_t                   *ecb,           [in]
    otma_grp_name_t         *group_name,    [in]
    otma_clt_name           *member_name,   [in]
    otma_srv_name           *partner_name,  [in]
    signed long int         *sessions,      [in]
    unsigned char           *tpipe_name);   [in]
```

戻り値 (rc 値)

rc および理由は、ECB がポストされた後に有効となります。各エラーの詳しい説明については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

0 呼び出しが正常に完了しました。

8 ユーザー・エラー。

12 ストレージの取得に失敗しました。

otma_open API

呼び出し元は、IMS が使用可能なときに、otma_open を呼び出して接続を行う必要があります。呼び出し元は、接続が完了したときまたは失敗したときにポストされる ECB を待つ必要があります。IMS が稼働していないか、または OTMA が始動していないと、接続は失敗します。

説明

呼び出し元は、いつでも otma_close 呼び出しを発行して、IMS との接続の試行を取り消すことができます。それに応じて ECB がポストされます。

この接続が確立された後に IMS で障害が発生すると、関数インターフェースに対するすべての呼び出しで、IMS がメッセージを listen しなくなったことを通知する戻りコードが受信されます。クローズが実行される前に IMS が再開すると、クライアントからのアクションなしに接続が再確立されます。otma_close および otma_open インターフェースを再度呼び出して、IMS との通信を再確立することもできます。既存の会話はすべて強制終了されます。この実装では OTMA 再同期プロトコルが使用されません。

otma_open API の拡張バージョンである otma_openx によって拡張機能が提供されます。

呼び出し

TCB モードのクライアントによって呼び出されます。

入力

***anchor**
アンカー・ワードへのポインター。otma_create でアンカー環境をセットアップしていない場合は、アンカー・ワードをゼロで初期設定する必要があります。

***group_name**
z/OS システム間カップリング・ファシリティー (XCF) グループ名が入っているストリングへのポインター。(char[8])

***member_name**
このメンバーの XCF メンバー名が入っているストリングへのポインター。(char[16])

***partner_name**
IMS の XCF メンバー名が入っているストリングへのポインター。(char[16])

***sessions**
IMS でサポートされるようになっている並列セッションの数。 001 から 999 までの長整数。

***tpipe_prefix**
T パイプ名の先頭の 1 文字から 4 文字。(char[4])

OTMA T パイプの命名規則についての詳細は、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

重要: 入力フィールド **group_name**、**member_name**、および **partner_name** については、指し示される XCF 名がいずれも左揃えされ、ブランクで埋められ、正しい大文字の EBCDIC 文字で構成されている必要があります。これらの命名規則のいずれかに違反している場合は、基礎となる XCF のエラーが報告されます。

出力

***anchor**
グローバル・ストレージのアドレスを受け取るアンカー・ワードへのポインター。

***retrsn**
戻りコード構造へのポインター。

***ecb** オープン完了時にポストされるイベント制御ブロックへのポインター。

C 言語の関数プロトタイプ

```
otma_open(
         otma_anchor_t   *anchor           [in/out]
         otma_retrsn_t   *retrsn,          [out]
         ecb_t           *ecb,             [out]
         otma_grp_name_t *group_name,      [in]
         otma_clt_name_t *member_name,     [in]
         otma_srv_name_t *partner_name,    [in]
         signed long int *sessions,        [in]
         unsigned char   *tpipe_name);     [in]
```

通知コード

OPEN ルーチンの呼び出し元は、OPEN に提供された ECB をチェックする必要があります。この ECB がまだポストされていない場合、呼び出し元は ECB を待つ (つまり OPEN プロトコルの完了を待つ) 必要があります。

0 XCF OPEN が正常に完了しました。

4 IMS が作動不能です。後で再試行してください。

8 XCF グループおよびメンバーはすでにアクティブです。

12 システム・エラーが発生しました。

戻り値 (rc 値)

rc および理由は、ECB がポストされた後に有効となります。

0 XCF JOIN が正常終了し、クライアント・ビッドが送信され、確認応答を受信しました。各エラーの詳しい説明については、「*IMS* バージョン *14* コミュニケーションおよびコネクション」を参照してください。

4 IMS が作動不能です。後で再試行してください。

8 XCF グループおよびメンバーはすでにアクティブです。

12 システム・エラーが発生しました。

関連資料:

『otma_openx API』

otma_openx API

otma_openx API は、otma_open API の拡張バージョンであり、追加機能があります。呼び出し元は、IMS が使用可能なときに、otma_openx を呼び出して接続を行う必要があります。呼び出し元は、接続が完了したときまたは失敗したときにポストされる ECB を待つ必要があります。IMS が稼働していないか、または OTMA が始動していないと、接続は失敗します。

説明

拡張機能には以下のものがあります。

- OTMA DRU 出口ルーチンを指定する機能
- API に対する将来の機能拡張に備えた追加機能

呼び出し

TCB モードのクライアントによって呼び出されます。

入力

otma_open API と同じ入力に以下の 2 つのパラメーターが追加されます。

***anchor**
アンカー・ワードへのポインター。otma_create でアンカー環境をセットアップしていない場合は、アンカー・ワードをゼロで初期設定する必要があります。

***group_name**
z/OS システム間カップリング・ファシリティー (XCF) グループ名が入っているストリングへのポインター。(char[8])

***member_name**
このメンバーの XCF メンバー名が入っているストリングへのポインター。(char[16])

***partner_name**
IMS の XCF メンバー名が入っているストリングへのポインター。(char[16])

***sessions**
IMS でサポートされるようになっている並列セッションの数。 001 から 999 までの長整数。

***tpipe_prefix**
T パイプ名の先頭の 1 文字から 4 文字。(char[4])

OTMA T パイプの命名規則についての詳細は、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

***ims_dru_name**
ユーザー定義の OTMA ユーザー・データ・フォーマット出口ルーチンが入っているストリングへのポインター。これは拡張 API パラメーターです。

***special_options**
非標準オプションを指定するための領域へのポインター。現時点で特別なオプションはサポートされていません。このパラメーターにはヌルを指定してください。これは拡張 API パラメーターです。

重要: 入力フィールド **group_name**、**member_name**、および **partner_name** については、指し示される XCF 名がいずれも左揃えされ、ブランクで埋められ、正しい大文字の EBCDIC 文字で構成されている必要があります。これらの命名規則のいずれかに違反している場合は、基礎となる XCF のエラーが報告されます。

出力

***anchor**
グローバル・ストレージのアドレスを受け取るアンカー・ワードへのポインター。

***retrsn**
戻りコード構造へのポインター。

***ecb** オープン完了時にポストされるイベント制御ブロックへのポインター。

C 言語の関数プロトタイプ

```
otma_openx(
          otma_anchor_t     *anchor,           [out]
          otma_retrsn_t     *retrsn,           [out]
          ecb_t             *ecb,              [out]
          otma_grp_name_t   *group_name,       [in]
          otma_clt_name_t   *member_name,      [in]
          otma_srv_name_t   *partner_name,     [in]
          signed long int   *sessions,         [in]
          tpipe_prfx_t      *tpipe_prefix,     [in]
          otma_dru_name_t   *ims_dru_name,     [in]
          otma_profile4_t   *special_options); [in]
```

通知コード

otma_open API と同じです。

戻り値 (rc 値)

otma_open API と同じです。

関連資料:

624 ページの『otma_open API』

otma_alloc API

otma_alloc API は、メッセージ通信を行う独立セッションを作成するために呼び出されます。

呼び出し

TCB モードのクライアントによって呼び出されます。

入力

***anchor**
otma_open によってセットアップされたアンカー・ワードへのポインター。

***username**
トランザクション・コマンドの RACF ユーザー名を保持しているストリングへのポインター。

許可プログラムからの呼び出しの場合は、入力したユーザー名が信頼されて IMS に渡されます。無許可プログラムからの呼び出しの場合は、OTMA C/I が現行のアクセス機能環境エレメント (ACEE) のコンテキストで

RACF 呼び出しを実行してユーザー名を取得します。入力したユーザー名 (存在する場合) は無視されます。無許可プログラムからの呼び出し元の場合は、ヌルを指定できます。

***transaction**
IMS に送信する IMS のトランザクションまたはコマンドの名前。

入力した IMS コマンドが 8 文字を超えている場合は、コマンドの最初の 8 文字がこのパラメーターに入力されます。コマンドの残りの文字は、後続の otma_send_receive API の送信バッファーの先頭に入力する必要があります。

このパラメーターをブランクのままにする場合は、後続の otma_send_receive API の送信バッファーの先頭に、IMS トランザクション名またはコマンドを (左揃えにして) 指定する必要があります。

***prfname**
トランザクション/コマンドの RACF グループ名を保持しているストリングへのポインター。

***special_options**
後続の otma_send_receive または otma_send_receivex API 呼び出しに対する処理オプションへのポインター。サポートされている処理オプションは次のとおりです。

ビット **0**
SyncOnReturn - このオプションでは、IMS は z/OS リソース・リカバリー・サービス (RRS) コンテキスト・トークンなしでメッセージを処理するように要求されます。この場合、ユーザー ID はRRS CTXRDTA が呼び出されると取得されます。

ビット **1**
SyncLevel1 - このオプションでは、OTMA C/I のデフォルトである同期レベル 0 の代わりに、OTMA send_then_commit 同期レベル 1 が使用されます。追加情報については、 DFSYCO ヘッダー・ファイルを参照してください。

出力

***retrsn**
戻りコード構造へのポインター。

***session_handle**
後続の otma_send_receive のセッションを一意的に識別するセッション・ハンドルへのポインター。

C 言語の関数プロトタイプ

```
otma_alloc(
          otma_anchor_t  *anchor,           [in]
          otma_retrsn_t  *retrsn,           [out]
          sess_handle_t  *session_handle,   [out]
          otma_profile_t *special_options,  [in]
          tran_name_t    *transaction,      [in]
          racf_uid_t     *username,         [in]
         racf_prf_t     *prfname);          [in]
```

戻り値 (rc 値)

rc および理由は、ECB がポストされた後に有効となります。各エラーの詳しい説明については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

0 成功しました。

4 セッション限度に達しました。

8 アンカーがヌルです。

otma_send_receive API

otma_send_receive API は、IMS とのメッセージ通信を開始するために呼び出されます。

説明

呼び出し元は、送信と受信の両方にバッファー定義を指定します。送信バッファーと受信バッファー両方の情報が提供されます。送信と同時に受信情報を提供すると、IMS から予期しないメッセージを受信することがないため、プロトコルが大幅に簡素化されます。IMS からの応答が届くと、ECB がポストされます。バッファー管理の作業は、すべてメッセージ出口ルーチンで処理されます。

otma_send_receive API の拡張バージョンである otma_send_receivex によって拡張機能が提供されます。

呼び出し

TCB モードのクライアントによって呼び出されます。

入力

***anchor**
otma_open によってセットアップされたアンカー・ワードへのポインター。

***session_handle**
otma_alloc から返された tpipe のセッション・ハンドルへのポインター。

***lterm**
lterm 名前フィールドへのポインター。入力時は IMS に渡されます。出力時は、IMS から返された lterm フィールドへと更新されます。どちらの場合も、ブランクにすることができます。

***modname**
MODname 名前フィールドへのポインター。入力時は IMS に渡されます。出力時は、IMS から返された MODname フィールドへと更新されます。どちらの場合も、ブランクにすることができます。

入力した modname が DFSM01、DFSMO2、または DFSM05 である場合、その値はブランクとして処理されます。

***send_buffer**
IMS に送信するデータへのポインター。トランザクション・パラメーター

にヌルを指定した場合は、IMS への送信時に、クライアント・コードでこのバッファー内のデータにトランザクション名またはコマンド、およびブランクを指定する必要があります。

***send_length**
送信データの長さ。

***send_segment_list**
IMS に送信するメッセージ・セグメントの長さの配列。最初のエレメントは、その後に続くセグメント長の数です。オプションとして、単一セグメントを送信する場合は、最初のエレメントまたは配列のアドレスをゼロにすることができます。

***receive_buffer**
IMS からの応答メッセージを受信するバッファーへのポインター。

***receive_length**
メッセージの受信に使用するバッファーの長さ。

***receive_segment_list**
IMS から送信されたセグメントの数を保持する配列。最初のエレメントは、配列内のエレメントの数に設定する必要があります。オプションとして、単一セグメントを受信する場合は、最初のエレメントまたは配列のアドレスをゼロにすることができます。いずれの場合も、すべてのセグメントがセグメントの境界が示されることなく連続して受信されます。

***context_id**
ヌルまたは z/OS リソース・リカバリー・サービス からの分散同期点コンテキスト ID。

- 許可された呼び出し元の場合、OTMA C/I はコンテキスト ID を IMS に直接渡し、コンテキスト ID データを検証しません。
- 無許可の呼び出し元の場合、OTMA C/I は CTXSWCH 呼び出しを実行してトークンの関連付けを解除し、そのトークンがタスクの現行トークンがどうかを検証します。OTMA C/I は、IMS から応答を受信すると、コンテキストを切り替えてタスクに戻してから、呼び出し元に制御を返します。

出力

***retrsn**
戻りコード構造へのポインター。

***ecb Event**
メッセージ通信完了時にポストされる制御ブロック。

***received_length**
`receive_buffer` に受信されたデータの長さを受け取るフィールド。セグメント長の合計と等しいはずです。

***receive_segment_list**
IMS から受信したメッセージ・セグメントの長さの配列。最初のエレメントはその後に続くセグメント長の数であり、クライアントによって配列の最大長を示す値に設定する必要があります。この値は受信時に変更されます。

***error_message**

エラー・メッセージ・フィールドへのポインターのアドレス。このフィールドは、IMS からエラー・メッセージや情報メッセージを受信するためにユーザーが用意します。通知コード 20 が返された場合は、このフィールドにデータが入ります。

C 言語の関数プロトタイプ

```
otma_send_receive(
            otma_anchor_t  *anchor,                [in]
            otma_retrsn_t  *retrsn,                [out]
            ecb_t          *ecb,                   [out]

            sess_handle_t  *session_handle,        [in]
            lterm_name_t   *lterm,                 [in/out]
            mod_name_t     *modname,               [in/out]

            char           *send_buffer,           [in]
            data_leng_t    *send_length,           [in]
            ioseg_list_t   *send_seg_list,         [in]

            char           *receive buffer,        [in]
            data_leng_t    *receive_length,        [in]
            data_leng_t    *received_length,       [out]
            ioseg_list_t   *receive_segment_list,  [in/out]
            context_t      *context_id,            [in]

            char           *error_message);        [out]
```

通知コード

0 正常に完了しました。

8 アンカーがないか、セッション・ハンドルが正しくないか、セグメントが大きすぎます。

12 送信が失敗しました。

16 受信が取り消されました。

20 IMS からエラーが返されました。

戻り値 (rc 値)

rc および理由は、ECB がポストされた後に有効となります。各エラーの詳しい説明については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

0 正常に完了しました。

8 アンカーがないか、セッション・ハンドルが正しくないか、セグメントが大きすぎます。

12 送信が失敗しました。

16 受信が取り消されました。

20 IMS からエラーが返されました。

otma_send_receivex API

otma_send_receivex API は、otma_send_receive API と同じ機能に加えて以下の拡張機能を提供します。

呼び出し

otma_send_receive API と同じです。

入力

otma_send_receive API と同じ入力に以下のパラメーターが追加されます。

***otma_user_data**

OTMA ユーザー・データへのポインター。OTMA ユーザー・データ・フィールドには、ユーザー入力を識別したり、入力を出力に関連付けたりするためのユーザー・データを入れることができます。このフィールドに値を指定すると、データが IMS に送信されます。IMS ユーザー出口 OTMAIOED および DFSYDRU0 で、このデータの読み取りや変更を行うことができます。otma_user_data を指定して otma_receive_async API を発行すると、データがユーザーに返されます。

OTMA ユーザー・データがない場合は、このフィールドにヌルを指定します。

出力

otma_send_receive API と同じです。

C 言語の関数プロトタイプ

```
otma_send_receivex(
             otma_anchor_t  *anchor,                [in]
             otma_retrsn_t  *retrsn,                [out]
             ecb_t          *ecb,                   [out]

             sess_handle_t  *session_handle,        [in]
             lterm_name_t   *lterm,                 [in/out]
             mod_name_t     *modname,               [in/out]

             char           *send_buffer,           [in]
             data_leng_t    *send_length,           [in]
             data_leng_t    *send_segment_list,     [in]

             char           *receive buffer,        [in]
             data_leng_t    *receive_length,        [in]
             data_leng_t    *received_length,       [out]
             data_leng_t    *receive_segment_list,  [in/out]
             context_t      *context_id,            [in]

             char           *error_message,         [out]
             otma_user_t    *otma_userdata);        [in/out]
```

通知コード

otma_send_receive API と同じです。

戻り値 (rc 値)

otma_send_receive API と同じです。

otma_send_async API

otma_send_async API は、IMS にトランザクションまたはコマンドを送信するために呼び出されます。

呼び出し

TCB モードのクライアントによって呼び出されます。

制約事項**:** この API を使用して、IMS 高速機能トランザクション、保護トランザクション (z/OS リソース・リカバリー・サービス コンテキスト ID を持つトランザクション)、および IMS 会話型トランザクションを送信することはできません。この 3 タイプのトランザクションには、代わり otma_send_receive API を使用してください。

入力

***anchor**
otma_open によってセットアップされたアンカー・ワードへのポインター。

***lterm**
lterm 名前フィールドへのポインター。lterm が入力されていない場合は、ヌルを指定します。

***modname**
MODname 名前フィールドへのポインター。MODname が入力されていない場合は、ヌルを指定します。

***otma_user_data**
OTMA ユーザー・データへのポインター。この 1022 バイトのフィールドはオプションです。OTMA ユーザー・データ・フィールドには、入力を識別したり、入力を出力に関連付けたりするためのデータを入れることができます。このフィールドに値を指定すると、データが IMS に送信されます。IMS ユーザー出口 OTMAIOED および DFSYDRU0 で、このデータの読み取りや変更を行うことができます。otma_user_data を指定して otma_receive_async API を発行すると、データが返されます。

OTMA ユーザー・データがない場合は、このフィールドにヌルを指定します。

***prfname**
トランザクション・コマンドの RACF グループ名を保持しているストリングへのポインター。このパラメーターはオプションです。 RACF グループ名が入力されていない場合は、ヌルを指定します。

***send_buffer**
IMS に送信するデータへのポインター。トランザクション・パラメーターにヌルを指定した場合は、IMS への送信時に、クライアント・コードでこのバッファー内のデータにトランザクション名またはコマンド、およびブランクを指定する必要があります。

***send_length**
送信データの長さ。

***send_segment_list**
IMS に送信するメッセージ・セグメントの長さの配列。複数セグメントの入力メッセージの場合、このパラメーターは必須です。このパラメーターを指定した場合は、最初のエレメントに入力セグメントの総数を入れる必要が

あります。 単一セグメントを入力する場合、このフィールドはオプションです。単一セグメントを送信する場合は、最初のエレメントまたは配列のアドレスをゼロにすることができます。

***special_options**
非標準オプションを指定するための領域へのポインター。現時点で特別なオプションはサポートされていません。このパラメーターにはヌルを指定してください。

***tpipe_name**
OTMA T パイプ名フィールドへのポインター。この名前は、`otma_create` および `otma_open` API に指定した T パイプ名とは異なっている必要があります。

***transaction**
IMS に送信する IMS のトランザクションまたはコマンドの名前。

入力した IMS コマンドが 8 文字を超えている場合は、コマンドの最初の 8 文字がこのパラメーターに入力されます。コマンドの残りの文字は、送信バッファーの先頭に入力する必要があります。

このパラメーターにヌルまたはブランクを指定すると、OTMA C/I は送信バッファーの先頭に IMS トランザクション名またはコマンドを入力するものと見なします。

***username**
トランザクション/コマンドの RACF ユーザー名を保持するストリングへのポインター。

許可プログラムからの呼び出しの場合は、入力したユーザー名が信頼されて IMS に渡されます。無許可プログラムからの呼び出しの場合は、OTMA C/I が現行のアクセス機能環境エレメント (ACEE) のコンテキストで RACF 呼び出しを実行してユーザー名を取得します。入力したユーザー名 (存在する場合) は無視されます。無許可プログラムからの呼び出し元の場合は、ヌルを指定できます。

出力

***ecb Event**
IMS が入力を受信または拒否したときにポストされるイベント制御ブロック。

***error_message**
エラー・メッセージ・フィールドへのポインターのアドレス。このアドレスは、IMS からエラー・メッセージや通知メッセージを受信するためにユーザーが提供します。通知コード 20 が返された場合は、このフィールドにデータが入ります。

***retrsn**
戻りコードおよび理由コード構造へのポインター。IMS OTMA が入力を拒否した場合は、NAK コードとその関連する理由コードが、OTMA C/I の理由コード 2 および 3 から入手できます。 NAK コードの説明については、「*IMS V14* メッセージおよびコード 第 *2* 巻*: DFS* 以外メッセージ」を参照してください。

C 言語の関数プロトタイプ

```
otma_send_async(
             otma_anchor_t  *anchor,                [in]
             otma_retrsn_t  *retrsn,                [out]
             ecb_t          *ecb,                   [out]

             tpipe_name_t   *tpipe_name,            [in]
             tran_name_t    *transaction,           [in]
             racf_uid_t     *username,              [in]
             racf_prf_t     *prfname,               [in]
             lterm_name_t   *lterm,                 [in]
             mod_name_t     *modname,               [in]
                 otma_user_t    *otma_userdata,         [in]

             char           *send_buffer,           [in]
             data_leng_t    *send_length,           [in]
             data_leng_t    *send_segment_list[],   [in]
             char           *error_message,         [out]
             void           *special_options);      [in]
```

通知コード

0 正常に完了しました。

8 入力が無効です。

12 入力が失敗しました。

16 入力が取り消されました (IMS がダウンしているか、OTMA が停止しています)。

20 IMS からエラー・メッセージまたは情報メッセージが返されました。

戻り値 (rc 値)

rc および理由は、ECB がポストされた後に有効となります。各エラーの詳しい説明については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

0 正常に完了しました。

8 アンカーがないか、入力が正しくありません。

12 送信が失敗しました。

16 入力が取り消されました (IMS がダウンしているか、OTMA が停止しています)。

20 IMS からエラー・メッセージまたは情報メッセージが返されました。

otma_receive_async API

otma_receive_async API は、IMS 出力メッセージまたは非送信請求メッセージを受信するために呼び出されます。呼び出し元は、IMS メッセージを受信するためのバッファー定義を指定します。IMS メッセージが受信されると、ECB がポストされます。

呼び出し

TCB モードのクライアントによって呼び出されます。

入力

***anchor**
otma_open によってセットアップされたアンカー・ワードへのポインター。

***tpipe_name**
OTMA T パイプ名フィールドへのポインター。この名前は、otma_create および otma_open API に指定した T パイプ名とは異なっている必要があります。

receive_length
メッセージの受信に使用するバッファーの長さ。

出力

***ecb Event**
IMS が応答を受信したときにポストされるイベント制御ブロック。

***error_message**
エラー・メッセージ・フィールドへのポインターのアドレス。このアドレスは、IMS からエラー・メッセージや通知メッセージを受信するためにユーザーが提供します。通知コード 20 が返された場合は、このフィールドにデータが入ります。

***lterm**
lterm 名前フィールドへのポインター。IMS から返される lterm 値で更新される可能性があります。

***modname**
MODname 名前フィールドへのポインター。IMS から返される MODname 値で更新される可能性があります。

***otma_user_data**
OTMA ユーザー・データへのポインター。この 1022 バイトのフィールドはオプションです。このフィールドが指定されている場合に IMS が OTMA ユーザー・データを返したときは、データが呼び出し元に戻されます。

受信する OTMA ユーザー・データは、otma_send_async API で提供されたものか、または IMS DRU 出口 DFSYDRU0 で作成されたものです。

***receive_buffer**
IMS からの応答メッセージを受信するバッファーへのポインター。

***received_length**
receive_buffer に受信されたデータの長さを受け取るフィールド。セグメント長の合計と等しいはずです。

***receive_segment_list**
IMS から受信したメッセージ・セグメントの長さの配列。クライアントは、最初のエレメントを、受信可能なメッセージ・セグメントの最大数を示すように設定する必要があります。すべてのセグメントを受信したあとは、最初の配列エレメントが受信したセグメントの実際の数を示し、残りの配列エレメントは受信した各セグメントの長さを示すようになります。

***retrsn**
戻りコードおよび理由コード構造へのポインター。

***special_options**
非標準オプションを指定するための領域へのポインター。現時点で特別なオプションはサポートされていません。このパラメーターにはヌルを指定してください。

C 言語の関数プロトタイプ

```
otma_receive_async(
            otma_anchor_t  *anchor,                 [in]
            otma_retrsn_t  *retrsn,                 [out]
            ecb_t          *ecb,                    [out]

            tpipe_name_t   *tpipe_name,             [in]
            lterm_name_t   *lterm,                  [out]
            mod_name_t     *modname,                [out]
                otma_user_t    *otma_userdata,          [out]

            char           *receive_buffer,         [out]
            data_leng_t    *receive_length,         [in]
            data_leng_t    *received_length,        [out]
            data_leng_t    *receive_segment_list[], [in/out]

            void           *special_options);       [in]
```

通知コード

0 正常に完了しました。

12 受信が失敗しました。

戻り値 (rc 値)

rc および理由は、ECB がポストされた後に有効となります。各エラーの詳しい説明については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

0 正常に完了しました。

8 アンカーがないか、セッション・ハンドルが正しくないか、セグメントが大きすぎます。

12 送信が失敗しました。

otma_free API

otma_free API は、otma_alloc で作成された独立セッションを解放するために呼び出されます。

呼び出し

TCB モードのクライアントによって呼び出されます。

入力

***anchor**
otma_open から返されたアンカー・ワードへのポインター。

***session_handle**
otma_alloc から返されたセッション・ハンドルへのポインター。

出力

***retrsn**
戻りコード構造へのポインター。

***session_handle**
セッション・ハンドルへのポインターは、otma_free によってヌルに設定されます。

C 言語の関数プロトタイプ

```
otma_free(
   otma_anchor_t  *anchor,          [in]
   otma_retrsn_t  *retrsn,          [out]
   sess_handle_t  *session_handle); [in/out]
```

戻り値 (rc 値)

rc および理由は、ECB がポストされた後に有効となります。各エラーの詳しい説明については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

0 成功しました。

4 割り振られたセッションではありません。

8 アンカーが正しくありません。

otma_close API

otma_close API は、通信用のストレージを解放し、z/OS システム間カップリング・ファシリティー (XCF) グループから抜けるために呼び出されます。この関数は、通信の実行中またはオープンの処理中に呼び出すことができます。このような場合は、関連するすべての ECB が、取り消しを示す通知コードとともにポストされます。

呼び出し

TCB モードのクライアントによって呼び出されます。

入力

***anchor**
otma_open から返されたアンカー・ワードへのポインター。

出力

***anchor**
otma_open から返されたアンカー・ワードへのポインター。

***retrsn**
戻りコードへのポインター。

C 言語の関数プロトタイプ

```
otma_close(
     otma_anchor_t  *anchor,  [in/out]
     otma_retrsn_t   *retrsn); [out]
```

戻り値 (rc 値)

rc および理由は、ECB がポストされた後に有効となります。各エラーの詳しい説明については、「*IMS V14* コミュニケーションおよびコネクション」を参照してください。

0 成功しました。

4 アンカーがヌルです。

8 XCF グループから抜けることができません。

OTMA C/I サンプル・プログラム

以下の 2 つのサンプル C プログラムは、表示のみを目的としています。

OTMA C/I サンプル・プログラムの保証と配布

OTMA C/I サンプル・プログラムには、保証と配布の制限事項があります。

このコードは、「現存するままの状態」で提供されます。IBM は、このコードの機能または性能に関して、明示的にも黙示的にも、法律上の瑕疵担保責任、商品性の保証および特定目的適合性の保証についての暗黙の保証を含め (ただし、これらに限定されない) いかなる保証も提供しません。 IBM は、このサンプル・コードの使用から生ずるいかなる損害に対しても、その予見の有無を問わず、責任を負いません。

以下の著作権および保証なしがそのまま表示されることを条件に、サンプル・コードは、自由に配布、複製、変更、および他のソフトウェアに組み込むことができます。

```
(c) Copyright IBM Corp.
2000 All Rights Reserved. Licensed Materials - Property of IBM
DISCLAIMER OF WARRANTIES.
以下の「同封された」コードは、IBM Corporation により作成されたサンプル・コードです。
This sample code is not part of any standard or IBM product and is provided
to you solely for the purpose of assisting you in the development
of your applications.
```

同期処理用の OTMA C/I サンプル・プログラム

次のプログラムは、OTMA C/I を同期 (1 入力 1 出力) 処理に使用する方法を示しています。

このサンプル・プログラムでは、otma_send_receive API を使用して IMS データの送受信を行います。

```
#pragma langlvl(extended)
/*********************************************************************/
/*                                                                   */
/* Callable Interface sample program using synchronous APIs          */
/*                                                                   */
/* Parameters:                                                       */
/*         Server Name                                               */
/*         Client Name                                               */
/*         User Name                                                 */
/*         Iterations                                                */
/*         Transaction                                               */
/*         User Group                                                */
```

```
/*         OTMA Data                                                  */
/*                                                                    */
/* Note:  The send buffer is sent as a file with a ddname of         */
/*        SENDBUFn in the invoking JCL.                               */
/*                                                                    */
/* Example:  //SENDBUF0 DD *,DLM=$$                                  */
/*           SEND OTMA TO SKS1                                        */
/*           $$                                                       */
/*                                                                    */
/* Note: COMPAR1 is the DDNAME of an input file used to compare      */
/*       actual output with expected output. '?' is used to delimit  */
/*       the compare string and '|' is used to ignore a char compare */
/*                                                                    */
/* Example:  //COMPAR0  DD *,DLM=$$                                  */
/*           SEND OTMA TO SKS1?                                       */
/*           $$                                                       */
/*                                                                    */
/**********************************************************************/

/**********************************************************************/
/*   Entry...                                                         */
/*                                                                    */
/*     This test program is callable from JCL                         */
/*                                                                    */
/* //NA1OTMA  JOB CLASS=A,MSGLEVEL=(1,1),MSGCLASS=H,REGION=2M       */
/* //*****************************************************************/
/* //* PARM=server_member_name tpipe_name client_member_name         */
/* //*      iterations command groupid OTMA_Data                     */
/* //MINISAMP EXEC PGM=NA1OTMA,                                      */
/* // PARM='TRAP(OFF)/IMS61CR1 IMSTESR G214992 1 /DISP groupid      */
/* //      OTMAData'                                                 */
/* //STEPLIB  DD DISP=SHR,DSN=OTMA.TEST.LOAD                        */
/* //SYSUDUMP DD SYSOUT=*                                           */
/* //STDOUT   DD SYSOUT=*                                           */
/* //STDERR   DD SYSOUT=*                                           */
/* //CEEDUMP  DD SYSOUT=*                                           */
/* //COMPAR1  DD *,DLM=$$                                           */
/* EXPECTED OUTPUT GOES HERE                                        */
/* $$                                                               */
/* //SENDBUF0 DD *,DLM=$$                                           */
/* SEND DATA GOES HERE                                              */
/* $$                                                               */
/*                                                                    */
/* Note: TRAP(OFF)/ Passes LE run-time option TRAP(OFF) which turns */
/*       off LE condition handling. To get a LE dump on abend set   */
/*       TRAP ON and provide a CEEDUMP DDNAME.                       */
/*                                                                    */
/* Note: COMPAR1 is the DDNAME of an input file used to compare     */
/*       actual output with expected output. '?' is used to delimit */
/*       the compare string and '|' is used to ignore a char compare*/
/*                                                                    */
/**********************************************************************/

/**********************************************************************/
/* An example for using the OTMA Client API in C lang.               */
/*   This program is broken into the following parts:                 */
/*      Declarations for special support                              */
/*      Process invocation parameters                                 */
/*      Setup for C signal handling                                   */
/*      Do XCF open processing and analysis                           */
/*      Do session allocate processing                                */
/*      Execute a command or transaction per invocation parm          */
/*      Do session free processing                                    */
/*      Do close                                                      */
/*      End                                                           */
/**********************************************************************/
```

```
/*********************************************************************/
/* API's for non-authorized OTMA caller                              */
/*********************************************************************/
#include "dfsyc0.h"             /* Non-authorized OTMA API's         */
#include <stdlib.h>             /* Standard C Header file            */
#include <stddef.h>             /* Standard C Header file            */
#include <stdio.h>              /* Standard C Header file            */

/*********************************************************************/
/* Internal functions                                                */
/*********************************************************************/
int  memc(char *comp_buf, char *rec_buf1 );

/* macro to move string to blank filled left justified char field    */
#define splat(t,s) ¥
  {\
  memset((char*)&(t),' ',sizeof(t));¥
  strncpy((char*)&(t), s ,strlen(s));}

/* standard math routines                                            */
#define   min(a,b)        ((a)<(b)?(a):(b))
#define   max(a,b)        ((a)>(b)?(a):(b))

main(int argc,char *argv[])
{

   /*         Following fields used by all Functions                 */

 otma_anchor_t    anchor;         /*  Handle returned by create      */
                                  /*  and used by all others.        */
 otma_retrsn_t    retrsn;         /*  Return code returned by all.   */
 long int         retsave;        /*  Return code save area          */

   /*     Following fields used by several Functions      */

 sess_handle_t     sess_handle;   /* Handle returned by allocate     */
                                  /* used by send_receive and free.  */
 otma_grp_name_t   grp_name;      /* API XCF Group Member Name.      */
 otma_clt_name_t   clt_name;      /* API XCF Client Member Name.     */
 otma_srv_name_t   srv_name;      /* API XCF Server Member Name.     */
                                  /*   (the IMS XCF member name).      */
 racf_uid_t        userid;        /* Our z/OS logon ID.              */
 racf_prf_t        groupid;       /* RACF Group ID                   */
 otma_user_t       otma_data;     /* Otma Data                       */

 lterm_name_t  lterm;                  /* Lterm name                */
 mod_name_t    modname;                  /* ModName                 */

 unsigned char error_message_text[120];/* IMS error msg field       */
                                  /* A place to receive any IMS      */
                                  /* DFS error messages.             */
 unsigned char *error_message = (unsigned char*)&error_message_text;
                                  /* a pointer to which is parameter */
                                  /* on send_receive.                */

 char         *tran;              /* Transaction Name / IMS Command  */
 tran_name_t  tran_name;          /* Transaction Name / IMS Command  */

#define BUFFER_LEN  4096          /* set our buffer sizes           */
#define NUM_BUFFER  60
#define COM_BUFFER  80
#define GROUP_NAME  "HARRY"       /* Set XCF group name to join     */

 char compare_buf[NUM_BUFFER + 1]; /* Compare buffer                */
 int long      buffer_length = 0;
 int long      rec_buffer_len = BUFFER_LEN;
 char          rec_buf[BUFFER_LEN];
```

```
 long int      rec_data_len = 0;
 char          send_buf[BUFFER_LEN];
 char          temp_buf[NUM_BUFFER];

 context_t     context = {0x00000000000000000000000000000000};
                       /* This test is not distributed sync point.   */
                       /* Too complicated for here.                  */
                       /* Normally this is obtained from RRS         */

/*********************************************************************/
/*  The callable interface makes use of z/OS Event Control Blocks.   */
/*  Any language which call the interface must deal with this.       */
/*********************************************************************/

 unsigned long  *(ecb_list[2]);        /* z/OS pause stuff           */
 unsigned long **pecb_list;

 ecb_t         ecbOPEN   = 0L;     /* ecb to be posted by OTMA API   */
 ecb_t         ecbIO     = 0L;     /* ecb to be posted by OTMA API   */
 ecb_t         signal    = 0L;     /* ecb to be posted by C runtime  */

 ecb_t         temp_ecb  = 0L;     /* used by compare and swap       */
 ecb_t         reset_ecb = 0L;     /* used by compare and swap       */

/*********************************************************************/
/*    Local variables                                                */
/*********************************************************************/

 int           iterations;
 int           loop_count;
 int           compare_result;
 long int      retcode;

 signed long   sessions;        /* number of sessions to support     */
 tpipe_prfx_t  tpipe_prefix;       /* first part of tpipe NAME       */

 FILE * stream;
 int num;       /* number of characters read from stream */


/*********************************************************************/
/* To support test functions - names of parms                        */
/* Print the parms out for documentation                             */
/*********************************************************************/

 char * argdefs[8]={ "pgm name",          /* 1                       */
                     "server name",       /* 2                       */
                     "client name",       /* 3                       */
                     "userid     ",       /* 4                       */
                     "iterations ",       /* 5                       */
                     "transaction",       /* 6                       */
                     "group id   ",       /* 7                       */
                     "otma data  ",       /* 8                       */
                     };

/*********************************************************************/
/*  Declare an array of compare file ddnames to                      */
/*  compare actual output received with expected output.             */
/*********************************************************************/

 char * infiledd[4]={"DD:COMPAR0",        /* 1                     */
                     "DD:COMPAR1" ,       /* 2                     */
                     "DD:COMPAR2" ,       /* 3                     */
                     "DD:COMPAR3" ,       /* 4                     */
                     };

/*********************************************************************/
```

```
/*  Declare an array of send file ddnames to                      */
/*  send application data to OTMA.                                */
/*****************************************************************/

 char * sndfiledd[4]= {"DD:SENDBUF0",        /* 1                  */
                       "DD:SENDBUF1" ,       /* 2                  */
                       "DD:SENDBUF2" ,       /* 3                  */
                       "DD:SENDBUF3" ,       /* 4                  */
                      };

/* ------------------------------------------------------------- */
/* Anounce the startup of the test program.                      */
/* ------------------------------------------------------------- */
   printf("Otmci01   Starting, version %s %s¥n" ,__DATE__,__TIME__ );

/* ------------------------------------------------------------- */
/* z/OS Pause Init - do this first, in case it fails bail out.   */
/* This sets up a C environment for signaling from the API.      */
/* ------------------------------------------------------------- */

  ecb_list[0] = (unsigned long *) &(signal); /* post by C signal    */
  ecb_list[1] = (unsigned long *)            /* post by OTMA        */
                ((unsigned long)&(ecbOPEN) |
                 (unsigned long)0x80000000);/* end of list         */
  pecb_list = &ecb_list[0];                 /* pointer to list     */
                                            /* define callable I/F */

/*****************************************************************/
/* Begin Test Case...                                            */
/* Anounce the startup of the test program.                      */
/*****************************************************************/
   printf("OTMCI01 Run Date: %s Run Time: %s¥n" ,__DATE__,__TIME__ );

/*****************************************************************/
/* Process parms/command line arguments.                         */
/*****************************************************************/

  /* First, print the parameters. */
  printf("Invocation parameters = ¥n");
   for (i=1 ; i<(min(8,argc));i++)
   {
     printf("%d   %s = ", i, argdefs[i]);
     printf("%s.¥n", argv[i]);
   }

   if (argc>1)  splat( srv_name, argv[1])      /* XCF memname of IMS */
   else         splat( srv_name, "IMS61CR1");  /* hard coded default */
   if (argc>2)  splat( clt_name, argv[2])      /* Client name        */
   else         splat( clt_name, "XCFTEST"  ); /* hard coded default */
   if (argc>3)  splat( userid  , argv[3])      /* ID to use          */
   else         splat( userid  , "XCFTEST"  ); /* hard coded default */
  if (argc>4)   iterations = atoi(argv[4]);    /* loop count         */
   else         iterations = 1;                /* hard coded default */
  if (argc>5)   tran = argv[5];                /* Transaction/IMS CMD*/
   else         tran = "";                     /* hard coded default */
   if (argc>6)  splat( groupid, argv[6])       /* Group ID to use    */
   else         splat( groupid, "        " );  /* hard coded default */
   if (argc>7)  splat( otma_data, argv[7])     /* OTMA Data          */
   else         splat( otma_data, "" );        /* hard coded default */

   /* ------------------------------------------------------------*/
   /* Open the file with the ddname SENDBUF0 supplied in the     */
   /* JCL which invoked this C driver.  Then read the file into  */
   /* temp_buf.                                                  */
   /* ------------------------------------------------------------*/

   if (( stream = fopen("DD:SENDBUF0","rb")) != NULL )
```

```
{
  num = fread( temp_buf, sizeof( char ), NUM_BUFFER, stream );
 printf("BUFF SIZE = %d.¥n", num);
  if (num == NUM_BUFFER) {
    printf( "Number of characters read = %i¥n", num );
    fclose( stream );
  }
  else {
    if ( ferror(stream) )
      printf( "Error reading DDNAME sendbuf0/n");
    else if ( feof(stream)) {
      printf( "EOF found¥n" );
      printf( "Number of characters read %d¥n", num );
      printf( "temp_buf = %.*s¥n", num, temp_buf);
      fclose( stream );
    }
  }
}
else
  printf( "ERROR opening DDNAME sendbuf0/n" );

/*  Initialize API parameters and buffers.                      */
splat( grp_name,GROUP_NAME );              /* XCF Group Name     */
splat( tpipe_prefix,"TPAS" );              /* tpipe Prefix Name  */
splat( tran_name,tran );                   /* do scan here       */
strncat(send_buf, temp_buf,num);   /* Copy temp_buf into send_buf */
buffer_length = strlen(send_buf);  /* Set send buffer length      */

/*********************************************************************/
/* Example of setting up parms to Open the XCF Link                  */
/*********************************************************************/

  retrsn.ret    = -1;
  retrsn.rsn[0] = -1;
  retrsn.rsn[1] = -1;
  retrsn.rsn[2] = -1;
  retrsn.rsn[3] = -1;

  sessions      = 10;    /* OTMA supports multiple parallel        */
                         /* sessions (TPIPES) How many do you want?*/

/*********************************************************************/
/*BEGIN:                                                             */
/* We have a CREATE function to set up storage and                   */
/* an OPEN function to start the protocol.                           */
/* If you do not need to customize the environment you can start    */
/* with the OPEN function, the CREATE will be done by OPEN.          */
/*********************************************************************/

  printf("-¥n");
  otma_create(&anchor,      /* (out) ptr to addr to receive ancho*/
         &retrsn,           /* (out) return code                 */
         (ecb_t *) &ecbOPEN,/* not posted by create but stored   */

         &grp_name,         /* (in) ptr to valid groupname       */
         &clt_name,         /* (in) Our member name              */
         &srv_name,         /* (in) Our server name              */

         &sessions,         /* (in) number of sessions to support*/
         &tpipe_prefix      /* (in) first part of tpipe name     */
         );

   printf("OTMA_CREATE issued. ret = %d rsn = %.8x,%.8x,%.8x,%.8x¥n"
          "  anchor is at %.8x.¥n",
          retrsn.ret,
          retrsn.rsn[0],
          retrsn.rsn[1],
```

```
          retrsn.rsn[2],
          retrsn.rsn[3],
          anchor);


 printf("-¥n");

/*****************************************************************/
/*  Connect to IMS                                               */
/*****************************************************************/

 otma_open(&anchor,        /* out ptr to addr to receive anchor  */
         &retrsn,          /* out return code                    */
         (ecb_t *)&ecbOPEN, /* out posted by open if failure     */
                           /*     else posted by exit pgm        */
         &grp_name,        /* in ptr to valid XCF groupname      */
         &clt_name,        /* in Our member name                 */
         &srv_name,        /* in Our server name                 */

         &sessions,        /* in number of sessions to support   */
         &tpipe_prefix     /* in first part of tpipe name        */
         );

  printf("OTMA_OPEN issued. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n"
          " Waiting for ecb at %.8x.=%.8x.¥n",
          retrsn.ret,
          retrsn.rsn[1],
          retrsn.rsn[2],
          retrsn.rsn[3],
          ecb_list[1],
          *ecb_list[1]
          );

 printf("-¥n");

/* ---------------------------------------------------------------- */
/* Here we wait for Open to signal complete                         */
/* ---------------------------------------------------------------- */
   DFSYCWAT(ecb_list[1]);   /* WAIT on ecb                         */

    printf("OPEN_OTMA done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x ¥n"
           "¥nEcb at %.8x.= %.8x.¥n",
           retrsn.ret,
           retrsn.rsn[0],
           retrsn.rsn[1],
           retrsn.rsn[2],
           retrsn.rsn[3],
           ecb_list[1], *ecb_list[1]
           );

    printf("Local Area Anchor at %8.8X = %8.8X¥n",
                &anchor, anchor);

    printf("-¥n");

 /* ------------------------------------------------------------*/
 /* The post code from open indicates success or failure       */
 /* ------------------------------------------------------------*/
    if (0!=(0x00ffffff & ecbOPEN))
      {
        printf("OPEN_OTMA ecb is posted failure.¥n");
        return(retrsn.rsn[0]);
      }

 /* ------------------------------------------------------------*/
 /* Set userid to blanks if userid = bobdavis                  */
 /* ------------------------------------------------------------*/
```

```
 printf(" Trans  = %.8s,¥n  ", tran_name );
 printf(" Userid = %.8s,¥n  ", userid );
 printf("Groupid = %.8s,¥n  ", groupid );

/************************************************************/
/* Like CREATE the ALLOC function just creates control blocks */
/* and stores data in them. Other functions may be invented   */
/* to modify these structures before the command-of-execution,*/
/* SEND_RECEIVE is issued.                                    */
/************************************************************/

 otma_alloc(
           &anchor,              /* in   ptr to global word   */
           &retrsn,              /* out  rc,reason(1-4)       */

           &sess_handle,         /* out  session id           */
           NULL,                 /* in   default overrides    */

           &tran_name,           /* in   IMS tp name or cmd   */
           &userid,              /* in   RACFid  or blanks    */
           &groupid              /* in   RACF group id or blnk*/
  );

  printf("OTMA_ALLOC done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n",
         retrsn.ret,
         retrsn.rsn[0],
         retrsn.rsn[1],
         retrsn.rsn[2],
         retrsn.rsn[3]
         );

/************************************************************/
/*   Even if ALLOC fails we go on here just to prove the      */
/*   API will reject the call.                                */
/************************************************************/

/************************************************************/
/* This is the call that sends the data and prepares to       */
/* receive the answer from IMS.                               */
/*                                                            */
/* This test program can iterate with multiple calls here.    */
/************************************************************/

/* ___Send message wait for reply_____________________        */
for (loop_count = 0 ; loop_count<iterations ; loop_count++)
{
/* ___Change the environment to wait for ecbIO                */
ecbIO = 0;                            /* clear ecb for reuse  */
ecb_list[1] = (unsigned long *)         /* posted by OTMA     */
                ((unsigned long)&(ecbIO) |
                 (unsigned long)0x80000000);  /* end of list */

 if (loop_count != 0)
  {

   /* ------------------------------------------------------------*/
   /* If looping more than once open the next file to send       */
   /* and read it into the send_buf.                             */
   /* ------------------------------------------------------------*/

   if (( stream = fopen(sndfiledd[loop_count],"rb")) != NULL )
    {
     num = fread( temp_buf, sizeof( char ), NUM_BUFFER, stream );
     printf("BUFF SIZE = %d.¥n", num);
     if (num == NUM_BUFFER) {
       fclose( stream );
```

```
      }
      else {
       if ( ferror(stream) )
        printf( "Error opening file
        else if ( feof(stream)) {
          printf( "EOF found¥n" );
          printf( "Number of characters read %d¥n", num );
          printf( "temp_buf = %.*s¥n", num, temp_buf);
          fclose( stream );
       }
      }
     }
    else
     printf( "Error opening file %s¥n", sndfiledd[loop_count]);
    /*  Initialize send and receiving buffers.       */
    memset(rec_buf ,0, sizeof(rec_buf));
    memset(send_buf ,0, sizeof(send_buf));
    strcat(send_buf, temp_buf );
    strcat(send_buf, " " );
    buffer_length = strlen(send_buf);
    printf("
    printf ("buffer length = %d¥n", buffer_length);
   }  /* end if loop_count != 0    */

   /* Print otma_send_receive parms and start of API  */
   memset(error_message_text ,0, sizeof(error_message_text));
   printf("Send buf at %.8x.¥n", &send_buf);
   printf("Send buf = %s.¥n", send_buf);
   printf("Receive buf at %.8x.¥n", &rec_buf);
   printf("Lterm = %.8s.¥n", lterm );
   printf("Modname = %.8s.¥n", modname );

   printf("-¥n");
   otma_send_receivex(
                    &anchor,          /* (in)  anchor block         */
                    &retrsn,          /* (out) return status        */
                    &ecbIO,           /* (in)  ecb address          */

                    &sess_handle,     /* (in)  session handle       */
                    &lterm,           /* (in/out)  logical terminal */
                    &modname,         /* (in/out)  module name      */

  (unsigned char *) &send_buf,        /* (in)  send buffer          */
                    &buffer_length,   /* (in)  size of send buffer  */
                    0,                /* (in)  send_segment_list    */

  (unsigned char *) &rec_buf,         /* (in)  receive buffer       */
                    &rec_buffer_len,  /* (in)  size of buffer       */
                    &rec_data_len,    /* (out) received data length */
                    0,                /* (in/out) receive seg list  */

                    &context,         /* (in)  context id           */
                    &error_message,   /* (out) ims message          */
                    &otma_data);      /* (in)  Otma Data            */

    printf("OTMA_SEND done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n",
            retrsn.ret,
            retrsn.rsn[0],
            retrsn.rsn[1],
            retrsn.rsn[2],
            retrsn.rsn[3]);

/* ------------------------------------------------------------- */
/* Here we wait for receive to signal complete                   */
/* An application can go do other thing while IMS is processing and */
/* while the XCF scheduled SRBs are returning data to the caller's  */
/* buffers. DO NOT DEALLOCATE THE BUFERS WHILE THIS IS GOING ON!    */
```

```
/* None of the output areas of the SEND_RECIEVE can be freed until  */
/* the ECB is posted complete.                                       */
/* ----------------------------------------------------------------- */

   DFSYCWAT(ecb_list[1]);      /* WAIT on ecb                    */

   retsave = retrsn.ret;       /* Save Receive return code       */

    printf("OTMA_RECEIVE done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n"
           "¥nEcb at %.8x.= %.8x.¥n",
           retrsn.ret,
           retrsn.rsn[0],
           retrsn.rsn[1],
           retrsn.rsn[2],
           retrsn.rsn[3],
           ecb_list[1],
           *ecb_list[1]
           );

    if (retrsn.ret != 0)
    {

     /* ___Error path Free allocated session _____________________ */
     printf("-error path retrsn.ret=
     printf("-¥n");
     printf( "Error message = %s¥n", error_message );
     otma_free(
                   & anchor,        /* (out)  ptr to global word   */
                   & retrsn,        /* (out)  rc,reason (1-4)      */
                   & sess_handle    /* (in)   unique path id       */
              );

     printf("OTMA_FREE done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x ¥n",
            retrsn.ret,
            retrsn.rsn[0],
            retrsn.rsn[1],
            retrsn.rsn[2],
            retrsn.rsn[3]
            );

     /* ___Sever IMS connection _________________________        */
     printf("-¥n");
     otma_close(
                   & anchor,        /* (in,out) tr to otma anchor */
                   & retrsn         /* (out) rc,reason (1-4)      */
             );

     printf("OTMA_CLOSE done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n",
            retrsn.ret,
            retrsn.rsn[0],
            retrsn.rsn[1],
            retrsn.rsn[2],
            retrsn.rsn[3]
            );

     return (retsave);      /* EXIT with receive API return code */
    }

  /* ----------------------------------------------------------*/
  /*  If SEND_RECEIVE worked ..                                */
  /* ----------------------------------------------------------*/


  /* ----------------------------------------------------------*/
  /*  Open the compare file containing the expected output     */
  /*  of the receive buffer.  Compare the expected output      */
  /*  with the actual output and return the result.            */
```

```
   /* ----------------------------------------------------------*/

  rec_buf[0] = ' ';         /* Remove possible NL ie x'15' */
  printf( "infiledd = %s¥n", infiledd[loop_count] );

  if (( stream = fopen(infiledd[loop_count],"rb")) != NULL )
  {
    num = fread( compare_buf, sizeof( char ), COM_BUFFER, stream );
    if (num == COM_BUFFER) {  /* fread success */
      printf( "compare_buf = %s¥n", compare_buf );
      printf( "    rec_buf = %s¥n", rec_buf );
      fclose( stream );
      compare_result = memc( compare_buf, rec_buf );
      printf( "compare_result =
      if (compare_result != 0)
        return(compare_result);        /* Exit if NO COMPARE    */
    }
    else {  /* fread() failed */
      if ( ferror(stream) )         /* possibility 1 */
        printf( "Error reading file %s¥n", infiledd[loop_count]);
      else if ( feof(stream)) {     /* possibility 2 */
        printf( "EOF found¥n" );
        printf( "Number of characters read %d¥n", num );
        printf( "compare_buf = %.*s¥n", num, compare_buf);
      }
    }
  }
  else
    printf( "Error opening file %s¥n", infiledd[loop_count]);
}      /* end of loop */

  /*****************************************************************/
  /* Once a message is sent to IMS and the answer received it      */
  /* is usual to release the tpipe for use by other transactions.  */
  /* For conversational trans an application would keep using      */
  /* the handle to continue a conversational transaction with IMS. */
  /* The Transaction name is specified in the ALLOC and it is      */
  /* intended that a FREE be done at the end of each transaction   */
  /* and a new ALLOC be done for the next one. This is not         */
  /* expensive.                                                    */
  /*****************************************************************/

    printf("-¥n");
    otma_free(
                    & anchor,        /* (out)  ptr to global word   */
                    & retrsn,        /* (out)  rc,reason (1-4)      */
                    & sess_handle    /* (in)   unique path id       */
             );

     printf("OTMA_FREE done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x ¥n",
            retrsn.ret,
            retrsn.rsn[0],
            retrsn.rsn[1],
            retrsn.rsn[2],
            retrsn.rsn[3]
            );

    printf("-¥n");

    /*                                                             */
    /* Finally, CLOSE severs the connection with IMS and frees the   */
    /* Storage used by the OTMA API.                               */
    /*  This will be done at job-step termination but its untidy.    */
    /*                                                             */

     otma_close(
                    & anchor,        /* (in,out) ptr to otma anchor */
```

```
                    & retrsn         /* (out)  rc,reason (1-4)      */
              );
    printf("OTMA_CLOSE done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x ¥n",
            retrsn.ret,
            retrsn.rsn[0],
            retrsn.rsn[1],
            retrsn.rsn[2],
            retrsn.rsn[3]
            );

    return (compare_result);          /* Retern return code         */
} /* end of main */

/*====================================================================*/
/*      Subroutine to compare expected results(compare_buf)           */
/*      with actual results(err_msg) the "|" is used to signify       */
/*      an ignore compare and "?" is used to mark the end of string. */
/* Note: Compare starts using an index i=1 ie. the 2nd character      */
/*       because the 1st character was blanked out. ( NL x'15' )      */
/*====================================================================*/


int memc(char *comp_buf, char *rec_buf1)
{

 int j;
 int i;

  j = 0;

  for (i=1;
      ( (j==0) && (comp_buf[i] != '?') );
       i++ )
    {
      if( comp_buf[i] != '|' )                    /* Ignore compare  */
      {
        if( comp_buf[i] != rec_buf1[i])           /* compare ok ?    */
        {
          j++;                                    /* No              */
          printf( "MISCOMPARE !!!   ¥n" );
          printf( "comp_buf[%d] = %c¥n", i, comp_buf[i] );
          printf( "rec_buf1[%d] = %c¥n", i, rec_buf1[i] );
        }
        else
          ;
      }
      else
        ;                                        /* Else null        */
    }

 return (j);
}
```

関連資料:

621 ページの『OTMA C/I に関するヒント』

非同期処理用の OTMA C/I サンプル・プログラム

次のプログラムは、OTMA C/I を非同期 (非送信請求) 処理に使用する方法を示しています。

このサンプル・プログラムでは、ループごとに otma_send_asynch 呼び出しと otma_receive_asynch 呼び出しが 1 回ずつ実行されます。

推奨事項: 同期 (1 入力 1 出力) 処理を排他的に使用する場合は、otma_send_receive API を使用してください。otma_send_receive API は、最も効率のよい同期処理の手段を提供します。

```
#pragma langlvl(extended)

/*********************************************************************/
/*                                                                   */
/* Callable Interface sample program using asynchronous APIs         */
/*                                                                   */
/* Parameters:                                                       */
/*         Server Name                                               */
/*         Client Name                                               */
/*         Transaction                                               */
/*         User Name                                                 */
/*         User Group                                                */
/*         Lterm                                                     */
/*         Mod Name                                                  */
/*         OTMA Data                                                 */
/*         Iterations                                                */
/*                                                                   */
/* Note:  The send buffer is sent as a file with a ddname of         */
/*        SENDBUFn in the invoking JCL.                              */
/*                                                                   */
/* Example:  //SENDBUF0 DD *,DLM=$$                                  */
/*           SEND OTMA TO SKS1                                       */
/*           $$                                                      */
/*                                                                   */
/* Note: COMPAR1 is the DDNAME of an input file used to compare      */
/*       actual output with expected output. '?' is used to delimit  */
/*       the compare string and '|' is used to ignore a char compare */
/*                                                                   */
/* Example:  //COMPAR0  DD *,DLM=$$                                  */
/*           SEND OTMA TO SKS1?                                      */
/*           $$                                                      */
/*                                                                   */
/* Note: TPIPBUFn is the DDNAME of an input file used to specify     */
/*       the tpipe name to be used for each iteration.               */
/*                                                                   */
/* Example:  //TPIPEBUF0  DD *,DLM=$$                                */
/*           TPIPE001                                                */
/*           $$                                                      */
/*                                                                   */
/*********************************************************************/

/*********************************************************************/
/*   Entry...                                                        */
/*                                                                   */
/*     This test program is callable from JCL                        */
/*                                                                   */
/* //NA1OTMA  JOB CLASS=A,MSGLEVEL=(1,1),MSGCLASS=H,REGION=2M       */
/* //*************************************************************  */
/* //* PARM=server_member_name client_member_name transaction       */
/* //*      user_name group_name lterm_name ModName OTMA_Data       */
/* //*      iterations                                              */
/* //*************************************************************  */
/* //MINISAMP EXEC PGM=NA1OTMA,                                      */
/* // PARM='TRAP(OFF)/IMS61CR1 IMSTESR G214992 /DISP user01 groupid */
/* //       Lterm ModName OTMAData 1'                                */
/* //STEPLIB  DD DISP=SHR,DSN=OTMA.TEST.LOAD                         */
/* //SYSUDUMP DD SYSOUT=*                                            */
/* //STDOUT   DD SYSOUT=*                                            */
/* //STDERR   DD SYSOUT=*                                            */
/* //CEEDUMP  DD SYSOUT=*                                            */
/* //COMPAR1  DD *,DLM=$$                                            */
/* EXPECTED OUTPUT GOES HERE                                         */
/* $$                                                                */
```

```
/* //SENDBUF0 DD *,DLM=$$                                            */
/* SEND DATA GOES HERE                                              */
/* $$                                                               */
/* //TPIPBUF0 DD *,DLM=$$                                           */
/* TPIPE NAME GOES HERE                                             */
/* $$                                                               */
/*                                                                  */
/* Note: TRAP(OFF)/ Passes LE run-time option TRAP(OFF) which turns */
/*       off LE condition handling. To get a LE dump on abend set   */
/*       TRAP ON and provide a CEEDUMP DDNAME.                      */
/*                                                                  */
/* Note: COMPAR1 is the DDNAME of an input file used to compare     */
/*       actual output with expected output. '?' is used to delimit */
/*       the compare string and '|' is used to ignore a char compare*/
/*                                                                  */
/********************************************************************/

/********************************************************************/
/* An example for using the OTMA Client API in C lang.              */
/*   This program is broken into the following parts:               */
/*      Declarations for special support                            */
/*      Process invocation parameters                               */
/*      Setup for C signal handling                                 */
/*      Do XCF open processing and analysis                         */
/*      Execute an API to send data per invocation parm             */
/*      Execute an API to receive data per invocation parm          */
/*      Do close                                                    */
/*      End                                                         */
/********************************************************************/

/********************************************************************/
/* Header Definitions.                                              */
/********************************************************************/
#include "dfsyc0.h"              /* Non-authorized OTMA API's       */
#include <stdlib.h>              /* Standard C Header file          */
#include <stddef.h>              /* Standard C Header file          */
#include <stdio.h>               /* Standard C Header file          */

/********************************************************************/
/* Internal functions                                               */
/********************************************************************/
/* memory comparison macro.                                         */
/* int  memc(char *comp_buf, char *rec_buf1 );                      */
/*                                                                  */
/*                                                                  */
/* macro to move string to blank filled left justified char field   */
/* #define splat(t,s) ¥                                             */
/* {\                                                               */
/* memset((char*)&(t),' ',sizeof(t));¥                              */
/* strncpy((char*)&(t), s ,strlen(s));}                             */
/*                                                                  */
/* standard math routines                                           */
/* #define   min(a,b)        ((a)<(b)?(a):(b))                      */
/* #define   max(a,b)        ((a)>(b)?(a):(b))                      */

/********************************************************************/
/*                                                                  */
/*     This OTMA C/I Program                                        */
/*                                                                  */
/* Note: TRAP(OFF)/ Passes LE run-time option TRAP(OFF) which turns */
/*       off LE condition handling. To get a LE dump on abend set   */
/*       TRAP ON and provide a CEEDUMP DDNAME.                      */
/*                                                                  */
/* Note: COMPAR1 is the DDNAME of an input file used to compare     */
/*       actual output with expected output. '?' is used to delimit */
/*       the compare string and '|' is used to ignore a char compare*/
/*                                                                  */
```

```
/*********************************************************************/
main(int argc,char *argv[])
{

/*********************************************************************/
/*  Fields used by OTMA C/I APIs.                                    */
/*********************************************************************/

/* The following fields used by all the OTMA C/I API's.              */

 otma_anchor_t    anchor;         /* Handle returned by create       */
                                  /* and used by all others.         */
 otma_retrsn_t    retrsn;         /* Return code returned by all.    */
                                                                     */
/* The following fields are used by the otma_create and              */
/* otma_open API's.                                                  */

otma_grp_name_t   grp_name;      /* API XCF Group Member Name.      */
 otma_clt_name_t   clt_name;      /* API XCF Client Member Name.     */
 otma_srv_name_t   srv_name;      /* API XCF Server Member Name.     */
                                  /*   (IMS's XCF member name).      */
 signed long       sessions;      /* number of sessions to support   */
 tpipe_prfx_t      tpipe_prefix;  /* first part of tpipe NAME        */
   /* The following fields are used by otma_send_async API.          */

 tpipe_name_t      tpipe;         /* User tpipe Name.                */
 tran_name_t       trans;         /* IMS Trancode or CMD.            */
 racf_uid_t        user_name;     /* RACF UserID.                    */
 racf_prf_t        user_prf;      /* RACF Groupname.                 */
 lterm_name_t      lterm;         /* Input Lterm.                    */
 mod_name_t        modname;       /* Input Modname.                  */
 otma_user_t       otma_data;     /* OTMA Userdata.                  */
 char             send_buf[BUFFER_LEN];
int long          buffer_length = 0;     /* Send Buffer length.        */
unsigned char error_message_text[120]; /*  IMS error msg field -     */
                                    /* A place to receive any IMS      */
                                  /* DFS error messages.             */
 unsigned char *error_message = (unsigned char*)&error_message_text;
                                  /* a pointer to which is parameter */
                                  /* on send_receive.                */
 otma_profile2_t   send_options;  /* Send Special Options.           */

   /* The following fields are used by otma_receive_async API.       */

 lterm_name_t      rec_lterm;     /* Output Lterm.                   */
 mod_name_t        rec_modname;   /* Output Modname.                 */
 otma_user_t       rec_otma_data; /* OTMA Userdata.                  */
 char          rec_buf[BUFFER_LEN];
 int long      rec_buffer_len = BUFFER_LEN;
 long int      rec_data_len = 0;
 otma_profile3_t   rec_options;   /* Receive Special Options.        */

/*********************************************************************/
/*  The callable interface makes use of z/OS Event Control Blocks.   */
/*  Any language which call the interface must deal with this.       */
/*********************************************************************/

 unsigned long  *(ecb_list[2]);        /* z/OS pause ecb list        */
 unsigned long **pecb_list;

 ecb_t         ecbOPEN   = 0L;     /* ecb to be posted by OTMA API   */
 ecb_t         ecbIO     = 0L;     /* ecb to be posted by OTMA API   */
 ecb_t         signal    = 0L;     /* ecb to be posted by C runtime  */

 ecb_list[0] = (unsigned long *) &(signal); /* post by C signal      */
 ecb_list[1] = (unsigned long *)            /* post by OTMA          */
               ((unsigned long)&(ecbOPEN) |
```

```
              (unsigned long)0x80000000);  /* end of list           */
 pecb_list   = &ecb_list[0];               /* pointer to list       */
                                           /* define callable I/F   */

/*********************************************************************/
/*  Local Variables                                                  */
/*********************************************************************/

 long int         retsave;        /* Return code save area           */
int           iterations;     /* Number of iterations to use     */
int           loop_count;     /* Number of iterations used       */
int            compare_result; /* Return Code result of the      */
                               /* comparison for buffers.         */

/*********************************************************************/
/*  Local Constants                                                  */
/*********************************************************************/

 #define BUFFER_LEN  4096          /* Set our buffer sizes          */
 #define NUM_BUFFER  80            /* Set the number of buffers     */
 #define GROUP_NAME  "HARRY"       /* Set XCF group name to join    */
 char    temp_buf[NUM_BUFFER];     /* Swapping buffer               */
 char    compare_buf[NUM_BUFFER + 1]; /* Compare buffer             */
 FILE * stream;
 int num;       /* number of characters read from stream */

/*********************************************************************/
/* To support test functions - names of parms in order to pring   */
/* the parms out for documentation.                                  */
/*********************************************************************/

char * argdefs[10]={"Program Name",  /* 1                  */
                    "Server Name",   /* 2                   */
                    "Client Name",   /* 3                   */
                    "Transaction",   /* 4                   */
                    "User Name  ",   /* 5                   */
                    "User Group ",   /* 6                   */
                    "Lterm      ",   /* 7                   */
                    "Mod Name   ",   /* 8                   */
                    "OTMA Data  ",   /* 9                   */
                    "Iterations ",   /* 10                  */
                    };

/*********************************************************************/
/*  Declare an array of compare file ddnames to                     */
/*  compare actual output received with expected output.            */
/*********************************************************************/

 char * infiledd[4]={"DD:COMPAR0",        /* 1                */
                      "DD:COMPAR1" ,        /* 2                */
                    "DD:COMPAR2" ,       /* 3                 */
                       "DD:COMPAR3" ,         /* 4                  */
                    };

/*********************************************************************/
/*  Declare an array of send file ddnames to                        */
/*  send application data to OTMA.                                  */
/*********************************************************************/

 char * sndfiledd[4]= {"DD:SENDBUF0",       /* 1                  */
                       "DD:SENDBUF1" ,      /* 2                  */
                       "DD:SENDBUF2" ,      /* 3                  */
                          "DD:SENDBUF3" ,       /* 4                   */
                      };


/*********************************************************************/
```

```
/*  Declare an array of tpipe names ddnames for the               */
/*  otma_send_async API.                                          */
/*****************************************************************/

 char * tpipefiledd[4]= {"DD:TPIPBUF0",    /* 1                  */
                         "DD:TPIPBUF1" ,   /* 2                  */
                         "DD:TPIPBUF2" ,   /* 3                  */
                         "DD:TPIPBUF3" ,   /* 4                  */
                       };

/*****************************************************************/
/* Begin Test Case...                                            */
/* Anounce the startup of the test program.                      */
/*****************************************************************/
   printf("OTMCI02 Run Date: %s Run Time: %s¥n" ,__DATE__,__TIME__ );

/*****************************************************************/
/* Process parms/command line arguments.                         */
/*                                                               */
/* Note:  If not a parameter is not used, then "NONE" is used in */
/*        its place.                                             */
/*                                                               */
/*****************************************************************/

  /* First, print the parameters. */
  printf("Invocation parameters = ¥n");
   for (i=1 ; i<(min(11,argc));i++)
   {
     printf("%d   %s = ", i, argdefs[i]);
     printf("%s.¥n", argv[i]);
   }

   printf("¥n");

   if (argc>1 && strcmp(argv[1],"NONE") != 0)
     splat( srv_name, argv[1])             /* Server Name.       */
   else
     splat( srv_name, "IMS61CR1");         /* Hard coded default */
   if (argc>2 && strcmp(argv[2],"NONE") != 0)
     splat( clt_name, argv[2])             /* Client name        */
   else
     splat( clt_name, "XCFTEST"  );        /* Hard coded default */
   if (argc>3 && strcmp(argv[3],"NONE") != 0)
     splat( trans, argv[3])                /* IMS Tran/Cmd to use*/
   else
     splat( trans, "");                    /* Hard coded default */
   if (argc>4 && strcmp(argv[4],"NONE") != 0)
     splat( user_name, argv[4])            /* RACF Username      */
   else
     splat( user_name, "");                /* Hard coded default */
   if (argc>5 && strcmp(argv[5],"NONE") != 0)
     splat( user_prf, argv[5])             /* RACF Group ID      */
   else
     splat( user_prf, "" );                /* Hard coded default */
   if (argc>6 && strcmp(argv[6],"NONE") != 0)
     splat( lterm , argv[6])               /* Lterm to use       */
   else
     splat( lterm , "" );                  /* Hard coded default */
   if (argc>7 && strcmp(argv[7],"NONE") != 0)
     splat( modname , argv[7])             /* ModName to use     */
   else
     splat( modname , "" );                /* Hard coded default */
   if (argc>8 && strcmp(argv[8],"NONE") != 0)
     splat( otma_data, argv[8])            /* OTMAData to use    */
   else
     splat( otma_data, "" );               /* Hard coded default */
   if (argc>9 && strcmp(argv[9],"NONE") != 0)
```

```
  iterations = atoi(argv[9]);           /* Loop count         */
else
  iterations = 1;                       /* Hard coded default */

/* -------------------------------------------------------------*/
/* Open the file with the ddname SENDBUF0 supplied in the     */
/* JCL which invoked this C driver.  Then read the file into  */
/* temp_buf.                                                  */
/* -------------------------------------------------------------*/

if (( stream = fopen("DD:SENDBUF0","rb")) != NULL )
{
  num = fread( temp_buf, sizeof( char ), NUM_BUFFER, stream );
  if (num == NUM_BUFFER) {
    printf( "Number of characters read = %i¥n", num );
    fclose( stream );
  }
  else {
    if ( ferror(stream) )
      printf( "Error reading DDNAME sendbuf0/n");
    else if ( feof(stream)) {
      printf( "EOF found¥n" );
      printf( "Number of characters read %d¥n", num );
      printf( "temp_buf = %.*s¥n", num, temp_buf);
      fclose( stream );
    }
  }
}
else
  printf( "ERROR opening DDNAME sendbuf0/n" );

/*-------------------------------------------------------------*/
/* Initialize parameters for the otma_create and otma_open   */
/* APIs.                                                      */
/*-------------------------------------------------------------*/

splat( grp_name,GROUP_NAME );               /* XCF Group Name     */
splat( tpipe_prefix,"TPAS" );               /* XCF Group Name     */
strcat(send_buf, temp_buf );       /* Copy temp_buf into send_buf */
strcat(send_buf, " " );            /* add a blank for strlen      */
buffer_length = strlen(send_buf);

/*****************************************************************/
/* Example of setting up parms to Open the XCF Link              */
/*****************************************************************/

  retrsn.ret    = -1;
  retrsn.rsn[0] = -1;
  retrsn.rsn[1] = -1;
  retrsn.rsn[2] = -1;
  retrsn.rsn[3] = -1;
  r             =  0;
  sessions      = 10;    /* OTMA supports multiple parallel        */
                         /* sessions (TPIPES) How many do you want?*/

/*****************************************************************/
/*BEGIN:                                                         */
/* We have a CREATE function to set up storage and               */
/* an OPEN function to start the protocol.                       */
/* If you don't need to customize the environment you can start  */
/* with the OPEN function, the CREATE will be done by OPEN.      */
/*****************************************************************/

  otma_create(&anchor,      /* (out) ptr to addr to receive ancho*/
         &retrsn,           /* (out) return code                */
         (ecb_t *) &ecbOPEN,/* not posted by create but stored   */
```

```
          &grp_name,         /* (in) ptr to valid groupname       */
          &clt_name,         /* (in) Our member name              */
          &srv_name,         /* (in) Our server name              */

          &sessions,         /* (in) number of sessions to support*/
          &tpipe_prefix      /* (in) first part of tpipe name     */
          );

    printf("OTMA_CREATE issued. ret = %d rsn = %.8x,%.8x,%.8x,%.8x¥n"
           "  anchor is at %.8x.¥n",
           retrsn.ret,
           retrsn.rsn[0],
           retrsn.rsn[1],
           retrsn.rsn[2],
           retrsn.rsn[3],
           anchor);


  printf("-¥n");


 /*********************************************************************/
 /*  Time to try to connect to IMS                                    */
 /*********************************************************************/

  /* ___start XCF connection_________________________ */

   otma_open(&anchor,        /* out ptr to addr to receive anchor */
          &retrsn,           /* out return code                   */
          (ecb_t *)&ecbOPEN, /* out posted by open if failure     */
                             /*     else posted by exit pgm       */
          &grp_name,         /* in ptr to valid XCF groupname     */
          &clt_name,         /* in Our member name                */
          &srv_name,         /* in Our server name                */

          &sessions,         /* in number of sessions to support  */
          &tpipe_prefix      /* in first part of tpipe name       */
          );

    printf("OTMA_OPEN issued. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n"
           " Waiting for ecb at %.8x.=%.8x.¥n",
           retrsn.ret,
           retrsn.rsn[1],
           retrsn.rsn[2],
           retrsn.rsn[3],
           ecb_list[1],
           *ecb_list[1]
           );

  printf("-¥n");

/* ---------------------------------------------------------------- */
/* Here we wait for Open to signal complete                         */
/* ---------------------------------------------------------------- */
    DFSYCWAT(ecb_list[1]);   /* WAIT on ecb                */

     printf("OTMA_OPEN done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x ¥n"
            "¥nEcb at %.8x.= %.8x.¥n",
            retrsn.ret,
            retrsn.rsn[0],
            retrsn.rsn[1],
            retrsn.rsn[2],
            retrsn.rsn[3],
            ecb_list[1], *ecb_list[1]
            );

     printf("Local Area Anchor at %8.8X = %8.8X¥n",
```

```
                &anchor, anchor);


 /* ------------------------------------------------------------*/
 /* The post code from open indicates success or failure       */
 /* ------------------------------------------------------------*/
    if (0!=(0x00ffffff & ecbOPEN))
      {
        printf("OPEN_OTMA ecb is posted failure.¥n");
        return(retrsn.rsn[0]);
      }

 /*************************************************************/
 /* This is the loop that sends and receives data.            */
 /*                                                           */
 /* This test program can iterate with multiple calls here.   */
 /*************************************************************/

for (loop_count = 0 ; loop_count<iterations ; loop_count++)
{

 /* Change the environment to wait for ecbIO               */
 ecbIO = 0;                            /* clear ecb for reuse  */
 ecb_list[1] = (unsigned long *)       /* posted by OTMA       */
               ((unsigned long)&(ecbIO) |
               (unsigned long)0x80000000);  /* end of list     */

  if (loop_count != 0)
   {

    /* ------------------------------------------------------------*/
    /* If looping more than once open the next file to send       */
    /* and read it into the send_buf.                             */
    /* ------------------------------------------------------------*/

    if (( stream = fopen(sndfiledd[loop_count],"rb")) != NULL )
     {
      num = fread( temp_buf, sizeof( char ), NUM_BUFFER, stream );
      if (num == NUM_BUFFER) {
        fclose( stream );
      }
      else {
       if ( ferror(stream) )
        printf( "Error opening file
        else if ( feof(stream)) {
          printf( "EOF found¥n" );
          printf( "Number of characters read %d¥n", num );
          printf( "temp_buf = %.*s¥n", temp_buf);
          fclose( stream );
       }
      }
     }
    else
     printf( "Error opening file %s¥n", sndfiledd[loop_count]);


    /*  Put data in to Send Buffer.  */
    memset(error_message_text ,0, sizeof(error_message_text));
    memset(send_buf ,0, sizeof(send_buf));
    strcat(send_buf, temp_buf );
    strcat(send_buf, " " );
    buffer_length = strlen(send_buf);

   }  /* end if loop_count != 0    */

    /* ------------------------------------------------------------*/
    /* If looping more than once open the next tpipe to use       */
```

```
  /* and read it into the tpipe.                                  */
  /* -----------------------------------------------------------*/

  if (( stream = fopen(tpipefiledd[loop_count],"rb")) != NULL )
   {
    num = fread( temp_buf, sizeof( char ), NUM_BUFFER, stream );
    if (num == NUM_BUFFER) {
      fclose( stream );
    }
    else {
     if ( ferror(stream) )
      printf( "Error opening file
      else if ( feof(stream)) {
        printf( "EOF found¥n" );
        printf( "Number of characters read %d¥n", num );
        printf( "temp_buf = %.*s¥n", temp_buf);
        fclose( stream );
      }
    }
   }
  else
   printf( "Error opening file %s¥n", sndfiledd[loop_count]);

  memcpy(tpipe, temp_buf, 8);


 /* Print announcement of send API.    */
 printf("-¥n-¥n- Iteration #%d  Send API ---------------¥n-¥n",
        loop_count+1);
 printf("tpipe Name = %.8s.¥n", tpipe);
 printf("Transaction = %.8s.¥n", trans);
 printf("RACF UserID = %.8s.¥n", user_name);
 printf("RACF Group = %.8s.¥n", user_prf);
 printf("Lterm = %.8s.¥n", lterm );
 printf("Modname = %.8s.¥n", modname );
 printf("OTMA Data = %.50s.¥n", otma_data );
 printf("Send buf = %s.¥n", send_buf);
 printf("Send buf at %.8x.¥n", &send_buf);
 printf ("Buffer length = %d.¥n", buffer_length);
 printf ("Waiting for ecb at %.8x.=%.8x.¥n", ecb_list[1],
        *ecb_list[1]);

  otma_send_async(
                &anchor,          /* (in)  anchor block         */
                &retrsn,          /* (out) return status        */
                &ecbIO,           /* (out) ecb address          */

                &tpipe,           /* (in)  user tpipe name      */
                &trans,           /* (in)  IMS trancode or cmd  */
                &user_name,       /* (in)  RACF userid          */
                &user_prf,        /* (in)  RACF group name      */
                &lterm,           /* (in)  logical terminal     */
                &modname,         /* (in)  module name          */
                &otma_data,       /* (in)  OTMA user data       */

 (unsigned char *) &send_buf,     /* (in)  send buffer          */
                &buffer_length,   /* (in)  size of send buffer  */
                0,                /* (in)  send_segment_list    */
                &error_message,   /* (out) IMS Error msg.       */
               &send_options);    /* (in)  send special options */

 DFSYCWAT(ecb_list[1]);           /* WAIT on ecb                */

 /* Print results of send API.    */
printf("OTMA_SEND_ASYNC done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n"
       "Ecb at %.8x.=%.8x.¥n",
           retrsn.ret,
```

```
           retrsn.rsn[0],
           retrsn.rsn[1],
           retrsn.rsn[2],
           retrsn.rsn[3],
           ecb_list[1],
           *ecb_list[1]
          );

  retsave = retrsn.ret;    /*  Save otma_send_async Return Code.  */


  /* Error Processing for OTMA_SEND_ASYNC API.          */
   if (retrsn.ret != 0)
   {

    /* ___Error path Free allocated session _____________________ */
    printf("-Error send_async API retrsn.ret=
    printf( "Error message = %s¥n", error_message );

if (( stream = fopen(infiledd[loop_count],"rb")) != NULL )
{
  num = fread( compare_buf, sizeof( char ), NUM_BUFFER, stream );
  if (num == NUM_BUFFER) {  /* fread success */
    printf( "Compare_buf = %.80s.¥n", compare_buf );
    printf( "Error_buf   = %.80s.¥n", error_message );
    fclose( stream );
    compare_result = memc( compare_buf, error_message );
    printf( "compare_result =
if (compare_result != 0)
  return(compare_result);        /* Exit if NO COMPARE    */
  }
  else {  /* fread() failed */
    if ( ferror(stream) )         /* possibility 1 */
      printf( "Error reading file %s¥n", infiledd[loop_count]);
    else if ( feof(stream)) {     /* possibility 2 */
      printf( "EOF found¥n" );
      printf( "Number of characters read %d¥n", num );
      printf( "Receive compare_buf = %.*s¥n", num, compare_buf);
    }
  }
}
else
  printf( "Error opening file %s¥n", infiledd[loop_count]);

    printf("-¥n");

    /* ___Sever IMS connection _____________________________ */
    printf("-¥n");
    otma_close(
                  & anchor,        /* (in,out) tr to otma anchor */
                  & retrsn         /* (out) rc,reason (1-4)      */
           );

    printf("OTMA_CLOSE done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n",
           retrsn.ret,
           retrsn.rsn[0],
           retrsn.rsn[1],
           retrsn.rsn[2],
           retrsn.rsn[3]
           );

    return (retsave);      /* EXIT with receive API return code */
   }

/* Initialize otma_receive_async parameters.            */
 splat( rec_lterm , "" );
 splat( rec_modname , "" );
```

```
  splat( rec_otma_data , "" );
  ecbIO = 0;                            /* clear ecb for reuse      */
  ecb_list[1] = (unsigned long *)       /* posted by OTMA           */
  ((unsigned long)&(ecbIO) |
                   (unsigned long)0x80000000);  /* end of list     */

   /* Print announcement of receive API.    */
   printf("-¥n-¥n- Iteration #%d  Receive API ---------------¥n-¥n",
           loop_count+1);
   printf("tpipe Name = %.8s.¥n", tpipe);
   printf("Waiting for ecb at %.8x=%.8x.¥n", ecb_list[1],
           *ecb_list[1]);

    otma_receive_async(
                     &anchor,          /* (in)  anchor block         */
                     &retrsn,          /* (out) return status        */
                     &ecbIO,           /* (out) ecb address          */

                     &tpipe,           /* (in)  user tpipe name      */
                     &rec_lterm,       /* (in)  logical terminal     */
                     &rec_modname,     /* (in)  module name          */
                     &rec_otma_data,   /* (in)  OTMA user data       */

   (unsigned char *) &rec_buf,         /* (out) Receive buffer       */
                     &rec_buffer_len,  /* (in)  size of rec buffer   */
                     &rec_data_len,    /* (in)  send_segment_list    */
                     0,                /* (in/out) rec multiple seg  */
                     &rec_options);    /* (in)  rec special options  */

DFSYCWAT(ecb_list[1]);                 /* WAIT on ecb                */
   /* Print results of receive API.    */
  printf("OTMA_REC_ASYNC done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n"
           "Ecb at %.8x.=%.8x.¥n",
             retrsn.ret,
             retrsn.rsn[0],
             retrsn.rsn[1],
             retrsn.rsn[2],
             retrsn.rsn[3],
             ecb_list[1],
             *ecb_list[1]);
   printf("Lterm = %.8s.¥n", rec_lterm );
   printf("Modname = %.8s.¥n", rec_modname );
   printf("OTMA Data = %.50s.¥n", rec_otma_data );
   printf("Receive buf = %.80s.¥n", rec_buf);
   printf("Receive buf at %.8x.¥n", &rec_buf);
   printf("Data length = %d.¥n", rec_data_len);
   printf("Buffer length = %d.¥n", rec_buffer_len);


   retsave = retrsn.ret;    /* Save otma_receive_async Return Code.  */

   /* Error Processing for OTMA_RECEIVE_ASYNC API.          */
    if (retrsn.ret != 0)
    {

     /* ___Error path Free allocated session _____________________ */
     printf("-error path retrsn.ret=
     printf("-¥n");

     /* ___Sever IMS connection _________________________________ */
     printf("-¥n");
     otma_close(
                    & anchor,        /* (in,out) tr to otma anchor */
                    & retrsn         /* (out) rc,reason (1-4)      */
             );

     printf("OTMA_CLOSE done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x¥n",
```

```
            retrsn.ret,
            retrsn.rsn[0],
            retrsn.rsn[1],
            retrsn.rsn[2],
            retrsn.rsn[3]
            );

     return (retsave);      /* EXIT with receive API return code */
    }


  /* ------------------------------------------------------------*/
  /*  Open the compare file containing the expected output      */
  /*  of the receive buffer.  Compare the expected output       */
  /*  with the actual output and return the result.             */
  /* ------------------------------------------------------------*/

   printf("-¥n-¥n- Iteration #%d  Data Validation -----------¥n-¥n",
           loop_count+1);

 if (( stream = fopen(infiledd[loop_count],"rb")) != NULL )
 {
   num = fread( compare_buf, sizeof( char ), NUM_BUFFER, stream );
   if (num == NUM_BUFFER) {  /* fread success */
     printf( "compare_buf = %.80s.¥n", compare_buf );
     printf( "    rec_buf = %.80s.¥n", rec_buf );
     fclose( stream );
     compare_result = memc( compare_buf, rec_buf );
     printf( "compare_result =
 if (compare_result != 0)
   return(compare_result);        /* Exit if NO COMPARE    */
   }
   else {  /* fread() failed */
     if ( ferror(stream) )         /* possibility 1 */
       printf( "Error reading file %s¥n", infiledd[loop_count]);
     else if ( feof(stream)) {     /* possibility 2 */
       printf( "EOF found¥n" );
       printf( "Number of characters read %d¥n", num );
       printf( "Receive compare_buf = %.*s¥n", num, compare_buf);
     }
   }
 }
 else
   printf( "Error opening file %s¥n", infiledd[loop_count]);

  memset(rec_buf ,' ', sizeof(rec_buf));

 printf( "End of loop ¥n" );
}      /* end of loop */


   printf("-¥n");

  /*****************************************************************/
  /* Finally, CLOSE severs the connection with IMS and frees the   */
  /* Storage used by the OTMA API.                                 */
  /* This will be done at job-step termination but its untidy.     */
  /*****************************************************************/

   otma_close(
                  & anchor,        /* (in,out) ptr to otma anchor */
                  & retrsn         /* (out)  rc,reason (1-4)      */
            );
   printf("OTMA_CLOSE done. ret = %.8x rsn = %.8x,%.8x,%.8x,%.8x ¥n",
           retrsn.ret,
           retrsn.rsn[0],
           retrsn.rsn[1],
```

```
            retrsn.rsn[2],
            retrsn.rsn[3]
            );

    return (compare_result);      /* We're done */
} /* end of main */

/*====================================================================*/
/*      Subroutine to compare expected results(compare_buf)           */
/*      with actual results(err_msg) the "|" is used to signify       */
/*      an ignore compare and "?" is used to mark the end of string. */
/* Note: Compare starts using an index i=1 ie. the 2nd character      */
/*       because the 1st character was blanked out. ( NL x'15' )      */
/*====================================================================*/


int memc(char *comp_buf, char *rec_buf1)
{

 int j;
 int i;

  j = 0;

  for (i=1;
      ( (j==0) && (comp_buf[i] != '?') );
       i++ )
    {
      if( comp_buf[i] != '|' )                   /* Ignore compare  */
      {
        if( comp_buf[i] != rec_buf1[i])          /* compare ok ?    */
        {
          j++;                                   /* No              */
          printf( "MISCOMPARE !!!   ¥n" );
          printf( "comp_buf[%d] = %c¥n", i, comp_buf[i] );
          printf( "rec_buf1[%d] = %c¥n", i, rec_buf1[i] );
        }
        else
          ;
      }
      else
        ;                                       /* Else null        */
    }

 return (j);
}
```

関連資料:

621 ページの『OTMA C/I に関するヒント』

第 7 章 Web サービス開発用の WSDL-to-PL/I セグメンテーション API

WSDL-to-PL/I セグメンテーション API は、WSDL-to-PL/I トップダウン開発シナリオで、IBM Developer for System z® が生成した PL/I アプリケーション・テンプレートによって使用および参照されます。

一般的に、各 API は @dfs_async_msg_header インスタンス上で作動し、XML または SOAP から派生した IMS メッセージを取り込んだり、XML または SOAP に変換する必要がある IMS メッセージを生成したりします。

これらの API は、IMS に付属の DFSPWSIO という名前の新規モジュールに実装されています。DFSPWSIO モジュールは、IBM Developer for System z WSDL2PLI シナリオで Web サービスを IMS Enterprise Suite SOAP Gateway にデプロイするために生成された PL/I-XML コンバーターおよびサービス・プロバイダー MPP に静的にリンクされている必要があります。 DFSPWSHK 出口ルーチンを使用して、現行の SOAP ヘッダー、本文、または障害データ構造が含まれるバッファーを送受信前に検査、変更、あるいは置換することができます。

重要: この API のセットを使用するには、IMS Enterprise Suite V3.1 SOAP Gateway、および IBM Developer for System z V9.0.1.1 が最小限必要です。

関連概念:

生成された PL/I テンプレートにビジネス・ロジックを追加するための WSDL - PL/I 間セグメンテーション API (アプリケーション・プログラミング)

関連資料:

WSDL - PL/I 間セグメンテーション API 出口ルーチン (DFSPWSHK) (出口ルーチン)

インクルード・ファイル DFSPWSH

インクルード・ファイル DFSPWSH は、WSDL - PL/I 間セグメンテーション API の DFSQGETS、DFSQSETS、DFSXGETS、および DFSXSETS で使用される PL/I 構造を定義します。

組み込みファイル DFSPWSH は、z/OS サーバー上の IMS がインストールしたデータ・セット DFSSSAMP 内にあります。

以下のコードは、DFSPWSH の最初の部分 (セグメンテーション API の定義の前) を示しています。

組み込みファイル DFSPWSH の最初の部分

```
/*********************************************************************
 * IBM IMS Web service segmentation APIs
 * IMS Connect and IMS MPP
 * DFSPWSH
 *
```

```
 * This file must be included by all IMS service provider MPPs
 * developed using the IBM Ratinal Developer for System z WSDL2PLI
 * support for IMS Enterprise Suite SOAP Gateway.
 *
 * @since   1.0.0.0, 1F64F288-F037-469F-987B-60BF1FBE4B4B
 * @version 2.0.0.0, 2FFA2F75-8D4F-4951-80D5-D2444181745C
 ********************************************************************/

%push;
%noprint;
%include CEEIBMCT;
%include CEEIBMAW;
%pop;

/********************************************************************
 * Required, symmetric asynchronous message header segment for use
 * with DFSPWSIO APIs: DFSQGETS, DFSQSETS, DFSXGETS, DFSXSETS.
 * @version 2.0.0.0, 2FFA2F75-8D4F-4951-80D5-D2444181745C
 ********************************************************************/
 dcl 01 @dfs_async_msg_header_ptr pointer;
 dcl 01 @dfs_async_msg_header unaligned
        based(@dfs_async_msg_header_ptr),
  02 ll          fixed bin (15) init(0),
  02 zz          fixed bin (15) init(0),
  02 trancode    char (08) init(''),
  02 header_guid char (36) init
     ('2FFA2F75-8D4F-4951-80D5-D2444181745C'),
  02 service_context,
  03 target_namespace wchar (1024) varying init(''),
  03 service_name     wchar (0512) varying init(''),
  03 port_name        wchar (0512) varying init(''),
  03 operation_name   wchar (0512) varying init(''),
  02 language_binding,
  03 struct_max_segment_size    fixed bin(31) init(32767),
  03 soap_header_bit  bit (1) aligned init('0'b),
  03 soap_header,
  04 header_struct_name         wchar (100) varying init(''),
  04 header_struct_segment_num  fixed bin (31) init(0),
  04 header_struct_segment_cnt  fixed bin (31) init(0),
  04 header_struct_size         fixed bin (31) init(0),
  04 header_struct_ptr          pointer,
  03 soap_body_bit    bit (1) aligned init('0'b),
  03 soap_body,
  04 body_struct_name           wchar (100) varying init(''),
  04 body_struct_segment_num    fixed bin (31) init(0),
  04 body_struct_segment_cnt    fixed bin (31) init(0),
  04 body_struct_size           fixed bin (31) init(0),
  04 body_struct_ptr            pointer,
  03 soap_fault_bit   bit (1) aligned init('0'b),
  03 soap_fault,
  04 fault_struct_name          wchar (100) varying init(''),
  04 fault_struct_segment_num   fixed bin (31) init(0),
  04 fault_struct_segment_cnt   fixed bin (31) init(0),
  04 fault_struct_size          fixed bin (31) init(0),
  04 fault_struct_ptr           pointer;

  dcl @dfs_async_msg_header_size fixed bin(31)
      value(storage(@dfs_async_msg_header));

/********************************************************************
 * IMS I/O Program Communication Block (IOPCB) declarations and
 * constants.
 ********************************************************************/
 dcl 01 @dfs_iopcb_mask_ptr pointer;
 dcl 01 @dfs_iopcb_mask unaligned based(@dfs_iopcb_mask_ptr),
  02 iopcb_lterm          char(8),
  02 resv                 char(2),
```

```
  02 iopcb_status_code    char(2),
  02 iopcb_date           decimal fixed(7,0),
  02 iopcb_time           decimal fixed(6,9),
  02 iopcb_msg_seq_number fixed bin(31),
  02 iopcb_mod_name       char(8),
  02 iopcb_user_id        char(8);

/*********************************************************************
 * @param @dfs_STRUCT_TYPE constants for use with DFSPWSIO APIs:
 * DFSQGETS, DFSQSETS.
 *********************************************************************/
 dcl @dfs_soap_header_struct fixed bin(31) value(1);
 dcl @dfs_soap_body_struct   fixed bin(31) value(2);
 dcl @dfs_soap_fault_struct  fixed bin(31) value(3);

/*********************************************************************
 * Return code constants for use with DFSPWSIO APIs:
 * DFSQGETS, DFSQSETS, DFSXGETS, DFSXSETS.
 *********************************************************************/
 dcl @dfs_success              fixed bin(31) value(000);
 dcl @dfs_omitted_parameter    fixed bin(31) value(100);
 dcl @dfs_invalid_pointer      fixed bin(31) value(101);
 dcl @dfs_invalid_struct_type  fixed bin(31) value(102);
 dcl @dfs_struct_not_found     fixed bin(31) value(103);
 dcl @dfs_struct_name_mismatch fixed bin(31) value(104);
 dcl @dfs_invalid_struct_order fixed bin(31) value(105);
 dcl @dfs_invalid_struct_size  fixed bin(31) value(106);
 dcl @dfs_invalid_struct_name  fixed bin(31) value(107);
 dcl @dfs_struct_already_set   fixed bin(31) value(108);
 dcl @dfs_invalid_segment_size fixed bin(31) value(109);

 dcl @dfs_icon_buf_exhausted   fixed bin(31) value(997);
 dcl @dfs_cee_call_failure     fixed bin(31) value(998);
 dcl @dfs_dli_call_failure     fixed bin(31) value(999);

/*********************************************************************
 * IMS CEETDLI interface declarations and constants.
 *********************************************************************/
 dcl @dfs_dli_get_unique     char (4) value('GU  ');
 dcl @dfs_dli_get_next       char (4) value('GN  ');
 dcl @dfs_dli_insert         char (4) value('ISRT');
 dcl @dfs_dli_message_exists char (2) value('CF');
 dcl @dfs_dli_end_segments   char (2) value('QD');
 dcl @dfs_dli_end_messages   char (2) value('QC');
 dcl @dfs_dli_status_ok      char (2) value('  ');

 dcl @dfs_message_max_data fixed bin(31) value(2147123205);
 dcl @dfs_segment_max_data fixed bin(31) value(32763);

/*********************************************************************
 * Language Environment declarations and constants.
 *********************************************************************/
 dcl 1 @dfs_cee_feedback feedback;

/*********************************************************************
 * Note: The remainder of this file contains declarations for
 * the APIs that enable the XML Converters
 * running in IMS Connect and the MPP running in an MPR to
 * exchange messages that conform to a protocol that provides
 * service invocation context and unique language bindings for
 * each part of a SOAP message: header, body, fault.
 *********************************************************************/
 dcl @dfs_icon_buf_ptr         pointer init(null());
 dcl @dfs_icon_buf_size        fixed bin(31) init(0);
 dcl @dfs_icon_buf_used        fixed bin(31) init(0);
 dcl @dfs_struct_name          wchar(100) varying init('');
 dcl @dfs_struct_ptr           pointer init(null());
```

```
dcl @dfs_struct_size          fixed bin(31) init(0);
dcl @dfs_cee_feedback_ptr     pointer init(null());
dcl @dfs_commit_structs       bit(1) init('0'b);
dcl @dfs_debug                bit(1) init('0'b);
dcl @return_code              fixed bin(31) init(0);

/*****************************************************************
 * DFSQGETS,
 *     Get a language structure that contains either a SOAP Header,
 *     SOAP Body, or SOAP Fault. Language structures are retrieved
 *     from the IMS Message Queue using the CEETDLI interface. All
 *     language structures must retrieved from the IMS Message Queue
 *     prior to setting language structures using API DFSQSETS.
 *
 * @param @dfs_async_msg_header_ptr,
 *     A pointer-by-value to the instance of @dfs_async_msg_header
 *     that was retrieved from the IMS Message Queue by issuing
 *     a GU using the CEETDLI interface prior to invoking the API.
 *     This same instance must be passed on subsequent calls to
 *     DFSQGETS and DFSQSETS.
 *
 * @param @dfs_iopcb_ptr,
 *     A pointer-by-value to the I/O PCB that was passed to the
 *     MPP by IMS. The I/O PCB will be used by the API when invoking
 *     CEETDLI to interact with the IMS Message Queue. If a return
 *     code of 999 is received from the API, inspect the I/O PCB
 *     to determine the cause of the error.
 *
 * @param @dfs_struct_type,
 *     An integer-by-value that specifies which language structure
 *     to retrieve from the MPP's input message.  The following
 *     constants defined in include file DFSPWSH may be used:
 *     @dfs_soap_header_struct, @dfs_soap_body_struct,
 *     @dfs_soap_fault_struct.
 *
 * @param @dfs_struct_name,
 *     A string-by-reference which contains the name of the
 *     language structure that the API should retrieve from the
 *     IMS Message Queue.  This value of this parameter must
 *     correspond to the value of parameter @dfs_struct_type.
 *
 * @param @dfs_struct_ptr,
 *     A pointer-by-reference into which the API will write the
 *     address of newly-allocated storage into which the requested
 *     language structure has been copied from the IMS Message
 *     Queue.  The storage allocated by the API resides in the
 *     same address space as the caller.  Therefore, it is highly
 *     recommended that the storage be explicilty freed by the
 *     caller when no longer needed.
 *
 * @param @dfs_struct_size,
 *     An integer-by-reference into which the API will write the
 *     size in bytes of the language structure.
 *
 * @param @dfs_cee_feedback_ptr,
 *     A pointer-by-value to an instance of @dfs_cee_feedback
 *     which defines a Language Environment Condition Token.
 *     The supplied instance is updated each time the API invokes
 *     Language Environment Callable Services. If a return code of
 *     998 is received from the API, use the publication Language
 *     Environment Run-Time Messages (SA22-7566-10) to inspect
 *     the contents of the condition token and determine the
 *     cause of the error.
 *
 * @param @dfs_debug,
 *     An optional bit that indicates whether or not
 *     trace information should be displayed by the API.
```

```
 *     Under normal circumstances trace information is written
 *     to standard out and therefore can be found in the
 *     job log of the Message Processing Region.
 *
 * @return One of the following codes will be returned by the API,
 *     o @dfs_success
 *     o @dfs_omitted_parameter
 *     o @dfs_invalid_pointer
 *     o @dfs_invalid_dfs_struct_type
 *     o @dfs_struct_not_found
 *     o @dfs_struct_name_mismatch
 *     o @dfs_invalid_struct_order
 *     o @dfs_cee_call_failure
 *     o @dfs_dli_call_failure
 ********************************************************************/
 dcl DFSQGETS entry(pointer byvalue, pointer byvalue,
     fixed bin(31) byvalue, wchar(100) varying byaddr,
     pointer byaddr, fixed bin(31) byaddr, pointer byvalue,
     bit(1) optional) returns(fixed bin(31));

/*********************************************************************
 * DFSQSETS,
 *     Set a language structure that contains either the SOAP
 *     Header, SOAP Body, or SOAP Fault.  This API does not
 *     insert language structures into the IMS Message Queue
 *     until instructed to do so via parameter @dfs_commit_structs.
 *     Therefore it is an error to deallocate or otherwise
 *     invalidate structure pointers passed to the API via parameter
 *     @dfs_struct_ptr before instructing the API to commit (insert)
 *     all structures to the IMS Message Queue.
 *
 * @param @dfs_async_msg_header_ptr,
 *     A pointer-by-value to the instance of @dfs_async_msg_header
 *     that was supplied on a previous call to DFSQGETS or DFSQSETS.
 *     Subsequent calls to this API must specify the same instance
 *     of @dfs_async_msg_header as it will be progressively updated.
 *
 * @param @dfs_iopcb_ptr,
 *     A pointer-by-value to the IOPCB that was passed to the
 *     MPP by IMS. The IOPCB will be used by the API when invoking
 *     CEETDLI to interact with the IMS Message Queue. If a return
 *     code of 999 is received from the API, inspect the I/O PCB
 *     to determine the cause of the error.
 *
 * @param @dfs_struct_type,
 *     An integer-by-value that specifies which language structure
 *     to set in the IMS Message Queue.  The following
 *     constants defined in include file DFSPWSH may be used:
 *     @dfs_soap_header_struct, @dfs_soap_body_struct,
 *     @dfs_soap_fault_struct.
 *
 * @param @dfs_struct_name,
 *     A string-by-reference which contains the name of the
 *     language structure that corresponds to the supplied value of
 *     the @dfs_struct_type parameter.
 *
 * @param @dfs_struct_ptr,
 *     A pointer-by-value to the language structure that
 *     corresponds to the values specified for parameters
 *     @dfs_struct_type and @dfs_struct_name.
 *
 * @param @dfs_struct_size,
 *     An integer-by-value that specifies the size in bytes of the
 *     language structure supplied via parameter @dfs_struct_ptr.
 *
 * @param @dfs_commit_structs,
 *     A bit-by-value that indicates whether the API should
```

```
*     insert the current and all previously supplied language
*     structures into the IMS Message Queue.
*
* @param @dfs_cee_feedback_ptr,
*     A pointer-by-value to an instance of @dfs_cee_feedback
*     which defines a Language Environment Condition Token.
*     The supplied instance is updated each time the API invokes
*     Language Environment Callable Services.  If a return code of
*     998 is received from the API, use the publication Language
*     Environment Run-Time Messages (SA22-7566-10) to inspect
*     the contents of the condition token and determine the
*     cause of the error.
*
* @param @dfs_debug,
*     An optional bit that indicates whether or not
*     trace information should be displayed by the API.
*     Under normal circumstances trace information is written
*     to standard out and therefore can be found in the
*     job log of the Message Processing Region.
*
* @return One of the following codes will be returned by the API,
*     o @dfs_success
*     o @dfs_omitted_parameter
*     o @dfs_invalid_pointer
*     o @dfs_invalid_dfs_struct_type
*     o @dfs_invalid_struct_order
*     o @dfs_invalid_struct_size
*     o @dfs_invalid_struct_name
*     o @dfs_struct_already_set
*     o @dfs_invalid_segment_size
*     o @dfs_cee_call_failure
*     o @dfs_dli_call_failure
*********************************************************************/
dcl DFSQSETS entry(pointer byvalue, pointer byvalue,
    fixed bin(31) byvalue, wchar(100) varying byaddr,
    pointer byvalue, fixed bin(31) byvalue, bit(1) byvalue,
    pointer byvalue, bit(1) optional) returns(fixed bin(31));

/*********************************************************************
* DFSXGETS,
*     Get a language structure that contains either the SOAP
*     Header, SOAP Body, or SOAP Fault.  Since the IMS Message
*     Queue is not available to XML Conversion in IMS Connect,
*     language structures are retrieved from the IMS Connect input
*     buffer.  The expected format of the IMS Connect input buffer
*     is an [LLZZDATA]+ byte stream.  This API is for use by PL/I
*     XML Converters running in IMS Connect.  It is not to be used
*     by an MPP.
*
* @param @dfs_async_msg_header_ptr,
*     A pointer-by-value to the instance of @dfs_async_msg_header
*     that was retrieved from the first segment of the IMS Connect
*     input buffer prior to invoking the API.
*
* @param @dfs_icon_buf_ptr,
*     A pointer-by-value to the IMS Connect input message buffer.
*     The expected format of the buffer is an array of LLZZDATA.
*
* @param @dfs_icon_buf_size,
*     An integer-by-value that specifies the length in bytes of
*     the buffer supplied in parameter @dfs_icon_buf_ptr.
*
* @param @dfs_struct_type,
*     An integer-by-value that specifies which language structure
*     to retrieve from the MPP's input message.  The following
*     constants defined in include file DFSPWSH may be used:
*     @dfs_soap_header_struct, @dfs_soap_body_struct,
```

```
 *     @dfs_soap_fault_struct.
 *
 * @param @dfs_struct_name,
 *     A string-by-reference which contains the name of the
 *     language structure that the API should retrieve from the
 *     IMS Connect input buffer.  This value of this parameter
 *     must correspond to the value of parameter @dfs_struct_type.
 *
 * @param @dfs_struct_ptr,
 *     A pointer-by-reference into which the API will write the
 *     address of a buffer that contains the bytes of the structure
 *     that corresponds to the values specified for parameters
 *     @dfs_struct_type and @dfs_struct_name.  This buffer must be
 *     freed by the XML Converter prior to returning to IMS Connect
 *     because the Language Environment enclave in which the XML
 *     Converters execute is persistent.
 *
 * @param @dfs_struct_size,
 *     An integer-by-reference into which the API will write the
 *     size in bytes of the structure that corresponds to the
 *     specified values of parameters @dfs_struct_type
 *     and @dfs_struct_name.
 *
 * @param @dfs_cee_feedback_ptr,
 *     A pointer-by-value to an instance of @dfs_cee_feedback
 *     which defines a Language Environment Condition Token.
 *     The supplied instance is updated each time the API invokes
 *     Language Environment Callable Services.  If a return code of
 *     998 is received from the API, use the publication Language
 *     Environment Run-Time Messages (SA22-7566-10) to inspect
 *     the contents of the condition token and determine the
 *     cause of the error.
 *
 * @param @dfs_debug,
 *     An optional bit that indicates whether or not
 *     trace information should be displayed by the API.
 *     Under normal circumstances trace information is written
 *     to standard out and therefore can be found in the
 *     IMS Connect job log.
 *
 * @return One of the following codes will be returned by the API,
 *     o @dfs_success
 *     o @dfs_omitted_parameter
 *     o @dfs_invalid_pointer
 *     o @dfs_invalid_dfs_struct_type
 *     o @dfs_struct_not_found
 *     o @dfs_struct_name_mismatch
 *     o @dfs_invalid_struct_order
 *     o @dfs_icon_buf_exhausted
 *     o @dfs_cee_call_failure
 ********************************************************************/
dcl DFSXGETS entry(pointer byvalue, pointer byvalue,
    fixed bin(31) byvalue, fixed bin(31) byvalue,
    wchar(100) varying byaddr, pointer byaddr, fixed bin(31) byaddr,
    pointer byvalue, bit(1) optional) returns(fixed bin(31));

/********************************************************************
 * DFSXSETS,
 *     Set a language structure that contains either the SOAP
 *     Header, SOAP Body, or SOAP Fault.  This API does not
 *     copy language structures into the IMS Connect output buffer
 *     until instructed to do so via parameter @dfs_commit_structs.
 *     Therefore it is an error to deallocate or otherwise
 *     invalidate structure pointers passed to the API via parameter
 *     @dfs_struct_ptr before instructing the API to commit (copy)
 *     all structures to the IMS Connect output buffer.
 *     This API is for use by PL/I XML Converters running in IMS
```

```
*     Connect.  It is not to be used by an MPP.
*
* @param @dfs_async_msg_header_ptr,
*     A pointer-by-value to the instance of @dfs_async_msg_header
*     that will be sent as the first segment of the IMS message.
*
* @param @dfs_icon_buf_ptr,
*     A pointer-by-value to the IMS Connect output message buffer.
*     The expected format of the buffer is an array of LLZZDATA.
*
* @param @dfs_icon_buf_size,
*     An integer-by-value that specifies the length in bytes of
*     the buffer supplied in parameter @dfs_icon_buf_ptr.
*
* @param @dfs_icon_buf_used,
*     An integer-by-reference into which the API will write
*     the number of bytes that were required to format the
*     language structure as a multi-segment IMS message
*     in the IMS Connect output buffer.  The value of this
*     parameter will always be greater than the actual size
*     of the language structure by at least 4 bytes.
*
* @param @dfs_struct_type,
*     An integer-by-value that specifies which language structure
*     to set in the IMS Connect output buffer.  The following
*     constants defined in include file DFSPWSH may be used:
*     @dfs_soap_header_struct, @dfs_soap_body_struct,
*     @dfs_soap_fault_struct.
*
* @param @dfs_struct_name,
*     A string-by-reference which contains the name of the
*     language structure that corresponds to the supplied value of
*     the @dfs_struct_type parameter.
*
* @param @dfs_struct_ptr,
*     A pointer-by-value to the language structure that
*     corresponds to the values specified for parameters
*     @dfs_struct_type and @dfs_struct_name.
*
* @param @dfs_struct_size,
*     An integer-by-value that specifies the size in bytes of the
*     language structure.
*
* @param @dfs_commit_structs,
*     A bit-by-value that indicates whether the API should
*     copy the current and all previously supplied language
*     structures into the IMS Connect output buffer.
*
* @param @dfs_cee_feedback_ptr,
*     A pointer-by-value to a Language Environment Condition Token
*     (@dfs_cee_feedback) that is updated by the API after each
*     invocation of a Language Environment Callable Service.  When
*     a RETURN_CODE of 998 is received from the API, use the
*     publication Language Environment Run-Time Messages
*     (SA22-7566-10) to inspect the contents of the condition
*     token and determine the cause of the error.
*
* @param @dfs_debug,
*     An optional bit that indicates whether or not
*     trace information should be displayed by the API.
*     Under normal circumstances trace information is written
*     to standard out and therefore can be found in the
*     IMS Connect job log.
*
* @return One of the following codes will be returned by the API,
*     o @dfs_success
*     o @dfs_omitted_parameter
```

```
 *     o @dfs_invalid_pointer
 *     o @dfs_invalid_dfs_struct_type
 *     o @dfs_invalid_struct_order
 *     o @dfs_invalid_struct_size
 *     o @dfs_invalid_struct_name
 *     o @dfs_struct_already_set
 *     o @dfs_invalid_segment_size
 *     o @dfs_icon_buf_exhausted
 *     o @dfs_cee_call_failure
 ********************************************************************/
dcl DFSXSETS entry(pointer byvalue, pointer byvalue,
    fixed bin(31) byvalue, fixed bin(31) byaddr,
    fixed bin(31) byvalue, wchar(100) varying byaddr,
    pointer byvalue, fixed bin(31) byvalue, bit(1) byvalue,
    pointer byvalue, bit(1) optional) returns(fixed bin(31));

/********************************************************************
 * DFSB64E,
 *     This API encodes an input buffer using the base64 encoding
 *     scheme specified by RFC 3548 available at
 *     http://tools.ietf.org/html/rfc3548.
 *
 * @param @bin_input_buf_ptr (input),
 *     A pointer-by-value to the binary buffer to encode in base64.
 *     The base64 sequence will be encoded in UTF-16.
 *
 * @param @bin_input_buf_len (input),
 *     An integer-by-value that specifies the length in bytes of
 *     the binary buffer supplied in parameter @bin_input_buf_ptr.
 *
 * @param @b64_output_buf_ptr (input),
 *     A pointer-by-value to a buffer in which to write the base64
 *     representation of the supplied binary buffer
 *     @bin_input_buf_ptr(1:@bin_input_buf_len). The buffer pointed
 *     to by this parameter must have a minimum length in bytes of
 *     [4 * floor( ( @bin_input_buf_len + 2 ) /3 ]. If this
 *     parameter is set to null, the API will write the length
 *     in bytes of the base64 result to parameter
 *     @b64_output_buf_len but will not actually perform encoding.
 *
 * @param @b64_output_buf_len (input),
 *     An integer-by-reference into which the API will write the
 *     length in bytes of the base64 sequence that was written to
 *     the buffer pointed to by parameter @b64_output_buf_ptr.
 *     Recall that result will be encoded in UTF-16.
 *
 * @return This API does not return any codes.
 ********************************************************************/
dcl DFSB64E entry(pointer byvalue, fixed bin(31) byvalue,
    pointer byvalue, fixed bin(31) byaddr);

/********************************************************************
 * DFSB64D,
 *     This API decodes a base64 input buffer by reversing the
 *     encoding scheme specified by RFC 3548
 *     available at http://tools.ietf.org/html/rfc3548.
 *
 * @param @b64_input_buf_ptr (input),
 *     A pointer-by-value to the base64 buffer to decode.
 *     The base64 sequence must be encoded in UTF-16.
 *
 * @param @b64_input_buf_len (input),
 *     An integer-by-value that specifies the length in bytes of
 *     the base64 buffer supplied in parameter @b64_input_buf_ptr.
 *
 * @param @bin_output_buf_ptr (input),
 *     A pointer-by-value to a buffer in which to write the decoded
```

```
 *     representation of the supplied base64 buffer
 *     @b64_input_buf_ptr(1:@b64_input_buf_len). If this
 *     parameter is set to null, the API will write the length
 *     in bytes of the decoded result to parameter
 *     @bin_output_buf_len but will not actually perform decoding.
 *
 * @param @bin_output_buf_len (input),
 *     An integer-by-reference into which the API will write the
 *     length in bytes of the decoded result that was written
 *     to the buffer pointed to by parameter @bin_output_buf_ptr).
 *
 * @return This API does not return any codes.
 ********************************************************************/
dcl DFSB64D entry(pointer byvalue, fixed bin(31) byvalue,
    pointer byvalue, fixed bin(31) byaddr);
```

関連概念:

生成された PL/I テンプレートにビジネス・ロジックを追加するための WSDL - PL/I 間セグメンテーション API (アプリケーション・プログラミング)

DFSQGETS

DFSQGETS API は、IMS メッセージ・キューから SOAP 構造をリトリーブし、呼び出し元に対して高級言語で情報を返します。

このトピックで参照している構造および変数は、組み込みファイル DFSPWSH で定義されています (665 ページの『インクルード・ファイル DFSPWSH』を参照)。

使用法**:**

- DFSQSETS を呼び出して構造を入れる前に、DFSQGETS を使用して IMS メッセージ・キュー内のすべての構造をリトリーブする必要があります。

制約事項:

- DFSQGETS は、SOAP 本体および SOAP 障害構造のリトリーブのみをサポートします。 SOAP ヘッダー構造はサポートされません。

パラメーター:

表 *149. DFSQGETS* のパラメーター

パラメーター	タイプ	使用法	説明
@dfs_async_msg_header_ptr	POINTER BYVALUE	入力	メッセージ処理プログラム (MPP) が事前に CEETDLI インターフェースを使用して Get Unique (**GU**) を発行することで IMS メッセージ・キューからリトリーブした @dfs_async_msg_header のインスタンスに対する pointer-by-value。 **重要:** このインスタンスは、DFSQGETS および DFSQSETS に対するすべての呼び出しで受け渡す必要があります。

表 149. DFSQGETS のパラメーター (続き)

パラメーター	タイプ	使用法	説明
@dfs_iopcb_ptr	POINTER BYVALUE	入力	IMS による入力でメッセージ処理プログラム (MPP) に受け渡された I/O PCB に対する pointer-by-value。 DFSQGETS は、CEETDLI を呼び出して IMS メッセージ・キューとの対話を行う際にこの I/O PCB を使用します。 **注:** DFSQGETS からの戻りコードが 999 である場合は、I/O PCB を検査してエラーの原因を判別してください。
@dfs_struct_type	SIGNED FIXED BIN(31) BYVALUE	入力	IMS メッセージ・キューからリトリーブする言語構造のタイプを指定する integer-by-value。組み込みファイル DFSPWSH で定義されている定数 @dfs_soap_body_struct を使用することができます。
@dfs_struct_name	WCHAR(100) VARYING BYADDR	入力	IMS メッセージ・キューからリトリーブする言語構造の名前が含まれる string-by-reference。このパラメーターの値は、パラメーター @dfs_struct_type の値に対応している必要があります。
@dfs_struct_ptr	POINTER BYADDR	出力	返された言語構造を含む、新規に割り振られたストレージ・ブロックに対する pointer-by-reference。 **重要:** このストレージ・ブロックは、呼び出し元と同じアドレス・スペース内にあります。したがって、ストレージ・ブロックが不要になった場合は、呼び出し元がこのストレージ・ブロックを解放することを強く推奨します。
@dfs_struct_size	SIGNED FIXED BIN(31) BYADDR	出力	返された言語構造のサイズ (バイト数) を含む integer-by-reference。

表 *149. DFSQGETS* のパラメーター *(*続き*)*

パラメーター	タイプ	使用法	説明
@dfs_cee_feedback_ptr	POINTER BYVALUE	入力	Language Environment®条件トークンを定義する @dfs_cee_feedback のインスタンスに対する pointer-by-value。このインスタンスは、DFSQGETS が言語処理環境呼び出し可能サービスを呼び出すたびに更新されます。 **注:** DFSQGETS からの戻りコードが 998 の場合は、資料「*Language Environment* ランタイム・メッセージ *(SA88-8554-09)*」を使用して、条件トークンの状態の内容を調査し、エラーの原因を判別してください。
@dfs_debug	BIT(1) OPTIONAL	入力	DFSQGETS がトレース情報を表示するかどうかを指定するオプション・ビット (WSDL - PL/I 間セグメンテーション API のトレース出力 (アプリケーション・プログラミング)を参照)。

戻りコード:

DFSQGETS の戻りコードは、DFSPWSH 組み込みファイルで定義されている定数です。

表 *150. DFSQGETS* の戻りコード

タイプ:	名前:	値:
SIGNED FIXED BIN(31)	@dfs_success	000
	@dfs_omitted_parameter	100
	@dfs_invalid_pointer	101
	@dfs_invalid_struct_type	102
	@dfs_struct_not_found	103
	@dfs_struct_name_mismatch	104
	@dfs_invalid_struct_order	105
	@dfs_cee_call_failure	998
	@dfs_dli_call_failure	999

DFSQGETS の呼び出し例

```
01: /* Invoke API DFSQGETS to retrieve the SOAP body
02: * language structure from the IMS Message Queue.
03: */
04: @dfs_struct_name      = 'RequestBodyStruct';
05: @dfs_cee_feedback_ptr = addr(@dfs_cee_feedback);
06: @dfs_debug = '0'b;
07:
```

```
08: @return_code =
09:    DFSQGETS(@dfs_async_msg_header_ptr,
10:    @dfs_iopcb_mask_ptr, @dfs_soap_body_struct,
11:    @dfs_struct_name, @dfs_struct_ptr,
12:    @dfs_struct_size, @dfs_cee_feedback_ptr,
13:    @dfs_debug);
14:
15: if (@return_code != @dfs_success) then do;
16:    display('MYMPP#handle_myOperation(): '
17:       || 'ERROR, DFSQGETS @dfs_soap_body_struct, '
18:       || '@return_code: '|| trim(@return_code) || '.');
19:    return;
20: end; else do;
21:    RequestBodyStruct_ptr = @dfs_struct_ptr;
22: end;
```

DFSQSETS

DFSQSETS API は、入力データとして受け渡された言語構造の情報から SOAP 構造を作成します。また、指定された場合、DFSQSETS は現行の SOAP 構造および以前に提供されたすべての SOAP 構造を IMS メッセージ・キューにコピーします。

このトピックで参照している構造および変数は、DFSPWSH で定義されています (665 ページの『インクルード・ファイル DFSPWSH』を参照)。

使用法**:**

- @dfs_commit_structs ビット・セットを指定して DFSQSETS を呼び出すことで IMS 複数セグメント・メッセージを IMS メッセージ・キューにコミットするまでは、パラメーター @dfs_struct_ptr を介して DFSQSETS に受け渡された構造ポインターの割り振り解除や無効化は行わないでください。
- DFSQSETS を呼び出して構造を入れる前に、DFSQGETS を使用して IMS メッセージ・キュー内のすべての構造をリトリーブする必要があります。

制約事項:

- DFSQSETS は、SOAP ヘッダー構造の保管をサポートしていません。

パラメーター:

表 151. DFSQSETS のパラメーター

パラメーター	タイプ	使用法	説明
@dfs_async_msg_header_ptr	POINTER BYVALUE	出力	メッセージ処理プログラム (MPP) が事前に CEETDLI インターフェースを使用して Get Unique (**GU**) を発行することで IMS メッセージ・キューからリトリーブした @dfs_async_msg_header のインスタンスに対する pointer-by-value。 **重要:** このインスタンスは、DFSQGETS および DFSQSETS に対するすべての呼び出しで受け渡す必要があります。

表 *151. DFSQSETS* のパラメーター *(*続き*)*

パラメーター	タイプ	使用法	説明
@dfs_iopcb_ptr	POINTER BYVALUE	入力	IMS による入力でメッセージ処理プログラム (MPP) に受け渡された I/O PCB に対する pointer-by-value。 DFSQSETS は、CEETDLI を呼び出して IMS メッセージ・キューとの対話を行う際にこの I/O PCB を使用します。 **注:** DFSQSETS からの戻りコードが 999 である場合は、I/O PCB を検査してエラーの原因を判別してください。
@dfs_struct_type	SIGNED FIXED BIN(31) BYVALUE	入力	IMS メッセージに設定する言語構造のタイプを指定する integer-by-value。組み込みファイル DFSPWSH で定義されている定数 @dfs_soap_body_struct を使用することができます。
@dfs_struct_name	WCHAR(100) VARYING BYADDR	入力	IMS メッセージに設定する言語構造の名前が含まれる string-by-reference。このパラメーターの値は、パラメーター @dfs_struct_type の値に対応している必要があります。
@dfs_struct_ptr	POINTER BYVALUE	入力	IMS メッセージに設定する言語構造に対する pointer-by-reference。 **注:** この値は、パラメーター @dfs_struct_type および @dfs_struct_name に対して指定された値に対応している必要があります。
@dfs_struct_size	SIGNED FIXED BIN(31) BYVALUE	入力	@dfs_struct_ptr が指す言語構造のサイズ (バイト数) を指定する integer-by-value。
@dfs_commit_structs	BIT(1) BYVALUE	入力	DFSQSETS が現行の言語構造および以前に提供されたすべての言語構造を IMS メッセージ・キューに挿入するかどうかを指定する bit-by-value。

表 *151. DFSQSETS* のパラメーター *(*続き*)*

パラメーター	タイプ	使用法	説明
@dfs_cee_feedback_ptr	POINTER BYVALUE	入力	言語処理環境条件トークンを定義する @dfs_cee_feedback のインスタンスに対する pointer-by-value。このインスタンスは、DFSQSETS が言語処理環境呼び出し可能サービスを呼び出すたびに更新されます。 **注:** DFSQSETS からの戻りコードが 998 の場合は、資料「*Language Environment* ランタイム・メッセージ *(SA88-8554-09)*」を使用して、条件トークンの状態の内容を調査し、エラーの原因を判別してください。
@dfs_debug	BIT(1) OPTIONAL	入力	DFSQSETS がトレース情報を表示するかどうかを指定するオプション・ビット (WSDL - PL/I 間セグメンテーション API のトレース出力 (アプリケーション・プログラミング)を参照)。

戻りコード:

DFSQSETS の戻りコードは、DFSPWSH 組み込みファイルで定義されている定数です。

表 *152. DFSQSETS* の戻りコード

タイプ:	名前:	値:
SIGNED FIXED BIN(31)	@dfs_success	000
	@dfs_omitted_parameter	100
	@dfs_invalid_pointer	101
	@dfs_invalid_struct_type	102
	@dfs_struct_not_found	103
	@dfs_struct_name_mismatch	104
	@dfs_invalid_struct_order	105
	@dfs_invalid_segment_size	109
	@dfs_cee_call_failure	998
	@irz_dli_call_failure	999

DFSQSETS の呼び出し例

```
01: /* Invoke API DFSQSETS to set the SOAP body language
02: * structure and commit it to the IMS Message Queue.
03: */
04: @irz_struct_name      = 'ResponseBodyStruct';
05: @irz_struct_ptr       = ResponseBodyStruct_ptr;
06: @dfs_struct_size      = storage(ResponseBodyStruct);
07: @dfs_commit_structs   = '1'b;
```

```
08: @dfs_cee_feedback_ptr = addr(@dfs_cee_feedback);
09: @dfs_debug            = '0'b;
10:
11: @return_code =
12:    DFSQSETS(@dfs_async_msg_header_ptr,
13:    @dfs_iopcb_mask_ptr, @dfs_soap_body_struct,
14:    @dfs_struct_name, @dfs_struct_ptr,
15:    @dfs_struct_size, @dfs_commit_structs,
16:    @dfs_cee_feedback_ptr, @dfs_debug);
17:
18: if (@return_code != @dfs_success) then do;
19:    display('MYMPP#handle_myOperation(): '
20:       || 'ERROR, DFSQSETS @dfs_soap_body_struct, '
21:       || '@return_code: '|| trim(@return_code) || '.');
22:    return;
23: end;
```

DFSXGETS

DFSXGETS API は、IMS Connect 入力バッファーから SOAP 構造をリトリーブし、呼び出し元に対して高級言語で情報を返します。

IMS Connect では XML 変換に IMS メッセージ・キューを使用することができないため、DFSXGETS は言語構造を IMS Connect 入力バッファーからリトリーブします。 IMS Connect 入力バッファーの予想される形式は、IMS メッセージ・セグメント (LLZZDATA) の配列です。

このトピックで参照している構造および変数は、組み込みファイル DFSPWSH で定義されています (665 ページの『インクルード・ファイル DFSPWSH』を参照)。

注**:** この API は、IMS Connect で稼働している PL/I XML コンバーターが使用するためのものです。メッセージ処理プログラム (MPP) では使用されません。

制約事項:

- DFSXGETS は、SOAP 本体および SOAP 障害構造のリトリーブのみをサポートします。 SOAP ヘッダー構造はサポートされません。

パラメーター:

表 *153. DFSXGETS* のパラメーター

パラメーター	タイプ	使用法	説明
@dfs_async_msg_header_ptr	POINTER BYADDR	出力	IMS Connect 入力バッファーで受信された @dfs_async_msg_header のインスタンスに対する pointer-by-reference。
@dfs_icon_buf_ptr	POINTER BYVALUE	入力	IMS Connect 入力メッセージ・バッファーに対する pointer-by-value。予想されるバッファーの形式は LLZZDATA の配列です。

表 153. DFSXGETS のパラメーター (続き)

パラメーター	タイプ	使用法	説明
@dfs_icon_buf_len	SIGNED FIXED BIN(31) BYVALUE	入力	@dfs_icon_buf_ptr が指すバッファーの長さ (バイト数) を指定する integer-by-value。
@dfs_struct_type	SIGNED FIXED BIN(31) BYVALUE	入力	IMS Connect 入力バッファーからリトリーブする構造のタイプを指定する integer-by-value。組み込みファイル DFSPWSH で定義されている定数 @dfs_soap_body_struct を使用することができます。
@dfs_struct_name	WCHAR(100) VARYING BYADDR	入力	IMS Connect 入力バッファーからリトリーブする言語構造の名前が含まれる string-by-reference。このパラメーターの値は、パラメーター @dfs_struct_type の値に対応している必要があります。
@dfs_struct_ptr	POINTER BYADDR	出力	DFSXGETS が、パラメーター @dfs_struct_type および @dfs_struct_name で要求された構造のバイトを含むバッファーのアドレスを書き込む pointer-by-reference。 **重要:** XML コンバーターが実行する言語処理環境エンクレーブが持続するため、このバッファーは、IMS Connect に返される前に XML コンバーターによって解放されなければなりません。
@dfs_struct_size	SIGNED FIXED BIN(31) BYADDR	出力	DFSXGETS がパラメーター @dfs_struct_ptr で返される構造のサイズ (バイト数) を書き込む integer-by-reference。

表 153. DFSXGETS のパラメーター (続き)

パラメーター	タイプ	使用法	説明
@dfs_cee_feedback_ptr	POINTER BYVALUE	入力	言語処理環境条件トークンを定義する @dfs_cee_feedback のインスタンスに対する pointer-by-value。このインスタンスは、DFSXGETS が言語処理環境呼び出し可能サービスを呼び出すたびに更新されます。 **注:** DFSXGETS からの戻りコードが 998 の場合は、資料「*Language Environment* ランタイム・メッセージ *(SA88-8554-09)*」を使用して、条件トークンの状態の内容を調査し、エラーの原因を判別してください。
@dfs_debug	BIT(1) OPTIONAL	入力	DFSXGETS がトレース情報を表示するかどうかを指定するオプション・ビット (WSDL - PL/I 間セグメンテーション API のトレース出力 (アプリケーション・プログラミング)を参照)。

戻りコード:

DFSXGETS の戻りコードは、DFSPWSH 組み込みファイルで定義されている定数です。

表 154. DFSXGETS の戻りコード

タイプ:	名前:	値:
SIGNED FIXED BIN(31)	@dfs_success	000
	@dfs_omitted_parameter	100
	@dfs_invalid_pointer	101
	@dfs_invalid_struct_type	102
	@dfs_struct_not_found	103
	@dfs_struct_name_mismatch	104
	@dfs_invalid_struct_order	105
	@dfs_icon_buf_exhausted	997
	@dfs_cee_call_failure	998

DFSXSETS

DFSXSETS API は、入力データとして受け渡された言語構造の情報から SOAP 構造を作成します。また、指定された場合、DFSXSETS は現行の SOAP 構造および以前に提供されたすべての SOAP 構造を IMS Connect 出力バッファーにコピーします。

IMS Connect では XML 変換に IMS メッセージ・キューを使用することができないため、DFSXSETS は言語構造を IMS Connect 出力バッファーに挿入します。IMS Connect 出力バッファーの形式は、IMS メッセージ・セグメント (LLZZDATA) の配列です。

このトピックで参照している構造および変数は、DFSPWSH で定義されています(665 ページの『インクルード・ファイル DFSPWSH』を参照)。

注**:** この API は、IMS Connect で稼働している PL/I XML コンバーターが使用するためのものです。メッセージ処理プログラム (MPP) では使用されません。

制約事項:

- DFSXSETS は、SOAP ヘッダー構造の保管をサポートしていません。

パラメーター:

表 *155. DFSXSETS* のパラメーター

パラメーター	タイプ	使用法	説明
@dfs_async_msg_header_ptr	POINTER BYVALUE	入力	IMS メッセージの最初のセグメントとして送信される @dfs_async_msg_header のインスタンスに対する pointer-by-value。
@dfs_icon_buf_ptr	POINTER BYVALUE	入力	IMS Connect 出力メッセージ・バッファーに対する pointer-by-value。バッファーの予想される形式は、IMS メッセージ・セグメント (LLZZDATA) の配列です。
@dfs_icon_buf_len	SIGNED FIXED BIN(31) BYVALUE	入力	@dfs_icon_buf_ptr が指すバッファーの長さ (バイト数) を指定する integer-by-value。
@dfs_icon_buf_used	SIGNED FIXED BIN(31) BYADDR	出力	IMS Connect 出力バッファー内で複数セグメント IMS メッセージとして言語構造をフォーマットするために必要なバイト数を DFSXSETS が書き込む integer-by-reference。このパラメーターの値は、必ず、言語構造の実際のサイズより 4 バイト以上大きくする必要があります。
@dfs_struct_type	SIGNED FIXED BIN(31) BYVALUE	入力	IMS Connect 出力バッファーに設定する言語構造のタイプを指定する integer-by-value。組み込みファイル DFSPWSH で定義されている定数 @dfs_soap_body_struct を使用することができます。

表 *155. DFSXSETS* のパラメーター *(*続き*)*

パラメーター	タイプ	使用法	説明
@dfs_struct_name	WCHAR(128) VARYING BYADDR	入力	パラメーター @dfs_struct_type の値に対応する言語構造の名前が含まれる string-by-reference。
@dfs_struct_ptr	POINTER BYVALUE	入力	パラメーター @dfs_struct_type および @dfs_struct_name で指定された構造に対応する構造に対する pointer-by-value。
@dfs_struct_size	SIGNED FIXED BIN(31) BYVALUE	入力	パラメーター @dfs_struct_ptr が指す構造のサイズ (バイト数) を指定する integer-by-value。
@dfs_commit_structs	BIT(1) BYVALUE	入力	DFSXSETS が現行の言語構造および以前に提供されたすべての言語構造を IMS Connect 出力バッファーにコピーするかどうかを指定する bit-by-value。
@dfs_cee_feedback_ptr	POINTER BYVALUE	入力	言語処理環境条件トークンを定義する @dfs_cee_feedback のインスタンスに対する pointer-by-value。このインスタンスは、DFSXSETS が言語処理環境呼び出し可能サービスを呼び出すたびに更新されます。 **注:** DFSXSETS からの戻りコードが 998 の場合は、資料「*Language Environment* ランタイム・メッセージ *(SA88-8554-09)*」を使用して、条件トークンの状態の内容を調査し、エラーの原因を判別してください。
@dfs_debug	BIT(1) OPTIONAL	入力	DFSXSETS がトレース情報を表示するかどうかを指定するオプション・ビット (WSDL - PL/I 間セグメンテーション API のトレース出力 (アプリケーション・プログラミング)を参照)。

戻りコード:

DFSXSETS の戻りコードは、DFSPWSH 組み込みファイルで定義されている定数です。

表 *156. DFSXSETS* の戻りコード

タイプ:	名前:	値:
SIGNED FIXED BIN(31)	@dfs_success	000
	@dfs_omitted_parameter	100
	@dfs_invalid_pointer	101
	@dfs_invalid_struct_type	102
	@dfs_invalid_struct_order	105
	@dfs_invalid_struct_size	106
	@dfs_invalid_struct_name	107
	@dfs_struct_already_set	108
	@dfs_invalid_segment_size	109
	@dfs_icon_buf_exhausted	997
	@dfs_cee_call_failure	998

DFSPWSIO API からの戻りコード

このトピックでは、DFSPWSIO API からの戻りコードについて説明します。

以下の表は、戻りコードについて説明しています。

表 *157. DFSPWSIO API* からの戻りコード

値	**DFSPWSH** 定数	説明
000	@dfs_success	API は正常に完了しました。
100	@dfs_omitted_parameter	必須パラメーターが API に対して指定されませんでした。
101	@dfs_invalid_pointer	API に提供されたポインターの値は、無効なメモリー・アドレスを指定しました。
102	@dfs_invalid_struct_type	API に対して指定された言語構造化タイプは、DFSPWSH.@dfs_soap_header_struct、DFSPWSH.@dfs_soap_body_struct、または DFSPWSH.@dfs_soap_fault_struct のいずれでもありませんでした。
103	@dfs_struct_not_found	指定されたタイプの言語構造が、IMS メッセージで見つかりませんでした。
104	@dfs_struct_name_mismatch	指定されたタイプの言語構造が IMS メッセージで見つかりましたが、指定された名前が一致しませんでした。
105	@dfs_invalid_struct_order	誤った順序での言語構造の取得または設定の試行が検出されました。例えば、IMS メッセージ内に SOAP ヘッダーが存在する場合、SOAP ヘッダー構造を取得する前に SOAP 本体構造の取得を試行するとエラーになります。
106	@dfs_invalid_struct_size	API に対して指定された言語構造のサイズが無効 (0 以下) であったか、最大値を超えていました (DFSPWSH.@dfs_message_max_data を参照)。

表 *157. DFSPWSIO API* からの戻りコード *(*続き*)*

値	**DFSPWSH** 定数	説明
107	@dfs_invalid_struct_name	指定された言語構造名が無効な PL/I ID でした。
108	@dfs_struct_already_set	指定された言語構造化タイプは、既に IMS メッセージに存在しています。
109	@dfs_invalid_segment_size	IMS Connect パラメーター XMPAMAXS で指定されたセグメント・サイズが無効です (5 以下または 32767 以上)。
995	@dfs_fetch_failure	この API は、使用可能なライブラリーから必要なロード・モジュールを取り出すことができませんでした。この戻りコードを受け取る前に、Enterprise PL/I ランタイム・メッセージ IBM0590S のインスタンスが生成されました。
997	@dfs_icon_buf_exhausted	IMS Connect の入力バッファーまたは出力バッファーが使い尽くされたため、API は言語構造の取得あるいは設定ができませんでした。このエラーは、コンパイル済み XML 変換が API を呼び出した場合にのみ発生します。
998	@dfs_cee_call_failure	API は、言語処理環境呼び出し可能サービスの呼び出し時にエラーを検出しました。パラメーター @dfs_cee_feedback_ptr で提供された言語処理環境条件トークンを検査して詳細を確認してください。
999	@dfs_dli_call_failure	API は、CEETDLI インターフェースの呼び出し時にエラーを検出しました。パラメーター @dfs_iopcb_ptr で提供された IOPCB を検査して詳細を確認してください。

以下の表は、各 API で使用される戻りコードを示しています。

表 *158.* 各 *API* で使用される戻りコード

値	**DFSPWSH** 定数	**DFSQGETS**	**DFSQSETS**	**DFSXGETS**	**DFSXSETS**
000	@dfs_success	X	X	X	X
100	@dfs_omitted_parameter	X	X	X	X
101	@dfs_invalid_pointer	X	X	X	X
102	@dfs_invalid_struct_type	X	X	X	X
103	@dfs_struct_not_found	X			
104	@dfs_struct_name_mismatch	X		X	
105	@dfs_invalid_struct_order	X	X	X	X
106	@dfs_invalid_struct_size		X		X
107	@dfs_invalid_struct_name		X		X
108	@dfs_struct_already_set		X		X
109	@dfs_invalid_segment_size		X		X
995	@dfs_fetch_failure	X	X	X	X
997	@dfs_icon_buf_exhausted			X	X
998	@dfs_cee_call_failure	X	X	X	
999	@dfs_dli_call_failure	X	X		X

[1]API がコンパイル済み XML 変換に対して内部的である場合でも、実行時にこれらのエラー・コードがメッセージ IRZ0500S および IRZ0501S に示されます。

関連情報:


IBM0590S および PL/I ランタイム・メッセージ

第 8 章 SQL プログラミングの参照情報

以下のトピックでは、IMS 用の構造化照会言語 (SQL) の参照情報を示します。

IMS での SQL の概念

構造化照会言語 (SQL) を使用する場合、特定の IMS 概念を理解していることが重要です。

構造化照会言語

IMS のデータへのアクセスに使用する言語の 1 つは SQL です。SQL はリレーショナル・データベースのデータ定義および操作するための標準化された言語です。

この言語は SQL ステートメントから構成されています。SQL ステートメントを使用することで、IMS データベース内のデータをリトリーブ、挿入、更新、あるいは削除することが可能になります。

SQL ステートメントをコーディングする場合、何を行いたいかを指定します。どのように行うかではありません。例えば、データにアクセスするのに必要なことは、そのデータを含むセグメントおよびフィールドの名前を指定することだけです。そのデータの入手方法を記述する必要はありません。

データのリレーショナル・モデルに従うと、

- データベースは 1 セットの表として理解されます。
- 関係は、表内の値により表現されます。
- SQL を使用することにより、1 つ以上の表から引き出すことができる結果表 を指定して、データがリトリーブされます。

IMS は、各 SQL ステートメント (すなわち、結果表の指定) を変換して、データのリトリーブおよび変更を行うための一連の操作にします。

すべての実行可能 SQL ステートメントは、準備して初めて実行することができます。

DDL SQL

IMS は、SQL データ定義言語 (DDL) をサポートし、拡張します。

IMS データベースは、階層データベース構造を使用します。標準 SQL DDL ステートメントは、リレーショナル・データベース構造に基づくパラメーター、キーワード、および概念を使用します。 DDL を使用して IMS データベースを作成または変更する場合、IMS データベースの階層構造および関連概念を常に理解する必要があるわけではありませんが、これらを理解することで、より効率が良くパフォーマンスの優れた IMS データベース構造を定義するのに役立つことが多くあります。リレーショナル・データベースの用語および概念を IMS の階層構造にマップする方法について詳しくは、階層データベースとリレーショナル・データベースの比較 (アプリケーション・プログラミング)を参照してください。

関連概念:

DDL を使用したデータベースおよびプログラム・ビューの定義 (データベース管理)

関連資料:

IMS DDL のセキュリティー (データベース管理)

静的 SQL

ソース形式の静的 SQL ステートメントは、COBOL などのホスト言語で書かれたアプリケーション・プログラムに組み込まれます。このステートメントはプログラムの実行前に準備されるもので、操作形式のステートメントはプログラムの実行後も存続します。

IMS バージョン 13 以降では、COBOL については静的 SQL がサポートされません。

動的 SQL

動的 SQL ステートメントを組み込んだプログラムは、静的 SQL を含むプログラムと同様にプリコンパイルしなければなりませんが、動的 SQL ステートメントは、静的 SQL とは違って、実行時に構成され、準備されます。

ソース形式の動的ステートメントは文字ストリングであり、アプリケーション・プログラムから SQL の PREPARE ステートメントを使用して IMS に渡されます。PREPARE ステートメントを使用して準備されたステートメントは、DECLARE CURSOR、DESCRIBE、または EXECUTE のいずれかのステートメント内で参照できます。

対話式 SQL

対話式 *SQL* は、IMS Enterprise Suite Explorer for Development を使用してサブミットされる SQL ステートメントを意味します。

SQL の IMS データ構造

SQL サポートにより、DL/I 呼び出しを使用する代わりに標準の SQL 照会を発行して IMS データにアクセスすることが可能になります。 IMS アプリケーションで SQL 呼び出しを使用するには、IMS データベースの階層モデルと標準的なリレーショナル・データベース・モデルの相違点について理解する必要があります。これは、SQL 呼び出しが、リレーショナル・データベースで使用されることが一般的であるためです。また、IMS データベース・エレメントがどのようにリレーショナル・データベース・エレメントにマップされているかを理解することも必要です。

データベース・セグメント定義では、セグメントのインスタンスにフィールドを定義します。この定義方法は、リレーショナル表で表内の行に列を定義する方法と類似しています。そのため、セグメントはテーブルに関連し、セグメント内のフィールドはテーブル内の列に関連します。データベース内のセグメントのオカレンスは、テーブル内の行に対応します。

次の表は、IMS データベース・エレメントとリレーショナル・データベース・エレメントとの間のマッピングを要約しています。

表 *159. IMS* データベース・エレメントとリレーショナル・データベース・エレメントとの間のマッピング

IMS の階層データベース・エレメント	同等のリレーショナル・データベース・エレメント
セグメント名	テーブル名
セグメント・インスタンス	テーブルの行
セグメント・フィールド名	列名
セグメントのユニーク・キー	テーブルの主キー
外部キー・フィールド	テーブルの外部キー
PCB	スキーマ
セグメント	表
フィールド	桁
レコード	行
データ・セット・グループまたは区域	表スペース

関連タスク:

データベースの設計と実装 (データベース管理)

階層データベースとリレーショナル・データベース

以下は、IMS データベースの階層モデルと標準的なリレーショナル・データベース・モデルの相違点について示しています。

IMS は、階層データベースのリレーショナル・モデルを提供します。用語の 1 対 1 のマッピングに加えて、IMS は、1 次キー制約あるいは外部キー制約全体での階層親子関係も示すことができます。

階層データベース・モデルとリレーショナル・データベース・モデルの比較については、階層データベースとリレーショナル・データベースの比較 (アプリケーション・プログラミング)を参照してください。

言語エレメント

SQL の基本構文および多くの SQL ステートメントに共通の言語エレメントを理解すると、IMS で SQL を使用するために役立ちます。

以下のトピックには、これらの言語エレメントに関する情報が記載されています。

文字

SQL 言語のキーワードと演算子の基本シンボルは文字 です。文字は、次のように英文字、数字、または特殊文字に分類されます。

- 英文字 は、英語のアルファベットの 26 個の大文字 (A から Z) と 26 個の小文字 (a から z) のいずれかです。
- 数字は、0 から 9 までのいずれかです。
- 特殊文字とは、英字または数字以外の任意の文字です。

トークン

SQL 言語の基本的な構文単位を、トークン と呼びます。トークンは 1 つ以上の文字で構成され、どの文字もブランクや、制御文字ではなく、またストリング定数や区切り ID 内の文字ではありません。

トークンは、通常 トークンと区切り トークンに分類されます。

- 通常トークン は、数値定数、通常 ID、ホスト ID、またはキーワードです。

  例:

  ```
  1       .1       +2       SELECT       E       3
  ```

- 区切りトークン は、ストリング定数、区切り ID、演算子記号、または構文図に示されている特殊文字です。疑問符 (?) も、 896 ページの『PREPARE』で説明しているパラメーター・マーカーの役割を果たす場合には、区切りトークンとなります。

  例:

  ```
  ,       'string'       "fld1"       =       .
  ```

スペース

スペースは、1 つ以上のブランク文字の連続です。

大文字と小文字

SQL ステートメント内のトークンには小文字を入れられますが、通常トークン内の小文字は大文字に変換されます。区切りトークンが大文字に変換されることはありません。

例: 以下の二つのステートメントは、大文字変換後は同一です。

```
select * from PCB01.HOSPITAL where hospname = 'Alexandria';
SELECT * FROM PCB01.HOSPITAL WHERE HOSPNAME = 'Alexandria';
```

ID

ID は、名前を形成するのに使われるトークンです。SQL ステートメントの ID は、SQL ID、またはホスト ID です。

SQL ID

SQL ID は、通常 *ID* または区切り *ID* です。

通常 **ID:**

通常 *ID* は、大文字の後に 0 個または 1 個以上の文字が続いたものです。2 文字目以降はそれぞれ、大文字、数字、または下線文字です。

通常 ID は、予約語であってはなりません。予約語を SQL の ID として使用する場合、大文字で指定する必要があり、区切り ID にするか、ホスト変数内で指定することが必要です。

例: 以下は通常 ID の例です。

```
HOSPITAL
```

ホスト ID

ホスト *ID* は、ホスト・プログラムで宣言された名前です。

命名規則

名前を作成する際の規則は、その名前を使って指定するオブジェクトのタイプによって異なります。

構文図では、異なるタイプの名前ごとに別々の用語を使っています。これらの用語の定義は、以下のとおりです。

column-name

表の列を示す修飾名または非修飾名。

修飾列名は、修飾子の後に 1 つのピリオドと SQL ID を続けた形です。修飾子は、表名、ビュー名、シノニム、別名、または相関名です。非修飾列名は SQL ID です。

cursor-name

SQL カーソルを指定する SQL ID。

descriptor-name

SQL 記述子域 (SQLIMSDA) を示すホスト ID。ホスト ID の説明については、 702 ページの『ホスト変数の参照』を参照してください。記述子名に標識変数を入れることはできません。

host-variable

ホスト変数を示すトークンの列。ホスト変数にはホスト ID が少なくとも 1 つは含まれます。この点については、 702 ページの『ホスト変数の参照』で説明します。

statement-name

準備済み SQL ステートメントを指定する SQL ID。

table-name

表 (IMS セグメント) を指定する修飾名または非修飾名。

1 部構成または非修飾の表名は、暗黙的な 2 つの修飾子を持つ SQL ID です。

データ・タイプ

IMS は、SQL からのデータ・タイプをサポートします。

SQL 内で取り扱うことができるデータの最小単位を、値 と呼びます。値がどのように解釈されるかは、そのソースのデータ・タイプによって決まります。 値のソースとしては、以下のものがあります。

- 列
- 定数
- 式
- 変数 (ホスト変数やパラメーター・マーカーなど)

以下の図は、IMS がサポートする組み込みデータ・タイプを示しています。

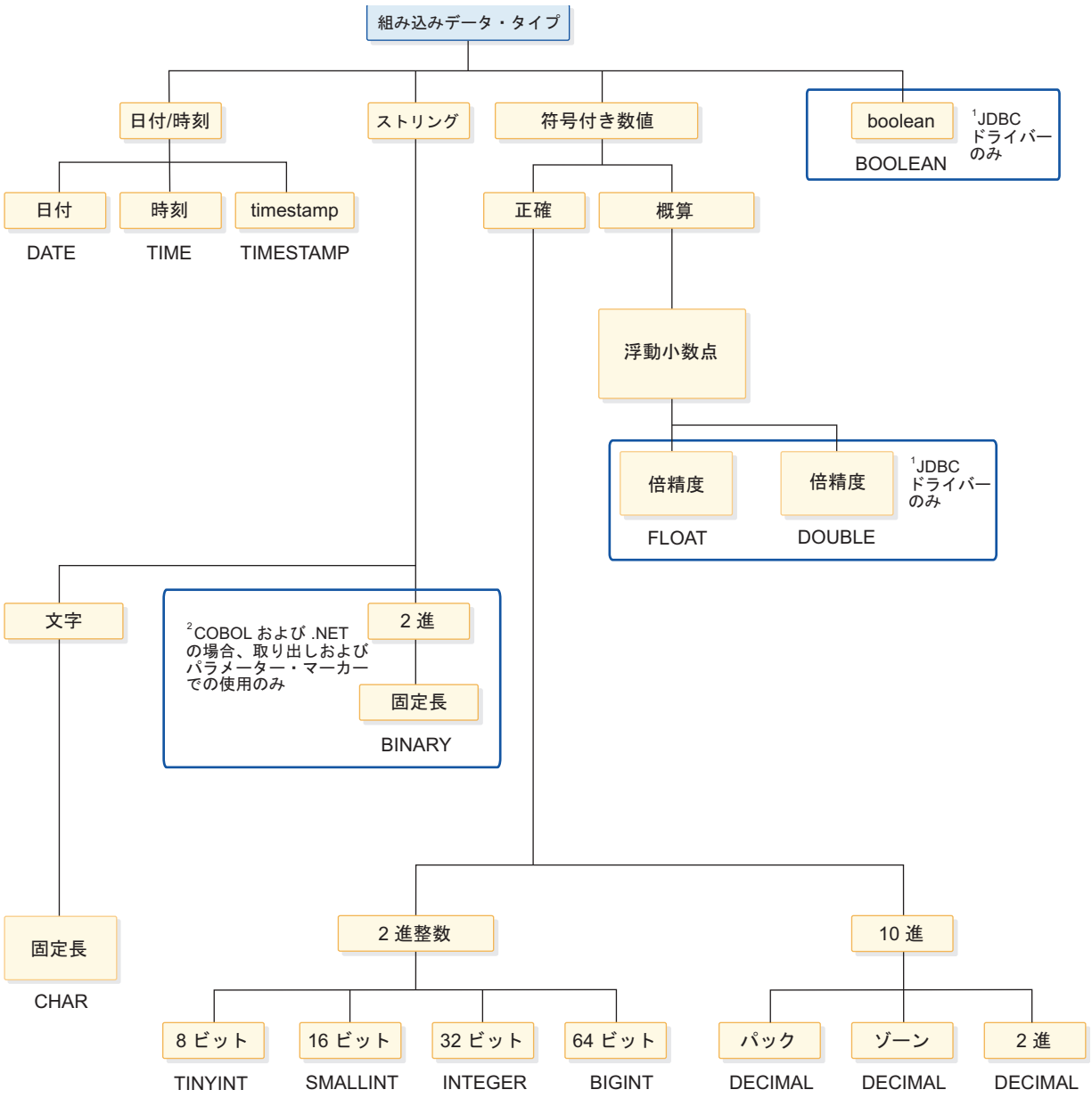


図 *36. IMS* によってサポートされる組み込みデータ・タイプ

重要:

1. BOOLEAN、FLOAT、DOUBLE、および固定長バイナリー・データ・タイプは、Java アプリケーション・プログラムでのみサポートされます。
2. 固定長バイナリー・データ・タイプは、COBOL および .NET アプリケーションの場合、取り出しおよびパラメーター・マーカーでの使用についてのみサポートされます。

関連概念:

JDBC のデータ変換サポート (アプリケーション・プログラミング)

関連資料:

 Java データ型にマップする COBOL コピーブックのタイプ (アプリケーション・プログラミング)

NULL

すべてのデータ・タイプに NULL 値が含まれています。NULL 値は、NULL 以外のあらゆる値から区別される特殊な値であり、(非 NULL の) 値がないことを示します。

IMS は、列を NULL に設定することはサポートしません。ただし、IMS は特定のケースを NULL として解釈します。例えば、フィールドがセグメント境界の外部にあり、特定のマッピングが適用されないマッピングが複数ある可変長セグメントは、NULL として解釈されます。

数値

数値データ・タイプは、厳密な数値 (2 進整数および 10 進数) と近似数値 (浮動小数点) としてカテゴリー化されます。

2 進整数には、短精度整数、長精度整数、および 64 ビット整数があります。 2 進数は整数の厳密な表現です。10 進数は実数の厳密な表現です。2 進数と 10 進数は、厳密な数値タイプと見なされます。浮動小数点数には、倍精度があります。浮動小数点数は実数の近似であり、近似数値タイプと見なされます。

すべての数値が、符号、精度、および位取りを持っています。列の値がゼロの場合、符号は正です。10 進浮動小数点は、ある数値の特殊値、およびさまざまな指数を持つ同じ数値 (例えば、0.0、0.00、0.0E5、1.0、1.00、1.0000) を持ちます。精度は、符号を除く 2 進数または 10 進数の合計桁数です。位取りは、小数点の右側にある 2 進数または 10 進数の合計桁数です。小数点がない場合、位取りはゼロです。

短精度整数 **(SMALLINT):**

短精度整数は、2 バイトを占有する 2 進整数です。短精度整数の範囲は、-32768 から +32767 までです。

長精度整数 **(INTEGER):**

長精度整数は、4 バイトを占有する 2 進整数です。

長精度整数の範囲は、 -2147483648 から +2147483647 までです。

64 ビット整数 **(BIGINT):**

64 ビット整数は、8 バイトを占有する 2 進整数です。

64 ビット整数の範囲は -9223372036854775808 から +9223372036854775807 です。

倍精度浮動小数点 **(DOUBLE** または **FLOAT):**

倍精度浮動小数点 数は、長 (64 ビット) 浮動小数点数です。

SQL for COBOL では、DOUBLE および FLOAT はサポートされません。

倍精度浮動小数点数の範囲は約 -7.2E+75 ～ 7.2E+75 です。この範囲で、最大の負の値は約 -5.4E-79 であり、最小の正の値は約 5.4E-079 になります。

10 進数**:**

10 進 数は、暗黙の小数点を持つパック 10 進数です。

小数点の位置は、数値の精度および位取りによって決まります。位取り (数値の小数部分の桁数) は、負の値あるいは精度より大きな値であってはなりません。最大精度は 31 桁です。

10 進列の値は、精度と位取りがすべて同じになります。10 進変数の範囲または 10 進列内の数値の範囲は、*-n* から *+n* です。*n* は、該当する精度と位取りで表すことのできる最大の正の値です。最大範囲は $1 - 10^{31}$ から $10^{31} - 1$ です。

数値ホスト変数**:**

2 進数値データの COBOL 形式は、USAGE BINARY です。 BINARY、COMP、および COMP-4 は同義語です。2 進数形式の数値は、2、4、または 8 バイトの記憶領域を占有します。

COBOL では、COMP-1 は短精度浮動小数点形式を指し、COMP-2 は長精度浮動小数点形式を指します。これらの形式は、それぞれ 4 バイトと 8 バイトの記憶領域を占有します。

COBOL では、10 進数は以下のフォーマットで表すことができます。

- パック 10 進数フォーマット (USAGE PACKED-DECIMAL または COMP-3 で示す)
- 外部 10 進フォーマット (USAGE DISPLAY と SIGN LEADING SEPARATE で示す)

文字ストリング

文字ストリング はバイトのシーケンスです。ストリングの長さはシーケンス内のバイト数です。長さがゼロの場合の値を、空ストリング と呼びます。空ストリングと NULL 値とを混同しないでください。

固定長文字ストリング**:**

固定長文字ストリング、列、および変数を定義する場合、長さ属性が指定され、すべての値が同じ長さになります。固定長文字ストリングの長さ属性の範囲は 1 から 255 です。

日時値

日時値は、ストリングでも数値でもありません。しかし、日時値は、特定の算術演算やストリング演算に使用することができ、特定のストリングとの互換性があります。

さらに、ストリングで日時値を表すこともできます。

日付:

日付 は、グレゴリオ暦を使用して 1 つの時点を示す 3 部構成の値 (年、月、日) です。グレゴリオ暦は西暦 1 年から有効であったものと想定しています。

年の部分の範囲は 0001 から 9999 までです。月の部分の範囲は 1 から 12 までです。日の部分の範囲は、月および年によって、1 から 28、29、30、または 31 までです。[1]

日付の内部表現は 4 バイトのストリングです。各バイトは、2 つのパック 10 進数から構成されます。最初の 2 バイトが年、3 バイト目が月、最後のバイトが日を表します。

文字ストリング表現の実際の長さは、255 バイト以下であることが必要です。

時間:

時刻 は、24 時間クロックを使って時刻を示す 3 部構成の値 (時、分、秒) です。時の部分の範囲は 0 から 24 までです。分および秒の部分の範囲は 0 から 59 までです。24 時の場合、分と秒の部分は両方ともゼロです。

時刻の内部表現は 3 バイトのストリングです。各バイトは、2 つのパック 10 進数から構成されます。1 バイト目が時、2 バイト目が分、最後のバイトが秒を表します。

文字ストリング表現の実際の長さは、255 バイト以下であることが必要です。

タイム・スタンプ:

タイム・スタンプ は、6 つの部分か 7 つの部分で構成される値 (年、月、日、時、分、秒、およびオプションの端数秒) と、オプションのタイム・ゾーン指定であり、日時を表します。

Java では、タイム・スタンプはタイプ java.sql.Timestamp にマップされます。

COBOL では、タイム・スタンプはアプリケーション定義の長さを持つ文字ストリングです。

日時ホスト変数:

日付、時刻、およびタイム・スタンプの値を格納するためには、通常は文字ストリング・ホスト変数が使用されます。

割り当てと比較

SQL の基本演算は割り当ておよび比較です。

割り当て演算は、INSERT および UPDATE の各ステートメントの実行時に行われます。さらに、関数またはストアード・プロシージャーが呼び出されると、関数ま

1. 歴史上の日付は、必ずしもグレゴリオ暦に従っていません。1582-10-04 から 1582-10-15 までの日付は、グレゴリオ暦には存在していませんが、有効な日付として受け入れられます。

たはストアード・プロシージャーの引数が割り当てられます。比較演算は、述部や ORDER BY などのその他の言語エレメントを含むステートメントの実行時に行われます。

この両方の演算の基本規則は、オペランドのデータ・タイプに互換性がなければならないということです。

下の表は、割り当ておよび比較におけるデータ・タイプの互換性を示します。

表 *160.* 割り当てと比較におけるデータ・タイプの互換性

オペランド	BYTES	SHORT	INT	LONG	DOUBLE	BIT	CHAR	PACKED	ZONED	日付	TIME	FLOAT	TIMESTAMP
BYTES	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
SHORT	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
INT	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
LONG	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
DOUBLE	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
BIT	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
CHAR	あり	あり	あり	あり	あり	あり	あり	あり	あり	あり	あり	あり	あり
PACKED	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
ZONED	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
日付	なし	なし	なし	なし	なし	なし	あり	なし	なし	あり	あり	なし	あり
TIME	なし	なし	なし	なし	なし	なし	あり	なし	なし	あり	あり	なし	あり
FLOAT	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし	なし	あり	なし
TIMESTAMP	なし	なし	なし	なし	なし	なし	あり	なし	なし	あり	あり	なし	あり

注**:**

1. LOB およびビット・データはサポートされません。
2. 日時を示す値の互換性は、割り当ておよび比較に限定されます。
   - 日時値は、ストリング列とストリング変数に割り当てることができます。
   - 日付の有効なストリング表現は、日付の列に割り当てることも、日付と比較することもできます。
   - 時刻の有効なストリング表現は、時刻の列に割り当てることも、時刻と比較することもできます。
   - タイム・スタンプの有効なストリング表現は、タイム・スタンプの列に割り当てることも、タイム・スタンプと比較することもできます。
3. 文字ストリングは、XML 列に割り当てることができます。

ストリングの割り当て

ストリング割り当てには、ストレージ割り当てと取り出し割り当ての 2 つのタイプがあります。

- ストレージ割り当ては、値を列に割り当てる場合です。
- 取り出し割り当て は、値を変数に割り当てる場合です。

規則は、ストレージ割り当てと取り出し割り当てで異なります。

文字ストリングの割り当て**:**

ソースと宛先の両方がストリングである場合、ストレージ割り当ておよび取り出し割り当ての規則が適用されます。

ストレージ割り当て*:*

文字のストレージ割り当ての基本的な規則は、関数の列またはパラメーターに割り当てるストリングの長さが、その列またはパラメーターの長さ属性を超えてはならないということです。

末尾ブランクはストリングの長さに含まれます。ストリングの長さが列またはパラメーターの長さ属性を超えている場合、以下のアクションがとられます。

- ストリングをターゲットに収めるために切り捨てる必要がある末尾の文字がすべてブランクであり、ストリングが文字ストリングまたはグラフィック・ストリングである場合、警告なしでストリングは切り捨てられ、割り当てられます。
- 他の場合、ストリングは割り当てられず、エラーが発生して、超過文字の少なくとも 1 つが非ブランクであることを示します。

固定長の列またはパラメーターにストリングを割り当てる場合に、ストリングの長さがターゲットの長さ属性より短い時は、ストリングの右側に、ブランクが必要な数だけ埋め込まれます。

取り出し割り当て*:*

COBOL では、ホスト変数に割り当てるストリングの長さは、変数の長さ属性を超えてもかまいません。ストリングの長さが変数の長さを超えている場合、ストリングの右側の文字が必要な数だけ切り捨てられます。

切り捨てが行われると、SQLIMSCA の SQLIMSWARN1 フィールドに W という値が割り当てられます。

固定長の変数に文字ストリングを割り当てる場合に、ストリングの長さがターゲットの長さ属性より短い時は、ストリングの右側に、ブランクが必要な数だけ埋め込まれます。

ストリング比較

ストリング比較は、バイナリー・ストリング、文字ストリングに関して行われる可能性があります。

文字ストリングの比較**:**

2 つのストリングの比較は、各ストリングの対応するバイトを比較することによって行います。ストリングの長さが同じでない場合、短い方のストリングには、その長さがもう一方のストリングと同じになるように右側に一時的にブランクを埋め込んでから、比較を行います。

2 つのストリングは、両方とも空の場合、あるいは対応するバイトがすべて等しい場合に等しくなります。空ストリングはブランクのストリングと同じです。2 つの

ストリングが等しくない場合、その関係 (すなわち、どちらが大きい値を持つか) は、ストリングの左端から見て等しくない最初のバイト同士を比較することによって判定されます。

定数

定数 (リテラル とも呼ばれる) は、値を指定します。定数を分類すると、ストリング定数と数値定数がある。 数値定数はさらに、整数、浮動小数点数、10 進数、または 10 進浮動小数点として分類されます。ストリング定数は、文字、またはバイナリーとして分類されます。

すべての定数が NOT NULL 属性を持っています。10 進浮動小数点定数を除いて、値がゼロの数値定数では、負の記号は無視されます。

整定数

整定数 は、整数を、小数点を含まない最大 19 桁の符号付きまたは符号なし数値として指定します。

値が長精度整数の範囲内にある場合、その整定数のデータ・タイプは長精度整数です。値が長精度整数の範囲外であるが、64 ビット整数の範囲内にある場合、その整定数のデータ・タイプは 64 ビット整数です。64 ビット整数値の範囲外に定義されている定数は、10 進定数と見なされます。

例:

```
64     -15     +100     32767     720176
```

構文図で使用している整数 という用語は、符号を含むことのできない長精度整数定数を表しています。

浮動小数点定数

浮動小数点定数 は、2 つの数値を E で分けた形式で倍精度の浮動小数点数を指定します。

最初の数値には、符号と小数点を入れることができます。2 番目の数値には、符号は入れられますが、小数点は入れられません。定数の値は、最初の数値と、2 番目の数値で指定された 10 の累乗値との積です。この値は、浮動小数点数の範囲内になければなりません。定数の中の文字数は、30 を超えてはなりません。先行ゼロを除いて、最初の数値の桁数は、17 桁を超えてはなりません。また 2 番目の数値の桁数は 2 桁を超えてはなりません。

例: 次の浮動小数点定数は、それぞれ「150」、「200000」、「-0.22」、「500」という数値を表します。

```
15E1     2.E5    -2.2E-1     +5.E+2
```

10 進定数

10 進定数 は、小数点を含むか、あるいは 2 進整数の範囲内にない、31 桁以下の符号付きまたは符号なしの数値です。

精度は総桁数であり、小数点より右の桁数があれば、これも含めたものとなります。総桁数には、先行ゼロと後続ゼロもすべて含まれます。位取りは、後続ゼロを含めた小数点の右側の桁数です。

例: 次の 10 進定数の精度の位取りは、それぞれ 5 と 2、4 と 0、2 と 0、23 と 2 です。

```
  025.50   1000.   -15.   +375893333333333333333.33
```

文字ストリング定数

文字ストリング定数 は可変長文字ストリングを指定します。文字ストリング定数には 1 つの形式があります。

- ストリング区切り文字で始まり、ストリング区切り文字で終わる文字シーケンス。

例:

```
'10/14/2013'    '32'    'DON''T CHANGE'        ''
```

この例の右端のストリング ('') は、空の文字ストリング定数を表します。これは、長さがゼロのストリングです。

COBOL の場合、文字ストリング EBCDIC 037 のみがサポートされます。

フィールド名

フィールド名の意味は、そのコンテキストによって変わります。

フィールド名は以下の目的で使うことができます。

- フィールドを識別する。
- フィールドの値を指定する (以下のコンテキストで使う場合など)。
  - *ORDER BY* 文節では、フィールド名は、その文節が適用される中間結果表内のすべての値を指定します。例えば、ORDER BY HOSPNAME は、HOSPNAME フィールドの値によって中間結果表を配列します。
  - 検索条件では、フィールド名は、その構成が適用される各行またはグループの値を指定します。例えば、ある行に対して検索条件 CODE=20 が適用された場合にフィールド名 CODE によって指定される値は、その行の CODE フィールドの値です。
- フィールドの名前を一時的に変更するための式のフィールド名を指定します (*select-clause* の AS 文節の場合など)。

修飾フィールド名

フィールド名の修飾子はセグメント名です。

修飾子の指定が任意のところでは、修飾子は 2 つの目的を果すことができます。詳しくは、 702 ページの『あいまいさを防ぐためのフィールド名修飾子』および を参照してください。

あいまいさを防ぐためのフィールド名修飾子

ORDER BY 文節、式、または検索条件のコンテキストでは、フィールド名は、DELETE ステートメントまたは UPDATE ステートメント、または FROM 文節の *table-reference* 内のいくつかのセグメントまたはビューのフィールドの値を参照します。

フィールド名を修飾する理由の 1 つは、そのフィールドがどのオブジェクトにあるフィールドなのかを示すことです。

表指定子*:* 特定のオブジェクト表を指定する修飾子を表指定子 と呼びます。オブジェクト表を識別する文節は、オブジェクト表の表指定子も設定します。例えば、SELECT ステートメント内の式のオブジェクト表は、次のステートメントのように、SELECT ステートメントの次にくる FROM 文節で指定します。

```
SELECT Z.HOSPCODE, WARDNO, WARDLL
FROM PCB01.HOSPITAL Z, PCB01.WARD
WHERE Z.HOSPNAME = 'ALEXANDRIA'
AND Z.HOSPCODE = 'R1210010000A'
```

FROM 文節内の表指定子は、次のように設定されます。

- 表名またはビュー名の後に続く名前は、相関名と表指定子の両方です。したがって、Z は表指定子であり、選択リストの最初の列名を修飾しています。
- 直接的な表名またはビュー名は、表指定子です。したがって、修飾された表名 PCB01.WARD は、は表指定子であり、選択リストの 2 番目の列名を修飾しています。

変数の参照

SQL ステートメント内の変数 は、SQL ステートメントの実行時に変更できる値を指定します。SQL ステートメント内で使用できる変数には、いくつかのタイプがあります。

ホスト変数
: ホスト変数は、ホスト言語のステートメントにより定義されます。ホスト変数を参照する方法について詳しくは、『ホスト変数の参照』を参照してください。

パラメーター・マーカー
: パラメーター・マーカーは、動的に準備された SQL ステートメント内でホスト変数の代わりに使用されます。パラメーター・マーカーについて詳しくは、PREPARE ステートメントのパラメーター・マーカーを参照してください。

特に断りのない限り、構文図の用語 *host variable* (ホスト変数) は、ホスト変数、またはパラメーター・マーカーを使用できる個所を示すために使用されます。

ホスト変数の参照

ホスト変数は、ホスト言語のステートメントによって直接定義されます。 SQL ステートメント内の *host-variable* は、ホスト変数を宣言する際の規則に従ってプログラム内で記述されたホスト変数を指定する必要があります。ホスト変数は、動的 SQL ステートメント内では参照できず、パラメーター・マーカーを代わりに使用する必要があります。

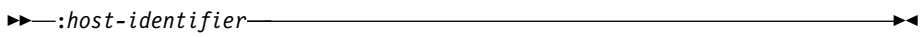
ホスト変数は、COBOL ではデータ・エレメントです。

SQL ステートメント内のホスト変数は、ホスト変数を宣言する際の規則に従ってプログラム内で記述されたホスト変数を識別するものでなければなりません。

構文図などで使っているホスト変数 という用語は、ホスト変数に対する参照を示します。FETCH ステートメントの INTO 文節では、ホスト変数は、IMS によって値が割り当てられる出力変数です。また、ホスト変数は、IMS に値を提供する入力変数である場合もあります。

変数参照

ホスト変数参照の一般形式は、以下のとおりです。



各ホスト ID は、ソース・プログラム内で宣言されている必要があります。

ホスト変数を参照する SQL ステートメントは、それらのホスト変数の宣言の有効範囲内になければなりません。カーソルの SELECT ステートメント内で参照されるホスト変数の場合は、OPEN ステートメントと DECLARE CURSOR ステートメントが同じ有効範囲内にあることが必要です。

ホスト変数へのすべての参照には、その前にコロンを付ける必要があります。先行するコロンがないホスト変数を SQL ステートメントが参照した場合、コプロセッサーはコロンの欠落に関するエラーを出します。あるいは、ホスト変数を非修飾の列名として解釈して、予測しない結果を生じる可能性があります。コロンがないホスト変数を列名として解釈することは、列名を参照できるコンテキストでホスト変数を参照した場合に発生します。

関連概念:

『動的 SQL 内のホスト変数』

動的 SQL 内のホスト変数

動的 SQL ステートメントでは、ホスト変数の代わりにパラメーター・マーカーを使用します。

パラメーター・マーカー は、動的 SQL ステートメント内での位置を表す疑問符 (?) です。この場所に、アプリケーションが値を提供します。

```
INSERT INTO PCB01.DOCTOR (hospital_HOSPCODE, patient_patnum, ward_wardno, doctno,docname) VALUES (?,?,?,?,?)
```

パラメーター・マーカーの置換:

準備済みステートメントが実行される前に、そのステートメント内の各パラメーター・マーカーは実際に、それに対応するホスト変数に置き換えられます。置換は、ソースがホスト変数の値であり、ターゲットが変数である割り当て演算です。割り当て規則は、割り当てと比較 (アプリケーション・プログラミング API)で列への割り当てに関して説明した規則です。

COBOL でのホスト構造

ホスト構造 は、SQL ステートメントで参照される COBOL グループです。

ここで使用しているホスト構造 という用語には、SQLIMSCA または SQIMSLDA は含まれません。

ホスト構造参照の形式は、ホスト変数参照の形式と同じです。S1 がホスト構造を指定する場合、:S1 という参照はホスト構造参照になります。

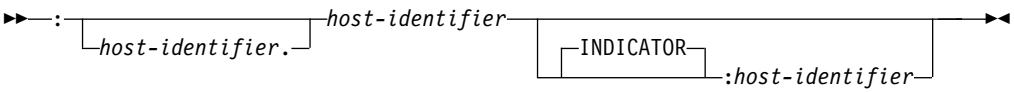
ホスト構造は、ホスト変数のリストを参照することができるコンテキストであれば、どこでも参照できます。ホスト構造参照は、ホスト言語の構造宣言内に定義された順序でその構造内に含まれるそれぞれのホスト変数に対する参照と同等です。標識配列の *n* 番目の変数は、ホスト構造の *n* 番目の変数の標識変数です。

例えば、*V1*、*V2*、および *V3* を構造 S1 内の変数として宣言した場合、次の 2 つのステートメントは同等です。

```
EXEC SQLIMS FETCH CURSOR1 INTO :S1;
EXEC SQLIMS FETCH CURSOR1 INTO :V1, :V2, :V3;
```

構造参照の他に、COBOL の個々のホスト変数も修飾名によって参照することができます。修飾された形式は、ホスト ID の後に、ピリオドと別のホスト ID を続けた形です。最初のホスト ID は構造を指定する必要があります。また、2 番目のホスト ID は、その構造内の次のレベルのホスト変数を指定する必要があります。

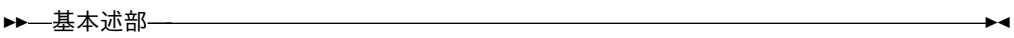
COBOL でのホスト変数 の構文は、以下のとおりです。



述部

述部 とは、特定の行またはグループに対して「真」、「偽」、または「不明」の条件を指定するものです。

いずれのタイプの述部にも、以下の規則が適用されます。



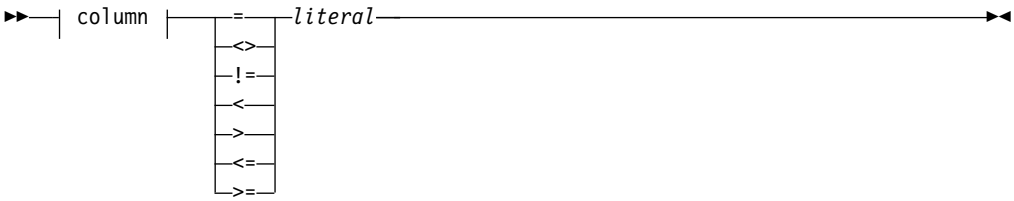
いずれのタイプの述部にも、以下の規則が適用されます。

- 同じ述部内で指定する値は、すべて互換であることが必要です。

基本述部

基本述部 は、2 つの値を比較するか、またはある 1 組の値を別の 1 組の値と比較します。

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。

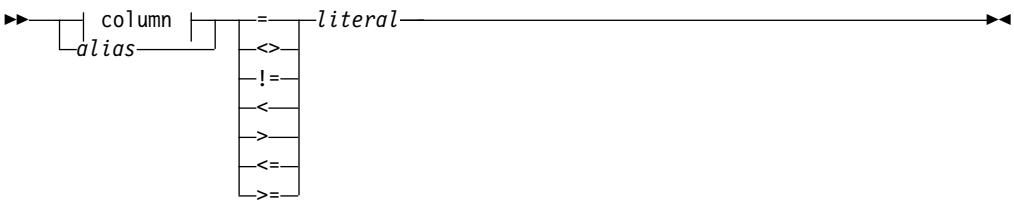


列:

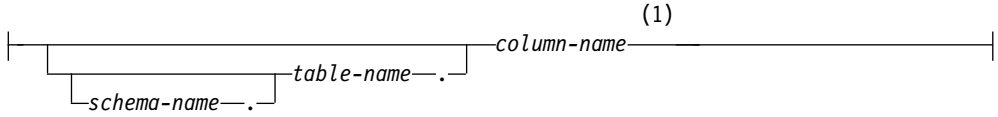
注:

1 複数のテーブルに同じ列名を使用できますが、テーブルが修飾されていない場合、各テーブルで列を検索する必要があります。

このステートメントがサポートされるのは、Java アプリケーション・プログラムにおいてのみです。



列:



注:

1 複数のテーブルに同じ列名を使用できますが、テーブルが修飾されていない場合、各テーブルで列を検索する必要があります。

述部の結果は、次の 2 つのケースに示すように、演算子によって異なります。

- 演算子が = の場合、述部の結果は次のとおりです。
  - 対応する値式の対がすべて真と評価される場合は、真。
  - 対応する値式のいずれか 1 対が偽と評価される場合は、偽。
- 演算子が <> の場合、述部 (x1,x2,...,xn) <> (y1,y2,...,yn) の結果は次のとおりです。
  - いずれかの i 値に対して xi=yi が偽と評価される (つまり、相互に等しくない NULL 以外の xi と yi のペアが少なくとも 1 つある) 場合に限り、真。
  - すべての i 値に対して xi=yi が真と評価される (つまり、(x1,x2,...,xn)=(y1,y2,...,yn) が真) 場合に限り、偽。

表 *161. x* と *y* という値の場合

述部	以下の場合にのみ真となる
x = y	*x* は *y* に等しい。
x <> y	*x* は *y* に等しくない。
x < y	*x* は *y* より小さい。
x > y	*x* は *y* より大きい。
x <= y	*x* は *y* より小さいか等しい。
x >= y	*x* は *y* より大きいか等しい。

値 *x* と *y* の例*:*

```
HOSPCODE = '528671'
XINTEGER < 20000
HOSPNAME <> :VAR1
```

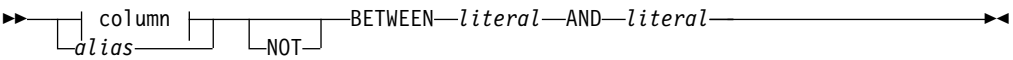
例*:* HOSPITAL セグメントから病院コードと病院名をリストします。ここでは、病院コードは H5140070000H です。

```
SELECT HOSPCODE, HOSPNAME
   FROM PCB01.HOSPITAL
   WHERE HOSPCODE = 'H5140070000H'
```

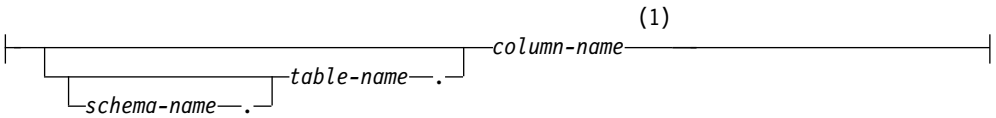
BETWEEN 述部

BETWEEN 述部は、指定した値が、昇順で指定した別の 2 つの値の間にあるかどうかを判別します。

BETWEEN 述部は、Java アプリケーション・プログラムでのみサポートされます。



列:



注**:**

1 複数のテーブルに同じ列名を使用できますが、テーブルが修飾されていない場合、各テーブルで列を検索する必要があります。

この述部の 2 つの形式はそれぞれ、下の表に示すように、等価の検索条件を持っています。

表 *162. BETWEEN* 述部および等価の検索条件

BETWEEN 述部	同等の検索条件
column1 BETWEEN *value1* AND *value2*	*column1* >= *value1* AND *columnn1* <= *value2*
column1 NOT BETWEEN *value1* AND *value2*	*column1* < *value1* OR *column1* > *value2*

検索条件については、 708 ページの『検索条件』で説明しています。

オペランドに日時値と日時値の有効なストリング表示の混合が含まれている場合、すべての値は日時オペランドのデータ・タイプに変換されます。

例: 以下の述部について考えます。

```
A BETWEEN B AND C
```

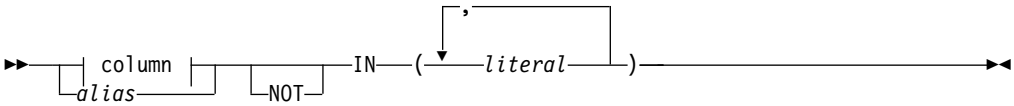
次の表は、*A*、*B*、および *C* のさまざまな値に対する述部の値を示しています。

A の値	*B* の値	*C* の値	述部の値
1、2、または 3	1	3	真
0 または 4	1	3	偽
ヌル	任意の値	任意の値	偽

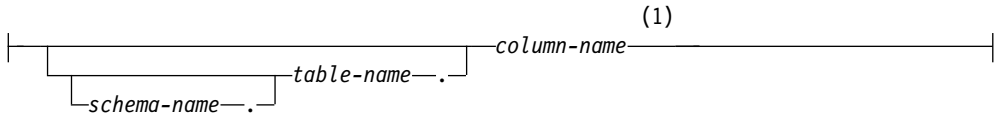
IN 述部

IN 述部は、1 つ以上の値を値のセットと比較します。

IN 述部は、Java アプリケーション・プログラムでのみサポートされます。



列:



注:

1 複数のテーブルに同じ列名を使用できますが、テーブルが修飾されていない場合、各テーブルで列を検索する必要があります。

IN 述部は、以下の比較述部に相当します。

表 *163. IN* 述部と等価の比較述部

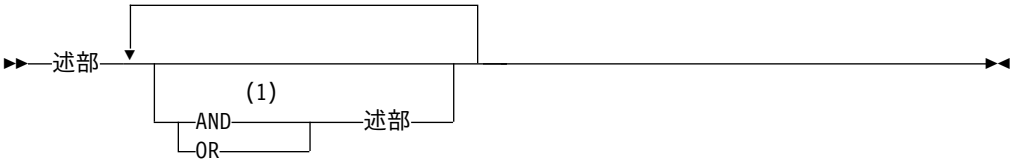
IN 述部	等価の比較述部
column1 IN (*value1*, *value2*, *valuen*)	*column1* = *value1* or *column1* = *value2* or *column1* = *valuen*
column1 NOT IN (*value1*, *value2*, *valuen*)	*column1* <> *value1* and *column1* <> *value2* and *column1* <> *valuen*

例: 次の述部は、部門 D11、B01、または C01 に属している従業員の行については真です。

```
WORKDEPT IN ('D11', 'B01', 'C01')
```

検索条件

検索条件 では、指定された行またはグループに関して、真または偽である条件を指定します。条件が真の場合、その行またはグループは結果に含まれる資格を持ちます。条件が偽または不明の場合、その行またはグループには資格がありません。



注:

1 別のテーブルとの間では、述部を AND で接続する必要があります。

説明

検索条件の結果は、指定したそれぞれの述部に指定した論理演算子 (AND、OR) を適用することによって得られます。論理演算子を指定しなければ、検索条件の結果は、指定した述部の結果となります。

AND と OR の定義は、次の表のとおりです。この表において、*P* と *Q* は任意の述部を表します。

表 *164. AND* および *OR* の真理値表

P	*Q*	*P* および *Q*	*P* または *Q*
真	真	真	真
真	偽	偽	真
偽	真	偽	真
偽	偽	偽	偽

括弧内の検索条件が先に評価されます。同じ順序レベルの演算子が評価される順序は、検索条件の最適化ができるように未定義となっています。

例

次の検索条件では、AND が先に適用されます。AND が適用された後、OR 演算子が適用されます。この 2 つの OR はどちらを先に適用しても結果が変わることはありません。したがって、IMS は OR 演算子を適用する順序を選ぶことができます。

```
PATNUM > ? AND AGE > ? OR HOSPCODE = ? OR HOSPNAME = ?
```

***COBOL** のみ:*

```
PATNUM>:VAR1 AND AGE>:VAR2 OR HOSPCODE=:VAR3 OR HOSPNAME=:VAR4
```

関連概念:

704 ページの『述部』

SQL ステートメント

このセクションでは、SQL ステートメントの構文図、セマンティックの説明、規則、および使用例を示します。

COBOL での SQL サポートは、IBM IMS Data Provider for Microsoft .NET の基礎となる SQL 関数を提供します。 COBOL アプリケーション・プログラムおよびその構文とルールでサポートされるすべての SQL ステートメントは、.NET アプリケーションにも適用されます。

表 *165. SQL* ステートメント

SQL ステートメント	機能	サポートされるアプリケーション・プログラム・タイプ
714 ページの『ALTER DATABASE』	既存のデータベースを変更する。	Java
764 ページの『ALTER TABLESPACE』	データベース内のデータ・セット・グループ、または DEDB のエリアの属性を変更する。	Java
729 ページの『ALTER TABLE』	データベース内の表の属性を変更する。	Java
772 ページの『CLOSE』	カーソルをクローズする。	COBOL、.NET
773 ページの『COMMENT ON』	リソースまたはオブジェクトの定義にコメントを追加する。	Java
776 ページの『CREATE DATABASE』	新規データベースを IMS に定義する。	Java
791 ページの『CREATE PROGRAMVIEW』	新規プログラム・ビューを定義する。	Java
812 ページの『CREATE TABLE』	新規表を定義する。	Java
862 ページの『CREATE TABLESPACE』	データ・セット・グループまたは高速機能エリアを定義する。	Java
878 ページの『DECLARE CURSOR』	SQL カーソルを定義する。	COBOL、.NET
879 ページの『DECLARE STATEMENT』	準備済み SQL ステートメントを識別するために用いる名前を宣言する。	COBOL、.NET
880 ページの『DELETE』	表から 1 つ以上の行を削除する。	COBOL、.NET、Java
881 ページの『DESCRIBE OUTPUT』	準備済みステートメントの結果列を記述する。	COBOL、.NET
882 ページの『DROP DATABASE』	IMS からデータベースを削除する。	Java
883 ページの『DROP PROGRAMVIEW』	プログラム・ビューを削除する。	Java

表 *165. SQL* ステートメント (続き)

SQL ステートメント	機能	サポートされるアプリケーション・プログラム・タイプ
884 ページの『DROP TABLE』	データベースから既存の表を削除する。	Java
885 ページの『DROP TABLESPACE』	データベース内のデータ・セット・グループ、または DEDB のエリアを削除する。	Java
886 ページの『EXECUTE』	準備済み SQL ステートメントを実行する。	COBOL、.NET
887 ページの『FETCH』	カーソルを位置付ける、データを戻す、またはカーソルの位置付けとデータを戻すことの両方を行う。	COBOL、.NET
889 ページの『INCLUDE』	ソース・プログラムに宣言を挿入する。	COBOL、.NET
890 ページの『INSERT』	表に 1 つ以上の行を挿入する。	COBOL、.NET、Java
894 ページの『OPEN』	カーソルをオープンする。	COBOL、.NET
896 ページの『PREPARE』	SQL ステートメント (オプショナル・パラメーターを含む) の実行準備を行う。	COBOL、.NET
899 ページの『SELECT』	カーソルの SELECT ステートメントを指定します。	COBOL、.NET、Java
909 ページの『UPDATE』	表の 1 つ以上の行にある 1 つ以上の列の値を更新する。	COBOL、.NET、Java
913 ページの『WHENEVER』	SQL 戻りコードに基づいて取るべき処置を定義する。	COBOL、.NET

SQL ステートメントの呼び出し方法

SQL ステートメントを呼び出す方法は、ステートメントが実行可能ステートメントであるか実行不能ステートメントであるか、あるいは *select-statement* であるかによって異なります。

SQL ステートメントは、実行可能 と実行不能 に分類されます。各ステートメントの説明の呼び出しに関する見出しのもとに、そのステートメントが実行可能であるかどうかを示してあります。

実行可能ステートメント は、以下の方法で呼び出すことができます。

- アプリケーション・プログラムで動的に準備および実行する
- 対話式に実行する

実行不能ステートメント は、アプリケーション・プログラムに組み込むことのみが可能です。

アプリケーション・プログラムでの SQL ステートメントの使用

SQL ステートメントは、ソース・プログラムに組み込むことができ、IMS コプロセッサーに対して実行依頼されます。 SQL ステートメントを入れることができる場所は、アプリケーション・プログラムの中でホスト言語ステートメントが使用できるところであれば、どこにでも構いません。それぞれのステートメントの前に、そのステートメントが SQL ステートメントであることを示すキーワード (複数語の場合もある) を付ける必要があります。

- COBOL の場合は、それぞれの SQL ステートメントの前にキーワード EXEC SQLIMS を付ける必要があります。

実行可能ステートメント*:* アプリケーション・プログラム内の実行可能 SQL ステートメントは、同じ場所にホスト言語のステートメントが指定されている場合、そのホスト言語ステートメントが実行されるたびに実行されます。 (したがって、例えば、ループ内のステートメントはループが実行されるたびに実行され、条件構成内のステートメントは条件が満たされるときにのみ実行されます。)

SQL ステートメントには、ホスト変数に対する参照を含めることができます。この方法で参照されるホスト変数は、以下の 2 つのいずれかの方法で使用することができます。

入力として
: ステートメントの実行において、ホスト変数の現行値が使用される。

出力として
: ステートメントを実行した結果として、変数に新しい値が割り当てられる。

特に、述部内のホスト変数に対するすべての参照は、実際には変数の現行値に置き換えられます。すなわち、変数が入力として使用されます。他の参照の取り扱いについては、ステートメントごとに個々に解説しています。

ステートメントの実行が成功したか失敗したかは、組み込み SQLIMSCA で SQLIMSCODE および SQLIMSSTATE フィールドを設定することで示されます。したがって、すべての実行可能ステートメントの後で、SQLIMSCODE または SQLIMSSTATE をテストしなければなりません。この代わりに、WHENEVER ステートメント (これ自体は実行不能) を使って、SQL ステートメントの実行直後に制御のフローを変えることもできます。

実行不能ステートメント*:* 実行不能のステートメントは、コプロセッサーによってのみ処理されます。コプロセッサーは、ステートメントで検出されたエラーを報告します。このステートメントが実行されることは決してなく、アプリケーション・プログラムの実行可能ステートメントの中に置かれた場合、「ノーオペレーション」として作用します。したがって、そのようなステートメントの後で SQL 戻りコードをテストしないでください。

動的準備と実行

ユーザーのアプリケーション・プログラムは、文字ストリングをホスト変数に入れた形で、SQL ステートメントを動的に作成することができます。

このように作成されたステートメントは、(組み込み) ステートメント PREPARE によって実行のために準備され、(組み込み) ステートメント EXECUTE によって実行することができます。

準備するステートメントには、ホスト変数に対する参照を含めてはなりません。代わりに、パラメーター・マーカーを入れることができます。(パラメーター・マーカーに関する規則については、PREPARE ステートメントの解説の中のパラメーター・マーカーの項を参照してください。) 準備済みステートメントを実行する場合、パラメーター・マーカーは、 EXECUTE ステートメントで指定したホスト変数の現在値によって、事実上置き換えられます。(この置換に関する規則については、EXECUTE ステートメントを参照してください。) ステートメントの準備が終わった後は、ホスト変数の異なる値を使って複数回実行することができます。

ステートメントの実行が成功したか失敗したかは、EXECUTE (または EXECUTE IMMEDIATE) ステートメントの後で、SQLIMSCA 内の SQLIMSCODE フィールドおよび SQLIMSSTATE フィールドに戻される値により示されます。組み込みステートメントのところで前述したように、これらのフィールドの検査が必要になります。

SELECT ステートメントの動的呼び出し

ユーザーのアプリケーション・プログラムは、文字ストリングをホスト変数に入れた形で、SELECT ステートメントを動的に作成することができます。

このように作成されたステートメントは、(組み込み) ステートメント PREPARE を使って実行準備をして、(実行不能) ステートメント DECLARE CURSOR によって参照することができます。このステートメントは、(組み込み) ステートメント OPEN を使ってカーソルをオープンするたびに実行されます。カーソルがオープンされた後は、(組み込み) SQL FETCH ステートメントを連続して実行し、結果表を一度に 1 行ずつ取り出していくことができます。

この方法で使用する SELECT ステートメントには、ホスト変数に対する参照を含めてはなりません。代わりに、パラメーター・マーカーを入れることができます。(パラメーター・マーカーに関する規則については、 896 ページの『PREPARE』の「注」を参照してください。) パラメーター・マーカーは、OPEN ステートメントで指定したホスト変数の値によって事実上置き換えられます。 (この置換に関する規則については、 894 ページの『OPEN』を参照してください。)

SELECT ステートメントの実行が成功したか失敗したかは、OPEN の後で SQLIMSCA 内の SQLIMSCODE と SQLIMSSTATE のフィールドに戻される値により示されます。組み込みステートメントのところで前述したように、これらのフィールドの検査が必要になります。

ホスト言語アプリケーションのエラー条件および警告条件の検出と処理

ホスト言語アプリケーションのエラー条件および警告条件は、SQLIMSCODE ホスト変数または SQLIMSSTATE ホスト変数、あるいは SQLIMSCA を使用して検査することができます。

それぞれのホスト言語は、診断情報を処理するための手段を提供しています。

COBOL では、実行可能 SQL ステートメントを含むアプリケーション・プログラムは、*SQLIMSCODE* という名前のスタンドアロン整変数を提供する必要があります。

SQLIMSSTATE:

IMS は、各 SQL ステートメントの実行後に SQLIMSSTATE を設定します。 IMS は、SQL 標準のエラー仕様に準拠した値を戻します。したがって、アプリケーション・プログラムは、SQLIMSCODE の代わりに SQLIMSSTATE をテストすることにより、SQL ステートメントの実行を検査できます。

SQLIMSSTATE は、共通のエラー状態については共通のコードをアプリケーション・プログラムに提供します (SQLIMSSTATE の値が製品固有の値となるのは、その製品固有のエラーまたは警告の場合だけです)。さらに、SQLIMSSTATE は、アプリケーション・プログラムが特定のエラーまたはエラーのクラスについてテストできるように設計されています。

LOOP ステートメントの場合は、LOOP ステートメントの END LOOP の部分が完了した後、SQLIMSSTATE が設定されます。REPEAT ステートメントを使用すると、REPEAT ステートメントの UNTIL および END REPEAT の部分が完了した後、SQLIMSSTATE が設定されます。

SQLIMSCODE:

SQLIMSCODE は、各 SQL ステートメントの実行後に IMS によっても設定されます。

IMS は、以下のように SQL 標準に準拠しています。

- SQLIMSCODE = 0 で SQLIMSWARN0 がブランクの場合、実行は成功しました。
- SQLIMSCODE = 100 の場合は、「データなし」が検出されました。例えば、カーソルが結果表の最後の行の後ろに位置していたために、FETCH ステートメントがデータを戻さなかった場合などです。
- SQLIMSCODE > 0 であり 100 ではない場合、実行は成功しましたが警告が出されました。
- SQLIMSCODE = 0 で SQLIMSWARN0 = 'W' の場合、実行は成功しましたが警告が出されました。
- SQLIMSCODE < 0 の場合、実行は失敗です。

LOOP ステートメントの場合は、LOOP ステートメントの END LOOP の部分が完了した後、SQLIMSSTATE が設定されます。REPEAT ステートメントを使用すると、REPEAT ステートメントの UNTIL および END REPEAT の部分が完了した後、SQLIMSSTATE が設定されます。

SQL 標準は、SQLIMSCODE のその他の特定の正または負の値の意味を定義していないため、これらの値の意味は製品固有です。

SQLIMSERRM:

SQLIMSERRM は、エラー・メッセージと長さを含む各 SQL ステートメントの後に IMS によって設定される可変長文字ストリング・セットです。

COBOL の場合は、SQLIMSERRM に SQLIMSERRML および SQLIMSERRMC が含まれます。SQLIMSERRMC には、IMS によって返される SQL エラー・メッセージが含まれます。最大 255 文字まで可能です。 SQLIMSERRML は、SQL エラー・メッセージの長さです。

関連資料:

SQL コード (メッセージおよびコード)

ALTER DATABASE

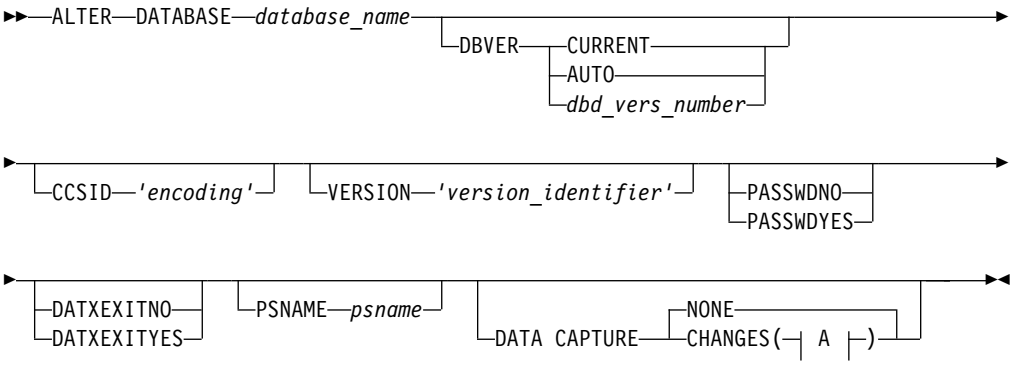
ALTER DATABASE ステートメントを使用して、既存のデータベースを変更できます。CREATE DATABASE ステートメントとは異なり、デフォルト値はありません。

呼び出し

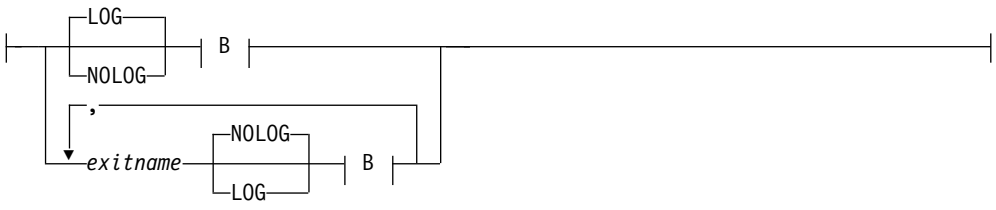
このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

- 『PHIDAM 構文』
- 715 ページの『HDAM 構文』
- 716 ページの『HIDAM 構文』
- 716 ページの『PHDAM 構文』
- 717 ページの『GSAM 構文』
- 717 ページの『HISAM 構文』
- 718 ページの『SHISAM 構文』
- 719 ページの『DEDB 構文』
- 719 ページの『HSAM 構文』
- 720 ページの『SHSAM 構文』
- 720 ページの『LOGICAL 構文』
- 720 ページの『INDEX 構文』
- 720 ページの『PSINDEX 構文』

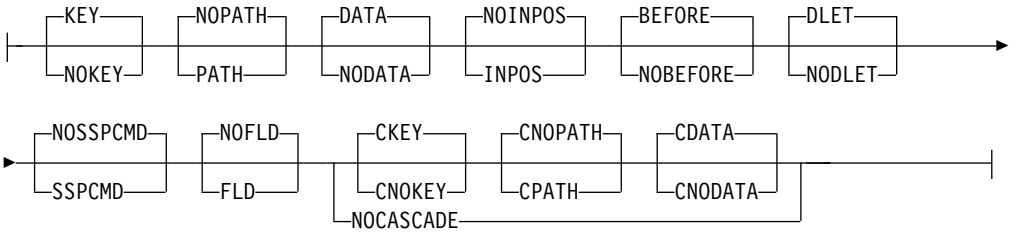
PHIDAM 構文



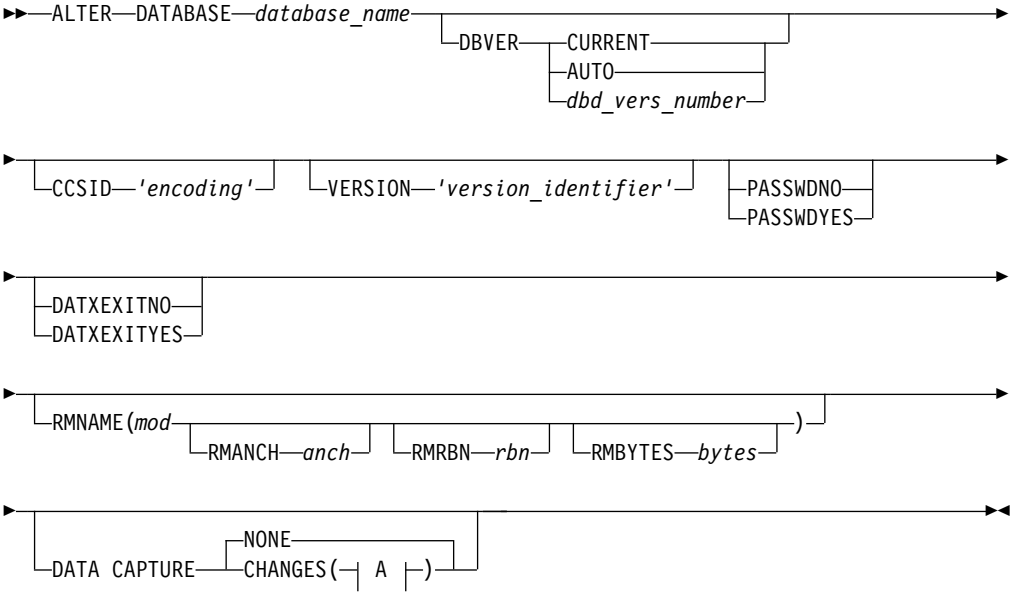
A:


LOG
NOLOG
B
,
exitname
NOLOG
LOG
B

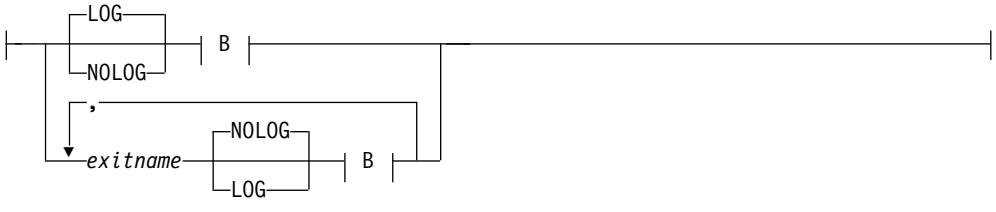
B:


KEY
NOKEY
NOPATH
PATH
DATA
NODATA
NOINPOS
INPOS
BEFORE
NOBEFORE
DLET
NODLET
NOSSPCMD
SSPCMD
NOFLD
FLD
CKEY
CNOKEY
CNOPATH
CPATH
CDATA
CNODATA
NOCASCADE

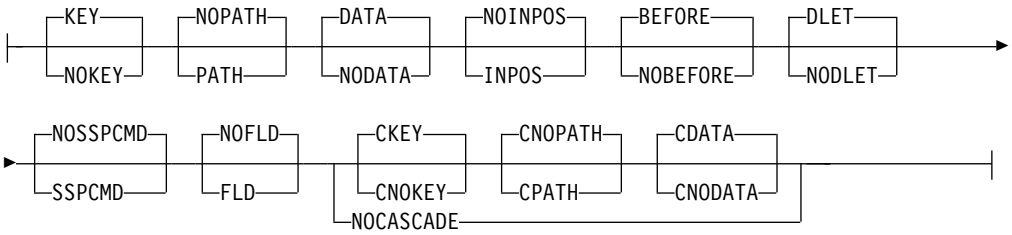
HDAM 構文


ALTER
DATABASE
database_name
DBVER
CURRENT
AUTO
dbd_vers_number
CCSID
'encoding'
VERSION
'version_identifier'
PASSWDNO
PASSWDYES
DATXEXITNO
DATXEXITYES
RMNAME(mod
RMANCH
anch
RMRBN
rbn
RMBYTES
bytes
)
DATA CAPTURE
NONE
CHANGES(
A
)

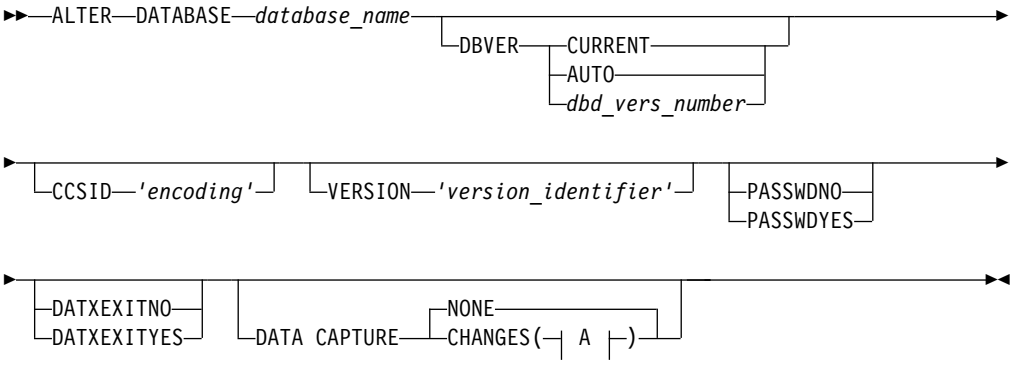
A:


LOG
NOLOG
B
,
exitname
NOLOG
LOG
B

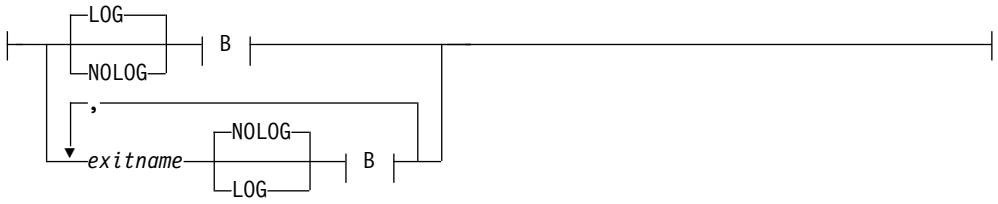
B:



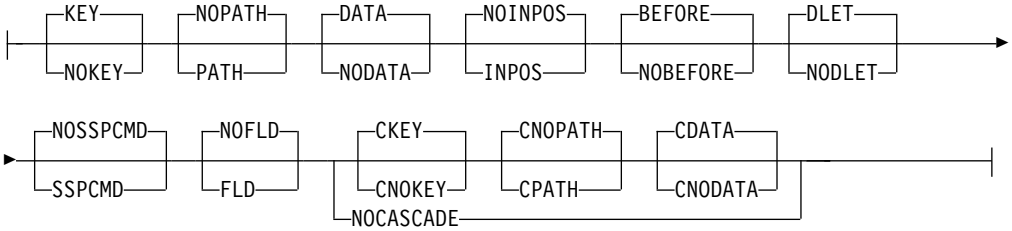
HIDAM 構文



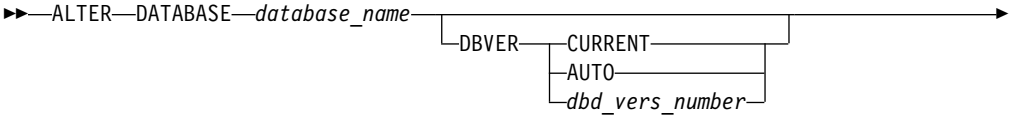
A:

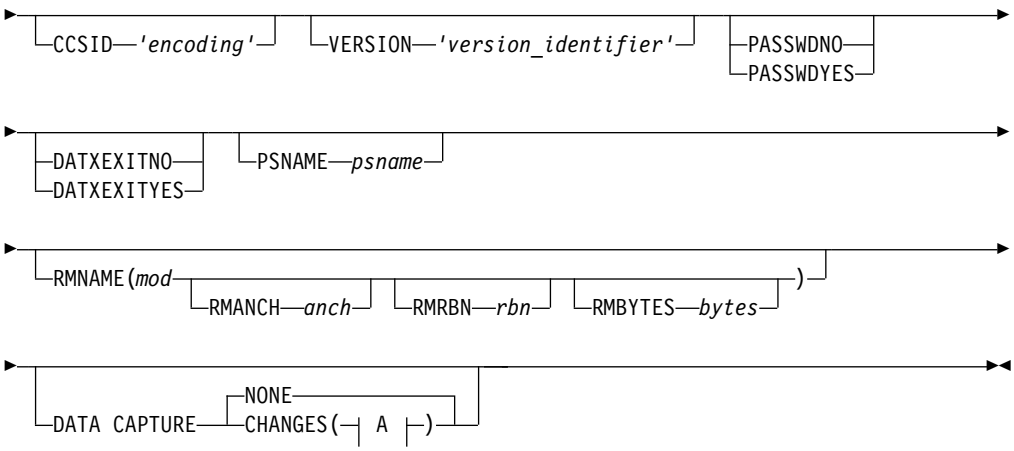


B:

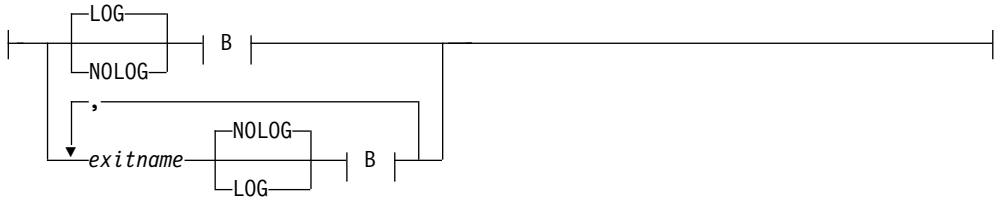


PHDAM 構文

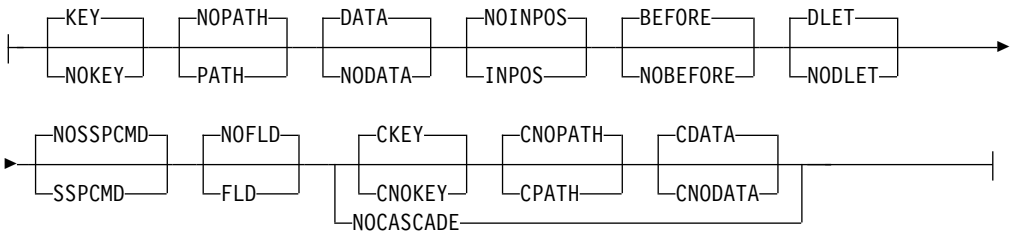




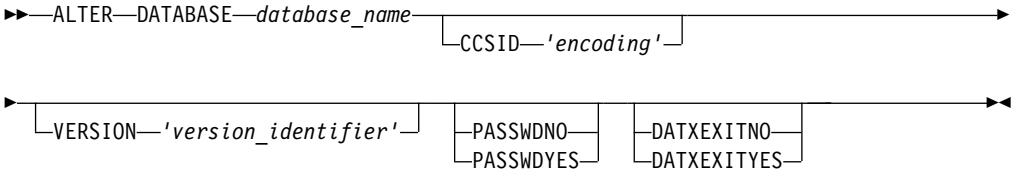
A:



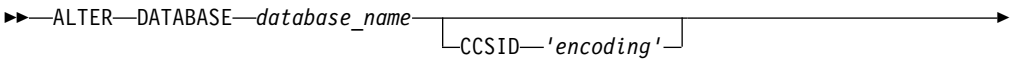
B:

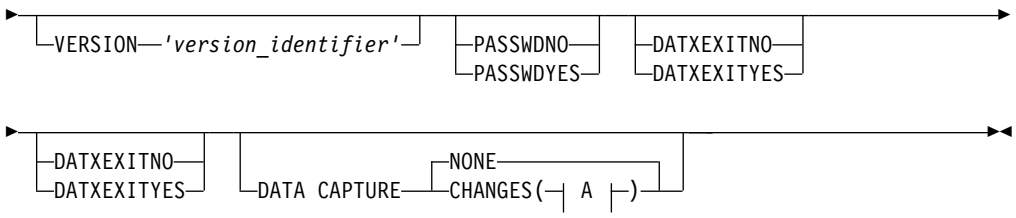


GSAM 構文

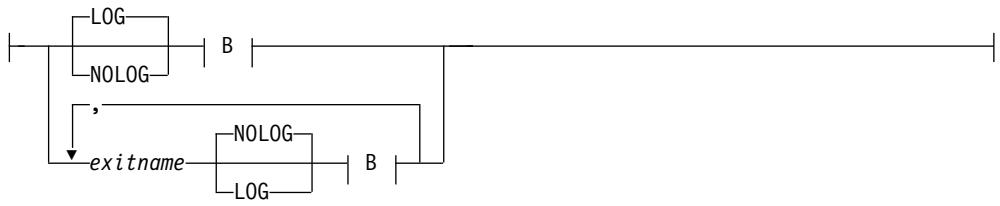


HISAM 構文

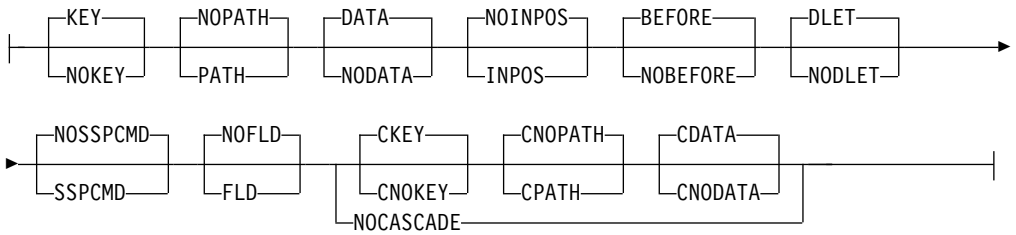




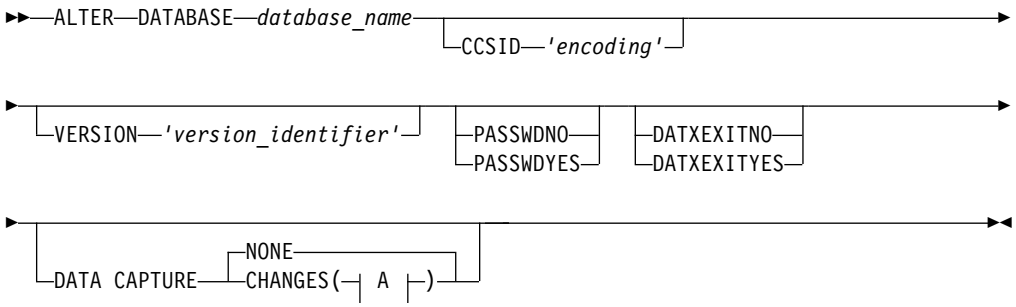
A:



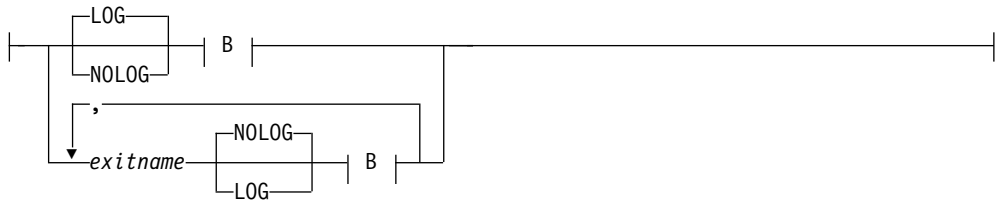
B:



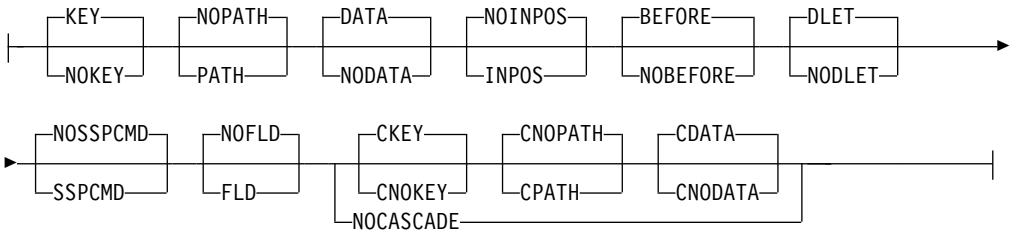
SHISAM 構文



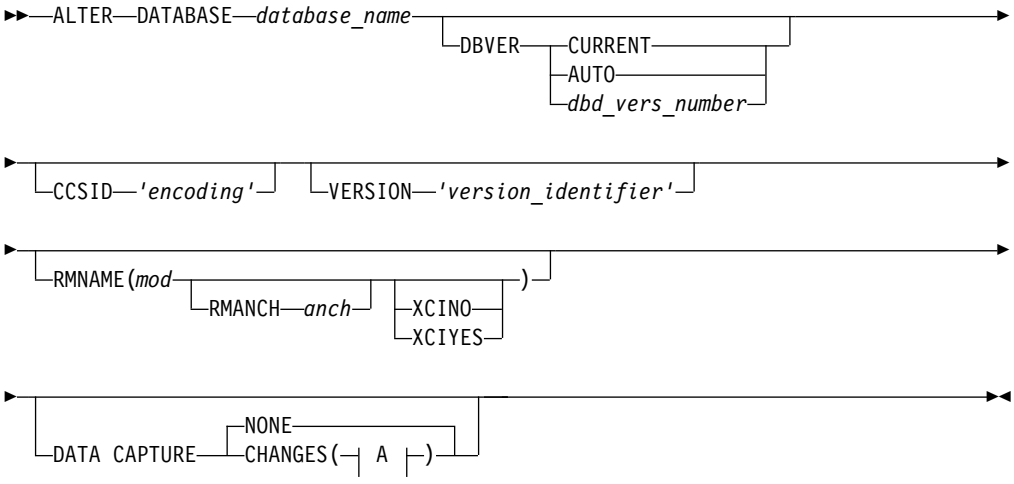
A:



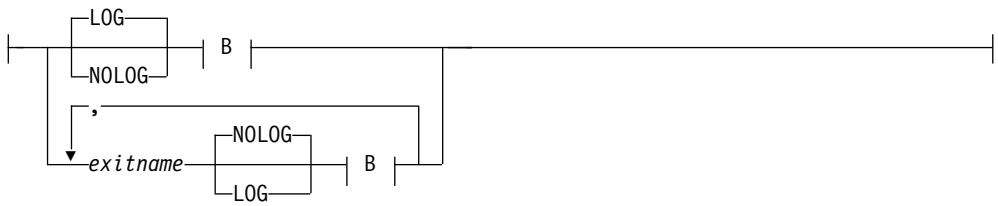
B:



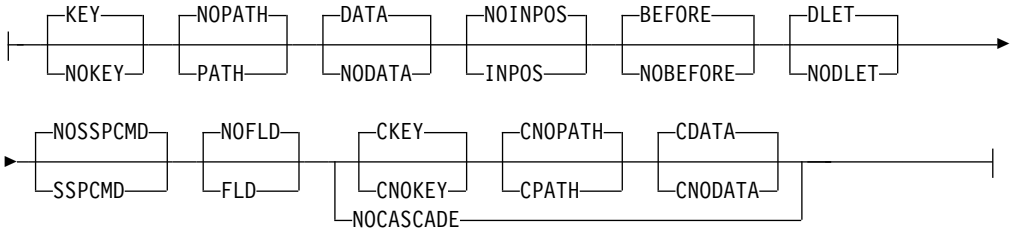
DEDB 構文



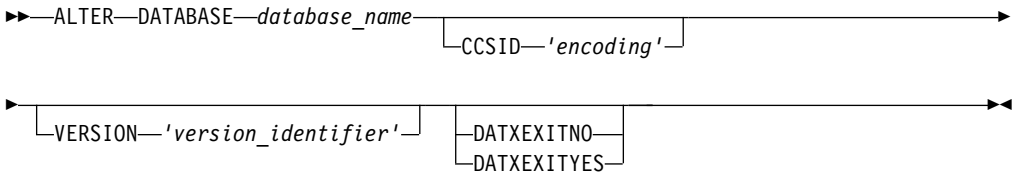
A:



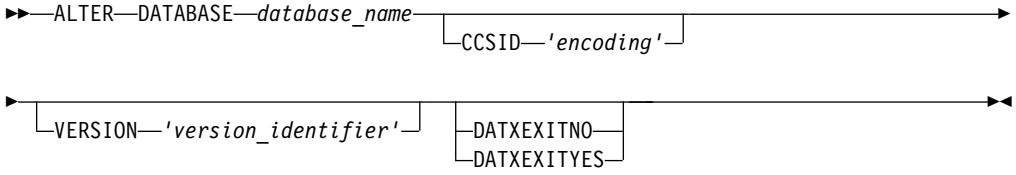
B:



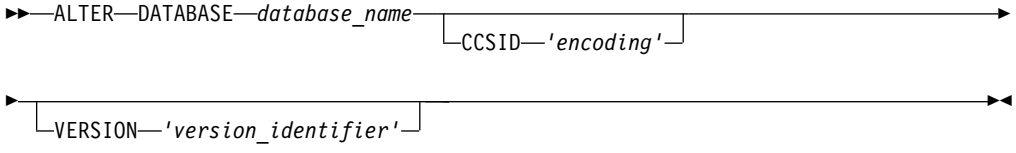
HSAM 構文



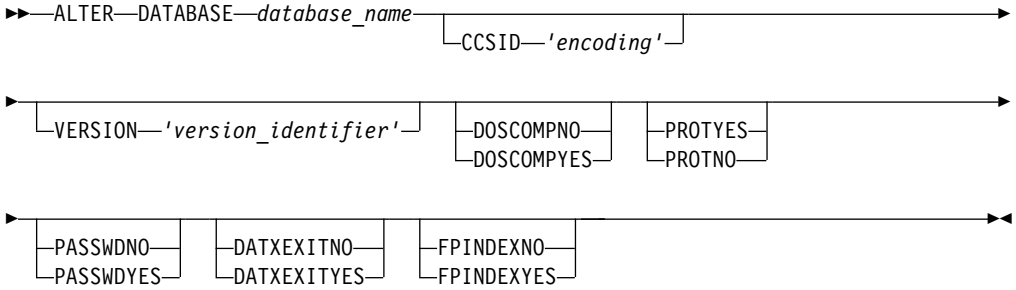
SHSAM 構文



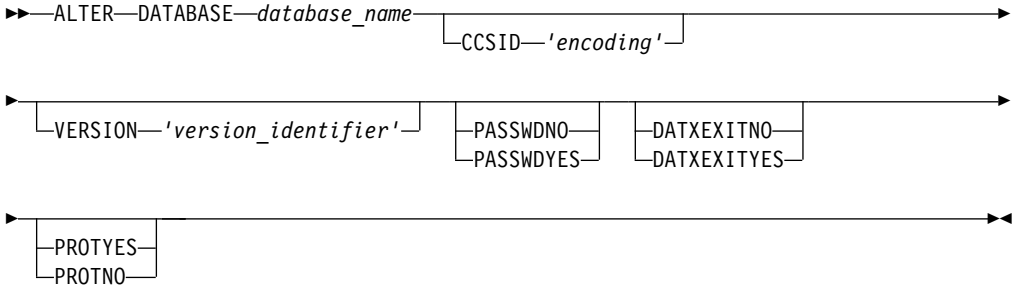
LOGICAL 構文



INDEX 構文



PSINDEX 構文



説明

ALTER DATABASE ステートメントには、以下のキーワード・パラメーターが定義されています。

database_name

変更するデータベースの名前。名前は、1 文字から 8 文字の英数字で指定することができます。

CCSID *'encoding'*

このデータベース内のすべての文字データのデフォルト・エンコードを指定する、1 文字から 25 文字のオプション・フィールド。 CCSID は、メタデータとしてカタログに保管されます。 OpenDatabase/JDBC ドライバーは、このメタデータを使用して、適切なエンコード・タイプによって結果セットを準備します。

この値に以下の文字を含むことはできません。

- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

この値は、個々のセグメントまたはフィールドでオーバーライドできます。

DATA CAPTURE

CREATE DATABASE ステートメントで DATA CAPTURE を指定すると、これらのオプションは物理データベース内のすべてのテーブルに適用されます。このパラメーターを CREATE ステートメントまたは ALTER TABLE ステートメントで指定した場合、このステートメントの指定はオーバーライドされます。

以下の物理データベースは DATA CAPTURE をサポートします。

- HISAM
- SHISAM
- HDAM
- PHDAM
- HIDAM
- PHIDAM
- DEDB

NONE

データ・キャプチャー・オプションがないことを示します。

CHANGES

任意の数の出口ルーチンを、各ルーチン独自の変更オプションのセットと一緒に指定できます。出口ルーチンを指定しない場合、ロギングに関する 1 セットの変更オプションのみを指定できます。このメソッドは、DBD マクロ・ステートメントの EXIT= パラメーターで出口ルーチン名の代わりにアスタリスク (*) を指定することと同じです。各セットは、コンマで分離します。NOCASCADE は、任意の組み合わせの C* (例えば、CKEY) オプションと相互に排他的です。

以下のオプションは DATA CAPTURE CHANGES に有効です。

BEFORE | NOBEFORE
REPL 呼び出しの場合に、変更前データが X'99' ログ・レコードに組み込まれます。BEFORE がデフォルトです。この属性は、DEDB の場合のみ有効です。

CDATA | CNODATA
カスケード削除の場合に、セグメント・データを出口ルーチンに渡します。また、CDATA は削除されるセグメントを識別します (物理連結キーで識別できない場合)。この属性は、NOCASCADE とは相互に排他的です。

CKEY | CNOKEY
物理連結キーを出口に渡します。このキーは、カスケード削除で削除されるセグメントを識別します。この属性は、NOCASCADE とは相互に排他的です。

CNOPATH | CPATH
出口ルーチンが物理ルートの階層パスにあるセグメント・データを必要としないことを示します。CNODATA は、カスケード削除に必要な相当量のパス・データを除去するために使用します。この属性は、NOCASCADE とは相互に排他的です。

DATA | NODATA
DATA は、更新用に物理テーブル・データが出口ルーチンに渡されることを指定します。DATA が指定され、EDITPROC 出口ルーチンもテーブルで使用されている場合、渡されるデータは拡張されたデータです。 DATA がデフォルトです。

DLET | NODLET
DLET 呼び出しの場合に、X'99' ログ・レコードが書き込まれます。DLET がデフォルトです。この属性は、DEDB の場合のみ有効です。

exitname
データを処理する出口ルーチンの名前を指定します。この名前は、ユーザーが IMS に対して定義したデータ・キャプチャー出口ルーチンの名前に一致する必要があります。最大 8 文字の英数字を使用できます。

KEY | NOKEY
KEY は、出口ルーチンに物理連結キーを渡すことを指定します。このキーは、アプリケーションによって更新される物理テーブルを識別します。 KEY がデフォルトです。

NOCASCADE
DL/I がこのセグメントを削除するときに、出口ルーチンは呼び出されないことを示します。従属セグメントを持たないセグメントを削除する場合には、カスケード削除は不要です。

NOFLD | FLD
FLD オプションは、DEDB FLD 呼び出しによって行われる更新をキャプチャーするよう要求します。このオプションは DEDB に対してのみ有効で、この情報は、オプション・ログが指定された場合に X'9904' ロ

グ・レコードにのみ記録されます。この情報はデータ・キャプチャー出口には渡されません。この属性は、DEDB の場合のみ有効です。

NOINPOS | INPOS
INPOS オプションは、HERE の挿入規則が使用されて F や L のコマンド・コードが使用されていないときに、ISRT がキーなしセグメントまたは一意でないキー付きセグメントに対して行われた場合に、ツイン・データを渡すよう要求します。ツイン・データ IMS は ISRT がキャプチャーされる前の時点に位置指定されます。

NOLOG | LOG
LOG オプションは、データ・キャプチャーの制御ブロックとデータを IMS システム・ログに書き込むよう要求します。

NOPATH | PATH
NOPATH は、出口ルーチンが物理ルートの階層パスにあるテーブルからデータを必要としないことを示します。NOPATH は、パス・データの検索に必要な処理時間を回避するための効率的な方法です。
NOPATH がデフォルトです。

PATH は、更新されたセグメントのために物理ルートの階層パスにある各セグメントからのデータを、出口ルーチンに渡す必要がある場合に指定できます。アプリケーションが、挿入、置換、または削除の目的で複数のセグメントを別々にアクセスできるようにするには、PATH を使用します。

DB2® for z/OS の 1 次キーを構成するためにパス内のテーブルからの情報が必要なときは、PATH オプションを使用できます。その場合、DB2 for z/OS の 1 次キーは、従属テーブルの更新のための伝搬要求で使用されます。一般に、この種のテーブル情報が必要になるのは、親がキー情報を含んでおり、従属テーブルが親テーブルには収まらない追加データを含んでいる場合です。

PATH は、追加処理が必要な場合にも使用できます。例えば、D コマンド・コードを呼び出さなかった場合などに、1 つの呼び出しで複数のテーブルにアクセスしないということがあります。その場合、アプリケーションが別々の呼び出しで各テーブルにアクセスする際、追加処理が必要になります。

NOSSPCMD | SSPCMD
SSPCMD オプションは、DEDB サブセット・ポインター・コマンド・コードをキャプチャーするよう要求します。このオプションは DEDB の場合にのみ有効です。

DATXEXITNO | DATXEXITYES

このデータベースを処理中に、アプリケーションがデータ変換ユーザー出口ルーチン (DFSDBUX1) を使用できるようにします。

YES を指定すると、各データベース呼び出しの始めと終わりにユーザー出口 DFSDBUX1 が呼び出されます。DFSDBUX1 がロードされない場合、IMODULE が呼び出されてこれをロードします。

NO を指定した場合でも、DFSDBUX1 が SDFSRESL にあれば、ユーザー DFSDBUX1 を呼び出すことができます。データベース定義に対して

DFSDBUX1 を再度呼び出す必要がない場合、X'FF' が JCB の SRCHFLAG フィールドに戻され、DFSDLA00 は、出口を必要としていないことを示すマークをデータベース定義に動的に付けます。この場合は、DMB が DMB プールから除去されない限り、IMS セッションの期間中に、そのデータベース定義に対してユーザー出口が再度呼び出されることはありません。

DBVER
DBD の特定のバージョンを識別する、0 から 2147483647 の範囲の数値。数値を指定すると、IMS が新規バージョンの DBD を生成します。このバージョンは、別のアプリケーション・プログラムで使用することができます。

- 指定されたバージョン番号が IMS カタログ内の現行のアクティブ・バージョンより 1 大きい値ではない場合、ALTER は失敗します。
- 指定されたバージョン番号が IMS カタログ内に既に存在し、現行バージョンではない場合、ALTER は失敗します。
- 指定されたバージョン番号が現行のアクティブ・バージョンである場合、IMS はその DBD バージョンの新規インスタンスを生成します。

数値のバージョン番号の代わりに、以下を指定することもできます。

DBVER AUTO
AUTO を指定すると、IMS は IMS カタログに保管されている現行のアクティブ・バージョン番号に基づいて、バージョン番号を自動的に 1 増やします。

DBVER CURRENT
CURRENT を指定すると、IMS は現行のアクティブ DBD を変更します。DBVER CURRENT がデフォルトです。

DOSCOMPNO | DOSCOMPYES
これが DLI/DOS 索引データベースであるかどうかを指定します。データベースが索引であって、DLI/DOS を使用して作成されたものである場合には、これを指定する必要があります。 DLI/DOS 索引データベースは、接頭部の一部としてセグメント・コードを含みます。データベースが DLI/DOS 索引データベースであることを指定すると、IMS は定義されたデータベース内にこのコードが存在することを見込んで、このコードを保存するような方法で処理を行います。これには、挿入される新しいセグメントにセグメント・コードを指定することが含まれます。DLI/DOS データベースは VSAM を使用する必要があり、PHDAM、PHIDAM、または PSINDEX データベースにすることはできません。

FPINDEXNO | FPINDEXYES
索引データベースが 1 次高速機能 DEDB データベースの副次索引であるかどうかを指定します。

PASSWDNO | PASSWDYES
PASSWDYES を指定すると、このデータベースのデータ・セットを開くときに、DL/I はデータベース名を VSAM パスワードとして使用します。このパラメーターは、VSAM をアクセス方式として使用するデータベースにのみ有効です。データベース名を LOGICAL または DEDB のデータベース・タイプのパスワードとして使用することはできません。ユーザーが、z/OS アクセス方式サービス・プログラムの DEFINE ステートメントを使用してこのデータベースの VSAM データ・セットを定義する場合には、制御レベル (CONTROLPW) また

はマスター・レベル (MASTERPW) のパスワードは、この DBD の DBDNAME と同じでなければなりません。この DBD に関連付けられたすべてのデータ・セットでは、同じパスワードを使用する必要があります。

IMS DB/DC システムでは、すべての VSAM OPEN がパスワード検査をう回するため、オペレーター・パスワード・プロンプトは回避されます。 IMS DB システムでは、VSAM パスワード検査が実行されます。バッチ環境では、自動パスワード保護が指定されていないときに、データベース名と等しくないパスワードにより制御レベル (CONTROLPW) でデータ・セットがパスワード保護されている場合、オペレーター・パスワード・プロンプトが出されます。

PROTYES | PROTNO

副次索引データベースで索引ポインター保護を使用するかどうかを指定します。このオプション・パラメーターは、IMS で使用される索引ポインター・セグメント内のすべてのフィールドの保全性を確保します。このパラメーターを使用すると、索引ポインター・セグメント内のフィールド (索引ポインター・セグメントのユーザー・データ部分のフィールドは除く) での置換操作をアプリケーション・プログラムは行えなくなります。索引ポインター・セグメントの削除操作は引き続き使用可能です。索引ポインター・セグメントに対する削除が出されると、索引ポインター・セグメント内の索引ターゲット・セグメント・ポインターが削除されます。しかし、最初に索引ポインター・セグメントの作成の原因となった索引ソース・セグメントは削除されません。

索引ポインター保護を使用しない場合、アプリケーション・プログラムでは、索引ポインター・セグメント内のすべてのフィールド (定数フィールド、検索フィールド、およびサブシーケンス・フィールドは除く) を置換できます。どの条件下でも、索引データベースへの挿入は無効です。

PSNAME *name*

PSINDEX、PHDAM、または PHIDAM データベースの HALDB 区画を選択するモジュールを指定します。このパラメーターは、HALDB 区画選択出口ルーチンのモジュール名です。このパラメーターは、データベースのアクセス・タイプが PSINDEX、PHDAM、または PHIDAM である場合のみ有効です。

例外: ルート・キー範囲で HALDB 区画メンバーシップを定義している場合は、ユーザー提供の HALDB 区画選択ルーチンは必要ありません。

RMNAME *name*

DEDB に格納されているデータ、あるいは HDAM または PHDAM データベースの 1 次データ・セット・グループに格納されているデータの管理に使用するモジュール名を指定します。このパラメーターが有効なのは、データベース・アクセス・タイプが HDAM、PHDAM、または DEDB である場合のみです。ランダム化モジュールは、DEDB、HDAM、または PHDAM データベースへのルート・セグメントの配置、またはそれらからのルート・セグメントの取り出しを制御します。ランダム化モジュールと呼ばれる 1 つ以上のモジュールは、IMS システム内で利用できます。ある特定のデータベースは、それに関連したランダム化モジュールを 1 つしか持つことができません。汎用モジュール (ユーザー提供のパラメーターを使用して、ある特定のデータベースのランダム化を実行するもの) を作成して、いくつかのデータベースで利用することができます。ランダム化モジュールの目的は、DEDB、 HDAM、または PHDAM データベースへのルート・セグメントの配置、またはそれらからのルート・セグメン

トの取り出しのためにアプリケーション・プログラムが指定する値を、相対ブロック番号およびアンカー・ポイント番号に変換することです。2 ステージ・ランダマイザーを選択すると、1 つの区域内でランダム化を実行することができます。 2 ステージ・ランダマイザーを選択すると、/DBRECOVERY コマンドで DEDB の区域をすべて停止させなくても、1 つの区域内のルート・アンカー・ポイント数を変更することができます。

PHDAM データベースでは、ランダマイザーのモジュール名および値が、区画ごとのデフォルトになります。HALDB 区画定義時に、区画ごとに異なるランダマイザー名と値を設定することができます。 HALDB 区画選択は、ランダム化モジュールを呼び出す前に行われます。ランダム化モジュールは、1 つの区画内でのみ位置を選択します。

モジュール名は、この DEDB、PHDAM、または HDAM データベース内のセグメントの格納およびアクセスに使用するユーザー提供ランダム化モジュールの名前 (1 文字から 8 文字の英数字) です。モジュール名パラメーターにランダマイザーの名前を指定し、アンカー・ポイント・パラメーターに 2 を指定して、2 ステージ・ランダマイザーを選択します。

RMANCH *number*

anch 値の目的は、高速機能 DEDB データベースを定義するか、全機能 HDAM または PHDAM データベースを定義するかによって異なります。

このパラメーターは符号なしの 10 進整数でなければなりません。

DEDB データベースの場合、*anch* の値はランダマイザーのタイプを指定します。値 1 は、単一ステージ・ランダマイザーを示します。値 2 は、2 ステージ・ランダマイザーを示します。その他の値は無効です。

HDAM および PHDAM データベースの場合、*anch* の値は、HDAM または PHDAM データベースのルート・アドレス可能域内の各制御インターバルまたは制御ブロックで必要なルート・アンカー・ポイントの数を指定します。標準的な値は 1 から 5 であり、この値は 255 を超えることはできません。

HDAM または PHDAM データベースにアクセスするときに、ユーザー・ランダム化ルーチンが、このパラメーターに指定された数値より大きいアンカー・ポイント番号を生成する場合、制御インターバルまたは制御ブロック内で最高の番号のアンカー・ポイントが使用されます。ランダム化ルーチンが IMS アンカー・ポイント番号 0 を生成した場合には、IMS は制御インターバルまたは制御ブロックのアンカー・ポイント 1 を使用します。

RMRBN *number*

このデータベースに関してランダム化モジュールに作成させる相対ブロック番号の最大値を指定します。このパラメーターは、HDAM または PHDAM データベースの場合のみ使用します。この値により、HDAM または PHDAM データベースのルート・アドレス可能域内の制御インターバルまたは制御ブロックの数が決まります。このパラメーターは、2^{24}-1 を超えない符号なしの 10 進整数にする必要があります。このパラメーターを省略すると、ランダム化モジュールが作成する相対ブロック番号で上限の検査は行われません。このパラメーターを指定したにもかかわらず、指定されたランダム化モジュールがこのパラメーターよりも大きい相対ブロック番号を生成する場合、ルート・アドレス可能域内の最高位の制御インターバルまたは制御ブロックが IMS により使用されます。ユーザ

ーのランダム化モジュールがブロック番号 0 を生成する場合、制御インターバルまたは制御ブロック 1 が IMS により使用されます。

HDAM または PHDAM データ・セットでは、最初のビットマップは、データ・セットの先頭にあるエクステントの先頭ブロックにあります。 HDAM または PHDAM データベースでは、データ・セット・グループごとに指定されるデータ・セットの先頭にあるエクステントの最初の制御インターバルまたは制御ブロックが、ビットマップに使用されます。 VSAM データ・セットでは、2 番目の制御インターバルがビットマップに使用され、最初の制御インターバルは予約されます。IMS は、ランダマイザーが計算したそのブロックに 1 を加えます。

RMBYTES *number*
別のデータベース・レコードの呼び出しによって分断されない一連の挿入でルート・アドレス可能域に格納できる、データベース・レコードの最大バイト数を指定します。このパラメーターは、HDAM または PHDAM データベースの場合のみ使用します。このパラメーターを省略すると、このデータベースのルート・セグメント・アドレス可能域に挿入できるデータベース・レコードの最大バイト数は、無制限になります。bytes パラメーターは、2^{24}-1 を超えない符号なしの 10 進整数にする必要があります。最大相対ブロック番号パラメーターを省略すると、このパラメーターは無視されます。この場合、ルート・アドレス可能域に挿入できるデータベース・レコードのバイト数に制限はありません。

このパラメーターを HDAM または PHDAM データベースに対して指定した場合に、データベース・レコードの長さがそれより大きいと、レコードの超過部分は、現行のファイルの終わり (EOF) に続くオーバーフロー域に挿入されます。この操作には、このパラメーターの値を超えるすべてのデータベース・レコードの超過部分を入れる十分なスペースが、現行の EOF の後に使用可能でなければなりません。現行の EOF の後のオーバーフロー域に十分なスペースがないと、データベース・レコードはデータベースにランダムに挿入されることになります。

XCINO | XCIYES
この DEDB が、ランダマイザーを呼び出すときに、拡張呼び出しインターフェースを使用するかどうかを指定します。このオプションでは、3 つの異なる方法でランダマイザーを呼び出すことができます。IMS の初期設定時、または /START DB コマンド実行時に、IMS は、まずランダマイザーをロードし、次にランダマイザーに INIT 呼び出しを行って、その初期設定ルーチンを呼び出します。 /DBR DB コマンド実行の過程で IMS は TERM 呼び出しを行い、終了ルーチンを呼び出してから、ランダマイザーをアンロードします。アプリケーションがルート・セグメントに GU または ISRT 呼び出しを出すと、通常のランダム化呼び出しがランダマイザーに対して行われます。XCI オプションは、DEDB の場合にのみ有効です。

VERSION '*version_identifier*'
バージョン ID ストリングを指定します。この ID を使用して、IMS カタログに対する後続の照会でリソースのバージョンを区別することができます。

使用上の注意

ALTER DATABASE ステートメントは、データベースを IMS に変更するため、ALTER DATABASE ステートメントで指定されたデータベースが存在しない場合、

このステートメントは -9000 メッセージで失敗します。

データベースのバージョン管理の注意事項

独自の DDL ストリームで ALTER DATABASE を使用すると、既存の DBD が変更されます。データベースのバージョン管理はオプションです。現行バージョンに対して変更を実施するには、DBVER キーワードで現行バージョン番号を指定します。また、「DBVER CURRENT」を指定することで、IMS がそれらの現行のアクティブ・バージョン番号を識別することもできます。

新規バージョンを生成するには、DBVER キーワードで次のバージョン番号を指定します。指定するバージョン番号は、現行のアクティブ・データベース・バージョンより 1 大きい値にする必要があります。

DBD に自動的にバージョン番号を割り当てるには、「DBVER AUTO」を指定します。DDL は、一時的に INIT トークンをバージョン番号として割り当てます。この割り当ては、複数のワークステーションが異なる DBD 変更を実施するケースを処理します。

オプションで、CREATE PROGRAMVIEW を指定して新規の PSB を生成したり、ALTER PROGRAMVIEW を指定して既存の PSB を更新したりすることができます。このストリームの一部として、PCB に対して 「DBVER AUTO」を指定すると、それらの PCB は生成される同じバージョン番号にロックされます。 IMS システムで DDL ストリームの変更がアクティブ化されると、IMS はバージョン番号を DBD および PSB に割り当てます。

ALTER DATABASE ステートメントで DBVER キーワードが省略されると、IMS は、IMS カタログに示されているように、現行のアクティブ・データベース・バージョンに対して変更を実施します。

例: 全機能データベース

DBD 生成ユーティリティーに対して以下を入力すると、基本全機能データベースが作成されます。

新規データベースを定義するためのオリジナル DBD ソース。

```
DBD   NAME=COGDBD,                                          C
         ENCODING=Cp1047,                                   C
         ACCESS=(HDAM,VSAM),                                C
         RMNAME=(DFSHDC40,3,3,25),                          C
         PASSWD=NO,                                         C
         VERSION='Latest version of COGDBD'
```

以下の例は、DBD ソースに同等の DDL です。

```
CREATE DATABASE COGDBD
  ACCESS HDAM VSAM
  RMNAME(DFSHDC40 RMANCH 3 RMRBN 3 RMBYTES 25)
  VERSION 'Latest version of COGDBD'
  CCSID 'Cp1047';
```

別のランダマイザー (PASSWD および VERSION) を提供する DBD ソース。

```
DBD   NAME=COGDBD,                                          C
         ENCODING=Cp1047,                                   C
         ACCESS=(HDAM,VSAM),                                C
```

```
         RMNAME=(DFSHDC20,3,3,25),                                  C
         PASSWD=YES,                                                 C
         VERSION='Latest version of COGDBD'
```

以下の例は、DBD ソースに同等の DDL です。

```
ALTER DATABASE COGDBD
  RMNAME(DFSHDC20 RMANCH 3 RMRBN 3 RMBYTES 25)
  PASSWDYES
  VERSION 'Latest version of COGDBD'
```

例: 高速機能高速処理データベース (DEDB)

上記の例と同様に、DBD 生成ユーティリティーに対して以下の入力を実行して、DEDB を作成することができます。

データベースを定義するオリジナル DBD ソース:

```
DBD       NAME=HOSPDBD1,                                           C
             ENCODING=Cp1047,                                      C
             ACCESS=(DEDB),                                        C
             RMNAME=(RMOD3,1,,,XCI)                                C
             PASSWD=NO
```

以下の例は、DBD ソースに同等の DDL です。

```
CREATE DATABASE HOSPDBD1
  ACCESS DEDB
  RMNAME( RMOD3 RMANCH 1 XCIYES)
  CCSID 'Cp1047';

COMMENT ON DATABASE HOSPDBD1 IS 'This describes database HOSPDBD1.'
```

別のランダマイザー (XCI、エンコード、および新規コメント) を提供する DBD ソース変更。

```
DBD       NAME=HOSPDBD1,                                           C
             ENCODING=Cp943C,                                      C
             ACCESS=(DEDB),                                        C
             RMNAME=(FPERNDM0,1,,,)                                C
             PASSWD=NO
```

以下の例は、DBD ソースに同等の DDL です。

```
ALTER DATABASE HOSPDBD1
  RMNAME(FPERNDM0 RMANCH 1 XCINO)
  CCSID 'Cp943C';
COMMENT ON DATABASE HOSPDBD1 IS 'Implemented change to database HOSPDBD1.'
```

ALTER TABLE

ALTER TABLE ステートメントを使用して、既存の表を変更できます。CREATE TABLE ステートメントとは異なり、それぞれのキーワード属性にデフォルト値はありません。その値を変更するには、キーワードまたは値を指定する必要があります。キーワードも値も指定しない場合、その属性に変更は行われません。

制約事項: 以下のいずれかのキーワードを CREATE TABLE ステートメントに指定した場合は、ALTER TABLE ステートメントを使用してキーワードとキーワード値を変更することはできません。キーワードとキーワード値を変更するには、まず

DROP TABLE ステートメントを使用して表を削除する必要があります。次に、CREATE TABLE ステートメントを使用して表を再作成し、キーワードとキーワード値を再び指定する必要があります。

- 定義するセグメント・タイプの内部名を指定する INTERNALNAME*internalname* キーワード。
- DIRECT DEPENDENT | SEQUENTIAL DEPENDENT

呼び出し

このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。 これは実行可能ステートメントですが、動的に準備することはできません。

- 『ALTER TABLE 構文』
- 『table-options 構文』
- 731 ページの『data capture changes 構文』
- 731 ページの『exit_attributes 構文』
- 732 ページの『alter-options 構文』
- 733 ページの『column-alteration 構文』
- 733 ページの『datatype 構文』
- 733 ページの『ims-column 構文』
- 734 ページの『inline-constraint 構文』
- 734 ページの『constraint 構文』
- 734 ページの『references-clause 構文』
- 735 ページの『map-definition 構文』
- 735 ページの『case-definition 構文』

ALTER TABLE 構文

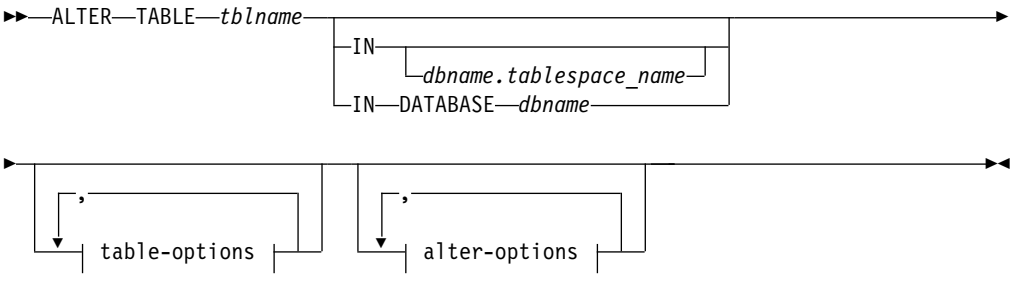
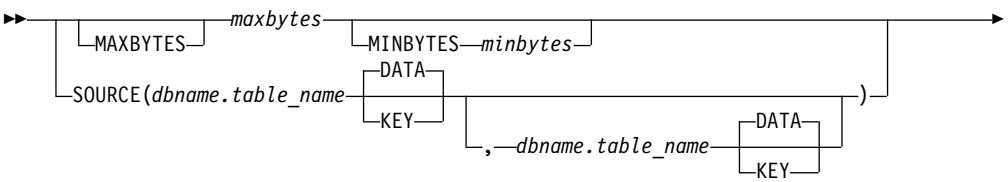
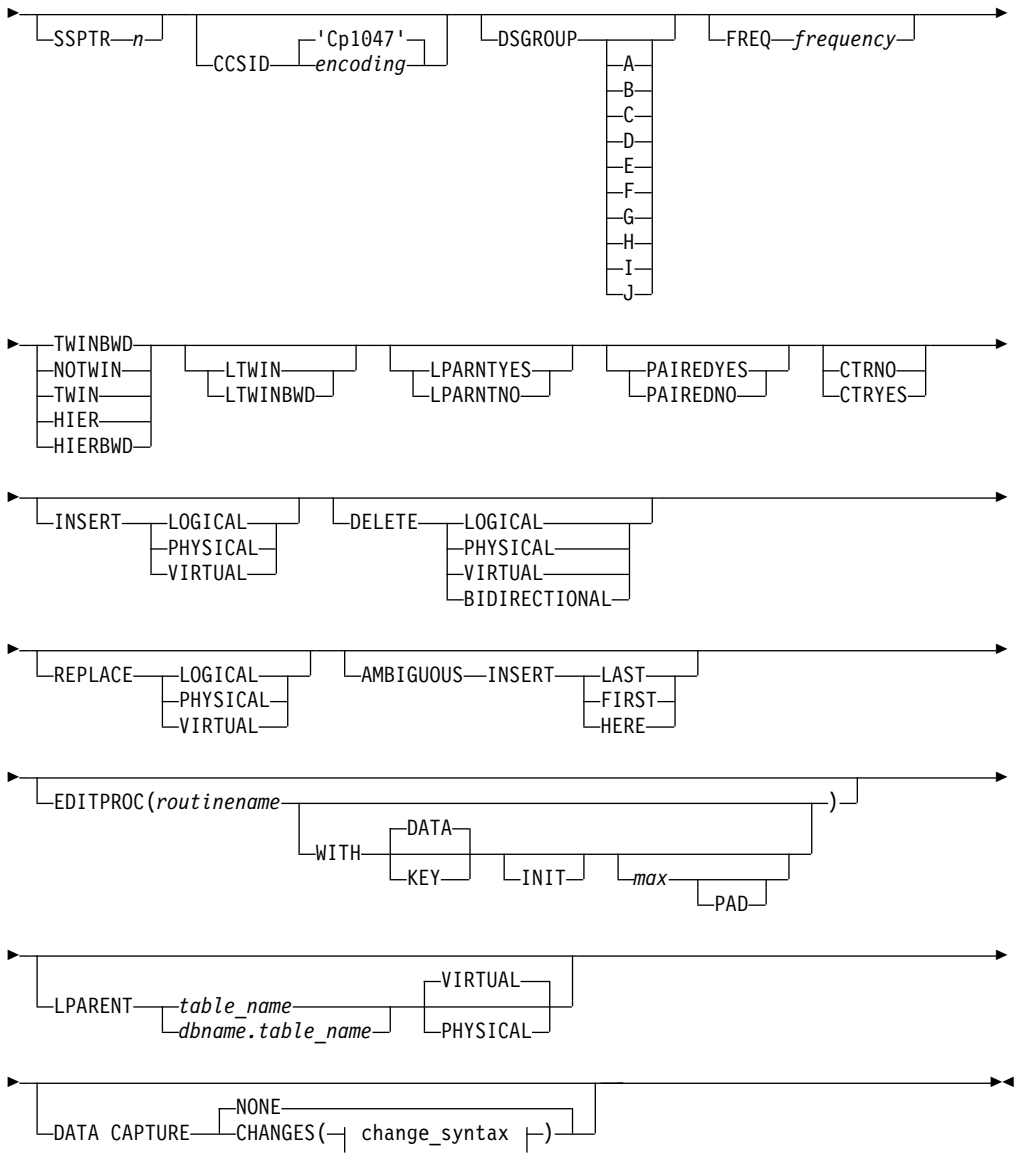
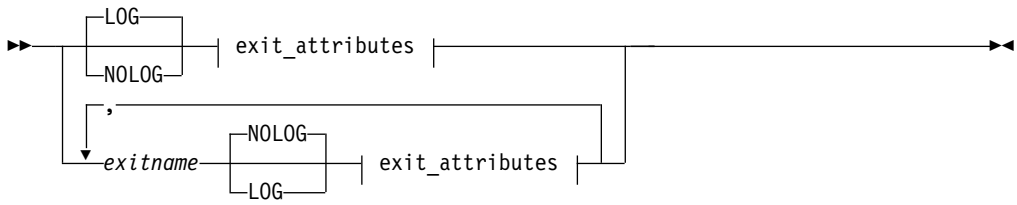


table-options 構文

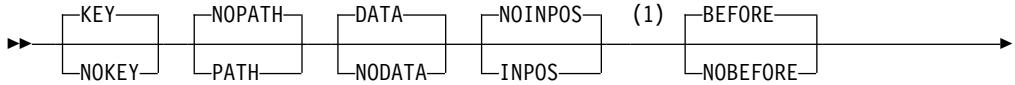


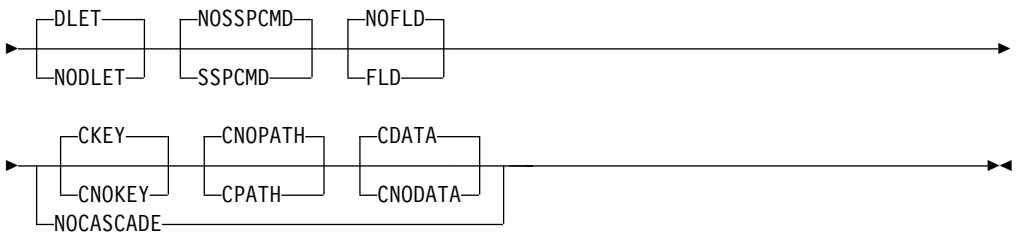


data capture changes 構文



exit_attributes 構文

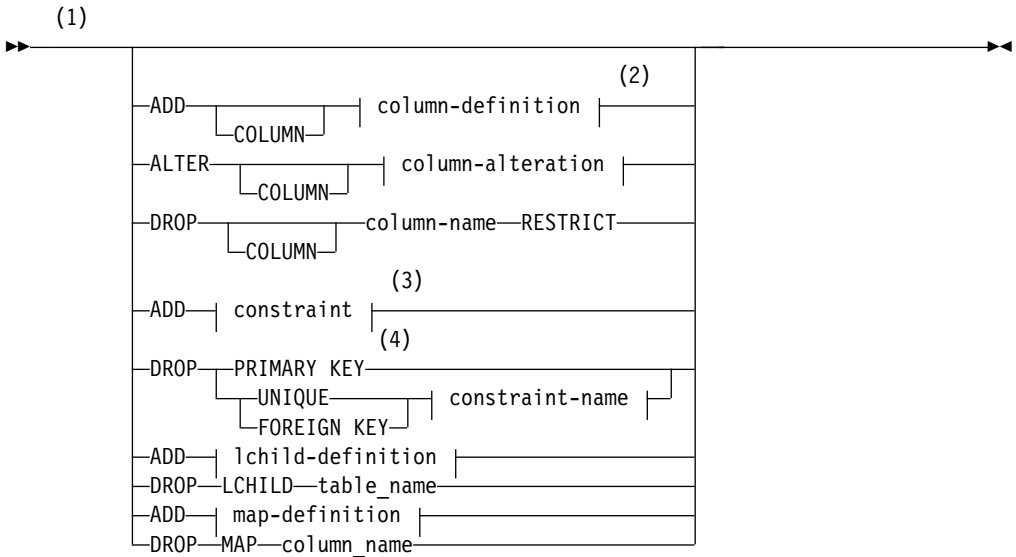




注:

1 BEFORE、NOBEFORE、DLET、NODLET、SSPCMD、NOSSPCMD、FLD、および NOFLD は、DEDB 専用です。

alter-options 構文



注:

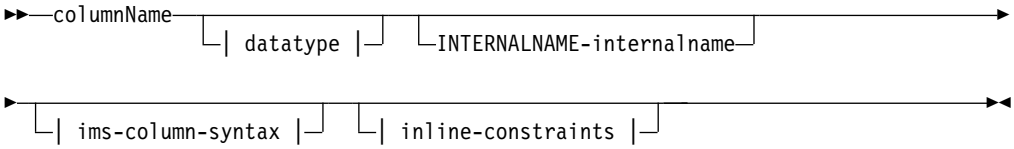
1 同じ節を複数回指定することはできません (ADD COLUMN または ALTER COLUMN 節の場合を除く)。複数の ADD COLUMN 節を同じステートメントに指定する場合、references-clause を含むことができる ADD COLUMN 節は多くて 1 つです。

2 CREATE TABLE セクションから column-definition の説明を参照してください。

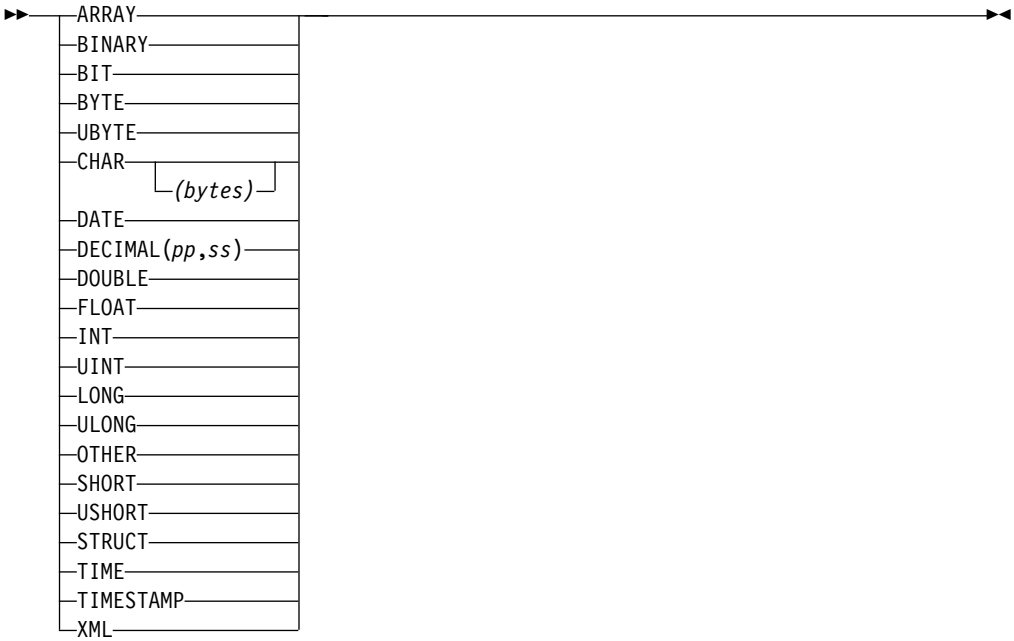
3 ADD 制約の DROP PRIMARY KEY 節と DROP FOREIGN KEY 節は相互に排他的であり、ALTER TABLE ステートメントごとに 1 つだけ指定することができます。

4 ADD 制約の DROP PRIMARY KEY 節と DROP FOREIGN KEY 節は相互に排他的であり、ALTER TABLE ステートメントごとに 1 つだけ指定することができます。

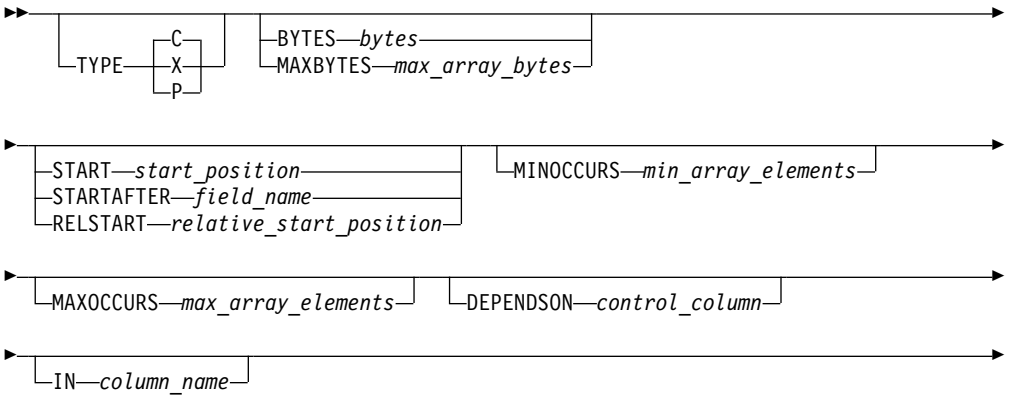
column-alteration 構文

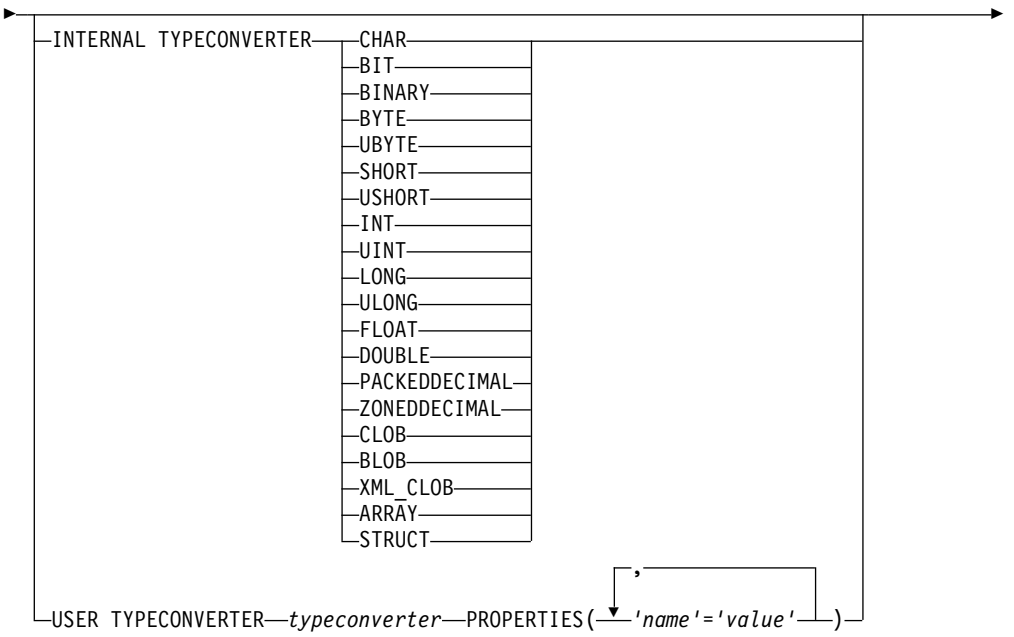


datatype 構文

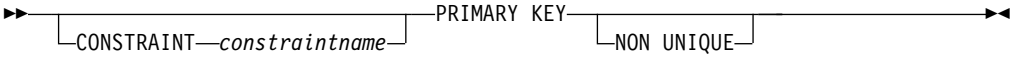


ims-column 構文

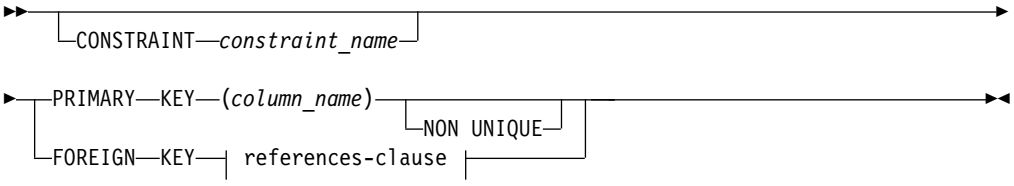




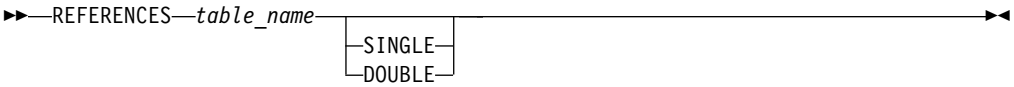
inline-constraint 構文



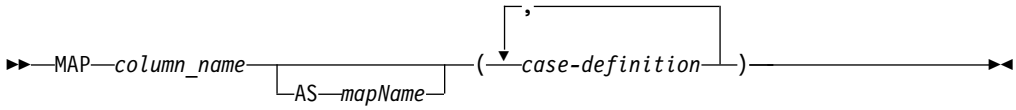
constraint 構文



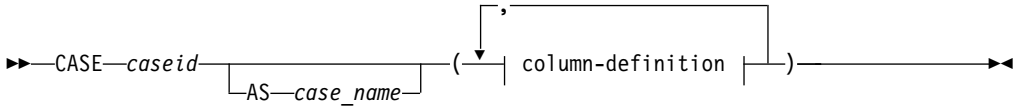
references-clause 構文



map-definition 構文



case-definition 構文



ALTER TABLE のキーワード・パラメーター

すべての table-options の説明は、CREATE TABLE セクションで定義されています。すべての table-options は、任意指定であり、指定されない場合にデフォルト値を意味するものではありません。 table-option の指定は、既存の値に置き換わる新規値が提供されていることを意味します。

DATA CAPTURE キーワードを指定すると、すべてのデータ・キャプチャー出口 (以前に提供されたものがある場合) が置き換わります。複数のデータ・キャプチャー出口が以前に提供されていた場合、必要に応じてそのすべてを再指定する必要があります。

ALTER TABLE ステートメントには、以下のキーワード・パラメーターが定義されています。

ALTER TABLE*tblname*
変更する表を指定します。この名前は、現行データベースに存在する表を示すものでなければなりません。

外部名は 1 文字から 128 文字の大文字の英数字ストリングとして指定します。表の名前には、下線文字を含めることができます。表の名前は、予約済みの SQL キーワードにすることはできません。また、DFS で開始することもできません。

ALTER TABLE (table-options) のキーワード・パラメーター

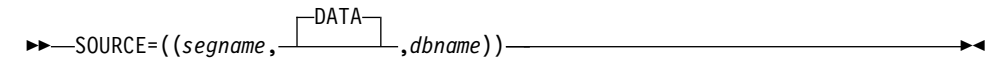
ALTER TABLE (table-options) ステートメントには、以下のキーワード・パラメーターが定義されています。

SOURCE
IMS 内部表の名前であり、以下の 2 つの目的で使用されます。

- 定義する仮想論理子セグメント・タイプで表される実論理子セグメント・タイプを識別するため。
- 論理データベース内に定義されるセグメント・タイプで表される、物理データベース内のセグメント・タイプ (複数も可) を識別するため。

制約事項: PHDAM および PHIDAM データベースでは物理対しかサポートしないため、SOURCE キーワードは使用できません。

仮想論理子を定義するステートメントは、次のようになります。

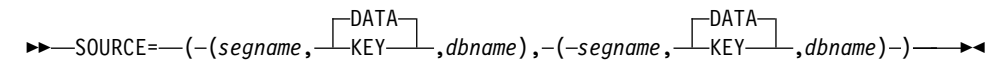


segname
実論理子の名前を指定します。

DATA
segname のキーとデータの両方の部分を、セグメントの組み立てに使用することを示します。このパラメーターは必須です。

dbname
実論理子が入っている物理データベースの名前を指定します。

論理データベースのセグメント・タイプを定義するステートメントは、次のようになります。



(*segname*, KEY | DATA,*dbname*)
最初のオカレンスは、論理セグメントとして定義されている物理データベース内のセグメントを指すか、またはこの論理データベースの連結セグメント・タイプの最初の部分として使用する物理データベース内の論理子セグメント・タイプを指します。

segname
物理データベース内の論理子セグメント・タイプを指します。

KEY
segname に指定されているセグメントのキー部分を、キー・フィードバック域に入れることを指定します。*segname* で表されている論理セグメント・タイプを処理するための呼び出しを発行するときは、このセグメントをユーザーの入出力域に入れてはなりません。

DATA
segname で指定したセグメントのキー部分をキー・フィードバック域に入れる必要のあることと、*segname* で表されている論理セグメント・タイプを処理するための呼び出しを発行するときは、このセグメントをユーザーの入出力域に入れる必要があることを指定します。

dbname
segname を含む物理データベースの名前を指定します。(*segname*, KEY|DATA, *dbname*) の 2 番目のオカレンスは、この論理データベース内の連結セグメントの目標親部分に使用される、物理データベース内の論理親セグメント・タイプまたは物理親セグメント・タイプを指し示します。この 2 番目のオカレンスの各パラメーターの説明は、最初のオカレンスの説明と同じです。

(*segname*, KEY | DATA, *dbname*) の最初のオカレンスが仮想論理子を指している場合、2 番目のオカレンス (指定する場合) は、実論理子の物理親を指していなければなりません。

ソース・セグメントを使用して、連結セグメントを表すときは、検索呼び出しの際にこの 2 つのセグメントのいずれを (または両方を) ユーザーの入出力域に入れるかを、KEY と DATA のパラメーターで指定します。DATA を指定すると、セグメントはユーザーの入出力域に入れられます。 KEY を指定すると、セグメントはユーザーの入出力域には入れられず、シーケンス・フィールド・キー (存在している場合) が PCB のキー・フィードバック域に入れられます。連結セグメントのキーは、論理子がどのパスからアクセスされるかによって、論理子のキー、つまり物理兄弟シーケンス・フィールドか論理兄弟シーケンス・フィールドのいずれかになります。KEY および DATA パラメーターは、検索タイプの呼び出しのみに適用されます。

挿入呼び出しでは、ユーザーの入出力域には常に論理子セグメントと、挿入規則が物理でない限り、論理親セグメントが入っていなければなりません。KEY がセグメントについて指定されている場合でも、参照されたセグメントを含んでいる論理データベースに対して呼び出しが出された場合には、そのセグメントを含んでいるデータベースを IMS が使用できなければなりません。SOURCE セグメント指定の最初のオカレンスが論理子を指している場合は、連結セグメントの目標親を指す 2 番目のオカレンスも指定する必要があります。明示的に指定されていなければ、デフォルトによりブロックの作成時に KEY パラメーターで組み込まれます。

論理 DBD 生成で定義するセグメントは、1 つ以上の物理 DBD 生成で前もって定義されているセグメントの物理定義を使わなければなりません。

SEGM ステートメントで INDEX データ・セット内のセグメントを定義する場合は、SOURCE パラメーターは無効です。

MAXBYTES *maxbytes*
MINBYTES *minbytes*

minbytes パラメーターを含む場合には、セグメント・タイプを可変長として定義します。maxbytes フィールドでは、このセグメント・タイプのオカレンスの最大長を指定します。maxbytes パラメーターに許される最大値と最小値は、固定長セグメントに関して説明した値と同じです。

セグメントが圧縮ルーチンによって処理される場合は、異常終了 0799 を回避するために、セグメント長を指定された最大定義より長くできるかどうかを示す制御情報を収容できるよう maxbytes フィールドを設定します。拡張を可能にするには、maxbytes に 10 バイトの任意の値を追加します。

minbytes パラメーターでは、可変長セグメントが使用する最小ストレージ量を指定します。minbytes の最大値は、maxbytes に指定する値です。minbytes の最小値は次のものでなければなりません。

- セグメント・タイプを編集/圧縮ルーチンで処理しない場合、または編集/圧縮ルーチンで処理はするがキー圧縮オプションを指定していない場合は、セグメント・タイプにシーケンス・フィールドを指定してあれば、minbytes にはシーケンス・フィールド全体を入れられる値を指定しなければなりません。
- キー圧縮オプションを使う編集/圧縮ルーチンで処理するセグメント・タイプ、または順序付けられていないセグメントの場合の最小値は、4 です。

HSAM、SHSAM、INDEX、PSINDEX、または SHISAM データベースのセグメントは可変長にすることができないため、これらのデータベースでは minbytes パラメーターは無効です。

高速機能 DEDB では、セグメントは、2 バイト・フィールド (この 2 バイト長フィールドを含むセグメントの長さを定義) で始まり、列で指定するユーザー・データが後に続きます。 minbytes の値には、4 (最小値) から maxbytes (最大値) を指定できますが、minbytes 値は、このセグメントのシーケンス・フィールドを入れるのに十分な大きさでなければなりません (つまり、minbytes ≥ START - 1 + 表に続くシーケンス・フィールドの BYTES)。例えば、シーケンス・フィールドの長さが 20 バイトで START= 7 のセグメントでは、最小 minbyte 値は 26 です。該当する任意の DL/I 呼び出しでは、実際のセグメント長は、シーケンス・フィールドを含んだ長さと maxbytes の値の間になります。maxbytes の値は、制御インターバル・サイズ - 120 を超えてはなりません。

TWINBWD | NOTWIN | TWIN | HIER | HIERBWD
定義するセグメント・タイプのオカレンスの接頭部に、ポインター・フィールドを予約することを指定します。 これらのフィールドは、このセグメントを隣接した親セグメントおよび兄弟セグメントに関連付けるために使用されます。

TWINBWD
定義するセグメントの接頭部に、4 バイトの物理兄弟順方向ポインター・フィールドと 4 バイトの逆方向物理兄弟ポインター・フィールドを予約します。逆方向物理兄弟ポインターを使用すると、削除パフォーマンスが向上します。

推奨事項: このオプションは、HIDAM および PHIDAM データベースのルート・セグメントに使用するようお勧めします。

NOTWIN
定義するセグメント・タイプのオカレンスの接頭部に、物理兄弟順方向ポインターのスペースを予約しないようにします。

NOTWIN は、次の場合に従属セグメント・タイプに対して指定できます。

- 物理親に階層ポインターが指定されていない。
- 物理親セグメント・タイプのオカレンスの物理子として保管されている従属セグメント・タイプのオカレンスが 1 つ以下である。

さらに、NOTWIN は、HDAM および PHIDAM データベースのルート・セグメント・タイプにも指定できますが、それはランダム化モジュールが同義語 (同じブロックおよびアンカー・ポイントを持っている異なる値のキー) を生成しない場合に限られます。

NOTWIN が従属セグメント・タイプについて指定されている場合に、従属セグメントの 2 番目のオカレンスを、ある与えられた物理親セグメントの物理子としてロードまたは挿入しようとすると、次のようになります。

- 初期ロード時に 2 番目のオカレンスを挿入しようとすると、LB 状況コードが戻されます。
- 初期ロード後に 2 番目のオカレンスを挿入しようとすると、II 状況コードが戻されます。

同義語をロードまたは挿入する試みはリジェクトされて、LB 状況コードまたは II 状況コードが出されます。

TWIN
定義するセグメントの接頭部に、4 バイトの物理兄弟順方向ポインター・フィールドを予約します。

HIER
定義するセグメント・タイプのオカレンスの接頭部に、4 バイトの階層順方向ポインター・フィールドを予約します。HALDB では HIER はサポートされません。

HIERBWD
定義するセグメント・タイプのオカレンスの接頭部に、4 バイトの階層順方向ポインター・フィールドと 4 バイトの階層逆方向ポインター・フィールドを予約します。逆方向階層ポインターを用いると、削除のパフォーマンスが上がります。 HALDB では HIERBWD はサポートされません。

LPARNTYES | LPARNTNO
論理親のタイプを指定します。

LPARNTYES
このパラメーターを指定できるのは、定義されるセグメント・タイプが論理子であり、しかも論理親が HDAM、HIDAM、PHDAM、または PHIDAM データベース内に存在する場合に限られます。論理親が HISAM データベース内にある場合は、このパラメーターを省略し、定義するセグメントの PARENT= パラメーターに PHYSICAL を指定してください。

HDAM、HIDAM、および HISAM データベースの場合、 LPARNT は、定義するセグメント・タイプのオカレンスの接頭部に 4 バイトの論理親ポインター・フィールドを予約します。

PHDAM および PHIDAM データベースの場合、LPARNT は、定義するセグメント・タイプのオカレンスの接頭部に 28 バイトの拡張ポインター・セットを予約します。

LPARNTNO
定義するセグメント・タイプが論理子ではなく、論理親が HDAM、HIDAM、PHDAM、および PHIDAM のデータベースのいずれにも存在しないことを指定します。

PAIREDYES | PAIREDNO
このセグメントが双方向論理関係を持つかどうかを指定します。

PAIREDYES
このセグメントが両方向論理関係を持つことを示します。このパラメーターは、以下のタイプに指定します。

- 仮想論理子セグメント・タイプ
- 両方向論理関係にある両方の物理対の論理子セグメント・タイプ

PAIRED を指定した場合、LTWIN および LTWINBWD パラメーターは無効になります。

PAIREDNO
このセグメントが双方向論理関係を持たないことを示します。

CTRNO | CTRYES
指定対象

CTRNO
定義するセグメント・タイプのオカレンスの接頭部に、4 バイトのカウンター・フィールドを予約しません。

CTRYES
定義するセグメント・タイプのオカレンスの接頭部に、4 バイトのカウンター・フィールドを予約します。カウンターが必要になるのは、HISAM、HDAM、または HIDAM データベースの中の論理親セグメントが、論理子ポインターで自分に接続されていない論理子セグメントを持っている場合です。ユーザーがこのパラメーターを指定しなくても、カウンターは、必要とされるすべてのセグメントの中に、DBD 生成時に自動的に入れられます。しかし、後で DBD 生成を行うのを避けるために、ユーザーは将来のカウンターの必要性を予想して、このパラメーターを使用してセグメント・タイプのオカレンスの接頭部の中にカウンター・フィールドを確保しておくことができます。HALDB では CTR はサポートされません。

INSERT {LOGICAL | PHYSICAL | VIRTUAL}
DELETE {LOGICAL | PHYSICAL | VIRTUAL | BIDIRECTIONAL}
REPLACE {LOGICAL | PHYSICAL | VIRTUAL}
定義するセグメント・タイプのオカレンスの挿入、削除、および置換に使用される規則を指定します。これらのパラメーターは、論理子セグメント、およびそれらの物理親セグメントと論理親セグメントに指定します。論理関係を持たないすべてのセグメント・タイプについては、これらを省略する必要があります。

AMBIGUOUS INSERT {LAST | FIRST | HERE}
この表が定義するセグメント・タイプの新しいオカレンスが、物理データベースに挿入される場所を指定します (物理兄弟順序を確立します)。この値は、シーケンス・フィールドを持たないセグメント、またはシーケンス・フィールドが固有でないセグメントを処理する場合にのみ使用します。固有のシーケンス・フィールドが定義されているセグメント・タイプに指定された場合、この値は無視されます。

HDAM および PHDAM のルートを除いて、FIRST、LAST、または HERE の規則は、データベースの初期ロードには適用されず、セグメントはロード・モードに示された順序でロードされます。初期ロードまたは HD 再ロードで、固有のシーケンス・フィールドが HDAM のルートに関して定義されていない場合は、FIRST、LAST、または HERE の挿入規則によりルートがチェーニングされる順序が決まります。したがって、HDAM または PHDAM データベースの再ロードでは、HERE または FIRST が使用されると、順序付けられていないルートの順序が反転されます。

DEDB セグメント以外は、LAST がデフォルトです。

高速機能順次従属セグメント処理の場合は、FIRST の挿入規則が常に使用され、これを変更することはできません。直接従属セグメント処理の場合は、FIRST、LAST、または HERE を指定できます。デフォルトは HERE です。

FIRST
シーケンス・フィールドが定義されていないセグメントの場合には、既存のすべての物理兄弟の前に新しいオカレンスが挿入されます。固有でないシー

ケンス・フィールドが定義されているセグメントの場合には、同じシーケンス・フィールド値を持つ既存のすべての物理兄弟の前に新しいオカレンスが挿入されます。

LAST
シーケンス・フィールドが定義されていないセグメントの場合には、既存のすべての物理兄弟の後ろに新しいオカレンスが挿入されます。固有でないシーケンス・フィールドが定義されているセグメントの場合には、同じシーケンス・フィールド値を持つ既存のすべての物理兄弟の後ろに新しいオカレンスが挿入されます。

HERE
シーケンス・フィールドのないセグメントの場合には、位置が確立されている物理兄弟の直前に新しいオカレンスが挿入されます。挿入するセグメントの物理兄弟に位置が確立されていなければ、既存のすべての物理兄弟の前に新しいオカレンスが挿入されます。固有でないシーケンス・フィールドが定義されているセグメントの場合には、同じシーケンス・フィールド値を持ち、位置が確立されている物理兄弟の直前に新しいオカレンスが挿入されます。同じシーケンス・フィールド値を持っている物理兄弟に位置が確立されていなければ、同じシーケンス・フィールド値を持つすべての物理兄弟の前に新しいオカレンスが挿入されます。挿入位置は、直前の DL/I 呼び出しで確立された位置により異なります。

物理パスに対して出される挿入呼び出しでは、指定した挿入規則よりコマンド・コードの L (last) が優先されるため、新しいオカレンスは LAST の挿入規則に従って挿入されます。

DSGROUP
PHDAM および PHIDAM データベースについて、複数データ・セット・グループを指定します。形式は DSGROUP c です。ここで、c は A から J の文字です。これによって、PHDAM および PHIDAM データベースを最大 10 個のデータ・セット・グループに分割できます。すべてのセグメントで、デフォルトは A (1 区分当たり 1 個のデータ・セット) です。ルート・セグメントに指定する場合、DSGROUP A でなければなりません。

制約事項: A から J のシーケンスにギャップがあってはなりません。例えば、DSGROUP C を CREATE TABLE ステートメントで指定した場合、DSGROUP B を指定した CREATE TABLE ステートメントが少なくとも 1 個必要で、各 HALDB 区画は A、B、および C データ・セットを持つことになります。

FREQ *frequency*
このセグメントが物理親の各オカレンスごとに何回現れる可能性があるかを示す見積回数を指定します。frequency パラメーターは、0.01 文字から 2^{24}-1 の符号なしの 10 進数でなければなりません。これがルート・セグメントの場合、「frequency」は、定義されるデータベース内に現れるデータベース・レコードの最大数の見積もりです。従属セグメントに適用される場合の FREQ パラメーターの値は、データベースの各データ・セット・グループの論理レコード長および物理ストレージ・ブロック・サイズを判別するのに使用されます。

CCSID *encoding*

セグメント内の文字データのエンコードを指定する、1 文字から 25 文字のオプション・フィールド。

CCSID パラメーターに指定する値には、以下の文字を含めることはできません。

- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

表内の **CCSID** パラメーターの値は、このセグメントのデータベース内の **CCSID** パラメーターの値をオーバーライドします。表で **CCSID** パラメーターが指定されていない場合、デフォルト値は、データベースの **CCSID** パラメーターの値、または (データベースで **CCSID** が指定されていない場合は) 値 Cp1047 (EBCDIC エンコードを指定) のいずれかです。

この値は、列定義の **CCSID** パラメーターによって個々のフィールドでオーバーライドすることができます。

SSPTR *n*

DEDB アクセス・タイプによって定義されたデータベース専用。サブセット・ポインターの数を指定します。0 から 8 の値を指定できます。0 を指定するか、あるいは SSPTR を指定しない場合は、サブセット・ポインターを使用しないことになります。

EDITPROC *routinename*

DEDB または全機能データベースにセグメント編集/圧縮の出口ルーチンを選択します。

全機能データベースのセグメント編集/圧縮の場合

SOURCE キーワードを指定している場合は、このキーワードを指定してはなりません。DL/I EDITPROC キーワードは、HSAM、SHSAM、SHISAM、INDEX のデータベース、および論理データベースには無効です。これは、すべてのデータベースの論理子セグメントについても無効です。HISAM データベースに使用する場合は、HISAM ルート・セグメントのシーケンス・フィールド・オフセットを変更してはなりません。さらに、セグメント編集/圧縮オプションが指定されているセグメント・タイプに指定できる最小セグメント長は、4 バイトです。

要確認**:** セグメント編集/圧縮出口ルーチンを使用しており、しかもセグメントを可変長として定義した場合は、可変長セグメントの圧縮時には、DBD で定義されたセグメントの最小の長さまでヌル・バイトが埋め込まれることに注意してください。最小のセグメントの長さが、圧縮長をオーバーライドします。これにより、過度に圧縮されたセグメントのロード時に、追加スペースが提供されます。

routinename

ユーザー提供の編集/圧縮出口ルーチンの名前を指定します。この名前は、1 文字から 8 文字の英数字値でなければならず、IMS.SDFSRESL 内の他の名前と同じであってはならず、また、データベース名と同じであってもなりません。

DATA
示された出口ルーチンがデータ・フィールドのみを圧縮または修正することを指定します。シーケンス・フィールドを修正してはならないだけでなく、セグメントの開始点に対してのシーケンス・フィールドの位置を変更するデータ・フィールドも修正してはなりません。圧縮ルーチンの名前が指定されていても、パラメーターが選択されていない場合は、DATA がデフォルトです。

KEY
指名されたセグメント内のどのフィールドも、出口ルーチンによって圧縮または修正が可能であることを指定します。このパラメーターは、HISAM データベースのルート・セグメントについては無効です。

INIT
セグメント出口ルーチンが初期設定と終了処理制御を必要とすることを示します。このパラメーターを指定すると、データベースのオープン後およびデータベースのクローズ後に編集/圧縮ルーチンに制御が渡されます。

max
圧縮出口ルーチンの間に固定長セグメントの長さを増やすことのできる最大バイト数を指定します。 1 バイトから 32 767 バイトを指定できます。 *max* のデフォルトは 10 です。

PAD
MAX で指定した数値は埋め込み用に使用し、MAX としては使用しないことを示します。1 から 32 767 までの数値は、挿入されたセグメントの圧縮後の長さが PAD 値よりも短いときに、そのサイズまで挿入セグメントが埋め込まれることを示します。

DEDB のセグメント編集/圧縮の場合

routinename
ユーザー提供のセグメント編集/圧縮出口ルーチンの z/OS ロード・モジュール名を指定します。ルーチン名が必要です。

DATA
セグメントのユーザー・データ部分のみが圧縮されることを指定します。 DATA がデフォルトです。

制約事項**:** KEY パラメーターは、DEDB についてはサポートされていません。KEY パラメーターを指定すると、エラー・メッセージが発行されます。

INIT
この指定により、データベースの最初の区域がオープンされた直後にセグメント圧縮出口ルーチンが制御を獲得し、データベースの最後の区域がクローズされる直前に制御を戻すことができます。指定された値の範囲内にセグメント長がある限り、セグメントの圧縮または拡張についてのフィールド修飾の検査中にエラーは起こりません。

制約事項**:** EDITPROC 節は、表の末尾に固有キー・フィールドが含まれる DEDB 表では禁止されています。

LPARENT *table_name* {VIRTUAL | PHYSICAL}
定義する表の論理親を指定します。

table_name
IMS 内部表の名前、および定義する表の論理親の名前を指定します。論理親が同じデータベース内に存在する場合、表の名前のみを指定するだけで構いません。論理親が別のデータベース内に存在する場合、データベースと表の名前の両方 (「database_name.tablename」など) を指定する必要があります。

VIRTUAL | PHYSICAL
論理親の連結キー (LPCK) を論理子セグメントの一部として保管するかどうかを指定します。論理子セグメントに対してのみパラメーターを指定してください。PHYSICAL を指定した場合は、LPCK は、各論理子セグメントと一緒に保管されます。VIRTUAL を指定した場合は、LPCK は論理子セグメントに保管されません。論理親が HISAM データベース内にある論理子セグメントに対しては、PHYSICAL を指定する必要があります。また、論理親の連結キーの任意の部分を使用して物理兄弟にチェーンに配列されている論理子セグメントに対しても指定する必要があります。

- PHDAM および PHIDAM
  - PHDAM および PHIDAM に対しては PHYSICAL がデフォルトです。
  - VIRTUAL を PHDAM または PHIDAM に対して指定した場合、それは無視され、PHYSICAL が使用されます。
- HDAM および HIDAM
  - HDAM および HIDAM に対しては VIRTUAL がデフォルトです。
  - HDAM および HIDAM データベース内のシンボリック・ポインターは、LPCK を使用し、PHYSICAL の指定を必要とします。

ALTER TABLE data capture changes (change_syntax) のキーワード・パラメーター

ALTER TABLE data capture changes (change_syntax) ステートメントには、以下のキーワード・パラメーターが定義されています。

DATA CAPTURE
CREATE DATABASE ステートメントで DATA CAPTURE を指定すると、これらのオプションは物理データベース内のすべてのテーブルに適用されます。このパラメーターを CREATE ステートメントまたは ALTER TABLE ステートメントで指定した場合、このステートメントの指定はオーバーライドされます。

以下の物理データベースは DATA CAPTURE をサポートします。

- HISAM
- SHISAM
- HDAM
- PHDAM
- HIDAM
- PHIDAM

- DEDB

NONE
データ・キャプチャー・オプションがないことを示します。

CHANGES
任意の数の出口ルーチンを、各ルーチン独自の変更オプションのセットと一緒に指定できます。出口ルーチンを指定しない場合、ロギングに関する 1 セットの変更オプションのみを指定できます。このメソッドは、DBD マクロ・ステートメントの EXIT= パラメーターで出口ルーチン名の代わりにアスタリスク (*) を指定することと同じです。各セットは、コンマで分離します。NOCASCADE は、任意の組み合わせの C* (例えば、CKEY) オプションと相互に排他的です。

以下のオプションは DATA CAPTURE CHANGES に有効です。

BEFORE | NOBEFORE
REPL 呼び出しの場合に、変更前データが X'99' ログ・レコードに組み込まれます。BEFORE がデフォルトです。この属性は、DEDB の場合のみ有効です。

CDATA | CNODATA
カスケード削除の場合に、セグメント・データを出口ルーチンに渡します。また、CDATA は削除されるセグメントを識別します (物理連結キーで識別できない場合)。この属性は、NOCASCADE とは相互に排他的です。

CKEY | CNOKEY
物理連結キーを出口に渡します。このキーは、カスケード削除で削除されるセグメントを識別します。この属性は、NOCASCADE とは相互に排他的です。

CNOPATH | CPATH
出口ルーチンが物理ルートの階層パスにあるセグメント・データを必要としないことを示します。CNODATA は、カスケード削除に必要な相当量のパス・データを除去するために使用します。この属性は、NOCASCADE とは相互に排他的です。

DATA | NODATA
DATA は、更新用に物理テーブル・データが出口ルーチンに渡されることを指定します。DATA が指定され、EDITPROC 出口ルーチンもテーブルで使用されている場合、渡されるデータは拡張されたデータです。 DATA がデフォルトです。

DLET | NODLET
DLET 呼び出しの場合に、X'99' ログ・レコードが書き込まれます。DLET がデフォルトです。この属性は、DEDB の場合のみ有効です。

exitname
データを処理する出口ルーチンの名前を指定します。この名前は、ユーザーが IMS に対して定義したデータ・キャプチャー出口ルーチンの名前に一致する必要があります。最大 8 文字の英数字を使用できます。

KEY | NOKEY
KEY は、出口ルーチンに物理連結キーを渡すことを指定します。このキーは、アプリケーションによって更新される物理テーブルを識別します。 KEY がデフォルトです。

NOCASCADE
DL/I がこのセグメントを削除するときに、出口ルーチンは呼び出されないことを示します。従属セグメントを持たないセグメントを削除する場合には、カスケード削除は不要です。

NOFLD | FLD
FLD オプションは、DEDB FLD 呼び出しによって行われる更新をキャプチャーするよう要求します。このオプションは DEDB に対してのみ有効で、この情報は、オプション・ログが指定された場合に X'9904' ログ・レコードにのみ記録されます。この情報はデータ・キャプチャー出口には渡されません。この属性は、DEDB の場合のみ有効です。

NOINPOS | INPOS
INPOS オプションは、HERE の挿入規則が使用されて F や L のコマンド・コードが使用されていないときに、ISRT がキーなしセグメントまたは一意でないキー付きセグメントに対して行われた場合に、ツイン・データを渡すよう要求します。ツイン・データ IMS は ISRT がキャプチャーされる前の時点に位置指定されます。

NOLOG | LOG
LOG オプションは、データ・キャプチャーの制御ブロックとデータを IMS システム・ログに書き込むよう要求します。

NOPATH | PATH
NOPATH は、出口ルーチンが物理ルートの階層パスにあるテーブルからデータを必要としないことを示します。NOPATH は、パス・データの検索に必要な処理時間を回避するための効率的な方法です。
NOPATH がデフォルトです。

PATH は、更新されたセグメントのために物理ルートの階層パスにある各セグメントからのデータを、出口ルーチンに渡す必要がある場合に指定できます。アプリケーションが、挿入、置換、または削除の目的で複数のセグメントを別々にアクセスできるようにするには、PATH を使用します。

DB2® for z/OS の 1 次キーを構成するためにパス内のテーブルからの情報が必要なときは、PATH オプションを使用できます。その場合、DB2 for z/OS の 1 次キーは、従属テーブルの更新のための伝搬要求で使用されます。一般に、この種のテーブル情報が必要になるのは、親がキー情報を含んでおり、従属テーブルが親テーブルには収まらない追加データを含んでいる場合です。

PATH は、追加処理が必要な場合にも使用できます。例えば、D コマンド・コードを呼び出さなかった場合などに、1 つの呼び出しで複数のテーブルにアクセスしないということがあります。その場合、アプリケーションが別々の呼び出しで各テーブルにアクセスする際、追加処理が必要になります。

NOSSPCMD | SSPCMD
SSPCMD オプションは、DEDB サブセット・ポインター・コマンド・コードをキャプチャーするよう要求します。このオプションは DEDB の場合にのみ有効です。

ALTER TABLE (alter-options) のキーワード・パラメーター

すべての table-options の説明は、ALTER TABLE ステートメントで定義されています。すべての table-options は、任意指定であり、指定されない場合にデフォルト値を意味するものではありません。 table-option の指定は、既存の値に置き換わる新規値が提供されていることを意味します。

DATA CAPTURE キーワードを指定すると、すべてのデータ・キャプチャー出口 (以前に提供されたものがある場合) が置き換わります。複数のデータ・キャプチャー出口が以前に提供されていた場合、必要に応じてそのすべてを再指定する必要があります。

制約事項: ALTER TABLE ステートメントでは、以下の節を 1 回だけ指定することができます。

- ADD LCHILD
- ADD MAP
- DROP COLUMN
- DROP LCHILD
- DROP MAP
- RENAME COLUMN

ADD COLUMN 節と ALTER COLUMN 節は、ALTER TABLE ステートメントで複数回指定することができます。

ALTER TABLE (alter-options) ステートメントには、以下のキーワード・パラメーターが定義されています。

ADD [COLUMN] ***column-definition***
表に列を追加します。表内の列は固有でなければなりません。

column_name
column_name は、IMS カタログのみに保管され、定義するデータベースには保管されていない外部名を表します。外部名は 1 文字から 128 文字の大文字の英数字ストリングとして指定します。外部名には下線文字を含めることができます。列名はセグメント内で固有でなければなりません。

制約事項: 列名は、予約済み SQL キーワードにすることも、DFS で開始することもできません。

IMS Universal ドライバーによって制限されている予約済み SQL キーワードのリストについては、IMS JDBC ドライバーにより制限されるポータブル SQL キーワード (アプリケーション・プログラミング)を参照してください。

INTERNALNAME ***internalname***
セグメント・タイプ内のこのフィールドの名前を指定します。この名前は、アプリケーション・プログラムによって DL/I 呼び出し SSA で参照することがで

きます。フィールド名はセグメント定義内で固有でなければなりません。
fldname1 値は、1 文字から 8 文字の英数字値にする必要があります。以下のタイプのフィールドでは INTERNALNAME パラメーターは必須です。

- SEQ パラメーターを指定するキー順フィールド・タイプ
- セグメント検索引数 (SSA) によって参照されるフィールド・タイプ
- センシティブ・フィールドとして PSB が参照するフィールド・タイプ。
- XDFLD によって参照されるフィールド・タイプ

その他のフィールド・タイプでは、INTERNALNAME パラメーターを省略することができます。 INTERNALNAME パラメーターを省略すると、データベースのデータ管理ブロック (DMB) 内のストレージを節約できます。ただし、このフィールドでの検索を可能にするには INTERNALNAME パラメーターを指定する必要があります。以下のタイプのフィールドでは INTERNALNAME パラメーターを指定できません。

- 配列として定義されているフィールド。配列として定義されているフィールドには、フィールド定義に ARRAY が含まれています。
- 配列エレメントとして定義されているフィールド。配列エレメントであるフィールドは、列内の IN キーワードに配列フィールドの名前を指定します。
- 1 つ以上のネストされた動的配列を含む構造として定義されたフィールド。構造として定義されているフィールドには、列に STRUCT が含まれています。
- 動的配列を含む構造内に一緒に含まれているフィールド。構造内に含まれているフィールドは、列内の IN キーワードに構造フィールドの名前を指定します。
- セグメント内で動的配列の後に続くフィールド。動的配列の後に続くフィールドは、STARTAFTER パラメーターを指定します。
- 別のフィールドの開始位置に相対させた開始位置を指定する RELSTART パラメーターを含むフィールド。
- XML によって定義されたフィールド。

/CK 列および /SX 列の場合、INTERNALNAME パラメーターが指定されている必要があります。 /CK 名または /SK 名を指定する場合、それらの名前は二重引用符 (") で囲む必要があります。

- HSAM、SHSAM、INDEX、PSINDEX、および DEDB は、/CK 列も /SX 列も許可しません。
- HISAM および SHISAM は、/CK 列のみを許可します。
- HDAM、HIDAM、PHDAM、および PHIDAM は、/CK 列と /SX 列を許可します。

ALTER [COLUMN] ***column-alteration***
既存の列の定義を変更します。指定した属性のみが変更され、その他の属性はそのまま変更されません。以後の列の値は、ALTER TABLE ALTER COLUMN ステートメントによる変更の影響を受けます。 column-alteration 属性は、CREATE TABLE セクションで定義されている column-definition 属性と類似していますが、属性の変更が必要な場合にのみ指定が必要であるという点で異なります。

RENAME [COLUMN] ***source-column-name*** **TO** ***target-column-name***
指定された列の名前を変更します。この名前は修飾してはなりません。

source-column-name
名前を変更する対象の列を指定します。この名前は表の既存の列を示すものでなければなりません。

target-column-name
列の新規名を指定します。この名前は、表内に既に存在する列を示すものであってはなりません。

DROP [COLUMN] ***column-name RESTRICT***
指定の列を表からドロップします。

ADD ***lchild-definition***
lchild を追加します。これらのオプションの説明については、 812 ページの『CREATE TABLE』の lchild 定義を参照してください。

DROP LCHILD ***table_name***
指定された lchild を表および関連する xdfld (存在する場合) からドロップします。table_name パラメーターは、論理子の IMS 内部名を指定します。

ALTER TABLE (column-definition) のキーワード・パラメーター

ALTER TABLE (column-definition) ステートメントには、以下のキーワード・パラメーターが定義されています。

column_name
column_name は、IMS カタログのみに保管され、定義するデータベースには保管されていない外部名を表します。外部名は 1 文字から 128 文字の大文字の英数字ストリングとして指定します。外部名には下線文字を含めることができます。列名はセグメント内で固有でなければなりません。

制約事項: 列名は、予約済み SQL キーワードにすることも、DFS で開始することもできません。

IMS Universal ドライバーによって制限されている予約済み SQL キーワードのリストについては、IMS JDBC ドライバーにより制限されるポータブル SQL キーワード (アプリケーション・プログラミング)を参照してください。

INTERNALNAME ***internalname***
セグメント・タイプ内のこのフィールドの名前を指定します。この名前は、アプリケーション・プログラムによって DL/I 呼び出し SSA で参照することができます。フィールド名はセグメント定義内で固有でなければなりません。
fldname1 値は、1 文字から 8 文字の英数字値にする必要があります。以下のタイプのフィールドでは INTERNALNAME パラメーターは必須です。
- SEQ パラメーターを指定するキー順フィールド・タイプ
- セグメント検索引数 (SSA) によって参照されるフィールド・タイプ
- センシティブ・フィールドとして PSB が参照するフィールド・タイプ。
- XDFLD によって参照されるフィールド・タイプ

その他のフィールド・タイプでは、INTERNALNAME パラメーターを省略することができます。 INTERNALNAME パラメーターを省略すると、データベースのデータ管理ブロック (DMB) 内のストレージを節約できます。ただし、この

フィールドでの検索を可能にするには INTERNALNAME パラメーターを指定する必要があります。以下のタイプのフィールドでは INTERNALNAME パラメーターを指定できません。

- 配列として定義されているフィールド。配列として定義されているフィールドには、フィールド定義に DATATYPE=ARRAY が含まれています。
- 配列エレメントとして定義されているフィールド。配列エレメントであるフィールドは、FIELD ステートメントの PARENT パラメーターに配列フィールドの名前を指定します。
- 1 つ以上のネストされた動的配列を含む構造として定義されたフィールド。構造として定義されているフィールドには、フィールド定義に DATATYPE=STRUCT が含まれています。
- 動的配列を含む構造内に一緒に含まれているフィールド。構造内に含まれているフィールドは、FIELD ステートメントの PARENT パラメーターに構造フィールドの名前を指定します。
- セグメント内で動的配列の後に続くフィールド。動的配列の後に続くフィールドは、STARTAFTER パラメーターを指定します。
- 別のフィールドの開始位置に相対させた開始位置を指定する RELSTART パラメーターを含むフィールド。
- DATATYPE=XML によって定義されたフィールド。

ALTER TABLE (datatype) のキーワード・パラメーター

ALTER TABLE (datatype) ステートメントには、以下のキーワード・パラメーターが定義されています。

ARRAY | BINARY | ...
フィールドの外部データ型を指定する、3 から 9 文字のオプションの英数字フィールド。

DATATYPE パラメーターに DECIMAL が指定されている場合、デフォルトの INTERNAL TYPECONVERTER は符号付き PACKEDDECIMAL です。

DATATYPE パラメーターに DATE、TIME、または TIMESTAMP が指定されている場合は、列定義の INTERNAL TYPECONVERTER パラメーターに LONG または CHAR を指定するか、USER TYPECONVERTER を指定する必要があります。このフィールドに列定義が含まれていない場合、INTERNAL TYPECONVERTER LONG がデフォルトになります。LONG が使用されると、その値は 1970 年 1 月 1 日からのミリ秒数として DASD に保管されます。

DATATYPE パラメーターに XML を指定すると、デフォルトの INTERNAL TYPECONVERTER は XML_CLOB です。これは、DATATYPE=XML が指定されている場合に有効な唯一の値です。

DATATYPE パラメーターに STRUCT または ARRAY が指定されている場合、デフォルトの INTERNAL TYPECONVERTER はそれぞれ STRUCT または ARRAY です。これらは、DATATYPE パラメーターにこのいずれかが指定されている場合に、それぞれ有効な唯一の値です。

その他の DATATYPE の値ではすべて、その値がデフォルトの INTERNAL TYPECONVERTER として使用されます。

TYPE=C の場合、DATATYPE のデフォルトは CHAR になります。TYPE パラメーターの他の指定ではすべて、DATATYPE のデフォルトは BINARY になります。

有効値は以下のとおりです。

ARRAY
ARRAY が指定された場合は、以下のようになります。

- INTERNALNAME パラメーターはサポートされません。
- BYTES または MAXBYTES パラメーターに指定するバイト値は、その配列に含まれるすべてのフィールドのバイト数の合計以上でなければなりません。

MSDB データベース・タイプでは ARRAY データ型はサポートされません。

ARRAY として定義されたフィールド、または ARRAY を含むフィールドは再定義できません。

配列として定義されているフィールドには、フィールド定義に DATATYPE=ARRAY が含まれています。

配列エレメントであるフィールドは、FIELD ステートメントの PARENT パラメーターに配列フィールドの名前を指定します。

BINARY
TYPE=P または TYPE=X が指定されている場合、BINARY が DATATYPE パラメーターのデフォルト値です。

BIT
BIT を指定する場合は、BYTES=1 も指定する必要があります。

BYTE
BYTE を指定する場合は、BYTES=1 も指定する必要があります。

UBYTE
UBYTE を指定する場合は、BYTES=1 も指定する必要があります。

CHAR
TYPE=C が指定されている場合、CHAR が DATATYPE パラメーターのデフォルト値です。

日付
DATE が指定されている場合、INTERNAL TYPECONVERTER CHAR または USER TYPECONVERTER *convertername* のいずれかが含まれている列定義も指定しない限り、BYTES=8 も指定する必要があります。

DECIMAL(*pp*,*ss*)

pp 精度。0 より大きい 1 バイトから 2 バイトの数値フィールド。

ss 位取り。0 以上の、1 バイトから 2 バイトの数値フィールド。 *ss* に指定する値は、*pp* の値より大きくてはなりません。

BYTES パラメーターには、使用されている 10 進数フォーマットと一致する値を指定する必要があります。

デフォルトの 10 進数フォーマットは符号付きパック 10 進数です。符号付きパック 10 進数フォーマットの場合に BYTES パラメーターとして必要な値を計算するには、*length* = ceiling ((*pp* + 1) / 2) という式を使用します。

デフォルトの 10 進数フォーマットは、INTERNAL TYPECONVERTER パラメーターを指定して変更できます。

INTERNAL TYPECONVERTER と指定してゾーン 10 進フォーマットを使用する場合は、*length* = *pp* という式を使用して、BYTES パラメーターの値を計算します。

DOUBLE
DOUBLE を指定する場合は、BYTES=8 も指定する必要があります。

FLOAT
FLOAT を指定する場合は、BYTES=4 も指定する必要があります。

INT
INT を指定する場合は、BYTES=4 も指定する必要があります。

UINT
UINT を指定する場合は、BYTES=4 も指定する必要があります。

LONG
LONG を指定する場合は、BYTES=8 も指定する必要があります。

ULONG
ULONG を指定する場合は、BYTES=8 も指定する必要があります。

OTHER
ユーザー定義のデータ型を使用することを指定します。OTHER を指定する場合は、ユーザー提供のタイプ・コンバーターを指定する USER TYPECONVERTER パラメーターを含む列定義も指定する必要があります。

SHORT
SHORT を指定する場合は、BYTES=2 も指定する必要があります。

USHORT
USHORT を指定する場合は、BYTES=2 も指定する必要があります。

STRUCT
STRUCT を指定するときは、この構造フィールドに子として動的配列フィールドが含まれる場合、SEQ パラメーターを同時に指定することはできません。動的配列フィールドは、DATATYPE=ARRAY と、さらに DEPENDSON や MAXBYTES パラメーターなどを指定して定義されます。

また、BYTES または MAXBYTES パラメーターに指定するバイト値は、その構造に含まれるすべてのフィールドのバイト数の合計以上でなければなりません。

MSDB データベース・タイプでは STRUCT データ型はサポートされません。

TIME
TIME が指定されている場合、INTERNAL TYPECONVERTER CHAR ま

たは USER TYPECONVERTER *convertername* のいずれかが含まれている列定義も指定しない限りは、BYTES=8 も指定する必要があります。

TIMESTAMP
TIMESTAMP が指定されている場合、INTERNAL TYPECONVERTER CHAR または USER TYPECONVERTER *convertername* のいずれかが含まれている列定義も指定しない限りは、BYTES=8 も指定する必要があります。

XML

制約事項: NAME パラメーターが指定されている場合、DATATYPE=XML はサポートされません。

ALTER TABLE (ims-column-syntax) のキーワード・パラメーター

ALTER TABLE (ims-column-syntax) ステートメントには、以下のキーワード・パラメーターが定義されています。

BYTES *bytes*
定義するフィールドの長さをバイトで指定します。システム関連フィールド以外のフィールドでは、BYTES は、値が 255 を超えない有効な自己定義項でなければなりません。

索引ソース・セグメント・タイプの連結キーまたはその連結キーの一部分をシステム関連フィールドとして定義する場合には、指定する値を 255 より大きくすることができますが、索引ソース・セグメントの連結キーの長さを超えてはなりません。

バイト長が 255 を超える可能性があるのは、IMS が検索できないように列が定義されている場合です。これらの列は、1 次キーとして定義することができず、INTERNALNAME キーワードを指定することができません。

/SX システム関連フィールドの長さは常に 4 バイトです。したがって、BYTES パラメーターは指定しても無視されます。

このフィールドが STRUCT または ARRAY によって構造または配列として定義されている場合、BYTES に指定する値は、その構造または配列に含まれるすべてのフィールドのバイト数の合計以上でなければなりません。

XML の場合、BYTES パラメーターはオプションであり、BYTES の有効値の範囲は 0 からそのセグメントの最大サイズまでです。XML の場合に BYTES パラメーターを省略すると、BYTES および MAXBYTES は許可されません。

CCSID *encoding*
列内の文字データのエンコードを指定する、単一引用符に囲まれた 1 文字から 25 文字のオプション・フィールド。これは、INTERNAL TYPECONVERTER が CHAR である場合にのみ有効です。

指定する値には、以下の文字を含めることができません。
- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

列に指定されていない場合、デフォルト値は、表で指定された値、あるいは (表で指定されない場合は) データベースで指定された値によって決まります。パラメーターが表でもデータベースでも指定されない場合、デフォルト値は Cp1047 であり、これは EBCDIC エンコードを指定します。

DEPENDSON=
動的配列内のエレメントの数を定義するフィールドの名前を指定します。参照されるフィールドの FIELD ステートメントは、DEPENDSON パラメーターを指定する FIELD ステートメントの前に配置する必要があります。指定する名前は、参照されるフィールドの定義内の EXTERNALNAME パラメーターの値 (明示的に定義された値であるか、デフォルトで受け入れた値であるかに関係なく) であることが必要です。

DEPENDSON パラメーターは、DATATYPE=ARRAY も指定されている場合にのみ有効です。MINOCCURS の値と MAXOCCURS の値が異なる場合、DEPENDSON は必須です。

DEPENDSON パラメーターによって参照されるフィールドは、以下のいずれかの DATATYPE 値を指定して定義する必要があります。

- INT
- SHORT
- LONG
- (pp) または (pp,ss) が指定された DECIMAL。ss は 0 または 00。

TYPE {C | X | P}
このフィールド内のデータのマスクまたは埋め込みに IMS が使用する、文字のタイプを決定します。

C 英数字データまたはデータのタイプの組み合わせを指定します。 C を指定すると、IMS がフィールドの未使用バイトを充てんする必要がある場合、IMS は値を左寄せし、値の右側の未使用バイトを X'40' によって充てんします。例えば、5 バイト・フィールド内の 3 バイト値 X'F5F4F3' は、X'F5F4F34040' として書き出されます。

X 16 進データを指定します。X を指定すると、IMS がフィールドの未使用バイトを充てんする必要がある場合、IMS は値を右寄せし、値の左側の未使用バイトを X'00' によって充てんします。例えば、5 バイト・フィールド内の 3 バイト値 X'543210' は、X'0000543210' として書き出されます。

P パック 10 進数データ。P を指定すると、IMS がフィールドの未使用バイトを充てんする必要がある場合、IMS は値を右寄せし、値の左側の未使用バイトを X'00' によって充てんします。例えば、5 バイト・フィールド内の 3 バイト値 X'54321C' は、X'000054321C' として書き出されます。

MAXBYTES *max_array_bytes*
フィールド・インスタンスのバイト長が動的配列内のエレメント数に応じて異なる可能性がある場合に、フィールドの最大サイズをバイト単位で指定します。MAXBYTES と BYTES は同時には指定できません。

MAXBYTES の値は、このフィールドの下にネストされている、すべてのフィールドのバイト値の最大合計値以上でなければなりません。

MAXBYTES パラメーターが必須かつ有効であるのは、以下の場合のみです。

- フィールドが動的配列として定義されている場合。フィールドが動的配列であるのは、配列内のエレメント数がそのフィールドのインスタンスごとに異なる可能性がある場合です。動的配列の定義内で DEPENDSON パラメーターは、動的配列のインスタンスの配列エレメント数を定義する、セグメント定義内の別のフィールドを参照します。
- 動的配列として定義済みのネストされたフィールドを含む、静的配列または構造として定義されたフィールドの場合。

MSDB データベース・タイプでは MAXBYTES パラメーターはサポートされません。

IN *column_name*
このフィールドを含む構造または配列として定義されているフィールドの名前を指定します。参照されるフィールドは、DATATYPE=ARRAY または DATATYPE=STRUCT のいずれかを指定して定義されている必要があります。

INTERNAL TYPECONVERTER
IMS データをアプリケーション・プログラムが要求するデータ型に変換するために IMS が使用する、内部変換ルーチンを指定します。

INTERNAL TYPECONVERTER または USER TYPECONVERTER のいずれかを指定することができますが、両方を指定することはできません。INTERNAL TYPECONVERTER と USER TYPECONVERTER は、相互に排他的です。

INTERNAL TYPECONVERTER パラメーターの有効な値は、以下のとおりです。

ARRAY
配列は、繰り返されるエレメントが含まれるデータ構造です。

BINARY
2 進整数には、短精度整数、長精度整数、および 64 ビット整数があります。 2 進数は整数の厳密な表現です。

BIT
BIT を指定する場合、対応する列に BYTES 1 も指定する必要があります。

BLOB
BLOB の値を INTERNALTYPECONVERTER パラメーターに指定できるのは、前の FIELD ステートメントに DATATYPE=BINARY が指定されている場合のみです。

BYTE
BYTE を指定する場合、対応する列に BYTES 1 も指定する必要があります。

UBYTE
UBYTE を指定する場合、対応する列に BYTES 1 と、DATATYPE BYTE または DATATYPE UBYTE のどちらかも指定する必要があります。

CHAR
CHAR の値を INTERNALTYPECONVERTER パラメーターに指定できるのは、前の FIELD ステートメントの DATATYPE パラメーターに CHAR、DATE、TIME、または TIMESTAMP が指定されている場合のみです。

DOUBLE
DOUBLE を指定する場合、対応する列に BYTES 8 も指定する必要があります。

FLOAT
FLOAT を指定する場合、対応する列に BYTES 4 も指定する必要があります。

INT
INT を指定する場合、対応する列に BYTES 4 も指定する必要があります。

UINT
UINT を指定する場合、対応する列に BYTES 4 および DATATYPE INT または DATATYPE UINT も指定する必要があります。

LONG
LONG を指定する場合、対応する列に BYTES 8 も指定する必要があります。

ULONG
ULONG を指定する場合、対応する列に BYTES 8 および DATATYPE LONG または DATATYPE ULONG も指定する必要があります。

PACKEDDECIMAL
PACKEDDECIMAL は、IMS Universal JDBC ドライバーおよび IMS Universal DL/I ドライバーのデータ・タイプ拡張子です。

SHORT
SHORT を指定する場合、対応する列に BYTES 2 も指定する必要があります。

USHORT
USHORT を指定する場合、対応する列に BYTES 2 および DATATYPE SHORT または DATATYPE USHORT も指定する必要があります。

STRUCT

XML_CLOB
文字ラージ・オブジェクト (CLOB) は、データベース管理システム内の文字データの集合です。

ZONEDDECIMAL
ZONEDDECIMAL は、IMS Universal JDBC ドライバーおよび IMS Universal DL/I ドライバーのデータ・タイプ拡張子です。 DATATYPE DECIMAL を指定する必要があります。

INTERNAL TYPECONVERTER を指定する場合、DATATYPE パラメーターも指定する必要があります。

INTERNAL TYPECONVERTER パラメーターで指定する値は、DATATYPE パラメーターで指定する値と整合している必要があります。ほとんどの場合、INTERNAL TYPECONVERTER には、DATATYPE パラメーターに指定した値と同じ値を指定する必要があります。

ISSIGNEDYES | ISSIGNEDNO
このパラメーターは、DATATYPE DECIMAL にのみ有効です。デフォルトは ISSIGNEDYES です。

MINOCCURS *min_array_elements*
DECIMAL データ・タイプの場合、デフォルトの INTERNAL TYPECONVERTER は符号付き PACKEDDECIMAL です。

MAXOCCURS *max_array_elements*
ARRAY の場合のみに、ARRAY 内のエレメントの最大数を指定する必須の数値。MAXOCCURS は必ず MINOCCURS 以上とし、ゼロであってはなりません。

OVERFLOW *table_name*
XML 文書を保持するために定義された列に収まらない XML 文書の部分を保管するために使用できる従属表の名前 (1 文字から 8 文字)。従属表の親は、XML データ列を含む表です。 OVERFLOW は、XML_CLOB データに DATATYPE XML を指定した列のみに適用されます。

PATTERN
日付、時刻、およびタイム・スタンプの Java データ・タイプ用のパターンを指定する、単一引用符で囲まれた 1 文字から 50 文字のオプション・フィールド。

PATTERN は、DATATYPE キーワードで DATE、TIME、または TIMESTAMP が指定され、INTERNAL TYPECONVERTER キーワードで CHAR が指定されている場合にのみ適用されます。 PATTERN はこれ以外のデータ型では無効です。

パターンは大/小文字が区別され、必ず単一引用符で囲む必要があります。

キーワード値の区切り文字として使用される単一引用符を除き、PATTERN キーワードに指定する値に以下の文字を含めることはできません。
- 単一引用符および二重引用符
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

指定できるパターンは、Java クラス java.text.SimpleDateFormat によって定義されます。DDL では、PATTERN に入力した値が Java で定義されたパターンに従っているかどうかはチェックされません。

例えば、`yyyy.MM.dd` という Java 形式を入力すると、結果として得られる時刻形式は「2013.01.01」になります。

PROPERTIES *name=value*
USER TYPECONVERTER パラメーターに指定されたユーザー・タイプ・コンバーターに対してプロパティーを指定します。これらのプロパティーが、ユーザー・タイプ・コンバーターに渡されます。

PROPERTIES パラメーターは、USER TYPECONVERTER が指定されている場合にのみ有効です。

PROPERTIES キーワードに指定される名前とプロパティーは、大/小文字を区別します。また、単一引用符で囲む必要があります。

PROPERTIES キーワードでは以下の文字はサポートされません。

- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

プロパティー名の最大長は 128 文字です。プロパティー値の最大長も 128 文字です。

形式は次のとおりです。

```
PROPERTIES=('name1' = 'value1' , 'name2' = 'value2')
```

例えば、次のとおりです。

```
PROPERTIES=('DOG' = 'BUTCH' , 'CAT' = 'LUCY')
```

RELSTART ***relative_start_position***
配列 (または状況によっては構造) のエレメントとして定義されているフィールドの開始位置を指定します。有効な値は 1 から 32767 です。

RELSTART に指定する値は、配列または構造の開始位置に相対させた、そのフィールドの開始バイト・オフセットです。例えば、配列内の最初のフィールドは、そのフィールドを含む配列がセグメントのバイト 50 で開始する場合でも、通常は RELSTART 1 を指定します。

配列フィールドを親として指定するフィールドでは、RELSTART は必須です。

構造を親として指定するフィールドでは、その構造フィールドが RELSTART または STARTAFTER を使用して定義されている場合、RELSTART は必須です。

次の例では、フィールド DYNARRAY は動的配列です。フィールド STRUCT01 は構造です。フィールド FLD03 および FLD04 は両方とも STRUCT01 を親として指定しています。セグメント内で動的配列が STRUCT01 の前に配置されているため、FLD03 と FLD04 の開始オフセットは STRUCT01 の開始位置との相対によってのみ指定できます。

```
FIELD EXTERNALNAME=ARRAYNUM,DATATYPE=DECIMAL(7,0),START=1,BYTES=4
FIELD EXTERNALNAME=DYNARRAY,DATATYPE=ARRAY,START=5,MAXBYTES=100
      MINOCCURS=10,MAXOCCURS=50,DEPENDSON=ARRAYNUM
FIELD EXTERNALNAME=FLD01,RELSTART=1,BYTES=2,PARENT=DYNARRAY
FIELD EXTERNALNAME=FLD02,STARTAFTER=DYNARRAY,BYTES=10
FIELD EXTERNALNAME=STRUCT01,DATATYPE=STRUCT,STARTAFTER=FLD02,BYTES=10
FIELD EXTERNALNAME=FLD03,RELSTART=1,BYTES=5,PARENT=STRUCT01
FIELD EXTERNALNAME=FLD04,RELSTART=6,BYTES=5,PARENT=STRUCT01
```

START、STARTAFTER、および RELSTART は同時には指定できません。

START ***start_position***
定義するフィールドの開始位置をセグメントの先頭からの相対的なバイトで指定します。 START の値は、値が 32767 を超えない数値項でなければなりません。セグメントの最初のバイトの開始位置は 1 です。可変長セグメントの場合、最初の 2 バイトはセグメントの長さを含みます。したがって、実際、ユーザー・データ・フィールドが始まるのは 3 バイトからです。フィールドをオーバーラップすることが許されています。論理子セグメントを定義する場合、セグメント・タイプの最初の *n* バイトは、論理親または物理親の連結キーです。位置 1 から開始するフィールドは、そのフィールドの全部または一部を定義します。位置 n+1 から開始するフィールドは、交差データで始まります。

START をシステム関連フィールドに使用すると、連結キーの一部を索引ソース・セグメント・タイプ内のフィールドとして記述できます。この方法で使用すると、START は、連結キーの関連部分の開始位置を、連結キーの先頭から相対的に指定します。この場合、連結キーの最初のバイトは、位置 1 であると見なされます。これは、値が、連結キーの長さ + 1 を超えない数値項でなければなりません。**BYTES** パラメーターに指定された値を引いてください。/SX システム関連フィールドの開始位置パラメーターは無視されます。

START、STARTAFTER、および RELSTART は同時には指定できません。

XML の場合、START パラメーターはオプションであり、START 0 の指定が可能です。XML の場合に START パラメーターを省略すると、START 0 がデフォルトになります。

STARTAFTER ***field_name***
フィールドが動的配列の後に開始するために、そのフィールドの開始バイト・オフセットを計算できない場合に、セグメント内でこのフィールドの直前に配置されるフィールドの名前を指定します。この名前は、INTERNALNAME キーワードで指定した名前にすることはできません。

STARTAFTER が必須かつ有効なのは、動的配列として定義されたフィールドが前のオフセットでそのフィールドの前に配置されているために、そのフィールドの開始位置を計算できない場合のみです。

動的配列があると、セグメント内の後続のフィールドの開始オフセットの計算が不可能になります。これは、動的配列のバイト長がセグメントのインスタンスごとに異なる場合があるためです。動的配列フィールドの列は、DEPENDSON および MAXBYTES パラメーターを含めることによって識別できます。

STARTAFTER パラメーターは、配列フィールドを親として定義するフィールドでは指定できません。 RELSTART パラメーターを代わりに指定してください。

START、STARTAFTER、および RELSTART は同時には指定できません。

URL ***xml_schema_url***
このフィールドを記述する XML スキーマを参照する URL の、単一引用符で囲まれた 1 文字から 256 文字のオプション・フィールド。

以下に例を示します。

```
URL='MySchema.xsd'
```

URL キーワードに指定する値には、以下の文字を含めることはできません。
- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

URL パラメーターは、XML_CLOB データに対して DATATYPE XML が指定されている場合にのみ適用されます。

USER TYPECONVERTER ***typeconverter***
型変換に使用されるユーザー提供 Java クラスの、単一引用符で囲まれた 1 文字から 256 文字の完全修飾 Java クラス名を指定します。

例えば、次のとおりです。

```
USER TYPECONVERTER 'class://com.ibm.ims.dli.types.PackedDateConverter'
```

USER TYPECONVERTER キーワードに指定する値には、以下の文字を含めることはできません。

- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

USER TYPECONVERTER は、INTERNAL TYPECONVERTER と相互に排他的です。

ALTER TABLE (constraint) のキーワード・パラメーター

ALTER TABLE (constraint) ステートメントには、以下のキーワード・パラメーターが定義されています。

CONSTRAINT *constraint_name*
制約の名前を指定します。制約名を指定しないと、ユニーク制約名が生成されます。名前を指定する場合には、以前に表で指定した制約のいずれの名前とも異なっていなければなりません。

PRIMARY KEY(*column_name*) NON UNIQUE
このフィールドが、セグメント・タイプ内のシーケンス・フィールドであることを表します。

NON UNIQUE
セグメント・タイプのオカレンスのシーケンス・フィールドで重複値を使用できることを示すオプション・キーワード。ルート・セグメント・タイプでは、各オカレンスのシーケンス・フィールドは、HDAM の場合を除いて、固有値を含んでいる必要があります。HDAM データベースのルート・セグメント・タイプはキー・フィールドを必要としません。キー・フィールドが定義されても、それが固有である必要はありません。

指定されない場合、セグメント・タイプのオカレンスのシーケンス・フィールドでは固有値のみを使用できます。従属セグメント・タイプでは、ある物理親セグメントのもとにある各オカレンスのシーケンス・フィールドは、固有値を含んでいなければなりません。

ALTER TABLE (references-clause) のキーワード・パラメーター

ALTER TABLE (references-clause) ステートメントには、以下のキーワード・パラメーターが定義されています。

FOREIGN KEY REFERENCES
従属セグメント・タイプの場合、このセグメントの物理親の名前を指定します。*column_name* は、形式 *x_y* に従う必要があります。ここで、*x* は親表の名前で、*y* はその表の 1 次キー列の名前です。

REFERENCES *table_name*
従属セグメントの親セグメントを指定するもので、IMS 外部表の名前です。

SINGLE|DOUBLE

現行表の物理親のすべてのオカレンスに入れる、物理子ポインターのタイプを指定します。 **SINGLE** および **DOUBLE** は、
PHDAM、PHIDAM、HDAM、HIDAM、または DEDB データベース内の表に対してのみ指定することができ、物理親が階層ポインター (HIER または HIERBWD) を指定している場合、それらは無視されます。

SINGLE を使用すると、4 バイトの物理子の先頭ポインターが、現行表の物理親のすべてのオカレンスに入れられます。 **SINGLE** はデフォルトです。

DOUBLE を使用すると、4 バイトの物理子の先頭ポインター、および 4 バイトの子の最後のポインターが、現行表の物理親のすべてのオカレンスに入れられます。

DROP PRIMARY KEY
1 次キーの定義をドロップします。

DROP FOREIGN KEY
外部キーの定義をドロップします。

ALTER TABLE (map-definition) のキーワード・パラメーター

ALTER TABLE (map-definition) ステートメントには、以下のキーワード・パラメーターが定義されます。

MAP
マップ定義の前には列定義がなければなりません。MAP ステートメントを使用すると、表内の列の代替マッピングが可能になります。制御列に関連する 1 つ以上の CASE ステートメントのグループが表内でネストされます。制御列は、表インスタンスで使用されているケースを識別します。

column_name
セグメント・インスタンスに使用されるマップ・ケースを判別する値が含まれる、この表内の列の外部名。列にこのマップの CASE ステートメントの *caseid* 値に対応する値が含まれていない場合、このマップは、この表インスタンスに使用されていません。

AS *map_name*
このマップの名前を定義する、1 文字から 128 文字のオプションの英数字フィールド。指定されない場合、IMS は、この表内で固有の名前を自動的に生成します。名前は、DFSMAP*xxxxxxxx* の形式でなければなりません。ここで、*xxxxxxxx* は増分番号です。 DFS 接頭部は、IMS によって予約されており、ユーザーが作成する名前の一部にすることはできません。

ALTER TABLE (case-definition) のキーワード・パラメーター

ALTER TABLE (case-definition) ステートメントには、以下のキーワード・パラメーターが定義されています。

CASE
CASE ステートメントは、マップ・ケースを定義します。マップ・ケースは、表内の特定のバイト範囲のオプションの代替フィールド・レイアウトを定義する列のセットです。

1 つのセグメント内の同じバイト範囲をマップするマップ・ケースは、MAP ステートメントによってグループ化されます。また MAP ステートメントは、マップ・ケースを、表定義で別個に定義された制御フィールドにリンクします。

各マップ・ケースには固有の ID があります。表のインスタンスでは、有効なマップ・ケースの ID が、セグメント作成時に制御フィールドに保管されます。

IMS Universal ドライバーが使用されていない限り、マップ・ケースによって定義されたフィールド・レイアウトを、このバイト範囲にアクセスするアプリケーション・プログラムに対し、COBOL コピーブックまたはその他のプログラミング成果物を使用して定義する必要があります。表インスタンスへのアクセス時に、アプリケーション・プログラムは制御フィールドの値を確認して、どのコピーブックを使用するかを決定します。

アプリケーション・プログラムが IMS Universal ドライバーを使用して IMS にアクセスする場合は、アプリケーション・プログラムに対してフィールド・レイアウトを定義するための追加のプログラミング成果物は必要ありません。

caseid
固有の文字または16 進ストリングを定義する 1 バイトから 128 バイトのフィールド。セグメントのフィールド構造の一部または全部がこのケースによってマップされている場合、表インスタンスはユーザー定義制御フィールドにこの *caseid* 値を指定します。

この値を文字ストリングとして指定する場合は、単一引用符内に指定する必要があります (例えば、'name01')。この値を 16 進ストリングとして指定する場合は、単一引用符内に指定し、その後に 16 進インディケーターを指定する必要があります (例えば、'00000001'x)。

caseid 値には、英数字、下線 (_)、@、$、および # を含めることができます。あるいは、16 進ストリングで指定することができます。値の長さは、ユーザー定義の制御フィールドの長さでサポートされているものでなければなりません。英数字の場合、値の長さは、必ず制御フィールドの BYTES パラメーターに指定された値以下とします。 16 進ストリングの場合、CASEID 値の長さは、必ず制御フィールドの BYTES パラメーターに指定された値の正確に 2 倍とします。

ケース ID は、そのケースの所属先マップ内で固有でなければなりません。

AS *case_name*
このケースの名前を定義する、1 文字から 128 文字のオプションの英数字フィールド。ケース名は表内で固有でなければなりません。指定されない場合、IMS は、この表内で固有の名前を自動的に生成します。名前は、DFSCASE*xxxxxxxx* の形式でなければなりません。ここで、*xxxxxxxx* は増分番号です。 DFS 接頭部は、IMS によって予約されており、ユーザーが作成する名前の一部にすることはできません。

例: COGDBD

```
DBD   NAME=COGDBD,                                          C
            ENCODING=Cp1047,                                  C
            ACCESS=(HDAM,OSAM),                               C
            RMNAME=(DFSHDC40,3,3,25),                         C
            PASSWD=NO
      DATASET  DD1=COGDATA,                                   C
```

```
|                                 DEVICE=3390,                                                C
|                                 SIZE=(8192)
|                       SEGM  NAME=ROOT,                                                      C
|                                 PARENT=0,                                                   C
|                                 BYTES=(28),                                                 C
|                                 RULES=(LLL,HERE)
|                       FIELD NAME=(ROOTKEY,SEQ,U),                                           C
|                                 BYTES=12,                                                   C
|                                 START=1,                                                    C
|                                 TYPE=C,                                                     C
|                                 DATATYPE=CHAR
|                       FIELD NAME=TABTYPE,                                                   C
|                                 BYTES=8,                                                    C
|                                 START=13,                                                   C
|                                 TYPE=C,                                                     C
|                                 DATATYPE=CHAR
|                       FIELD NAME=NEWFLD01,                                                  C
|                                 EXTERNALNAME=New_Field_01,                                  C
|                                 BYTES=8,                                                    C
|                                 START=21,                                                   C
|                                 TYPE=X,                                                     C
|                       DFSMARSH ENCODING=Cp1047,                                             C
|                                 INTERNALTYPECONVERTER=DOUBLE
|                       SEGM      NAME=TSINT,                                               C
|                                 PARENT=ROOT,                                                C
|                                 BYTES=(8,6),                                                C
|                                 REMARKS='This describes table TSINT for testing.',          C
|                                 RULES=(LLL,HERE)
|                       FIELD     NAME=RNUM,                                                C
|                                 BYTES=4,                                                    C
|                                 START=3,                                                    C
|                                 DATATYPE=INT
|                       FIELD     NAME=LL,                                                  C
|                                 BYTES=2,                                                    C
|                                 START=1,                                                    C
|                                 DATATYPE=SHORT
|                       FIELD     NAME=CSINT,                                               C
|                                 EXTERNALNAME=CSINT,                                         C
|                                 BYTES=2,                                                    C
|                                 START=7,                                                    C
|                                 DATATYPE=SHORT
|                       SEGM      NAME=TINT,                                                C
|                                 EXTERNALNAME=TESTINTEGER,                                   C
|                                 PARENT=ROOT,                                                C
|                                 BYTES=(10,6),                                               C
|                                 RULES=(PPP,FIRST)                                           C
|                                 REMARKS='This describes table TINT.',                       C
|                                 RULES=(LLL,HERE)
|                       FIELD     NAME=RNUM,                                                C
|                                 BYTES=4,                                                    C
|                                 START=3,                                                    C
|                                 DATATYPE=INT
|                       FIELD     NAME=LL,                                                  C
|                                 BYTES=2,                                                    C
|                                 START=1,                                                    C
|                                 DATATYPE=SHORT
|                       FIELD     NAME=CINT,                                                C
|                                 EXTERNALNAME=CINTEGER,                                      C
|                                 BYTES=4,                                                    C
|                                 START=7,                                                    C
|                                 DATATYPE=INT
|
|                     ALTER TABLE root (
|                         ADD COLUMN New_Field_01 DOUBLE INTERNALNAME newfld01
|                     ) IN DATABASE COGDBD
|                       MAXBYTES 28;
```

例: データベース・バージョン管理の変更

データベースバージョン管理について詳しくは、データベースのバージョン管理 (データベース管理)を参照してください。

```
DBD   NAME=COGDBD,                                              C
         ENCODING=Cp1047,                                       C
         ACCESS=(HDAM,OSAM),                                    C
         RMNAME=(DFSHDC40,3,3,25),                              C
         PASSWD=NO
DATASET  DD1=COGDATA,                                           C
         DEVICE=3390,                                           C
         SIZE=(8192)
SEGM  NAME=ROOT,                                                C
         PARENT=0,                                              C
         BYTES=(28),              <-- From 20 to 28             C
         RULES=(LLL,HERE)
FIELD NAME=(ROOTKEY,SEQ,U),                                     C
         BYTES=12,                                              C
         START=1,                                               C
         TYPE=C,                                                C
         DATATYPE=CHAR
FIELD NAME=TABTYPE,                                             C
         BYTES=8,                                               C
         START=13,                                              C
         TYPE=C,                                                C
         DATATYPE=CHAR
FIELD NAME=NEWFLD01,                                            C
         EXTERNALNAME=New_Field_01,                             C
         BYTES=8,                                               C
         START=21,                                              C
         TYPE=X,                                                C
DFSMARSH ENCODING=Cp1047,                                       C
         INTERNALTYPECONVERTER=DOUBLE


ALTER DATABASE COGDBD
  DBVER 1;

ALTER TABLE root (
    ADD COLUMN New_Field_01 DOUBLE INTERNALNAME newfld01
) IN DATABASE COGDBD
  MAXBYTES 28;
```

ALTER TABLESPACE

ALTER TABLESPACE ステートメントは、データベース内のデータ・セット・グループ、または DEDB のエリアの属性を変更します。表スペースの変更は、データベース・リソースに対する変更です。

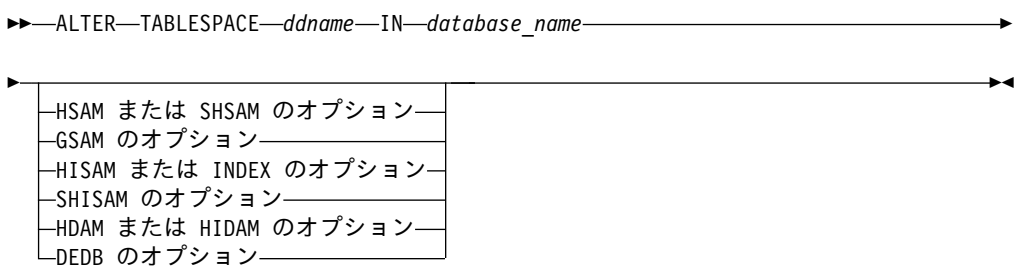
呼び出し

このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

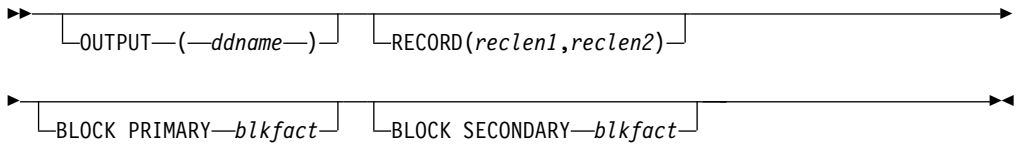
- 765 ページの『ALTER TABLESPACE 構文』
- 765 ページの『HSAM または SHSAM 構文』
- 765 ページの『GSAM 構文』
- 765 ページの『HISAM または INDEX 構文』
- 765 ページの『SHISAM 構文』

- 766 ページの『HDAM または HIDAM 構文』
- 766 ページの『DEDB 構文』

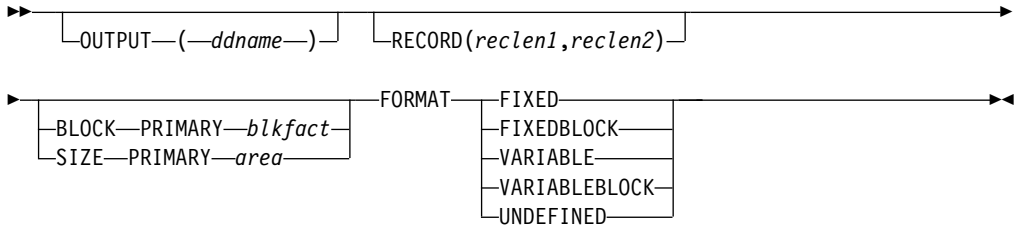
ALTER TABLESPACE 構文



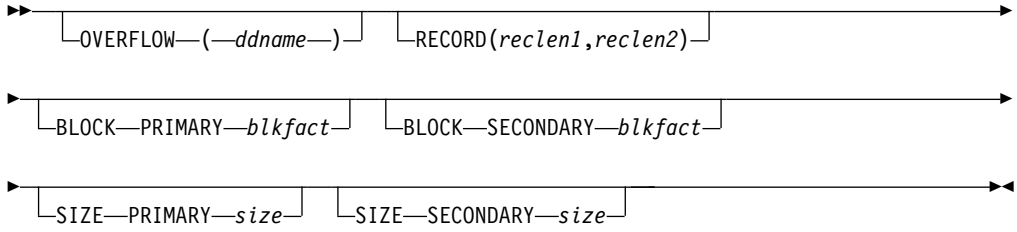
HSAM または SHSAM 構文



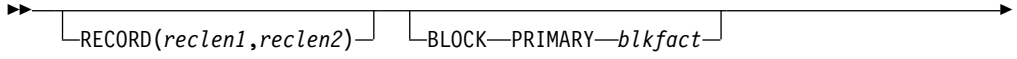
GSAM 構文

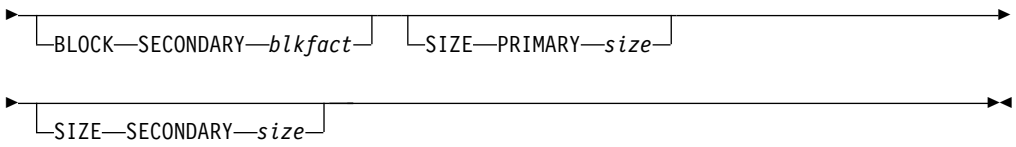


HISAM または INDEX 構文

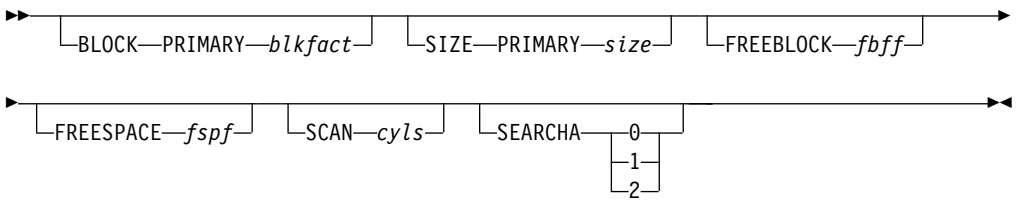


SHISAM 構文

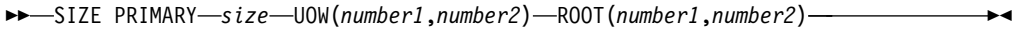




HDAM または HIDAM 構文



DEDB 構文



説明

ALTER TABLESPACE ステートメントには、以下のキーワード・パラメーターが定義されています。

ddname
変更する高速機能データベース区域またはデータ・セット・グループの 1 文字から 8 文字の DD 名。名前には、英数字文字および特殊文字 (#、@、および $) を含めることができます。

データベース区域の場合、この名前は、1 つのデータベース区域のみが含まれるデータ・セットの場合は区域名でも DD 名でも構いませんが、データ・セットに複数のデータベース区域が含まれる場合は区域名でなければなりません。データベースが DBRC に登録されている場合は、区域名を使用します。

HSAM | SHSAM | GSAM
入力データ・セットの DD 名。入力データ・セットは、アプリケーション・プログラムがデータベースからデータを取得する際に使用されます。

HISAM | SHISAM | INDEX
データ・セット・グループ内の 1 次データ・セットの DD 名。

HIDAM | HDAM
データ・セット・グループ内のデータ・セットの DD 名。

DEDB
単一エリア・データ・セットの場合は区域名または DD 名ですが、多重エリア・データ・セットの場合は区域名のみが可能です。データベースが DBRC に登録されている場合は、このパラメーターに区域名を指定してください。

IN*database_name*
このデータベース区域またはデータ・セット・グループが属しているデータベースの名前。

database_name
このデータ・セット・グループが属するデータベース。

データベースが動的に割り振られるデータベースの DBD 名を指定します。この名前は、このデータベース・パラメーター・リストを識別するための IMS.SDFSRESL 内のメンバー名として使用されます。この名前が決して IMS.SDFSRESL 内の既存メンバーと競合しないように注意する必要があります。それには IMS モジュールおよびユーザー提供の出口ルーチンが含まれます (ただし、これらに限定されません)。

BLOCK PRIMARY
BLOCK SECONDARY
ブロック化因数を指定して、HSAM、SHSAM、GSAM、HISAM、SHISAM、および索引データベースのデータ・セット・グループのデータ・セットで使用できるようにするか、またはオーバーヘッドを含まない制御インターバル・サイズまたはブロック・サイズを指定して HDAM および HIDAM データベースのデータ・セット・グループのデータ・セットを使用できるようにします。

アクセス方式として VSAM を使用する HISAM、SHISAM、および索引データベースの場合は、BLOCK パラメーターの代わりに SIZE パラメーターを使用して制御インターバル・サイズを指定してください。HISAM、SHISAM、または索引データベースに SIZE キーワードを指定した場合は、BLOCK キーワードは無効です。

RECORD および BLOCK オペランドを使用する場合は、結果として得られる制御インターバル・サイズは、8192 バイトより小さい場合は 512 の倍数でなければなりません。指定したレコード長と指定したブロック化因数の積に VSAM オーバーヘッドを加えたものが 512 の倍数でなく、8192 バイトより小さい場合には、制御インターバル・サイズは、それより大きな最も近い 512 の倍数の値に切り上げて得られます。8192 から 30720 バイト (最大許容サイズ) の制御インターバル・サイズは、2048 バイトの倍数でなければなりません。RECORD オペランドと BLOCK オペランドの積に VSAM オーバーヘッドを加えたものが、8192 から 30720 バイトの値であっても、2048 の倍数ではない場合は、制御インターバル・サイズは、それより大きな最も近い 2048 の倍数の値に切り上げて得られます。

VSAM オーバーヘッドは、ブロック化因数が 1 の場合は 7 バイトであり、そうでない場合は 10 バイトです。OSAM データ・セットの最大ブロック・サイズは 32KB です。

HDAM および HIDAM データベースの場合は、BLOCK PRIMARY パラメーターを使用して、IMS の制御インターバルやブロック・サイズの計算をオーバーライドすることができます。ただし、BLOCK PRIMARY パラメーターで指定された値のほかに、IMS は、ルート・アンカー・ポイント、フリー・スペース・アンカー・ポイント、およびアクセス方式オーバーヘッドのためのスペースを追加します。結果として得られるブロックまたは制御インターバルのサイズは、SIZE PRIMARY パラメーターの説明の中の式を参照するか、IMS の出力を調べることで判別することができます。 SIZE パラメーターが指定されておらず、アクセス方式が VSAM である場合、IMS は、CI 内の未使用スペースを CI 内の各論理レコードに均等に分配することで、最適な VSAM LRECL 値を計算します。 SIZE PRIMARY パラメーターが指定されている場合、これは実行されません。

以下の表は、BLOCK および RECORD オペランドの使用について説明します。

表 *166. BLOCK* および *RECORD* オペランド

データベース・タイプ	**BLOCK** および **RECORD** オペランドの使い方
HSAM/SHSAM	**BLOCK** BLOCK PRIMARY は、入力データ・セットに適用され、常に 1。 BLOCK SECONDARY は、出力データ・セットに適用され、常に 1。 **RECORD** *recordlength1* は、入力レコードの長さ。 *recordlength2* は、出力レコードの長さ。 HSAM/SHSAM は、常に非ブロック化される。LRECL と BLKSIZE は等しい。
GSAM	**BLOCK** BLOCK PRIMARY は、入出力データ・セットに適用される。 BLOCK SECONDARY は、無効なサブパラメーター。 **RECORD** *recordlength1* は、LRECL の長さであるか、可変長レコードの最大サイズ。 *recordlength2* は、可変長レコードの最小サイズ。 **SIZE** SIZE PRIMARY は、入出力データ・セットの BLKSIZE。 SIZE SECONDARY は、無効なサブパラメーター。
HISAM/SHISAM	**BLOCK** BLOCK PRIMARY は、1 次データ・セットのブロック化因数。 BLOCK SECONDARY は、オーバーフロー・データ・セットのブロック化因数。 **RECORD** *recordlength1* は、データ・セットの論理レコード長。 *recordlength2* は、オーバーフロー・データ・セットの論理レコード長。
HIDAM、HDAM	**BLOCK** size0 は、OSAM または VSAM データ・セット・グループのオーバーヘッドを含まないサイズ。 **RECORD** 無視されます。
DEDB	BLOCK および RECORD オペランドは無効。

表 166. *BLOCK* および *RECORD* オペランド (続き)

データベース・タイプ	**BLOCK** および **RECORD** オペランドの使い方
INDEX	**BLOCK** BLOCK PRIMARY は、1 次データ・セットのブロック化因数。 BLOCK SECONDARY は、オーバーフロー・データ・セットのブロック化因数。 **RECORD** *recordlength1* は、1 次データ・セットの論理レコード長。 *recordlength2* は、オーバーフロー・データ・セットの論理レコード長。

注**:** TABLESPACE ステートメントで *recordlength1* と *recordlength2* の両方を指定するときは、GSAM の場合を除き、*recordlength2* は *recordlength1* と等しいかそれ以上でなければなりません。

FORMAT
データ・セット内のレコードのフォーマットを指定します。有効なレコード・フォーマットは以下のとおりです。

FIXED
固定長。

FIXEDBLOCK
固定長およびブロック化。

VARIABLE
可変長。

VARIABLEBLOCK
可変長およびブロック化。

UNDEFINED
未定義長。

このキーワードは必須で、GSAM データベースのみに有効です。

FREEBLOCK
フリー・ブロック頻度因子を指定します。データベースのロードまたは再編成中に、このデータ・セット・グループ内の *n* 番目の制御インターバルまたはブロックごとに、それがフリー・スペースとして残されます。有効な範囲は 0 から 100 (1 を除く) です。デフォルトは 0 です。

値を小さくすると、データベース内のフリー・スペースの頻度が増えます。例えば、値を 2 にすると、各データの後にフリー・スペース・ブロックが生じることになります。この場合には、再編成ユーティリティーまたはロード・ユーティリティーの実行時に、フリー・スペース・ブロックのために余分の処理が必要になるので、システム・パフォーマンスが低下することになります。

FREEBLOCK は、IMS keyword FRSPC=(*fbff,*)) と同等です。

このキーワードはオプションで、HDAM または HIDAM にのみ有効です。

FREESPACE
フリー・スペース・パーセント係数を指定します。これは、フリー・スペースと

してこのデータ・セット・グループに残す各制御インターバルまたはブロックの最小パーセントです。有効な範囲は 0 から 99 です。デフォルトは 0 です。

このキーワードはオプションで、HDAM または HIDAM にのみ有効です。

FREESPACE は、IMS keyword FRSPC=(*fspf*)) と同等です。

OUTPUT (*ddname*)
出力データ・セットの DD 名 (1 文字から 8 文字の英数字) を指定します。HSAM または SHSAM データベースには必須ですが、GSAM データベースにはオプションです。この出力データ・セットは、データベースのロード時に IMS が使用します。このキーワードは、他のデータベース・アクセス・タイプには無効です。

OUTPUT は、IMS キーワード DD2= と同等です。

OVERFLOW (*ddname*)
このデータ・セット・グループ内のオーバーフロー・データ・セットの DD 名 (1 文字から 8 文字の英数字) を指定します。次のものについてはこのパラメーターを指定する必要があります。

- 固有でないキーが付いている索引ポインター・セグメントが入っている INDEX データベース。
- HISAM データベースのすべてのデータ・セット・グループ。ただし、HISAM データベース内に 1 つのセグメント・タイプしか定義されていない場合は除きます。

以下の条件が適用されます。

- 単純 HISAM (SHISAM) データベースには無効です。
- 1 つのセグメント・タイプのみが含まれる HISAM データベースには必須ではありません。
- すべての索引セグメントが索引のキー順データ・セット内に挿入されるため、索引 DBD には必須ではありません。
- アクセス・タイプ SHISAM で定義された INDEX データベースには無効です。
- HISAM および INDEX データベース・アクセス・タイプにのみ有効です。

RECORD(*recordlength1*,*recordlength2*)
このデータ・セット・グループに使用するデータ管理論理レコード長を指定します。このキーワードはオプションであり、
HSAM、SHSAM、GSAM、HISAM、SHISAM、INDEX にのみ有効です。

SCAN *cylinders*
セグメント挿入操作時に使用可能ストレージ・スペースを検索するときに、スキャンする直接アクセス装置のシリンダーの数を指定します。このパラメーターはオプションであり、HIDAM または HDAM データベースにのみ有効です。指定する場合、この値は、255 を超えない 10 進整数でなければなりません。一般的な値は 0 から 5 です。デフォルトは 3 です。0 を指定した場合、現行シリンダーのスペースのみがスキャンされます。

スキャンは、現行シリンダー位置から両方向に実行されます。スキャン限界値が原因で、現行エクステントの外側の区域がスキャンに含まれる場合には、IMS は、スキャンが現行エクステントの境界を超えないようにスキャン限界を調整し

ます。このパラメーターで定義されたシリンダー範囲内にセグメント挿入のためのスペースが見つからない場合は、データベースのデータ・セット・グループの現在の終わりのスペースが使用されます。

SEARCHA 0 | 1 | 2
IMS が HD データベースにセグメントを挿入するときに使用する、HD スペース検索アルゴリズムのタイプを指定します。

0 IMS が使用する HD スペース検索アルゴリズムを選択することを指定します。 0 がデフォルトです。

1 IMS が使用する HD スペース検索アルゴリズムは、2 番目に望ましいブロックまたは CI 内のスペースを検索しないという指定です。

2 IMS が使用する HD スペース検索アルゴリズムは、2 番目に望ましいブロックまたは CI 内のスペースを検索するという指定です。

このキーワードはオプションで、HDAM または HIDAM データベースにのみ有効です。

SIZE PRIMARY *size1*
HISAM、SHISAM、INDEX の場合、このキーワードは、データ・セット・グループ内の 1 次データ・セットの制御インターバルまたはブロック・サイズを指定します。

HDAM、HIDAM の場合、このキーワードは、データ・セット・グループ内のデータ・セットの制御インターバルまたはブロック・サイズを指定します。
GSAM の場合、このキーワードは、入出力データ・セットのブロック・サイズを指定します。

DEDB の場合、このキーワードは必須で、制御インターバルを指定します。

このキーワードは、他のすべてのデータベース・タイプには無効です。

SIZE PRIMARY は、IMS keyword SIZE=(*size1*,)) と同等です。

SIZE SECONDARY *size2*
HISAM、SHISAM、INDEX の場合、このキーワードは、オーバーフロー・データ・セットの制御インターバルまたはブロック・サイズを指定します。

このキーワードは、HISAM、SHISAM、および INDEX にのみ有効です。

SIZE SECONDARY は、IMS keyword SIZE=(,*size2*)) と同等です。

ROOT(*number1*,*number2*)
区域のルート・アドレス可能部分と独立オーバーフロー用の予約区域に割り振られる合計スペースを指定します。

number1
区域のルート・アドレス可能部分に割り振られる合計スペースを指定します。これは、UOW で表されます。VSAM データ・セットの残りの部分は、順次従属データ用に予約されます。

有効な範囲は 2 から 32767 です。VSAM データ・セットのスペース量より大きくすることはできません。

number2
独立オーバーフロー用に予約するスペース (UOW) を指定します。これは、1 以上で、かつ *number1* で指定された値未満でなければなりません。

独立オーバーフローは UOW を含んでいませんが、スペース割り振りの単位として UOW サイズが使用されます。

再編成 UOW は、DEDB 初期設定ユーティリティーにより自動的に割り振られます。VSAM スペース定義には、この追加の UOW を含めてください。つまり、必要な合計スペースは、ルート・アドレス可能域、独立オーバーフロー、および再編成用の 1 つの追加 UOW になります。再編成 UOW は、高速 DEDB 直接再編成ユーティリティーでは使用されませんが、IMS の他の機能で使用される場合があります。

ROOT キーワードは必須で、DEDB にのみ有効です。

UOW(*number1*,*number2*)
必須で、DEDB にのみ有効です。*number1* は、作業単位の制御インターバルの数を指定します。有効な範囲は 2 から 32767 です。 *number2* は、オーバーフロー・セクション中の制御インターバルの数を指定します。 1 以上だが、*number1* より少なくとも 1 小さい任意の値。

CLOSE

CLOSE ステートメントは、カーソルをクローズします。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。 これは実行可能ステートメントですが、動的に準備することはできません。

構文

説明

CLOSE ステートメントには、以下のキーワード・パラメーターが定義されています。

cursor-name
クローズするカーソルを指定します。カーソル名は、 878 ページの『DECLARE CURSOR』で説明している宣言済みのカーソルを示すものでなければなりません。

例

カーソル C1 は、一度に行を 1 つずつフェッチして、アプリケーション・プログラム変数 HOSPCODE、HOSPNAME、WARDNAME、および PATNAME に入れるために用いられます。最後にカーソルをクローズします。カーソルを再オープンすると、フェッチされる行の最初の位置に再びカーソルが置かれます。

```
EXEC SQLIMS
  DECLARE C1 CURSOR FOR DYSQL
END-EXEC.

EXEC SQLIMS
   PREPARE DYSQL FROM :SELECT-STATEMENT
```

```
END-EXEC

EXEC SQLIMS OPEN C1 END-EXEC.

EXEC SQLIMS
  FETCH C1 INTO :HOSPCODE, :HOSPNAME, :WARDNAME, :PATNAME
END-EXEC.

IF SQLIMSCODE = 100
PERFORM DATA-NOT-FOUND
ELSE
PERFORM GET-REST-OF-HOSP
UNTIL SQLIMSCODE IS NOT EQUAL TO ZERO.

EXEC SQLIMS CLOSE C1 END-EXEC.
```

COMMENT ON

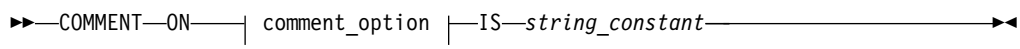
COMMENT ON ステートメントは、IMS カタログ内のリソースまたはオブジェクトの定義にコメントを追加します。COMMENT ON ステートメントを再発行することにより、コメントを変更できます。

呼び出し

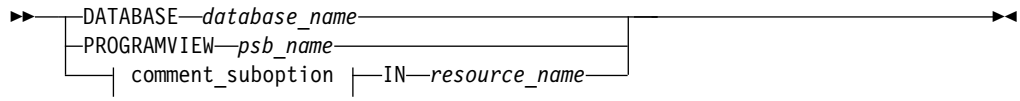
このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

- 『COMMENT ON 構文』
- 『comment_option の構文』
- 『comment_suboption の構文』

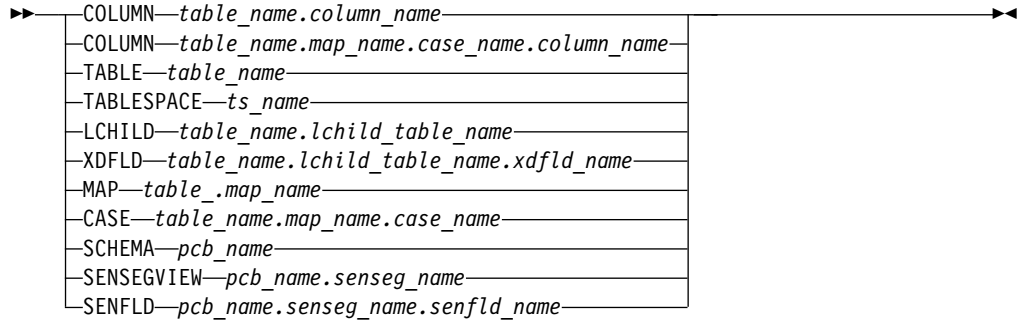
COMMENT ON 構文



comment_option の構文



comment_suboption の構文



説明

COMMENT ON ステートメントには、以下のキーワード・パラメーターが定義されています。

IS *string_constant*
オプションのユーザー・コメント。単一引用符に囲まれた 1 文字から 256 文字のストリング。この値には、以下の文字を含めることができません。

- 単一引用符 (コメント・ストリング全体を囲むために使用されている場合を除く)。以下の例では、単一引用符の正しい使い方と誤った使い方を示します。

```
CORRECT
    IS 'These remarks apply to the XYZ application'
INCORRECT
    IS 'These remarks apply to the 'XYZ' application'
```

- 二重引用符
- より小 (<) 記号
- より大 (>) 記号
- アンパーサンド (&)

IN *resource_name*
コメントを適用するリソースを指定します。リソースは、データベース (DBD) またはプログラム・ビュー (PSB) のいずれかです。名前は、 1 文字から 8 文字の英数字とすることができます。

以下の場合は、IN *resource_name* キーワードを省略します。

- COMMENT ON DATABASE *database_name*
- COMMENT ON PROGRAMVIEW *psb_name*

DATABASE *database_name*
データベースを識別します。名前は、 1 から 8 文字の英数字とすることができます。データベース・リソースは、事前に定義されている必要があります。

PROGRAMVIEW *psb_name*
PROGRAMVIEW または PSB を識別します。名前は、 1 文字から 8 文字の英数字とすることができます。PROGRAMVIEW は、事前に定義されている必要があります。

COLUMN *table_name.column_name*
列および列が属する表を識別します。名前は、1 文字から 128 文字の英数字とすることができます。列および表は、事前に定義されている必要があります。

COLUMN *table_name.map_name.case_name.column_name*
代替列マッピングおよび列が属しているマップ・ケースを識別します。名前は、1 文字から 128 文字の英数字とすることができます。表、マップ、ケース、および列は、事前に定義されている必要があります。

TABLE *table_name*
表を識別します。名前は、1 文字から 128 文字の英数字とすることができます。表は、事前に定義されている必要があります。

TABLESPACE *ts_name*
表スペースを識別します。名前は、 1 から 8 文字の英数字とすることができます。表スペースは、事前に定義されている必要があります。

LCHILD *table_name.lchild_table_name*
LCHILD ステートメントで指定された論理子表の名前を識別します。名前は、1 文字から 128 文字の英数字とすることができます。LCHILD 表の名前とその表が属している表は、事前に定義されている必要があります。

XDFLD *table_name.child_table_name.xdfld_name*
XDFLD ステートメントで指定された索引付きデータ・フィールドを識別します。名前は、1 文字から 26 文字の英数字とすることができます。XDFLD 名およびそのフィールドが属している表は、事前に定義されている必要があります。

MAP *table_name.map_name*
マップを識別します。名前は、1 文字から 128 文字の英数字とすることができます。マップおよび表の名前は、事前に定義されている必要があります。

CASE *table_name.map_name.case_name*
ケースを識別します。名前は、1 文字から 128 文字の英数字とすることができます。ケース、マップ、および表の名前は、事前に定義されている必要があります。

SCHEMA *pcb_name*
PCB を識別します。名前は、 1 から 8 文字の英数字とすることができます。PCB 名は、事前に定義されている必要があります。

SENSEGVIEW *pcb_name.senseg_name*
SENSEG を識別します。名前は、 1 から 8 文字の英数字とすることができます。SENSEG および PCB の名前は、事前に定義されている必要があります。

SENFLD *pcb_name.senseg_name.senfld_name*
SENFLD を識別します。名前は、 1 から 8 文字の英数字とすることができます。SENFLD、SENSEG、および PCB の名前は、事前に定義されている必要があります。

使用上の注意

このステートメントを使用して送信されたテキストは、リソースの IMS カタログ定義内の REMARKS セグメントに保管されます。リソースまたはオブジェクトに関連する REMARKS セグメントは、カタログ内にあるそのリソースまたはオブジェクトの直接の子です。

例

データベースに対するコメント指定の例:

```
COMMENT ON DATABASE dhvntz02 IS 'This describes database DHVNTZ02.'
```

表に対するコメント指定の例:

```
COMMENT ON TABLE root IN dhvntz02 IS 'This describes the root table.'
```

表スペースに対するコメント指定の例:

```
COMMENT ON TABLESPACE hidam IN dhvntz02 IS 'Dataset Group 1'
```

CREATE DATABASE

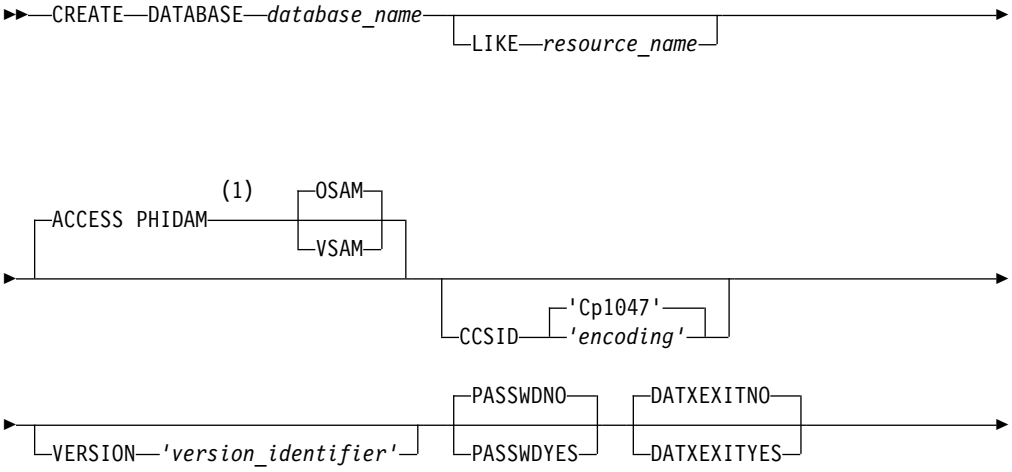
CREATE DATABASE ステートメントは、IMS に対して新規データベースを定義します。

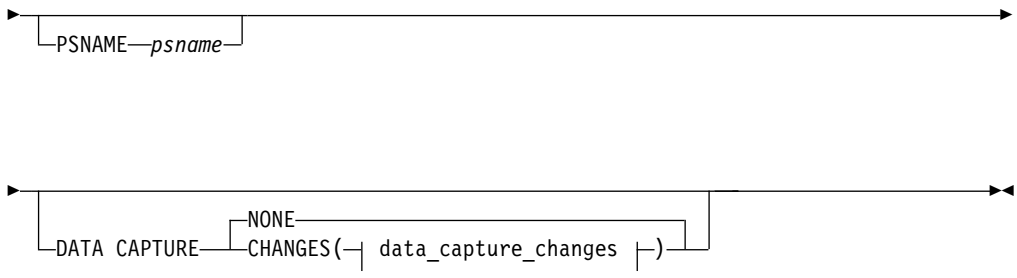
呼び出し

このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

- 『PHIDAM 構文』
- 777 ページの『HDAM 構文』
- 777 ページの『HIDAM 構文』
- 778 ページの『PHDAM 構文』
- 778 ページの『GSAM 構文』
- 778 ページの『HISAM 構文』
- 779 ページの『SHISAM 構文』
- 779 ページの『DEDB 構文』
- 779 ページの『HSAM 構文』
- 780 ページの『SHSAM 構文』
- 780 ページの『LOGICAL 構文』
- 780 ページの『INDEX 構文』
- 780 ページの『PSINDEX 構文』
- 781 ページの『data capture changes 構文』
- 781 ページの『exit_attributes 構文』

PHIDAM 構文

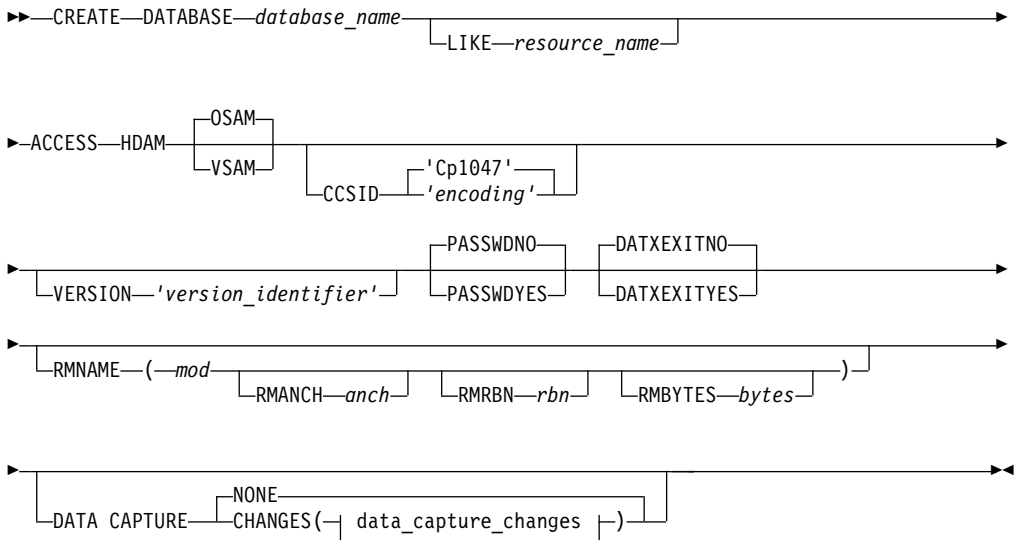




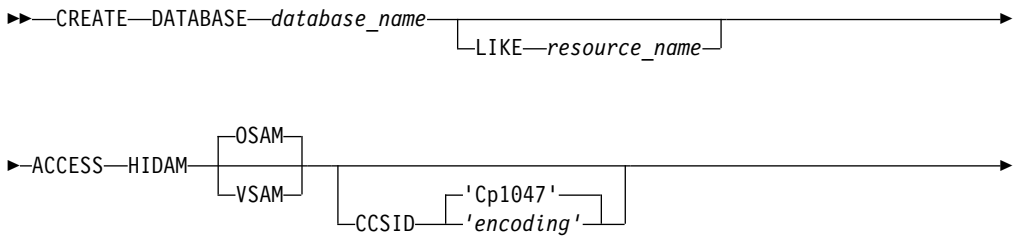
注:

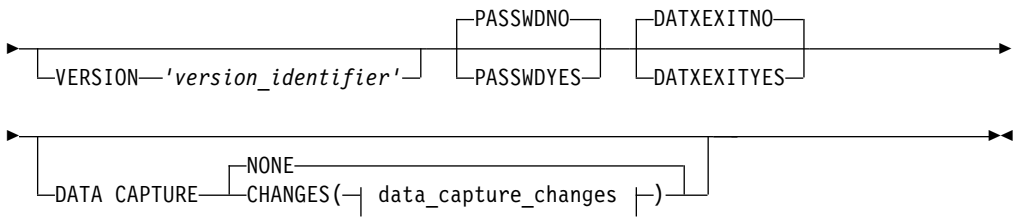
1 ACCESS キーワードを指定しない場合、PHIDAM OSAM がデフォルトのデータベース・アクセス・タイプです。特定のデータベース・アクセス・タイプが必要な場合、ユーザーは ACCESS キーワードとそれに続くアクセス・タイプを指定する必要があります。

HDAM 構文

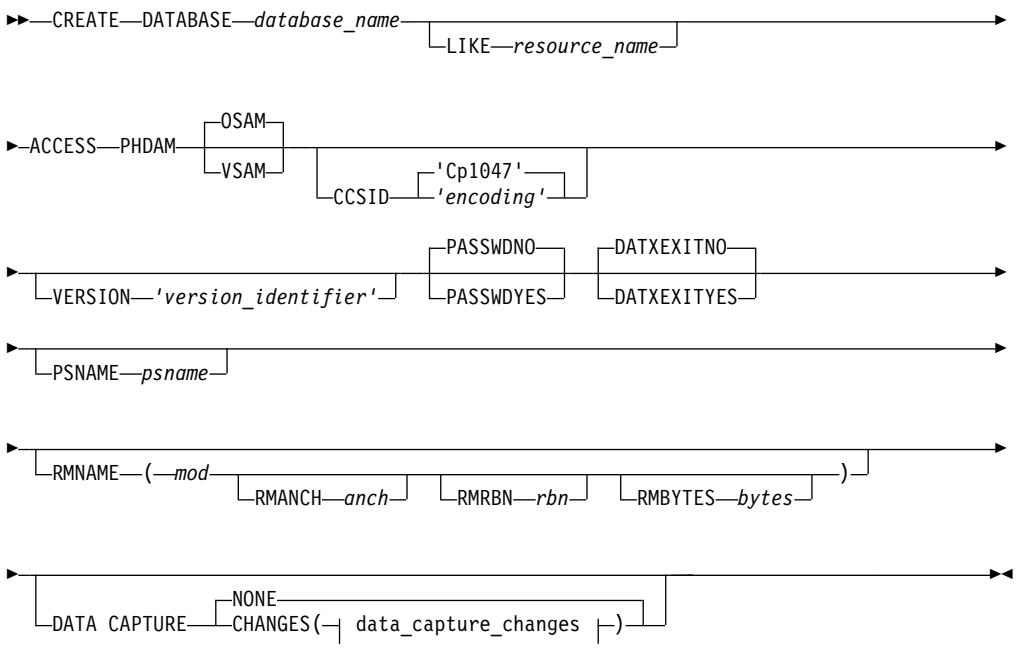


HIDAM 構文

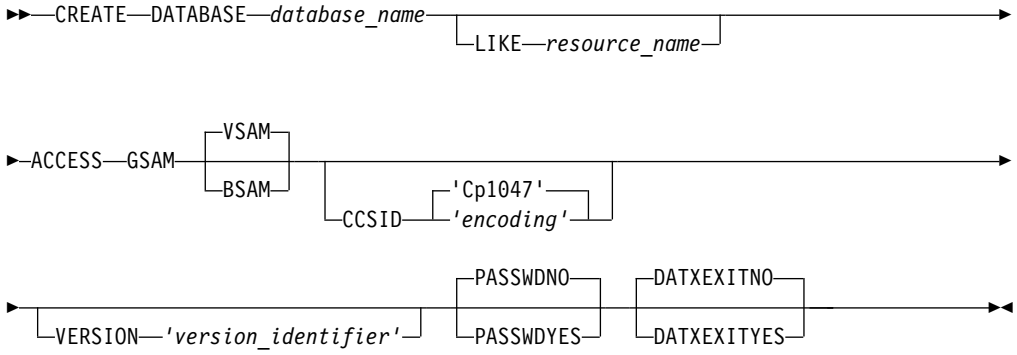




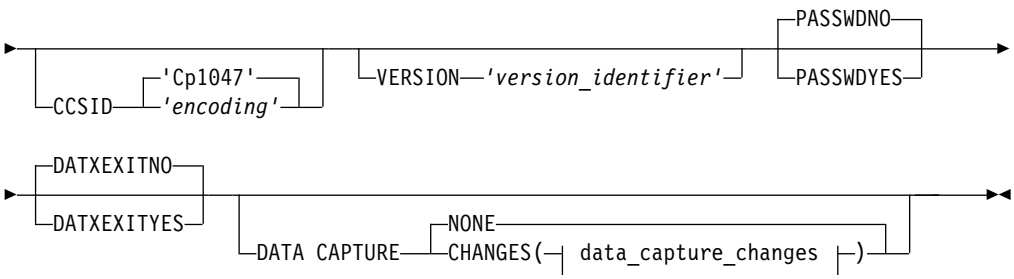
PHDAM 構文



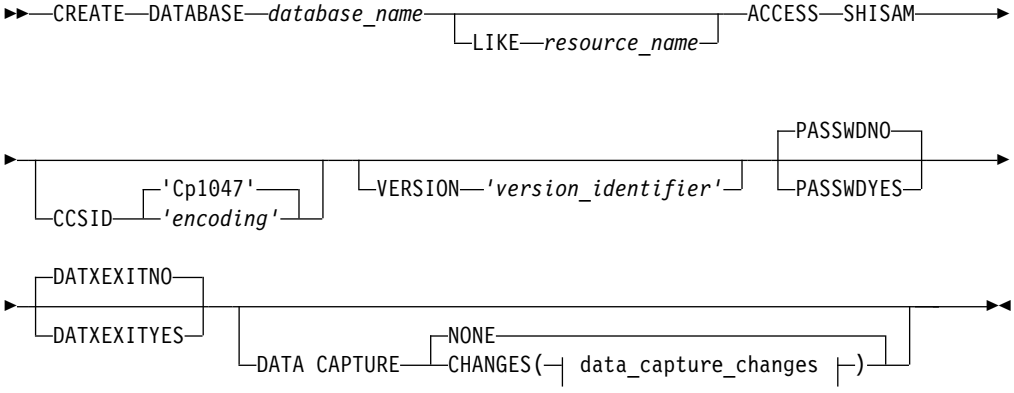
GSAM 構文



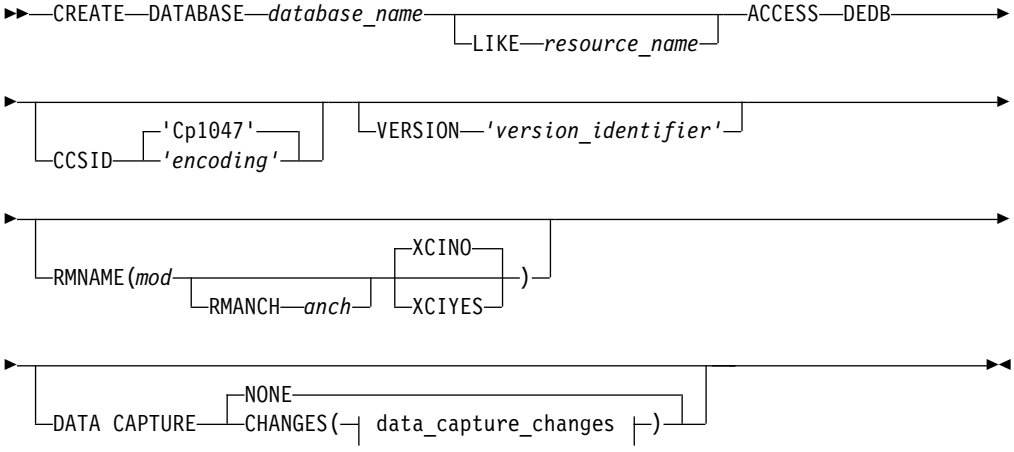
HISAM 構文



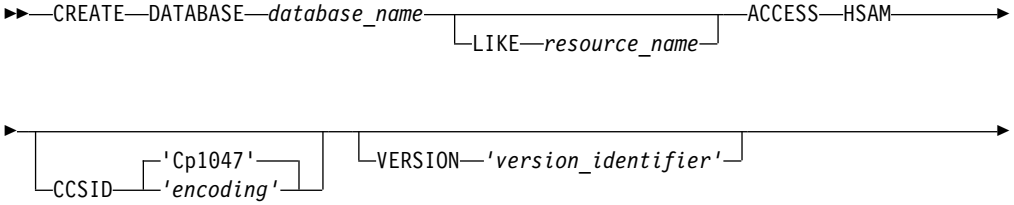
SHISAM 構文

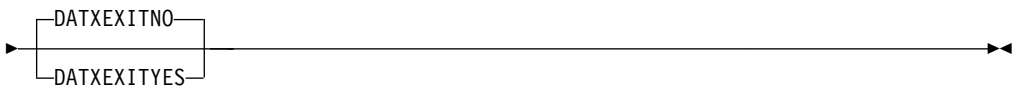


DEDB 構文

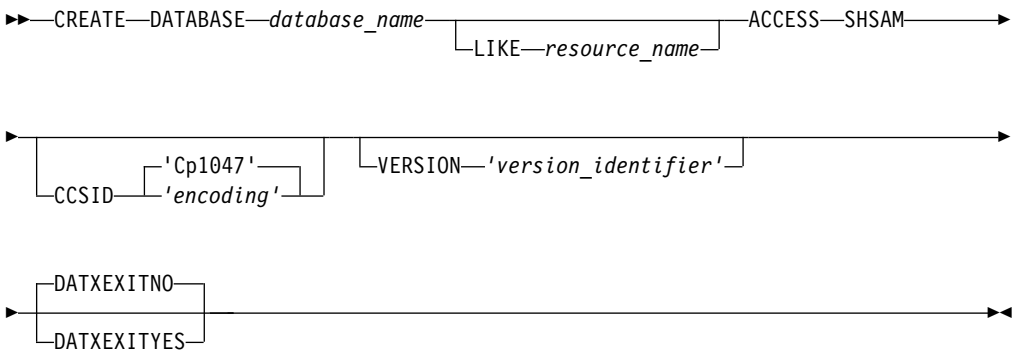


HSAM 構文

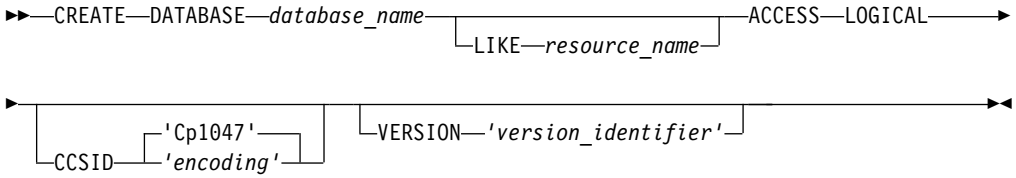




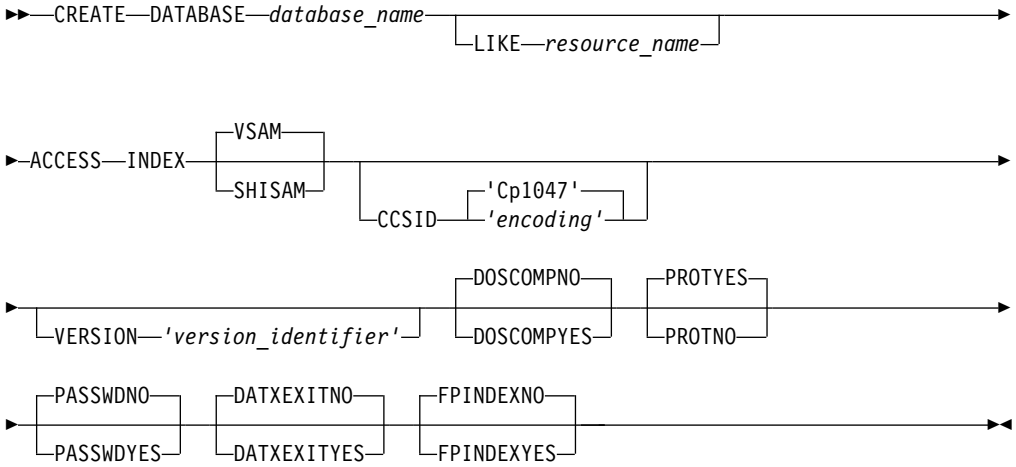
SHSAM 構文



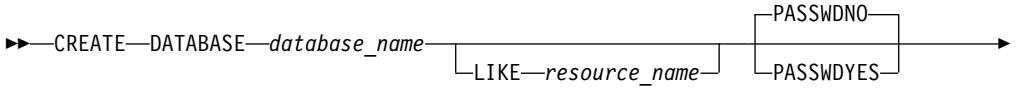
LOGICAL 構文

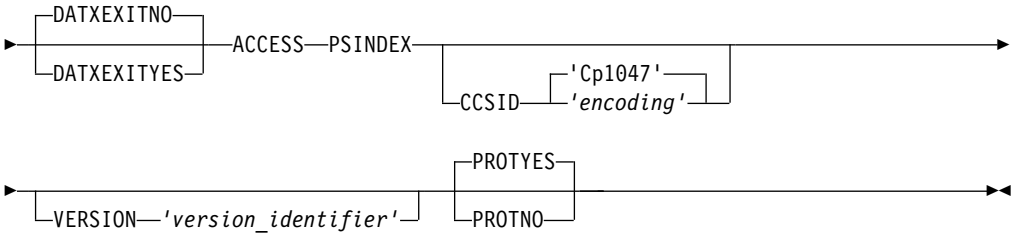


INDEX 構文

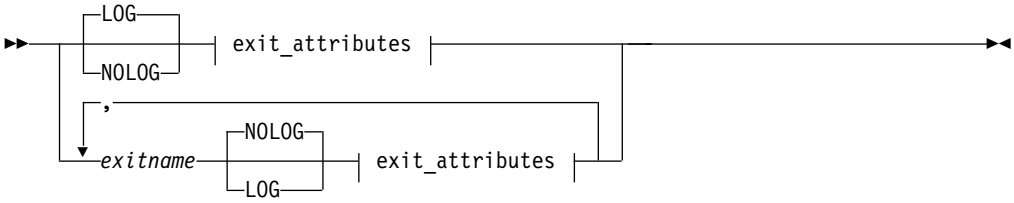


PSINDEX 構文

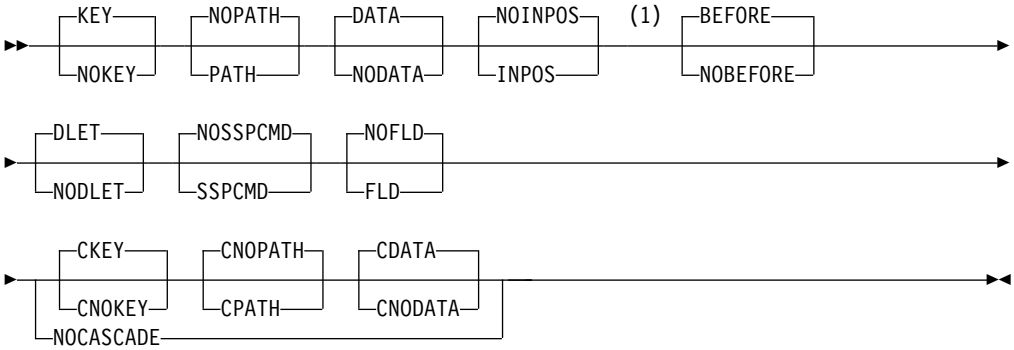




data capture changes 構文



exit_attributes 構文



注:

1 BEFORE、NOBEFORE、DLET、NODLET、SSPCMD、NOSSPCMD、FLD、および NOFLD は、DEDB 専用です。

説明

CREATE DATABASE ステートメントには、以下のキーワード・パラメーターが定義されています。

database_name

記述するデータベースの名前を指定します。名前は、 1 から 8 文字の英数字とすることができます。既存のデータベースまたはプログラム・ビューと同じ名前をデータベースに付けないでください。
そのデータベース名を持つリソースが既に存在している場合、-9002 エラー・メッセージが返されます。

ACCESS

このデータベースに使用する DL/I のアクセス方式およびオペレーティング・システムのアクセス方式を指定します。また、このキーワードは、副次索引データベースを HALDB として定義します。別のアクセス方式は次のとおりです。

HSAM
階層順次アクセス方式 (HSAM)。HSAM を指定しても、HSAM データベースに定義されているセグメント・タイプが 1 つだけしかない場合には、このパラメーターはデフォルトの SHSAM になります。

SHSAM
1 つの固定長セグメント・タイプのみを含む単純 HSAM データベース。単純 HSAM データベースが定義されている場合には、セグメント・タイプのオカレンスに接頭部を付けて IMS がデータベースを処理できるようにする必要はありません。

GSAM
汎用順次アクセス方式 (GSAM)。オペレーティング・システム・アクセス方式として BSAM または VSAM を指定できます。デフォルトは VSAM です。 GSAM を指定した場合、DBD で物理セグメントは許可されません。

HISAM
階層索引順次アクセス方式 (HISAM)。IMS は、VSAM オペレーティング・システム・アクセス方式を使用して HISAM データベースを作成します。

SHISAM
1 つの固定長セグメント・タイプのみを含む単純 HISAM データベース。IMS は、VSAM オペレーティング・システム・アクセス方式を使用して SHISAM データベースを作成します。単純 HISAM データベースが定義されている場合には、セグメント・タイプのオカレンスに接頭部を付けて IMS がデータベースを処理できるようにする必要はありません。

HDAM
階層直接アクセス方式 (HDAM)。オペレーティング・システムのアクセス方式として、OSAM または VSAM を指定できます。OSAM がデフォルトです。

PHDAM
区分階層直接アクセス方式 (PHDAM)。オペレーティング・システムのアクセス方式として、OSAM または VSAM を指定できます。OSAM がデフォルトです。

HIDAM
階層索引直接アクセス方式 (HIDAM)。オペレーティング・システムのアクセス方式として、OSAM または VSAM を指定できます。OSAM がデフォルトです。

PHIDAM
区分階層索引直接アクセス方式 (PHIDAM) は、デフォルトのアクセス方式です。オペレーティング・システムのアクセス方式として、OSAM または VSAM を指定できます。OSAM がデフォルトです。

DEDB
高速処理データベース (DEDB)。

INDEX
HIDAM データベース内のルート・セグメント・タイプのオカレンスの 1 次索引が作成されます。あるいは HISAM、HDAM、または HIDAM データベース内のセグメント・タイプの副次索引が作成されます。HIDAM データベースの 1 次索引または副次索引の場合は、オペレーティング・システム・アクセス方式として VSAM を指定する必要があります。

INDEX パラメーターは、DEDB データベースの副次索引を作成する場合にも使用されます。このような場合、VSAM および SHISAM はどちらも有効なオペレーティング・システム・アクセス・タイプです。

PHIDAM データベースの 1 次索引を定義する場合、INDEX パラメーターは使用しません。

PSINDEX
PHDAM データベースおよび PHIDAM データベースのセグメント・タイプの区分副次索引を作成します。VSAM オペレーティング・システム・アクセス方式を使用して PSINDEX が作成されます。

LOGICAL
論理データベースは、1 つ以上の物理データベースの一部または全部の論理連結で構成されます。論理データベースは、既存の物理データベースを参照する必要があります。

CCSID *'encoding'*
このデータベース内のすべての文字データのデフォルト・エンコードを指定する、1 文字から 25 文字のオプション・フィールド。

デフォルトのコード・ページは Cp1047 です。これは、EBCDIC エンコードを指定します。

この値に以下の文字を含むことはできません。

- 単一引用符または二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

この値は、個々の表または列でオーバーライドすることができます。

DATA CAPTURE
CREATE DATABASE ステートメントで DATA CAPTURE を指定すると、これらのオプションは物理データベース内のすべてのテーブルに適用されます。このパラメーターを CREATE ステートメントまたは ALTER TABLE ステートメントで指定した場合、このステートメントの指定はオーバーライドされます。

以下の物理データベースは DATA CAPTURE をサポートします。

- HISAM
- SHISAM
- HDAM
- PHDAM
- HIDAM
- PHIDAM

- DEDB

NONE
データ・キャプチャー・オプションがないことを示します。

CHANGES
任意の数の出口ルーチンを、各ルーチン独自の変更オプションのセットと一緒に指定できます。出口ルーチンを指定しない場合、ロギングに関する 1 セットの変更オプションのみを指定できます。このメソッドは、DBD マクロ・ステートメントの EXIT= パラメーターで出口ルーチン名の代わりにアスタリスク (*) を指定することと同じです。各セットは、コンマで分離します。NOCASCADE は、任意の組み合わせの C* (例えば、CKEY) オプションと相互に排他的です。

以下のオプションは DATA CAPTURE CHANGES に有効です。

BEFORE | NOBEFORE
REPL 呼び出しの場合に、変更前データが X'99' ログ・レコードに組み込まれます。BEFORE がデフォルトです。この属性は、DEDB の場合のみ有効です。

CDATA | CNODATA
カスケード削除の場合に、セグメント・データを出口ルーチンに渡します。また、CDATA は削除されるセグメントを識別します (物理連結キーで識別できない場合)。この属性は、NOCASCADE とは相互に排他的です。

CKEY | CNOKEY
物理連結キーを出口に渡します。このキーは、カスケード削除で削除されるセグメントを識別します。この属性は、NOCASCADE とは相互に排他的です。

CNOPATH | CPATH
出口ルーチンが物理ルートの階層パスにあるセグメント・データを必要としないことを示します。CNODATA は、カスケード削除に必要な相当量のパス・データを除去するために使用します。この属性は、NOCASCADE とは相互に排他的です。

DATA | NODATA
DATA は、更新用に物理テーブル・データが出口ルーチンに渡されることを指定します。DATA が指定され、EDITPROC 出口ルーチンもテーブルで使用されている場合、渡されるデータは拡張されたデータです。 DATA がデフォルトです。

DLET | NODLET
DLET 呼び出しの場合に、X'99' ログ・レコードが書き込まれます。DLET がデフォルトです。この属性は、DEDB の場合のみ有効です。

exitname
データを処理する出口ルーチンの名前を指定します。この名前は、ユーザーが IMS に対して定義したデータ・キャプチャー出口ルーチンの名前に一致する必要があります。最大 8 文字の英数字を使用できます。

KEY | NOKEY
KEY は、出口ルーチンに物理連結キーを渡すことを指定します。このキーは、アプリケーションによって更新される物理テーブルを識別します。 KEY がデフォルトです。

NOCASCADE
DL/I がこのセグメントを削除するときに、出口ルーチンは呼び出されないことを示します。従属セグメントを持たないセグメントを削除する場合には、カスケード削除は不要です。

NOFLD | FLD
FLD オプションは、DEDB FLD 呼び出しによって行われる更新をキャプチャーするよう要求します。このオプションは DEDB に対してのみ有効で、この情報は、オプション・ログが指定された場合に X'9904' ログ・レコードにのみ記録されます。この情報はデータ・キャプチャー出口には渡されません。この属性は、DEDB の場合のみ有効です。

NOINPOS | INPOS
INPOS オプションは、HERE の挿入規則が使用されて F や L のコマンド・コードが使用されていないときに、ISRT がキーなしセグメントまたは一意でないキー付きセグメントに対して行われた場合に、ツイン・データを渡すよう要求します。ツイン・データ IMS は ISRT がキャプチャーされる前の時点に位置指定されます。

NOLOG | LOG
LOG オプションは、データ・キャプチャーの制御ブロックとデータを IMS システム・ログに書き込むよう要求します。

NOPATH | PATH
NOPATH は、出口ルーチンが物理ルートの階層パスにあるテーブルからデータを必要としないことを示します。NOPATH は、パス・データの検索に必要な処理時間を回避するための効率的な方法です。
NOPATH がデフォルトです。

PATH は、更新されたセグメントのために物理ルートの階層パスにある各セグメントからのデータを、出口ルーチンに渡す必要がある場合に指定できます。アプリケーションが、挿入、置換、または削除の目的で複数のセグメントを別々にアクセスできるようにするには、PATH を使用します。

DB2® for z/OS の 1 次キーを構成するためにパス内のテーブルからの情報が必要なときは、PATH オプションを使用できます。その場合、DB2 for z/OS の 1 次キーは、従属テーブルの更新のための伝搬要求で使用されます。一般に、この種のテーブル情報が必要になるのは、親がキー情報を含んでおり、従属テーブルが親テーブルには収まらない追加データを含んでいる場合です。

PATH は、追加処理が必要な場合にも使用できます。例えば、D コマンド・コードを呼び出さなかった場合などに、1 つの呼び出しで複数のテーブルにアクセスしないということがあります。その場合、アプリケーションが別々の呼び出しで各テーブルにアクセスする際、追加処理が必要になります。

NOSSPCMD | SSPCMD
SSPCMD オプションは、DEDB サブセット・ポインター・コマンド・コードをキャプチャーするよう要求します。このオプションは DEDB の場合にのみ有効です。

DATXEXITNO | DATXEXITYES

このデータベースを処理中に、アプリケーションがデータ変換ユーザー出口ルーチン (DFSDBUX1) を使用できるようにします。デフォルトは DATXEXITNO です。

DATXEXITYES を指定すると、各データベース呼び出しの始めと終わりにユーザー出口 DFSDBUX1 が呼び出されます。DFSDBUX1 がロードされない場合、IMODULE が呼び出されてこれをロードします。

DATXEXITNO を指定した場合でも、DFSDBUX1 が SDFSRESL にあれば、DFSDBUX1 ユーザー出口ルーチンを呼び出すことができます。データベース定義に対して DFSDBUX1 を再度呼び出す必要がない場合、X'FF' が JCB の SRCHFLAG フィールドに戻され、DFSDLA00 は、出口を必要としていないことを示すマークをデータベース定義に動的に付けます。この場合は、DMB が DMB プールから除去されない限り、IMS セッションの期間中に、そのデータベース定義に対してユーザー出口が再度呼び出されることはありません。

DOSCOMPNO | DOSCOMPYES
これが DLI/DOS 索引データベースであるかどうかを指定します。データベースが索引であって、DLI/DOS を使用して作成されたものである場合には、これを指定する必要があります。 DLI/DOS 索引データベースは、接頭部の一部としてセグメント・コードを含みます。データベースが DLI/DOS 索引データベースであることを指定すると、IMS は定義されたデータベース内にこのコードが存在することを見込んで、このコードを保存するような方法で処理を行います。これには、挿入される新しいセグメントにセグメント・コードを指定することが含まれます。DLI/DOS データベースは VSAM を使用する必要があり、PHDAM、PHIDAM、または PSINDEX データベースにすることはできません。

FPINDEXNO | FPINDEXYES
索引データベースが 1 次高速機能 DEDB データベースの副次索引であるかどうかを指定します。デフォルトでは、索引データベースは副次索引ではありません。

LIKE *resource_name*
新規リソースのベースとなるモデル・リソースの名前を指定します。

PASSWDNO | PASSWDYES
PASSWDYES を指定すると、このデータベースのデータ・セットを開くときに、DL/I はデータベース名を VSAM パスワードとして使用します。このパラメーターは、VSAM をアクセス方式として使用するデータベースにのみ有効です。データベース名を LOGICAL または DEDB のデータベース・タイプのパスワードとして使用することはできません。ユーザーが、z/OS アクセス方式サービス・プログラムの DEFINE ステートメントを使用してこのデータベースの VSAM データ・セットを定義する場合には、制御レベル (CONTROLPW) またはマスター・レベル (MASTERPW) のパスワードは、この DBD の

DBDNAME と同じでなければなりません。この DBD に関連付けられたすべてのデータ・セットでは、同じパスワードを使用する必要があります。

IMS DB/DC システムでは、すべての VSAM OPEN がパスワード検査をう回するため、オペレーター・パスワード・プロンプトは回避されます。 IMS DB システムでは、VSAM パスワード検査が実行されます。バッチ環境では、自動パスワード保護が指定されていないときに、データベース名と等しくないパスワードにより制御レベル (CONTROLPW) でデータ・セットがパスワード保護されている場合、オペレーター・パスワード・プロンプトが出されます。

PASSWDNO を指定すると、データベース名を VSAM パスワードとして使用してはならないことを示します。これがデフォルトの動作です。

PROTYES | PROTNO

副次索引データベースで索引ポインター保護を使用するかどうかを指定します。このオプション・パラメーターは、IMS で使用される索引ポインター・セグメント内のすべてのフィールドの保全性を確保します。このパラメーターを使用すると、索引ポインター・セグメント内のフィールド (索引ポインター・セグメントのユーザー・データ部分のフィールドは除く) での置換操作をアプリケーション・プログラムは行えなくなります。索引ポインター・セグメントの削除操作は引き続き使用可能です。索引ポインター・セグメントに対する削除が出されると、索引ポインター・セグメント内の索引ターゲット・セグメント・ポインターが削除されます。しかし、最初に索引ポインター・セグメントの作成の原因となった索引ソース・セグメントは削除されません。

索引ポインター保護を使用しない場合、アプリケーション・プログラムでは、索引ポインター・セグメント内のすべてのフィールド (定数フィールド、検索フィールド、およびサブシーケンス・フィールドは除く) を置換できます。どの条件下でも、索引データベースへの挿入は無効です。

デフォルトで、副次索引データベースでは索引ポインター保護を使用します。

PSNAME *psname*

PSINDEX、PHDAM、または PHIDAM データベースの HALDB 区画を選択するモジュールを指定します。このパラメーターは、HALDB 区画選択出口ルーチンのモジュール名です。このパラメーターは、データベースのアクセス・タイプが PSINDEX、PHDAM、または PHIDAM である場合のみ有効です。

例外: ルート・キー範囲で HALDB 区画メンバーシップを定義している場合は、ユーザー提供の HALDB 区画選択ルーチンは必要ありません。

RMNAME *mod*

DEDB に格納されているデータ、あるいは HDAM または PHDAM データベースの 1 次データ・セット・グループに格納されているデータの管理に使用するモジュール名を指定します。このパラメーターが有効なのは、データベース・アクセス・タイプが HDAM、PHDAM、または DEDB である場合のみです。ランダム化モジュールは、DEDB、HDAM、または PHDAM データベースへのルート・セグメントの配置、またはそれらからのルート・セグメントの取り出しを制御します。ランダム化モジュールと呼ばれる 1 つ以上のモジュールは、IMS システム内で利用できます。ある特定のデータベースは、それに関連したランダム化モジュールを 1 つしか持つことができません。汎用モジュール (ユーザー提供のパラメーターを使用して、ある特定のデータベースのランダム化を

実行するもの) を作成して、いくつかのデータベースで利用することができます。ランダム化モジュールの目的は、DEDB、 HDAM、または PHDAM データベースへのルート・セグメントの配置、またはそれらからのルート・セグメントの取り出しのためにアプリケーション・プログラムが指定する値を、相対ブロック番号およびアンカー・ポイント番号に変換することです。2 ステージ・ランダマイザーを選択すると、1 つの区域内でランダム化を実行することができます。 2 ステージ・ランダマイザーを選択すると、/DBRECOVERY コマンドで DEDB の区域をすべて停止させなくても、1 つの区域内のルート・アンカー・ポイント数を変更することができます。

PHDAM データベースでは、ランダマイザーのモジュール名および値が、区画ごとのデフォルトになります。HALDB 区画定義時に、区画ごとに異なるランダマイザー名と値を設定することができます。 HALDB 区画選択は、ランダム化モジュールを呼び出す前に行われます。ランダム化モジュールは、1 つの区画内でのみ位置を選択します。

モジュール名は、この DEDB、PHDAM、または HDAM データベース内のセグメントの格納およびアクセスに使用するユーザー提供ランダム化モジュールの名前 (1 文字から 8 文字の英数字) です。モジュール名パラメーターにランダマイザーの名前を指定し、アンカー・ポイント・パラメーターに 2 を指定して、2 ステージ・ランダマイザーを選択します。

RMANCH *anch*

anch 値の目的は、高速機能 DEDB データベースを定義するか、全機能 HDAM または PHDAM データベースを定義するかによって異なります。

このパラメーターは符号なしの 10 進整数でなければなりません。このパラメーターのデフォルト値は 1 です。

DEDB データベースの場合、*anch* の値はランダマイザーのタイプを指定します。値 1 は、単一ステージ・ランダマイザーを示します。値 2 は、2 ステージ・ランダマイザーを示します。その他の値は無効です。

HDAM および PHDAM データベースの場合、*anch* の値は、HDAM または PHDAM データベースのルート・アドレス可能域内の各制御インターバルまたは制御ブロックで必要なルート・アンカー・ポイントの数を指定します。標準的な値は 1 から 5 であり、この値は 255 を超えることはできません。

HDAM または PHDAM データベースにアクセスするときに、ユーザー・ランダム化ルーチンが、このパラメーターに指定された数値より大きいアンカー・ポイント番号を生成する場合、制御インターバルまたは制御ブロック内で最高の番号のアンカー・ポイントが使用されます。ランダム化ルーチンが IMS アンカー・ポイント番号 0 を生成した場合には、IMS は制御インターバルまたは制御ブロックのアンカー・ポイント 1 を使用します。

RMRBN *rbn*

このデータベースに関してランダム化モジュールに作成させる相対ブロック番号の最大値を指定します。このパラメーターは、HDAM または PHDAM データベースの場合のみ使用します。この値により、HDAM または PHDAM データベースのルート・アドレス可能域内の制御インターバルまたは制御ブロックの数が決まります。このパラメーターは、2^{24}-1 を超えない符号なしの 10 進整数にする必要があります。このパラメーターを省略すると、ランダム化モジュールが作成する相対ブロック番号で上限の検査は行われません。このパラメーターを指

定したにもかかわらず、指定されたランダム化モジュールがこのパラメーターよりも大きい相対ブロック番号を生成する場合、ルート・アドレス可能域内の最高位の制御インターバルまたは制御ブロックが IMS により使用されます。ユーザーのランダム化モジュールがブロック番号 0 を生成する場合、制御インターバルまたは制御ブロック 1 が IMS により使用されます。

HDAM または PHDAM データ・セットでは、最初のビットマップは、データ・セットの先頭にあるエクステントの先頭ブロックにあります。 HDAM または PHDAM データベースでは、データ・セット・グループごとに指定されるデータ・セットの先頭にあるエクステントの最初の制御インターバルまたは制御ブロックが、ビットマップに使用されます。 VSAM データ・セットでは、2 番目の制御インターバルがビットマップに使用され、最初の制御インターバルは予約されます。IMS は、ランダマイザーが計算したそのブロックに 1 を加えます。

RMBYTES *bytes*
別のデータベース・レコードの呼び出しによって分断されない一連の挿入でルート・アドレス可能域に格納できる、データベース・レコードの最大バイト数を指定します。このパラメーターは、HDAM または PHDAM データベースの場合のみ使用します。このパラメーターを省略すると、このデータベースのルート・セグメント・アドレス可能域に挿入できるデータベース・レコードの最大バイト数は、無制限になります。bytes パラメーターは、2^{24}-1 を超えない符号なしの 10 進整数にする必要があります。最大相対ブロック番号パラメーターを省略すると、このパラメーターは無視されます。この場合、ルート・アドレス可能域に挿入できるデータベース・レコードのバイト数に制限はありません。

このパラメーターを HDAM または PHDAM データベースに対して指定した場合に、データベース・レコードの長さがそれより大きいと、レコードの超過部分は、現行のファイルの終わり (EOF) に続くオーバーフロー域に挿入されます。この操作には、このパラメーターの値を超えるすべてのデータベース・レコードの超過部分を入れる十分なスペースが、現行の EOF の後に使用可能でなければなりません。現行の EOF の後のオーバーフロー域に十分なスペースがないと、データベース・レコードはデータベースにランダムに挿入されることになります。

XCINO | XCIYES
この DEDB が、ランダマイザーを呼び出すときに、拡張呼び出しインターフェースを使用するかどうかを指定します。このオプションでは、3 つの異なる方法でランダマイザーを呼び出すことができます。IMS の初期設定時、または /START DB コマンド実行時に、IMS は、まずランダマイザーをロードし、次にランダマイザーに INIT 呼び出しを行って、その初期設定ルーチンを呼び出します。 /DBR DB コマンド実行の過程で IMS は TERM 呼び出しを行い、終了ルーチンを呼び出してから、ランダマイザーをアンロードします。アプリケーションがルート・セグメントに GU または ISRT 呼び出しを出すと、通常のランダム化呼び出しがランダマイザーに対して行われます。XCI オプションは、DEDB の場合にのみ有効です。

VERSION '*version_identifier*'
ID ストリングを指定します。これをデータベース変更のコメント記述子として使用することができます。

使用上の注意

デフォルト・オプション (CREATE DATABASE database_name ステートメントを使用し、他のパラメーターは指定しない) を使用してデータベースを定義すると、OSAM データ・セット・アクセス・タイプを使用する PHIDAM データベースが作成されます。また、次のように CREATE ステートメントにいずれかのキーワードを含めることで、PHIDAM データベースが VSAM または OSAM データ・セット・アクセス・タイプを使用するように明示的に指定することもできます: `CREATE DATABASE database_name ACCESS PHIDAM OSAM` または `CREATE DATABASE database_name ACCESS PHIDAM VSAM`

データ・バージョン管理の注意事項

CREATE DATABASE ステートメントでは、データベース・バージョン番号 (DBVER) は常に 0 です。 CREATE は IMS に対して新規データベースを定義し、常に 0 が基本バージョンです。同じ DDL ストリームにおける CREATE ステートメントおよび ALTER ステートメント (アクティブ化コマンドの前) はすべて、バージョン 0 で処理されます。

自動的に生成されたすべてのダミー PSB は、デフォルトで現行バージョン (バージョン 0) を参照します。オプションで CREATE SENSEGVIEW を発行して手動で PSB を作成し、PCB に対して "DBVER 0" を指定することで、バージョン 0 で固定することができます。このストリームにはバージョン 0 のみが存在するため、これより大きいバージョン番号を指定することはできません。オプションで "DBLEVEL CURR | BASE" 設定を指定することもできます。バージョン 0 を参照する PSB がアクティブ化されているが、データベースのバージョン管理が有効にされていない場合、IMS はその PSB からのアプリケーション呼び出しを拒否するので注意してください。

例: 全機能データベース

以下の入力を使用して、データ・キャプチャー出口を使用せずに DATA CAPTURE CHANGES キーワードを指定することができます。これは、ユーザーがロギングのみを必要としていることを IMS に指示します。

DBD ソースと同等:

```
DBD   NAME=DHVNTZ02,ACCESS=(PHIDAM,OSAM),                       X
         EXIT=((*,KEY,DATA,NOPATH,(CASCADE,KEY,DATA,NOPATH),    X
         LOG))
```

DDL と同等:

```
CREATE DATABASE DHVNTZ02
  DATA CAPTURE CHANGES(
   LOG KEY DATA NOPATH CKEY CDATA CNOPATH
  )
```

以下の入力を使用して、複数のデータ・キャプチャー出口を使用して DATA CAPTURE CHANGES キーワードを指定することができます。

DBD ソースと同等:

```
DBD   NAME=DHVNTZ02,ACCESS=(PHIDAM,OSAM),                          X
         EXIT=((EXIT1A,(CASCADE,KEY,DATA,PATH),                    X
         KEY,DATA,PATH,NOLOG),                                     X
         (EXIT1B,NOKEY,NOPATH,NOLOG,(CASCADE,NOKEY,DATA,NOPATH)),X
         (EXIT1C,(CASCADE,NOKEY,NODATA,NOPATH),                    X
         NOKEY,DATA,PATH,NOLOG),                                   X
         (EXIT1D,KEY,NODATA,PATH,NOLOG,                            X
         (CASCADE,NOKEY,NODATA,PATH)))
```

DDL と同等**:**

```
CREATE DATABASE DHVNTZ02
  DATA CAPTURE CHANGES(
   EXIT1A NOLOG KEY DATA PATH CKEY CDATA CPATH,
   EXIT1B NOLOG NOKEY NOPATH CNOKEY CDATA CNOPATH,
   EXIT1C NOLOG NOKEY DATA PATH CNOKEY CNODATA CNOPATH,
   EXIT1D NOLOG KEY NODATA PATH CNOKEY CNODATA CPATH
  )
```

DBD 生成ユーティリティーに対して以下を入力すると、基本全機能データベースが作成されます。

```
DBD   NAME=COGDBD,                                                 C
         ENCODING=Cp1047,                                          C
         ACCESS=(HDAM,OSAM),                                       C
         RMNAME=(DFSHDC40,3,3,25),                                 C
         PASSWD=NO,                                                C
         VERSION='Latest version of COGDBD'
```

以下の CREATE DATABASE ステートメントを使用して、同じデータベースを作成することができます。

```
CREATE DATABASE COGDBD
  ACCESS HDAM OSAM
  RMNAME(DFSHDC40 RMANCH 3 RMRBN 3 RMBYTES 25)
  VERSION 'Latest version of COGDBD'
  CCSID 'Cp1047';
```

例**:** 高速機能高速処理データベース **(DEDB)**

上記の例と同様に、DBD 生成ユーティリティーに対して以下の入力を実行して、DEDB を作成することができます。

```
DBD       NAME=HOSPDBD1,                                           C
             ENCODING=Cp1047,                                      C
             ACCESS=(DEDB),                                        C
             RMNAME=(RMOD3,1,,,XCI)                                C
             PASSWD=NO
```

CREATE DATABASE ステートメントを使用して、同等のデータベースを作成することができます。

```
CREATE DATABASE HOSPDBD1
  ACCESS DEDB
  RMNAME( RMOD3 RMANCH 1 XCIYES)
  CCSID 'Cp1047';
```

CREATE PROGRAMVIEW

IMS のもとでアプリケーション・プログラムを実行する前に、アプリケーション PROGRAMVIEW を作成して、プログラムが論理端末と論理データ構造をどのように使用できるか記述する必要があります。PROGRAMVIEW は、IMS ではプログラ

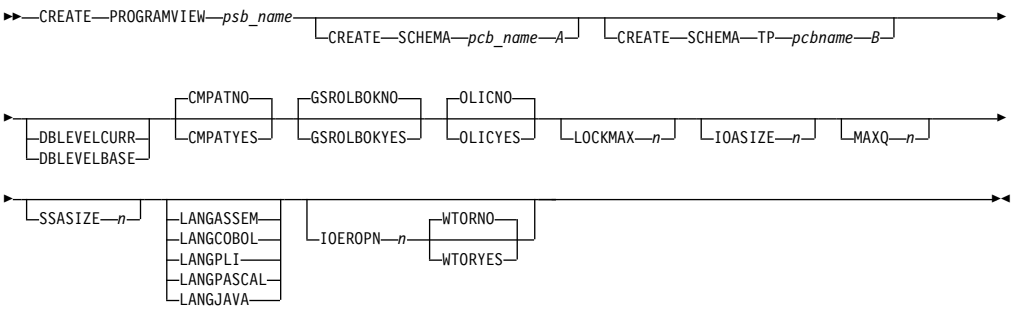
ム仕様ブロック (PSB) と呼ばれます。CREATE PROGRAMVIEW ステートメントは、IMS 固有のリソースである PSB を作成します。

呼び出し

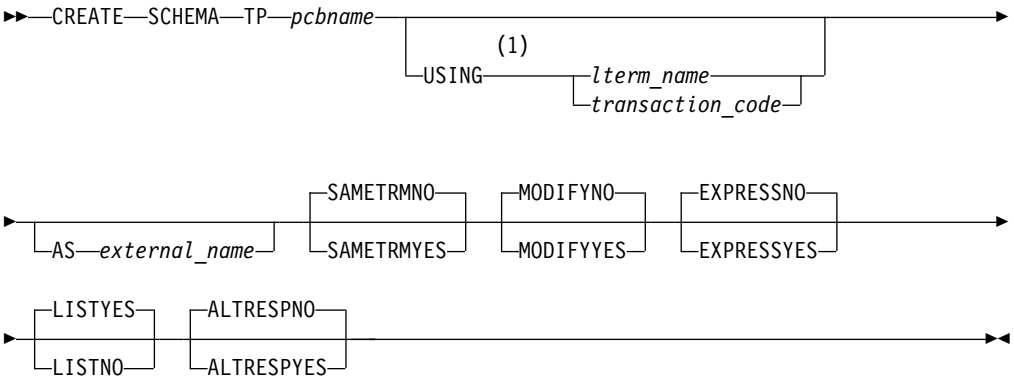
このステートメントは、IMS JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。 CREATE PROGRAMVIEW ステートメントには、CREATE SENSEGVIEW ステートメントおよび CREATE SCHEMA ステートメントが必要です。

- 『CREATE PROGRAMVIEW 構文』
- 『alternate_schema_statement 構文』
- 793 ページの『DB_schema_statement 構文』
- 793 ページの『GSAM_schema_statement 構文』
- 793 ページの『sch_procopt 構文』
- 794 ページの『ssview_statement 構文』
- 794 ページの『sf_statement 構文』
- 795 ページの『ssv_procopt 構文』

CREATE PROGRAMVIEW 構文



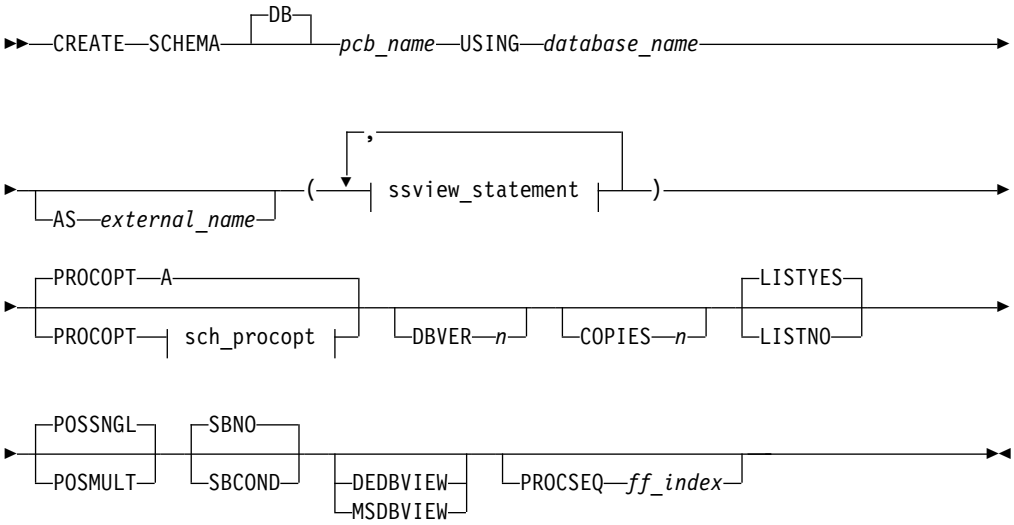
alternate_schema_statement 構文



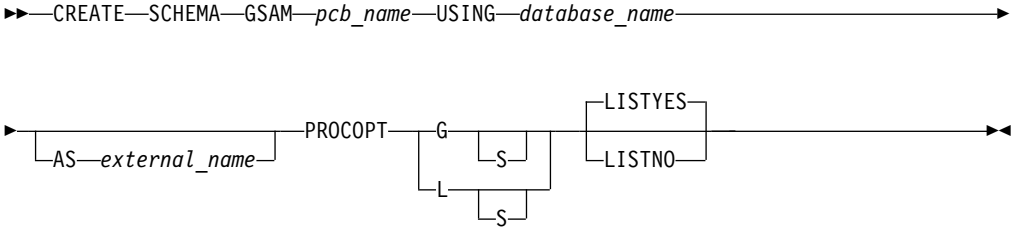
注:

1 MODIFYYES が指定されている場合を除き、*lterm_name* または *transaction_code* が必要です。

DB_schema_statement 構文

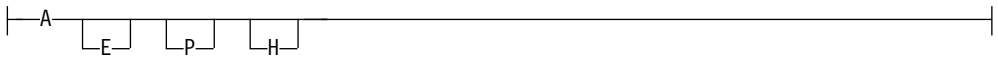


GSAM_schema_statement 構文

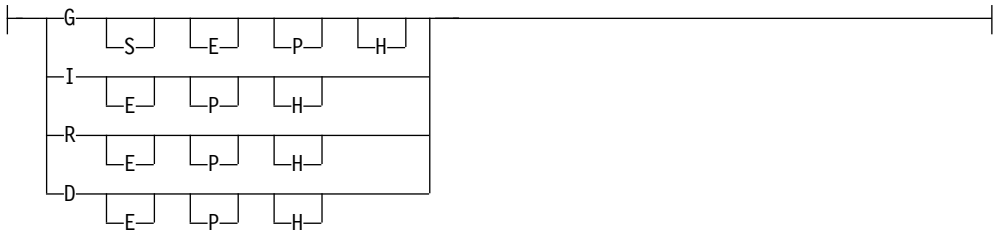


sch_procopt 構文

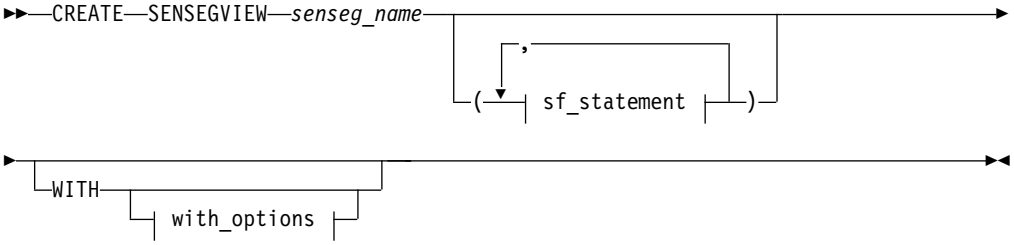
a:



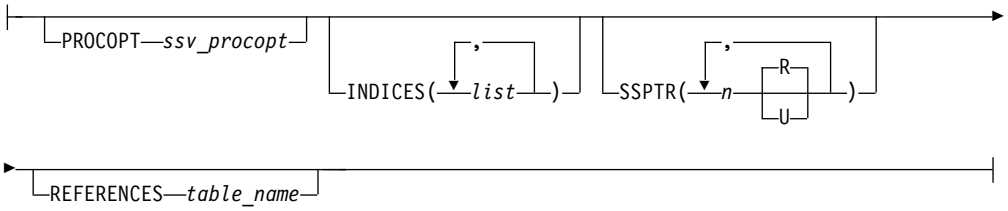
b:



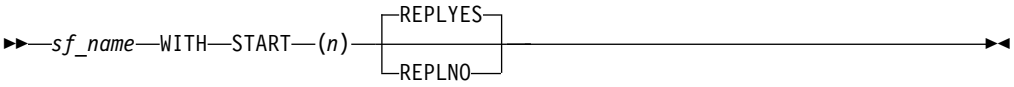
ssview_statement 構文



with_options:



sf_statement 構文



ssv_procopt 構文

CREATE PROGRAMVIEW のキーワード・パラメーター

CREATE PROGRAMVIEW ステートメントでは、以下のキーワード・パラメーターを指定することができます。

psb_name
プログラム・ビューの名前を指定します。プログラム・ビューは、IMS の PSB に相当します。この名前は、カタログ内に存在するデータベースや PSB の名前と同じであってはなりません。

DBLEVELCURR | DBLEVELBASE
データベースのバージョン管理が使用可能な場合、特定のデータベース・バージョンを要求しないアプリケーション・プログラムに対してデータを返すのに使用するデータベース定義のバージョンを指定します。このプログラム仕様を使用するすべてのアプリケーション・プログラムでは、ここで指定された値が、IMS.PROCLIB データ・セットの DFSDF*xxx* メンバーで指定された DBLEVEL パラメーターのシステム・デフォルトをオーバーライドします。

DBLEVELCURR は、アプリケーションがデータベース定義の最新バージョンに関連するデータを受信することを指定します。

DBLEVELBASE は、アプリケーションがデータベース定義のバージョン 0 に関連するデータを受信することを指定します。

使用されるデータベース仕様のデフォルト・バージョンに関係なく、アプリケーションは、プログラムをスケジュールする際に特定のバージョンのデータベース仕様を使用するように要求することができます。

CMPATNO | CMPATYES
BMP または MSG と、バッチ DL/I パラメーター・リストとの間の互換性を提供します。 YES に設定される場合、この PSB は、どのように使用されているかにかかわらず、常に入出力 PCB があるように取り扱われます。NO に設定される場合、BMP または MSG 領域についてのみ入出力 PCB が PSB に追加されます。デフォルトは NO です。

GSROLBOKNO | GSROLBOKYES
次の場合に、内部 ROLB 呼び出しを行って非 GSAM データベース更新をロールバックするかどうかを制御します。

- アプリケーションが非メッセージ・ドリブン BMP であるとき。
- PSB に、GSAM データベースの PCB が含まれているとき。
- スレッド作成時または SQL 呼び出し時に、Db2® for z/OS がデッドロックを報告するとき。

YES は、内部 ROLB 呼び出しが行われること、およびデッドロックに関する SQL コードがアプリケーション・プログラムに戻されることを意味します。NO は、ROLB 呼び出しではなく、ユーザー異常終了 777 が発生することを意味します。デフォルトは NO です。

OLICNO | OLICYES
この PSB のユーザーが、この PSB に指定されているデータベースに対して BMP として実行される調査ユーティリティー機能またはオンライン・データベース・イメージ・コピー・ユーティリティーを実行する許可を与えられているかどうかを示します。YES を指定すると、オンライン・イメージ・コピー・ユーティリティーおよび調査ユーティリティー機能を使用でき、NO を指定すると、オンライン・イメージ・コピー・ユーティリティーおよび調査ユーティリティー機能を使用できません。NO がデフォルトです。 PSB 内の任意のデータベース PCB (TYPE パラメーターに DB が指定されている PCB) が L または LS 処理オプションを指定する場合、このパラメーターは無効です。

例外: このパラメーターは、CICS、GSAM、HSAM、MSDB、または DEDB データベースに対して使用できません。

LOCKMAX *n*
アプリケーション・プログラムが一度に獲得できるロックの最大数を示します。*n* は、0 から 255 の数値です。*n* は、1000 の単位で指定します。例えば、LOCKMAX=5 と指定すると、一度に最大 5000 個のロックが可能であることを示します。

デフォルトは 0 です。これは、一度に可能なロックの最大数がないことを示します。

コミットなしでアプリケーション・プログラムが延長して実行される場合は、データベースのレコードおよび変更に関して IMS が行うロッキングは累積可能です。 LOCKMAX パラメーターを使用すれば、単一のアプリケーション・プログラムがすべてのロック・ストレージを使い切り、他のプログラムの異常終了が発生することを防ぐことができます。

LOCKMAX=0 (限界を完全にオフにする) を指定するか、従属領域 (BMP、MPP、または IFP) またはバッチ (DBB あるいは DLI) に対して LOCKMAX=1 から 32767を指定することで、プログラム実行時に指定された LOCKMAX 値をオーバーライドすることができます。値は、1000 の単位です。この方法を使用すれば、LOCKMAX パラメーターに指定できる最大値 255 を超えることができます。

IOASIZE *n*
アプリケーション・プログラムで使用する最大入出力域のサイズ (バイト数) を指定します。このサイズ指定を使用して、このアプリケーション・プログラムのスケジューリング時に、ユーザー入出力域のデータに関する制御領域のコピーを保持するために PSB プールに予約する主記憶域の量が決められます。この値を指定しない場合、IMS は最大のデフォルト入出力域サイズを計算します。デフォルトのサイズは、可能な限り最長のパス CALL 内にあるすべてのセンシティブ・セグメントの合計の長さです。 (アプリケーション・プログラムが、セグメント中のすべてのフィールドにセンシティブでなくても、セグメントの全長を使用する必要があります。) 指定する値はバイト単位で、最大が 256000 です。た

だし、1 つのパス CALL でアプリケーションに戻される全連結セグメントの合計の長さは、65535 バイトを超えてはなりません。

PSB がフィールド・センシティブ・セグメントを含んでいる場合に、IOASIZE が指定されていると、指定値は、ACBGEN ユーティリティーが計算した IOASIZE よりも大きい場合にのみ使用されます。使用される IOASIZE の値は、指示されます。このプール所要量の主要なコンポーネントは、IOASIZE と SSASIZE です。

この PSB で STAT 呼び出しまたはテスト・プログラム (DFSDDLT0) を使用する場合は、IOASIZE を 600 バイトより大きくしなければなりません。

この PSB で CMD または GCMD 呼び出し (自動化操作プログラム・インターフェース・アプリケーション・プログラムから) を使用する場合には、IOASIZE は 132 バイト以上にする必要があります。

拡張チェックポイント・リスタートを使用する場合は、IOASIZE には、下記の大きい方の値以上の値を指定しなければなりません。

- 以前のチェックポイントがある DL/I データベース (この PSB にチェックポイントがある場合) の位置変更を行っているときに、再始動時に出される GU 呼び出しからのデータを受け取るために必要な入出力域。
- 以前のチェックポイントがある GSAM データ・セットで使用される最大 LRECL。

XRST CALL の第 3 パラメーター (I/O AREA LEN) が指す値、またはこのパラメーターの値のいずれか大きい方の値が使用されます。

MAXQ *n*
同期点と同期点の間に出すことができる、Qx コマンド・コードによるデータベース呼び出しの最大数。最大数は 32,767 です。デフォルトはゼロです。

SSASIZE *n*
アプリケーション・プログラムで使用するすべての SSA の最大合計長を指定します。IMS は、このサイズ指定を使用して、このアプリケーション・プログラムの実行中に、ユーザーの SSA ストリングのコピーを保持するために PSB 作業プールに予約する主記憶域量を決めます。この値を指定しない場合、IMS は、デフォルトとして使用される最大 SSA サイズを計算します。計算されたサイズは、この PSB 内の任意の PCB 内の最大レベル数に 280 を掛けたものです。指定する値はバイト単位で、最大が 256000 です。

制約事項: DBCTL なしの CICS のもとで IMS を実行するとき、PSB 作業プール所要量は 64 KB を超えることはできません。

このプール所要量の主要なコンポーネントは、IOASIZE と SSASIZE です。

重要: 高速機能副次索引の呼び出しでは、SSASIZE ワークエリアが、SUBSEQ フィールドからの追加ストレージおよび修飾子の数を収容する変換済み SSA を保持します。DL/I 呼び出しが開始されると、この変換済み SSA は全機能データベースに渡されます。

デフォルトの SSASIZE は、ACBGEN で定義されているデフォルトの SSA サイズに 840 バイトを加えた値に指定されます。

SSASIZE を指定した場合、またはデフォルトを使用していて SSASIZE の大きさが不足している場合、AU 状況コードが発行されます。この問題を修正するには、PSB での SSASIZE の値を大きくして PSBGEN および ACBGEN を再実行します。

LANGASSEM | LANGCOBOL | LANGPLI | LANGPASCAL | LANGJAVA
メッセージ処理プログラムまたはバッチ処理プログラムの作成に使用するコンパイラー言語を示すオプションのキーワード。OLICYES を指定した場合、LANGPLI は無効です。アプリケーション・プログラムが C 言語で作成されている場合には、LANGASSEM を指定します。

CICS および z/OS 用の Language Environment (言語環境プログラム) は LANGPASCAL をサポートしていません。

アプリケーションが JMP 領域内で IMS 用の Java クラス・ライブラリーを使用している場合は、LANGJAVA を指定する必要があります。

PLICALLA エントリー・ポイントを使用して互換モードで実行される IMS PL/I アプリケーションを使用している場合は、LANGPLI を指定する必要があります。

IOEROPN *n*
バッチ・タイプ領域 (DLI または DBB) の場合にのみ適用できます。このパラメーターは、CICS には無効です。 n サブパラメーターは条件コードであり、IMS が正常終了し、アプリケーション・プログラムの実行中にデータベースで 1 つ以上の入力エラーまたは出力エラーが起きたときにオペレーティング・システムに戻されます。 n サブパラメーターは 0 から 4095 の数値です。

n=451 の場合は、IMS は条件コードをオペレーティング・システムに渡さずに、U451 異常終了を出して終了します。n=451 であって、IMS またはアプリケーション・プログラムが U451 以外で異常終了し、しかも入出力エラーが起きた場合には、プログラマー向け書き込みのメッセージ DFS0426I が出されます。このメッセージは、実行時に入出力エラーが起きたこと、および U451 異常終了が起きたこと (実際の異常終了が起きていない場合) を示します。

n=451 の場合、オペレーターが DFS0451A メッセージに対して CONT を応答したとしても、IMS は異常終了 U0451 で終了します。

IOEROPN パラメーターを使用すれば、入出力エラーが起きたときの固有の JCL 条件コードを設定でき、後続のジョブ・ステップでその条件コードをテストできます。このパラメーターを指定しない場合には、アプリケーション・プログラムから渡される戻りコードはオペレーティング・システムに渡され、データベースの入出力エラーを示すのは状況コードおよびコンソール・メッセージのみになります。

WTORNO | WTORYES
WTORYES を指定すると、DFS0451A 入出力エラー・メッセージについての WTOR が出され、DL/I はオペレーターによる応答を待ってから続行します。ABEND と応答すると、IMS は U0451 異常終了で終了します。 CONT と応答すると、IMS は続行します。他の応答をすると、DFS0451A メッセージが再び出されます。

WTORYES または WTORNO を指定する場合は、IOEROPN も指定する必要があります。

CREATE SCHEMA のキーワード・パラメーター

CREATE SCHEMA ステートメントには、以下のキーワード・パラメーターが定義されています。

ALTRESPNO | ALTRESPYES
応答モード、会話型モード、または排他モードで端末に応答するために、入出力 PCB の代わりにこの代替スキーマを使用できるかどうかを指定します。代替スキーマ (TP PCB) のみに有効です。デフォルトは ALTRESPNO です。

EXPRESSNO | EXPRESSYES
アプリケーション・プログラムが異常終了した場合に、この代替スキーマからのメッセージを送信するか、バックアウトするかを指定します。代替スキーマ (TP PCB) のみに有効です。デフォルトは EXPRESSNO です。

EXPRESSYES が指定されている場合、プログラムが異常終了するか、ROLL または ROLB 呼び出しを出した場合でも、メッセージを宛先端末に送信できることを示します。これらの条件下にあるすべての PCB について (高速でも非高速でも)、挿入されても伝送可能になっていないメッセージは取り消されますが、伝送可能にされたメッセージは取り消されません。

非高速 PCB の場合は、プログラムが同期点 (コミット・ポイント) に達するまで、メッセージを宛先に伝送することはできません。同期点が起こるのは、プログラムが終了するか、CHKP 呼び出しを出すか、あるいは次の入力メッセージを要求する (トランザクションが MODE=SNGL で定義されているとき) 場合です。

高速 PCB の場合は、IMS が完全なメッセージを持っていることを認識している場合に、メッセージは宛先に伝送できます。メッセージは、PURG 呼び出しがその PCB を使用して行われるか、あるいはプログラムが次の入力メッセージを要求するときに、利用可能です。

IMS システム定義で PSB が高速機能アプリケーションとして定義されている場合、EXPRESSYES を指定しても、応答代替 PCB の実行時には無視されます。

EXPRESSNO が指定されている場合、アプリケーション・プログラムが異常終了すると、メッセージがバックアウトされることを示します。

MODIFYNO | MODIFYYES
代替スキーマが変更可能であるかどうかを指定します。この機能により、このスキーマに関連する宛先名を動的に修正することができます。 MODIFYYES を指定する場合は、USING 節を省略します。代替スキーマ (TP PCB) のみに有効です。デフォルトは MODIFYNO です。

SAMETRMNO | SAMETRMYES
応答代替スキーマに名前を指定された論理端末が、入力メッセージの発信元となった論理端末と同じ物理端末に割り当てられていることを IMS が検証するかどうかを指定します。応答モードの端末で動作するプログラムおよび会話型プログラムが使用する応答代替スキーマについては、SAMETRMYES を指定する必要があります。排他モードの出力専用装置にメッセージを送信するために代替応答スキーマを使用する場合は、SAMETRMNO を指定します。代替スキーマ (TP PCB) のみに有効です。デフォルトは SAMETRMNO です。

DBVER *n*

データベースのバージョン管理が使用可能な場合、このアプリケーション・プログラムで必要なデータベース定義 (DBD) のバージョン番号を指定します。

指定される数値は、DBD で定義され、IMS カタログに保管されているバージョン番号と一致している必要があります。データベース・バージョン番号の有効値は、0 から 2147483647 です。

PSB 内の複数の PCB が同じデータベースを参照している場合、それぞれの PCB で、同じ DBD バージョン番号を指定する必要があります。

LISTYES | LISTNO

入り口でアプリケーション・プログラムに渡される PCB リストに、名前を指定した PCB を含めるかどうかを指定します。指定した PCB を PCB リストに含めるときは、YES を指定します。指定した PCB を PCB リストから除外するときは、NO を指定します。YES がデフォルトです。

PCB リストから PCB を除外するときには、label または PCBNAME= パラメーターで PCB に名前を指定する必要があります。アプリケーション・プログラムが PCB のアドレスを必要としない場合には、LIST=NO を指定します。

POSSNGL | POSMULT

論理データ構造の単一または複数の位置付けを指定します。単一または複数の位置付けは、呼び出し機能にバリエーションを与えます。

単一位置付けと複数位置付けの間のパフォーマンスの変化はほとんどありません。HSAM は複数位置付けをサポートしていません。

POS=SINGLE または S がデフォルトです。

例外: 従属セグメントが 2 つを超える DEDB では、デフォルトは POS=MULTPLE または M です。

DEDB に PCB ステートメントで POS の値をコーディングしても、従属セグメントの個数に基づいて選択されるデフォルトが変更されることはありません。

SBNO | SBCON

順次バッファリング (SB) を使用してバッファーに入れる PCB を指定します。これはオプション・パラメーターです。バッチおよび BMP に関して DFSSBUX0 でデフォルト・オプションを SB=COND に変更していない限り、デフォルトは SB=NO です。

COND

SB を条件付きで活動化することを指定します。 IMS は、DB データ・セットに対するこの PCB の入出力参照パターンの統計をモニターします。IMS は、順次入出力参照パターンと妥当なアクティビティー率を検出する場合、SB をアクティブにし、必要なバッファーを獲得します。

NO この DB PCB に SB を使用しないことを指定します。

ヒント: 短時間実行の MPP、高速機能プログラム、および CICS プログラムでは、SB= キーワードを省略するか、SB=NO を指定する必要があります。

DEDBVIEW | MSDBVIEW

MSDB コミット・ビューの指定に使用します。既存のアプリケーションでは、MSDB コミット・ビューまたはデフォルトの DEDB コミット・ビューを使用

することができます。 DEDB に関して MSDB コミット・ビューを使用するには、ステートメントで VIEW=MSDB を指定します。VIEW=MSDB を指定しなかった場合は、DEDB では DEDB コミット・ビューを使用します。MSDB を DEDB にマイグレーションする場合は、既存のアプリケーション・プログラムのいずれにも変更を加える必要はありません。

VIEW=MSDB を指定する PCB を使用して REPL 呼び出しを発行する場合は、セグメントにキーが必要です。コマンド・コード 'D' が指定されている場合は、パス内のいずれのセグメントにもこの条件が当てはまります。そうでない場合は、状況 AM が戻されます。

PROCSEQ *index_dbname*

DBDNAME パラメーターに名前が指定されたデータベースを 2 次処理シーケンスで処理するときに使用する 2 次索引の名前を指定します。このパラメーターはオプションです。これが有効であるのは、このデータベースに 2 次索引が存在している場合のみです。このパラメーターを使用する場合は、後に続く SENSEG ステートメントが、索引付きデータベースの中のセグメント・タイプの 2 次処理シーケンス階層を反映している必要があります。例えば、最初の SENSEG ステートメントでは、PARENT=0 パラメーターで索引先セグメントの名前を指定する必要があります。

full_function_index_dbname は副次索引 DBD の名前でなければなりません。

2 次処理シーケンスの場合には、処理オプション L および LS は無効です。索引ターゲット・セグメントや、その逆親のいずれかを、挿入したり削除したりすることはできません。ブロックを作成する際に、これらのセグメントの処理オプションに I または D が含まれていると、処理オプションがこの制限を反映するよう変更されていることを示す警告メッセージが出されます。

PROCOPT *sch_procopt*

この PCB で宣言されているセンシティブ・セグメントに関する処理オプションを指定します。指定したこれらの処理オプションは、関連するアプリケーション・プログラムで使用できます。このパラメーターで最大 4 つのオプションを使用できます。パラメーター中の文字の意味は次のとおりです。

A すべてのオプション。PROCOPT=A には、デフォルトで G (取得)、I (挿入)、R (置換)、および D (削除) オプションが含まれます。PROCOPT=A はデフォルトの設定値です。

G Get オプション。

I 挿入オプション。PROCOPT=I には、デフォルトで高速機能 DEDB に対する G (取得) オプションが含まれます。PROCOPT=I には、他のデータベース・タイプに対する G オプションは含まれません。

R 置換オプション。PROCOPT=R には、デフォルトで G (取得) オプションが含まれます。

D 削除オプション。PROCOPT=D には、デフォルトで G (取得) オプションが含まれます。

P パス CALL です。コマンド・コード D を使用する場合は必須です。ただし、フィールドに依存しないバッチ・プログラムでの ISRT 呼び出しの場合は除きます。 DEDB を処理する際にコマンド・コード D を

使用する場合は、PROCOPT=P は必須ではありません。P は、A (すべて)、G (取得)、I (挿入)、D (削除)、および L (ロード) オプションと組み合わせて使用されます。

O PCB に O オプションを使用すると、IMS は戻されたセグメントの所有権を検査しません。したがって、保全性なしの読み取りプログラムでは、別のプログラムによって更新されたセグメントを取得する可能性があります。更新プログラムが異常終了してバックアウトした場合は、現在も過去もデータベースに存在していないセグメントを保全性なしの読み取りプログラムが取得することになります。あるセグメントが削除され、同じ位置に同じタイプの別のセグメントが挿入された場合は、セグメント・データ、およびアプリケーションへ返された後続のすべてのデータが、異なるデータベース・レコードから取得される可能性があります。したがって、O オプションを使用する場合は、そのオプションで読み取られたデータに基づいて更新を行わないでください。O は、GO、GON、GONP、GOT、GOTP、または GOP いずれかのみのとして指定する必要があります。

IMS はこれらのエラー・タイプのいくつかを認識し、それらを異常終了 U0849 に変換します。ただし、PROCOPT GOx の場合に起きるその他の条件は、保全性なしの読み取りが原因であるために検出されません。この場合、ループ、ハング、およびシステムの異常終了が発生するおそれがあります。この PROCOPT を使用する場合は、システム設計を慎重に検討して、並行更新アクティビティーがこのような種類の条件の発生リスクを高める可能性があるかどうかを判断する必要があります。

N 読み取り専用アプリケーション・プログラムが起こす異常終了の数を減らします。読み取り専用アプリケーション・プログラムは、別のアプリケーション・プログラムが更新しているデータを参照できます。このことが行われるときには、そのデータを指す無効なポインターが存在する場合があります。無効なポインターが検出されると、読み取り専用アプリケーション・プログラムは異常終了します。N を指定すると、この状態を回避できます。代わりに、GG 状況コードがプログラムに戻されます。プログラムは、処理を終了するか、別のセグメントを読み取って処理を続行するか、あるいは別のパスを使用してデータにアクセスするかを決定する必要があります。N を指定する場合は、GON、GONH、または GONP として指定する必要があります。

T T を指定すると DL/I により自動的に操作が再試行されることを除いて、N オプションと同様です。再試行が失敗すると、GG 状況コードがアプリケーション・プログラムに戻されます。T を指定する場合、GOT、 GOTH、または GOTP として指定する必要があります。

E これを指定すると、オンライン・プログラムがデータベースまたはセグメントを排他使用できるようになります。これは、G、I、D、R、および A と共に使用します。

制約事項**:** DEDB の場合、PROCOPT=E は許可されません。

L データベースのロードのためのロード・オプション (HIDAM および PHIDAM を除く)。

GS 昇順でのみセグメントを読み取ります (HSAM の場合のみ)。HSAM デ

ータベースの場合に GS を指定すると、DL/I IMS 領域において基本順次アクセス方式 (BSAM) ではなく、待機順次アクセス方式 (QSAM) を使用してセグメントが読み取られます。

LS 昇順でのみセグメントをロードします (HIDAM、HDAM、PHIDAM、PHDAM)。このロード・オプションは、HIDAM および PHIDAM の場合は必須です。HIDAM および PHIDAM データベースの場合には LS を指定する必要があるため、ルート・セグメント・シーケンス・フィールドの索引がデータベースのロード時に作成されます。

H 特定の PSB を使用するアプリケーション・プログラムについて高速順次処理を指定します。PROCOPT=H の使用には、以下の制限が適用されます。

- DEDB のみに使用できます。
- PCB レベルでは使えますが、セグメント・レベルでは使えません。
- 他の高速機能処理オプションと一緒に使用する必要があります。
- PROCOPT には、H を含めて最大 4 つまでオプションを指定できます。
- BMP の場合にのみ指定できます。
- 1 つの PSB につき、データベース当たり 1 つの PROCOPT=H PCB しか使用できません。 HSSP を使用する BMP が、同じ PSB 内の同じデータベースに関して、PROCOPT=H を持つ複数の PCB を使用すると、最初に使用された PCB 以外の PCB を使用するすべてのデータベース呼び出しで FH 状況コードを受け取ります。SETO ステートメントでキーワード NOPROCH を使用すると、この制限を軽減することができます。
- PROCSEQD=*Fast_Path_index_dbdname* が指定されている場合、PROCOPT=H は使用できません。
- PROCOPT=H は、PROCOPT=GO と一緒に使用することができません。

H は A、G、I、R および D と共に使用されます。

PROCOPT 値は、最大 4 文字の長さで指定することができます。これには、A、G、I、R、D、および L のうち少なくとも 1 つのオプションが必要です。つまり、A、G、I、R、D、および L のグループからのオプションがない場合、オプション E、S、P、O、N、T、または H が存在することはできません。

デフォルトの PROCOPT 値は A です。PROCOPT GOx および Lx のグループは、ダイアグラムに示されたシーケンスに従う必要があります。

CREATE SENSEGVIEW のキーワード・パラメーター

CREATE SENSEGVIEW ステートメントには、以下のキーワード・パラメーターが定義されています。

PROCOPT *ssv_procopt*
関連付けられているアプリケーション・プログラムがこのセンシティブ・セグメントを使用する場合の有効な処理オプションを表します。このパラメーターは、PCB ステートメントの PROCOPT= パラメーターと同じ意味を持ちます。この

パラメーターの有効オプションの他に、PCB ステートメントでは使用できず、SENSEG ステートメントで使用できるオプションが 1 つあります。PROCOPT が K である場合は、キー・センシティビティーのみを示します。 SSA を持たない GN 呼び出しは、データ・センシティブ・セグメントのみにアクセスできます。キー・センシティブ・セグメントが SSA での検索用に指定されている場合は、そのセグメントはユーザーの入出力域には移されません。キーは、PCB のキー・フィードバック域の該当するオフセット位置に置かれます。この PROCOPT= パラメーターを指定しない場合、PCB PROCOPT パラメーターがデフォルトとして使用されます。 PCB ステートメントと SENSEG ステートメントに指定した処理オプションが異なる場合、それらのオプションに互換性があれば、SENSEG PROCOPT が PCB PROCOPT をオーバーライドします。先行する PCB ステートメントに PROCOPT= L または LS を指定した場合には、このパラメーターを省略する必要があります。

PROCOPT= L または LS を指定する場合は、仮想論理子のセグメント・タイプには SENSEG ステートメントを指定してはなりません。置換機能と削除機能は、GET 機能も暗黙指定します。

セグメントに PROCOPT=K の指定があれば、非修飾 GN (GET NEXT) 呼び出しは、PROCOPT が K 以外のセンシティブ・セグメントにスキップします。

SENSEG PROCOPT は PCB PROCOPT を変更します。PROCOPT=E が PCB の中で指定されている場合は、SENSEG PROCOPT にも E を指定する必要があります (その SENSEG 専用にスケジュールする場合)。

SENSEG ステートメントの中で処理オプション N または T を指定しても無効です。これらの処理オプションは PCB ステートメントの中でしか指定できません。

DEDB 順次従属セグメントの処理オプションは、G または I のいずれかでなければなりません。これらの値のいずれかを PCB ステートメントで指定しない場合は、SENSEG PCB ステートメントに PROCOPT=G または I を指定する必要があります。

連結セグメントの場合、PROCOPT= パラメーターは、連結セグメントの論理子セグメントを制御します。連結セグメントの論理親は、SEGM PCB ステートメントの RULES= パラメーターで制御されます。

INDICES

どの 2 次索引に検索フィールドを含めるかを指定します。この検索フィールドは索引先セグメント・タイプの SSA を修飾するために使用されます。INDICES= パラメーターは、索引先セグメント・タイプにのみ指定できます。これを指定すると、索引先セグメント・タイプの呼び出しの SSA を、指定された各 2 次索引に含まれている索引先セグメント・タイプの検索フィールド上で修飾できます。

制約事項:

- 索引先セグメント・タイプの呼び出しの SSA は、副次索引の検索フィールドでは修飾できません。ただし、索引セグメント・タイプの場合の SENSEG ステートメントの INDICES= パラメーターか、または PCB ステートメントの PROCSEQ= パラメーターに、その副次索引が指定されている場合は別です。

- INDICES= パラメーターは、高速機能副次索引ではサポートされません。

list1 には、最大 32 個の副次索引の DBD 名を指定できます。複数の名前を指定する場合には、それらの名前をコンマで区切って、そのリストを括弧で囲む必要があります。

REFERENCES *table_name*
論理関係に関与している表、あるいは副次索引がアクセスするデータベース内の表は、REFERENCES を使用してこの表の親表を識別します。

論理関係に関与している表の場合、名前は、表の論理親の IMS 内部名でなければなりません。

副次索引によって指されるソース・セグメントの物理親である表の場合、ソース・セグメントの名前を使用します。

副次索引によって指されるソース・セグメントの従属である表の場合、従属セグメントの物理親の名前を使用します。

REFERENCES キーワードは、副次索引ソース・セグメントには適用されません。

物理 (非論理) データベースを参照し、副次索引を持つデータベース内にないスキーマあるいは PCB は、REFERENCES キーワードをサポートしません。CREATE SENSEGVIEW 節がルートを重要であると定義する場合、REFERENCES キーワードを省略します。

REPLYES | REPLNO
このフィールドを置き換え呼び出しで変更するかどうかを指定します。NO または N を指定できます。指定しない場合は、REPLACE=YES (または Y) がデフォルトになります。

SSPTR
サブセット・ポインターの数とサブセット・ポインターのセンシティビティーを指定します。最大 8 個のサブセット・ポインターを定義できます。サブセット・ポインターの数 (第 1 パラメーター) は、1 から 8 でなければなりません。サブセット・ポインターのセンシティビティー (第 2 パラメーター) は、R (読み取りセンシティブ) または U (更新) でなければなりません。第 1 パラメーターと第 2 パラメーターを指定しない場合、ポインターにはセンシティビティーがありません。n のみを指定した場合は、ポインターは読み取りセンシティブです。SSPTR=R はデフォルトです。

処理オプションが A、R、I、または D でない場合は、U (更新センシティビティー) を使用できません。

sf_name
FIELD ステートメントで定義したこのフィールドの名前。このフィールドは、1 文字から 8 文字の英数字です。

START(*n*)
ユーザーの入出力域内のセグメントの先頭に対応するこのフィールドの開始位置を指定します。セグメントの最初のバイトの *startpos* は 1 です。*startpos* は値が 32 767 を超えない 10 進数でなければなりません。

ステートメントのタイプ

以下は、CREATE PROGRAMVIEW ステートメントのスキーマのタイプをリストしています。

Alternate_schema_statement

代替スキーマ (PCB) では、現行入力メッセージのソース以外の宛先を記述します。

このステートメント命令により、アプリケーション・プログラムでは、入力メッセージのソース以外の宛先に出力メッセージを送信できます。

注: 出力を送る宛先ごとに 1 つのスキーマ・ステートメントが必要です。

出力メッセージは、出力端末か、あるいは別のプログラムが処理する入力トランザクション・キューのいずれかに送信することができます。それぞれの出力メッセージの宛先に、別個の代替スキーマ (PCB) 宛先が必要です。出力で応答する必要のあるものが入力ソース端末のみの場合、このタイプのスキーマ・ステートメントを含めてはなりません。メッセージ処理プログラム、バッチ・メッセージ処理プログラム、および高速機能プログラムは、それぞれ関連する PROGRAMVIEW の中に代替スキーマ・ステートメントを持つことができます。代替スキーマは、高速機能トランザクションへのメッセージの送信に使用することはできません。しかし、高速機能アプリケーション・プログラムは、代替スキーマを使用して、任意の端末または IMS トランザクションにメッセージを経路指定することができます。

DB_schema_statement

DL/I、高速機能、または GSAM データベースのアプリケーション・プログラム・アクセスを記述するスキーマ・ステートメント。

通常、これらのスキーマ・ステートメントの 1 つ以上が含まれますが、常に必要なわけではありません。例えば、メッセージ通信プログラムまたは会話型メッセージ・プログラムでは、データベースへのアクセスを要求することができません。そのため、データベース・スキーマは不要です。

PROGRAMVIEW で定義できるスキーマの最大数は 2500 です。これは、すべての IMS 領域タイプ (MSG、DL/I など) 内で実行されるアプリケーション・プログラムに対する最大値です。

使用上の注意

CREATE PROGRAMVIEW ステートメントは PSB を記述します。 PCB を記述するには、CREATE PROGRAMVIEW ステートメントに 1 つ以上の CREATE SCHEMA ステートメントがネストされる必要があります。

SENSEG を記述するには、各 CREATE SCHEMA ステートメントに 1 つ以上の CREATE SENSEGVIEW ステートメントがネストされる必要があります。SENFIELD は、各 SENSEGVIEW にネストすることができます。 SCHEMA ステートメントおよび SENSEGVIEW ステートメントを指定する順序は重要です。

```
CREATE PROGRAMVIEW ... (
 CREATE SCHEMA ... (
  CREATE SENSEGVIEW ... ,
  CREATE SENSEGVIEW ... WITH ...,
  CREATE SENSEGVIEW ...
 )
```

```
 CREATE SCHEMA ... (
  CREATE SENSEGVIEW ... (
   senfld WITH ...
   senfld WITH ...
   senfld WITH ...
  ) WITH ... ,
  CREATE SENSEGVIEW ... (
   senfld WITH ...
   senfld WITH ...
   senfld WITH ...
),
  CREATE SENSEGVIEW ... (
   senfld WITH ...
   senfld WITH ...
   senfld WITH ...
  )
 )
 ...
)
```

例

以下の例は、複数の PCB を持つ従来の IMS プログラム仕様ブロック・マクロ・ステートメントのサンプルを示しており、その後に、CREATE PROGRAMVIEW、CREATE SCHEMA、および CREATE VIEW の各ステートメントを使用する同等の DDL のサンプルを示しています。

PSB ユーティリティー・ソース:

```
*********************************************************************
*    DB  PCB NUMBER 1      DB   DEDBJN21
*********************************************************************
  PCB       TYPE=DB,DBDNAME=DEDBJN21,POS=M,PROCOPT=A,KEYLEN=26,        C
               PCBNAME=PCB01,EXTERNALNAME=PCB01
    SENSEG  NAME=HOSPITAL,PARENT=0
    SENSEG  NAME=PAYMENTS,PARENT=HOSPITAL,PROCOPT=GI
    SENSEG  NAME=WARD,PARENT=HOSPITAL
    SENSEG  NAME=PATIENT,PARENT=WARD
    SENSEG  NAME=ILLNESS,PARENT=PATIENT
    SENSEG  NAME=TREATMNT,PARENT=ILLNESS
    SENSEG  NAME=DOCTOR,PARENT=TREATMNT
    SENSEG  NAME=BILLING,PARENT=PATIENT
    SENSEG  NAME=ARRAY,PARENT=HOSPITAL
    SENSEG  NAME=STRUCT,PARENT=HOSPITAL
    SENSEG  NAME=REDEFINE,PARENT=HOSPITAL
    SENSEG  NAME=MAP,PARENT=HOSPITAL
    SENSEG  NAME=EXFLDSEG,PARENT=HOSPITAL
    SENSEG  NAME=NUMSEGM,PARENT=HOSPITAL
*********************************************************************
*    DB  PCB NUMBER 2      DB   DEDBJN21
*********************************************************************
  PCB       TYPE=DB,DBDNAME=DEDBJN21,POS=M,PROCOPT=GO,KEYLEN=26,       C
               PCBNAME=PCB10,EXTERNALNAME=PCB10
    SENSEG  NAME=HOSPITAL,PARENT=0
    SENSEG  NAME=PAYMENTS,PARENT=HOSPITAL
    SENSEG  NAME=WARD,PARENT=HOSPITAL
    SENSEG  NAME=PATIENT,PARENT=WARD
    SENSEG  NAME=ILLNESS,PARENT=PATIENT
    SENSEG  NAME=TREATMNT,PARENT=ILLNESS
    SENSEG  NAME=DOCTOR,PARENT=TREATMNT
    SENSEG  NAME=BILLING,PARENT=PATIENT
    SENSEG  NAME=ARRAY,PARENT=HOSPITAL
    SENSEG  NAME=STRUCT,PARENT=HOSPITAL
    SENSEG  NAME=REDEFINE,PARENT=HOSPITAL
    SENSEG  NAME=MAP,PARENT=HOSPITAL
```

```
...
*********************************************************************
*    DB  PCB NUMBER 11          HIDAM HOSPITAL DB DH41SK01
*********************************************************************
  PCB       TYPE=DB,DBDNAME=DH41SK01,POS=M,PROCOPT=AP,KEYLEN=26,       C
               PCBNAME=PCB11,EXTERNALNAME=PCB11
     SENSEG  NAME=HOSPITAL,PARENT=0
     SENSEG  NAME=PAYMENTS,PARENT=HOSPITAL,PROCOPT=GI
     SENSEG  NAME=WARD,PARENT=HOSPITAL
     SENSEG  NAME=PATIENT,PARENT=WARD
     SENSEG  NAME=ILLNESS,PARENT=PATIENT
     SENSEG  NAME=TREATMNT,PARENT=ILLNESS
     SENSEG  NAME=DOCTOR,PARENT=TREATMNT
     SENSEG  NAME=BILLING,PARENT=PATIENT
     SENSEG  NAME=PHARMACY,PARENT=HOSPITAL
     SENSEG  NAME=BACKORDR,PARENT=PHARMACY
     SENSEG  NAME=ARRAY,PARENT=HOSPITAL
     SENSEG  NAME=STRUCT,PARENT=HOSPITAL
     SENSEG  NAME=REDEFINE,PARENT=HOSPITAL
     SENSEG  NAME=MAP,PARENT=HOSPITAL
     SENSEG  NAME=SFTEST,PARENT=HOSPITAL
      SENFLD  NAME=SF1,START=1
      SENFLD  NAME=SF2,START=40
      SENFLD  NAME=SF3,START=30
*********************************************************************
*        PSBGEN PSBNAME=BMP255
*********************************************************************
     PSBGEN  PSBNAME=BMP255,LANG=ASSEM,CMPAT=YES,IOASIZE=32000,         C
                SSASIZE=32000
```

同等の DDL ステートメント:

```
CREATE PROGRAMVIEW bmp255 (
 CREATE SCHEMA pcb01.dedbjn21 AS pcb01 (
  CREATE SENSEGVIEW hospital,
  CREATE SENSEGVIEW payments.hospital WITH PROCOPT GI,
  CREATE SENSEGVIEW ward.hospital,
  CREATE SENSEGVIEW patient.ward,
  CREATE SENSEGVIEW illness.patient,
  CREATE SENSEGVIEW treatmnt.illness,
  CREATE SENSEGVIEW doctor.treatmnt,
  CREATE SENSEGVIEW billing.patient,
  CREATE SENSEGVIEW array.hospital,
  CREATE SENSEGVIEW struct.hospital,
  CREATE SENSEGVIEW redefine.hospital,
  CREATE SENSEGVIEW map.hospital,
  CREATE SENSEGVIEW exfldseg.hospital,
  CREATE SENSEGVIEW numsegm.hospital
 )
 PROCOPT A
POS=MULTIPLE|SINGLE,

 CREATE SCHEMA pcb10.dedbjn21 AS pcb10 (
  CREATE SENSEGVIEW hospital,
  CREATE SENSEGVIEW payments.hospital,
  CREATE SENSEGVIEW ward.hospital,
  CREATE SENSEGVIEW patient.ward,
  CREATE SENSEGVIEW illness.patient,
  CREATE SENSEGVIEW treatmnt.illness,
  CREATE SENSEGVIEW doctor.treatmnt,
  CREATE SENSEGVIEW billing.patient,
  CREATE SENSEGVIEW array.hospital,
  CREATE SENSEGVIEW struct.hospital,
  CREATE SENSEGVIEW redefine.hospital,
  CREATE SENSEGVIEW map.hospital
 )
 PROCOPT GO
```

```
 POSMULTI,

 CREATE SCHEMA pcb11.dh41sk01 AS pcb11 (
  CREATE SENSEGVIEW hospital,
  CREATE SENSEGVIEW payments.hospital WITH PROCOPT GI,
  CREATE SENSEGVIEW ward.hospital,
  CREATE SENSEGVIEW patient.ward,
  CREATE SENSEGVIEW illness.patient,
  CREATE SENSEGVIEW treatmnt.illness,
  CREATE SENSEGVIEW doctor.treatmnt,
  CREATE SENSEGVIEW billing.patient,
  CREATE SENSEGVIEW pharmacy.hospital,
  CREATE SENSEGVIEW backordr.pharmacy,
  CREATE SENSEGVIEW array.hospital,
  CREATE SENSEGVIEW struct.hospital,
  CREATE SENSEGVIEW redefine.hospital,
  CREATE SENSEGVIEW map.hospital,
  CREATE SENSEGVIEW sftest.hospital (
sf1 WITH START(1),
sf2 WITH START(40),
sf3 WITH START(30)
  )
 )
 PROCOPT AP
 POSMULTI
)
LANGASSEM
CMPATYES
IOASIZE 32000
SSASIZE 32000
```

TPPCB ソース:

```
PCB    TYPE=TP,NAME=OUTPUT1
        PCB    TYPE=TP,NAME=OUTPUT2
        PCB    TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,KEYLEN=100
        SENSEG NAME=PARTMAST,PARENT=0,PROCOPT=A
        SENSEG NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
        PCB    TYPE=GSAM,DBDNAME=REPORT,PROCOPT=LS
        PSBGEN LANG=COBOL,PSBNAME=APPLPGM3
        END
```

同等の DDL ステートメント:

```
CREATE PROGRAMVIEW applpgm3 (
 CREATE SCHEMA TP pcb01.output1,
 CREATE SCHEMA TP pcb02.output2,

 CREATE SCHEMA pcb03.partmstr AS pcb03 (
  CREATE SENSEGVIEW partmast WITH PROCOPT 'A',
  CREATE SENSEGVIEW cpws.partmast WITH PROCOPT 'A'
 )
 PROCOPT 'A',

 CREATE SCHEMA pcb04.report
 PROCOPT 'LS',
)
LANGCOBOL
```

TPPCB ソース:

```
PCB      TYPE=DB,NAME=FISDBD1,PROCOPT=GRP,KEYLEN=20
        SENSEG   NAME=EMPLOYEE,PARENT=0
        SENFLD   NAME=EMPLNAME,START=13,REPL=NO
        SENFLD   NAME=EMPFNAME,START=1,REPL=NO
        SENFLD   NAME=EMPMI,START=11
        SENSEG   NAME=OFFICE,PARENT=EMPLOYEE
```

```
        SENSEG   NAME=EMPLPROJ,PARENT=EMPLOYEE
        SENFLD   NAME=PROJNUM,START=1
        SENFLD   NAME=PROJTITLE,START=10
        SENFLD   NAME=EPFUNCTN,START=35
        SENFLD   NAME=EPTIMEST,START=60
        SENFLD   NAME=EPTIMCUR,START =70
        PSBGEN   LANG=ASSEM,PSBNAME=APPLPGM1
        END

CREATE SENSEGVIEW APPLPGM1 (
 FISDBD1 DB FOR FISDBD1 PROCOPT 'GRP'
 EMPLOYEE SENSEG IN FISDBD1
 EMPLNAME SENSEG IN FISDBD1 START (13) REPLACEN
 ...
) LANGASSEM

CREATE PROGRAMVIEW applpgm1 (
 CREATE SCHEMA pcb01.fisdbd1 AS pcb01 (
  CREATE SENSEGVIEW employee (
   emplname WITH START (13) REPLNO,
   empfname WITH START (1) REPLNO,
   empmi WITH START (11)
  )

  CREATE SENSEGVIEW office.employee,

  CREATE SENSEGVIEW emplproj.employee (
   projnum WITH START (1),
   projtitle WITH START (10),
   epfunctn WITH START (35),
   eptimest WITH START (60),
   eptimcur with START (70)
  )
 )
)
LANGASSEM
```

高速機能の例:

```
PCB    TYPE=DB,DBDNAME=MSDBLM01,PROCOPT=R,KEYLEN=4 NONTERMINAL-RELATED
SENSEG NAME=LDM,PARENT=0                        (DEFAULT)
PCB    TYPE=DB,DBDNAME=MSDBLM02,PROCOPT=R,KEYLEN=1 NONTERMINAL-RELATED
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM03,PROCOPT=R,KEYLEN=2 NONTERMINAL-RELATED
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM04,PROCOPT=R,KEYLEN=8 NONTERMINAL-RELATED
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM05,PROCOPT=R,KEYLEN=8 FIXED RELATED
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=A,KEYLEN=8 DYNAMIC RELATED
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=R,KEYLEN=8 DYNAMIC RELATED
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=G,KEYLEN=8 DYNAMIC RELATED
SENSEG NAME=LDM,PARENT=0
PSBGEN LANG=ASSEM,PSBNAME=APPLPGM1              END OF PSBGEN MACRO
END    END OF PSB GEN

CREATE PROGRAMVIEW applpgm1 (
 CREATE SCHEMA pcb01.msdblm01 AS pcb01 (
  CREATE SENSEGVIEW lim
 )
 PROCOPT 'R',
 CREATE SCHEMA pcb02.msdblm02 AS pcb02 (
  CREATE SENSEGVIEW lim
 )
 PROCOPT 'R',
```

```
 CREATE SCHEMA pcb03.msdblm03 AS pcb03 (
  CREATE SENSEGVIEW lim
 )
 PROCOPT 'R',
 CREATE SCHEMA pcb04.msdblm04 AS pcb04 (
  CREATE SENSEGVIEW lim
 )
 PROCOPT 'R',
 CREATE SCHEMA pcb05.msdblm05 AS pcb05 (
  CREATE SENSEGVIEW lim
 )
 PROCOPT 'R',
 CREATE SCHEMA pcb06.msdblm06 AS pcb06 (
  CREATE SENSEGVIEW lim
 )
 PROCOPT 'A',
 CREATE SCHEMA pcb06.msdblm06 AS pcb06 (
  CREATE SENSEGVIEW lim
 )
 PROCOPT 'R',
 CREATE SCHEMA pcb06.msdblm06 AS pcb06
  CREATE SENSEGVIEW lim
 )
 PROCOPT 'G'
)
LANGASSEM
```

TPPCB ソース:

```
DEDB SSPTR
PCB    TYPE=DB,DBDNAME=MSDBLM01,PROCOPT=R,     NONTERMINAL-RELATED     X
PCB TYPE=DB,DBDNAME=X,PROCOPT=A,KEYLEN=100
SENSEG NAME=A,PARENT=0
SENSEG NAME=B,PARENT=A,SSPTR=((1,R),(2,U),(5))
SENSEG NAME=C,PARENT=B
SENSEG NAME=D,PARENT=A,SSPTR=((2,R))
PSBGEN LANG=COBOL,PSBNAME=APPLPGM1
END

CREATE PROGRAMVIEW applpgm1 (
 CREATE SCHEMA pcb01.msdblm01 AS pcb01
 PROCOPT 'R',

 CREATE SCHEMA pcb02.x AS pcb02 (
  CREATE SENSEGVIEW a,

  CREATE SENSEGVIEW b.a
  SSPTR (1 R,2 U,5),

  CREATE SENSEGVIEW c.b,

  CREATE SENSEGVIEW d.a
  SSPTR (2 R)
 )
 PROCOPT 'A'
)
LANGCOBOL
```

TPPCB ソース:

```
PCB        TYPE=DB,DBDNAME=DTA3,PROCOPT=A,KEYLEN=15,PROCSEQ=X4
SENSEG     NAME=DA,PARENT=0
SENSEG     NAME=DB,PARENT=DA
SENSEG     NAME=DC,PARENT=DA,INDICES=X5
SENSEG     NAME=DD,PARENT=DC
SENSEG     NAME=DE,PARENT=DC,INDICES=X6
PSBGEN     LANG=COBOL,PSBNAME=APPLPGM1
```

```
END


CREATE PROGRAMVIEW applpgm1 (
 CREATE SCHEMA pcb01.dta3 AS pcb01 (
  CREATE SENSEGVIEW da,

  CREATE SENSEGVIEW db.da,

  CREATE SENSEGVIEW dc.da WITH INDICES (X5)

  CREATE SENSEGVIEW dd.dc

  CREATE SENSEGVIEW de.dc WITH INDICES (X6)
 )
 PROCOPT 'A'
 PROCSEQ X4
)
LANGCOBOL
```

CREATE TABLE

CREATE TABLE ステートメントは、新規の表を定義します。

制約事項: 以下のいずれかのキーワードを CREATE TABLE ステートメントに指定した場合は、ALTER TABLE ステートメントを使用してキーワードとキーワード値を変更することはできません。キーワードとキーワード値を変更するには、まず DROP TABLE ステートメントを使用して表を削除する必要があります。次に、CREATE TABLE ステートメントを使用して表を再作成し、キーワードとキーワード値を再び指定する必要があります。

- 定義するセグメント・タイプの内部名を指定する INTERNALNAME*internalname* キーワード。
- DIRECT DEPENDENT | SEQUENTIAL DEPENDENT

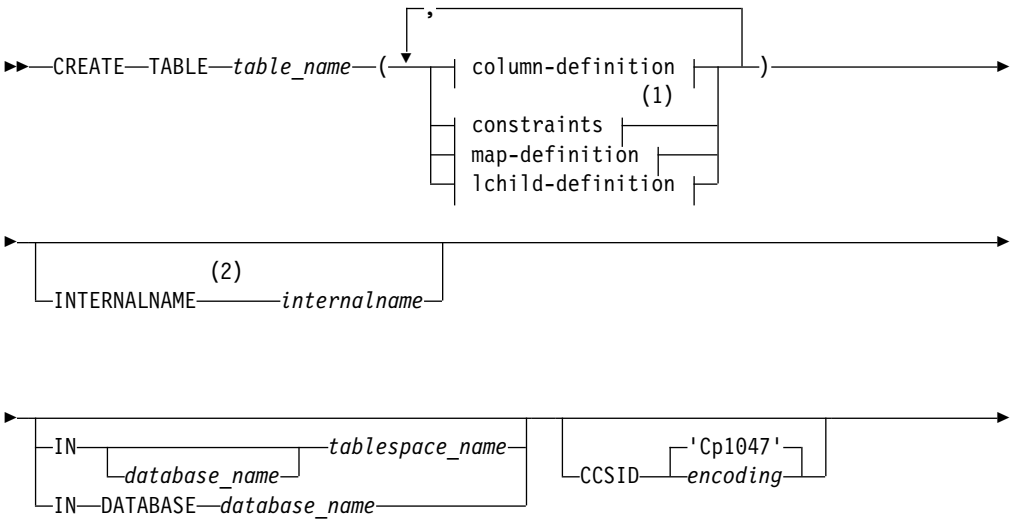
呼び出し

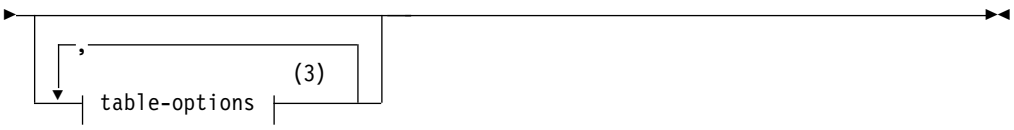
このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。 これは実行可能ステートメントですが、動的に準備することはできません。

- 813 ページの『CREATE TABLE 構文』
- 814 ページの『column-definition 構文』
- 814 ページの『data-type 構文』
- 815 ページの『ims-column 構文』
- 815 ページの『inline-constraints 構文』
- 816 ページの『constraint 構文』
- 816 ページの『references-clause 構文』
- 816 ページの『map-definition 構文』
- 816 ページの『case-definition 構文』
- 816 ページの『lchild-definition 構文』
- 816 ページの『lchild-option 構文 (HISAM)』
- 817 ページの『lchild-option 構文 (HDAM)』
- 817 ページの『lchild-option 構文 (HIDAM)』

- 817 ページの『lchild-option 構文 (PHDAM または PHIDAM)』
- 817 ページの『lchild-option 構文 (全機能副次索引データベースの INDEX)』
-
- 817 ページの『lchild-option 構文 (PSINDEX)』
- 817 ページの『xdfld-options 構文 (HISAM、SHISAM、HDAM、HIDAM、PHDAM、または PHIDAM)』
- 818 ページの『table-options 構文 (PHIDAM または PHDAM)』
- 818 ページの『table-options 構文 (HIDAM または HDAM)』
- 819 ページの『table-options 構文 (DEDB)』
- 819 ページの『table-options 構文 (HISAM または SHISAM)』
- 820 ページの『table-options 構文 (HSAM または SHSAM)』
- 820 ページの『table-options 構文 (INDEX)』
- 820 ページの『table-options 構文 (LOGICAL)』
- 820 ページの『source-clause 構文』
- 820 ページの『editproc-clause 構文』
- 821 ページの『lparent-clause 構文』
- 821 ページの『data_capture 構文』
- 821 ページの『exit_changes 構文』
- 821 ページの『exit_attributes 構文』

CREATE TABLE 構文





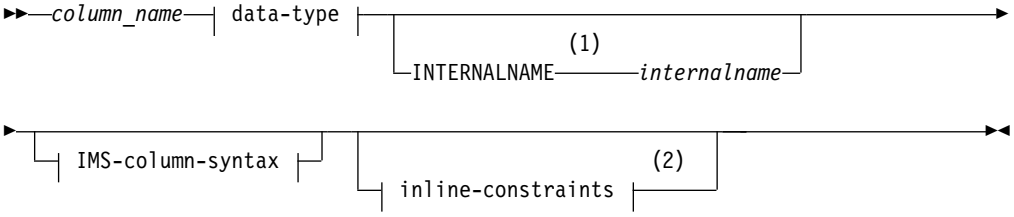
注:

1 constraints フラグメントおよび lchild-definition フラグメントは、GSAM データベースでは無効です。

2 INTERNALNAME は、GSAM データベースでは無効です。

3 table-options フラグメントは、GSAM データベースでは無効です。

column-definition 構文

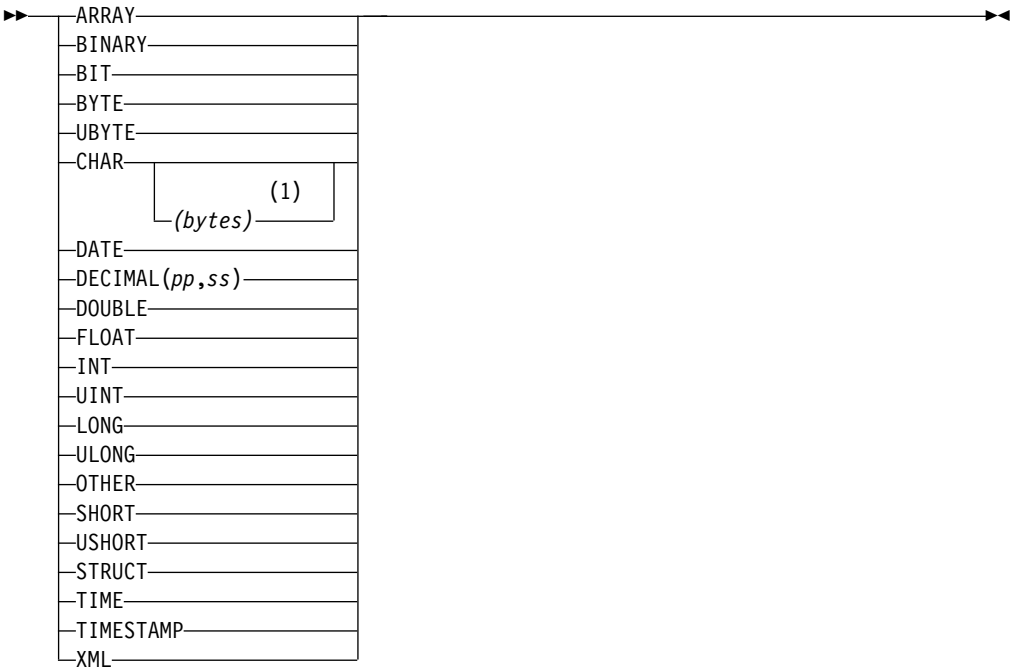


注:

1 INTERNALNAME は、GSAM データベースでは無効です。

2 inline-constraints フラグメントは、GSAM データベースでは無効です。

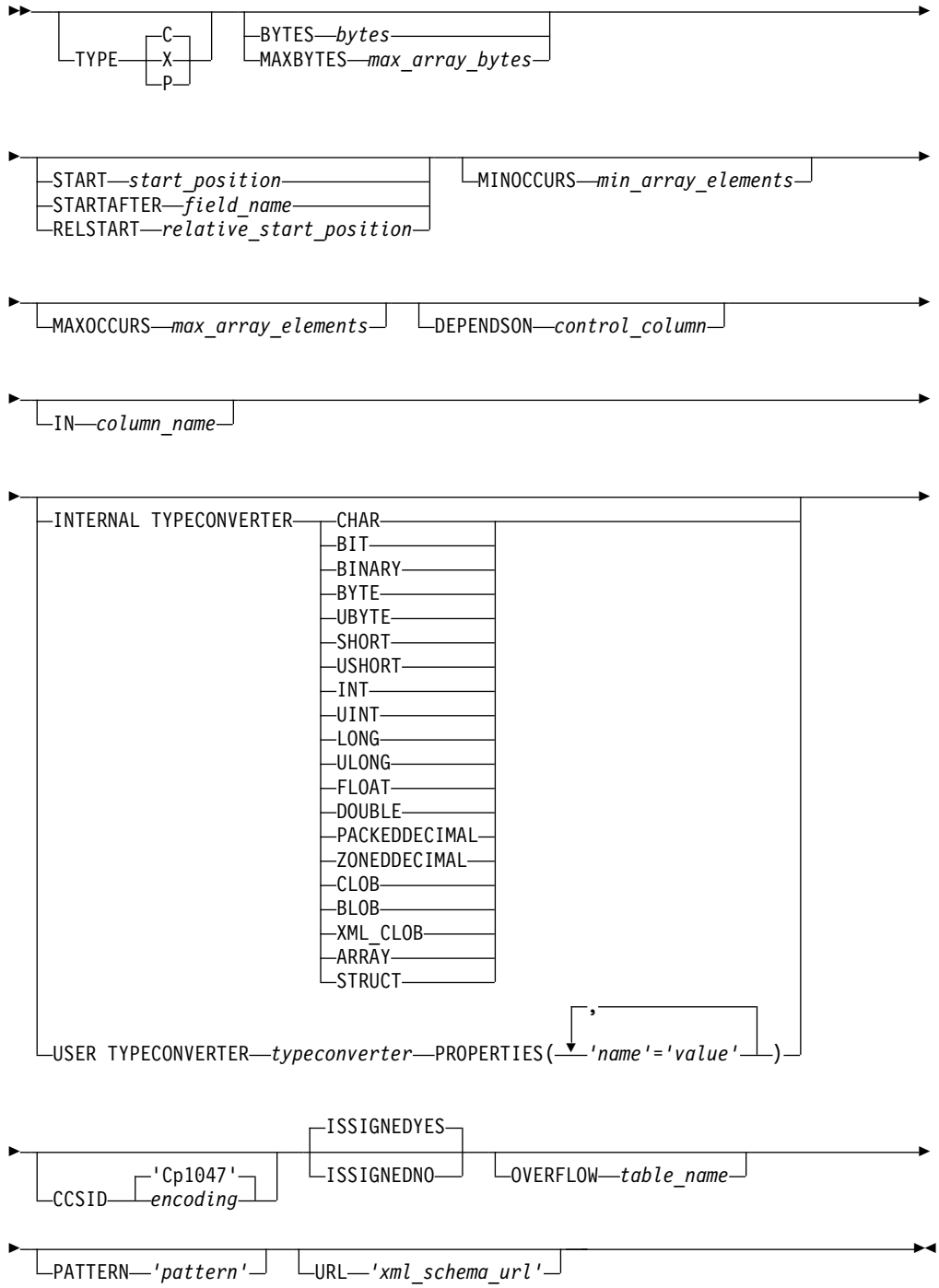
data-type 構文



注:

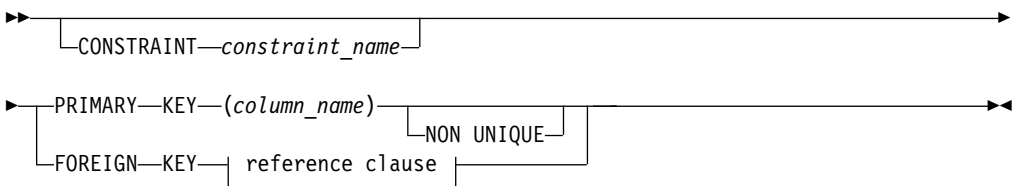
1 CHAR のデフォルトのバイト数は 1 です。

ims-column 構文

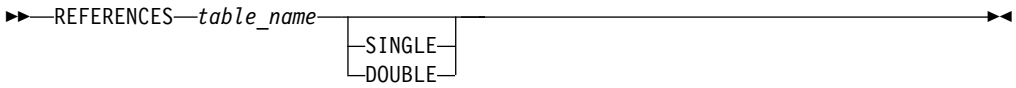


inline-constraints 構文

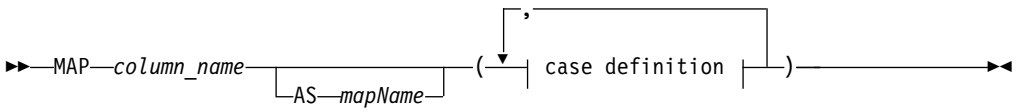
constraint 構文



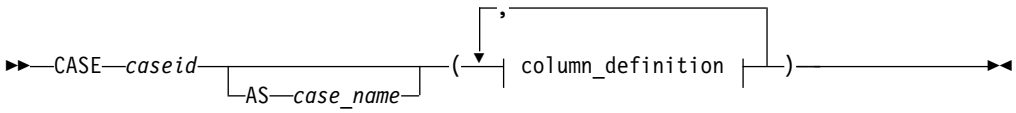
references-clause 構文



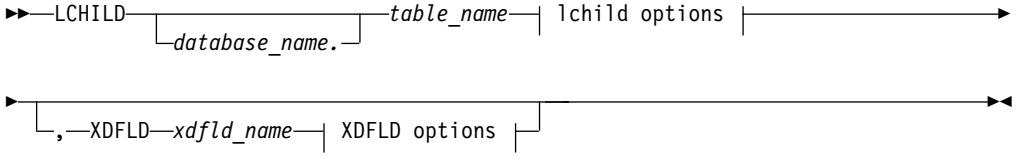
map-definition 構文



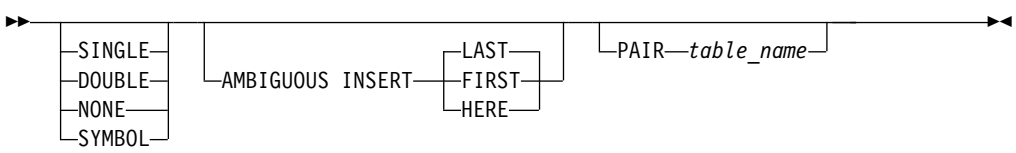
case-definition 構文



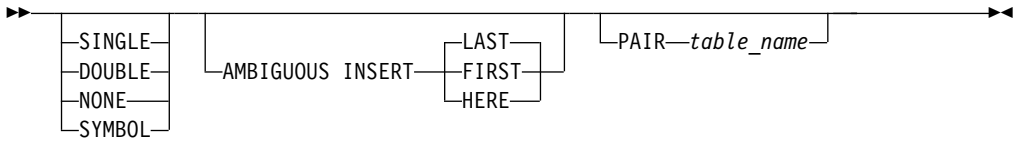
lchild-definition 構文



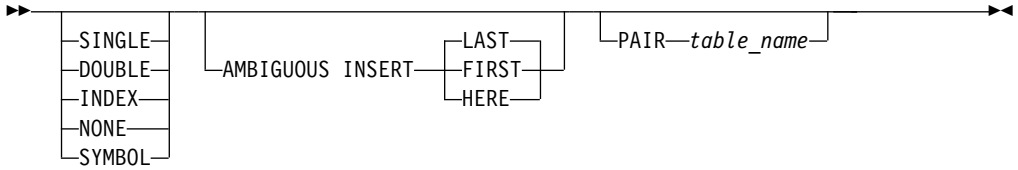
lchild-option 構文 (HISAM)



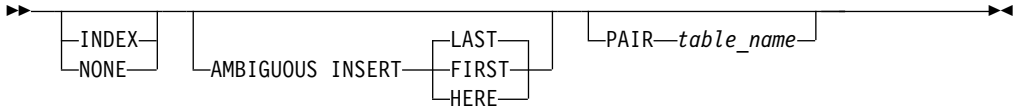
lchild-option 構文 (HDAM)



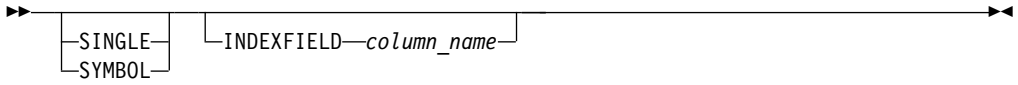
lchild-option 構文 (HIDAM)



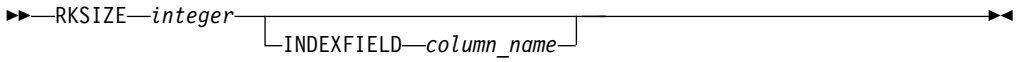
lchild-option 構文 (PHDAM または PHIDAM)



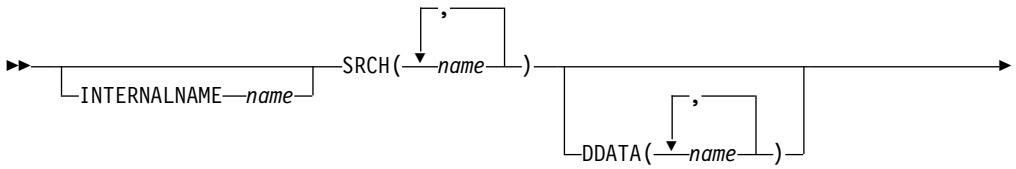
lchild-option 構文 (全機能副次索引データベースの INDEX)



lchild-option 構文 (PSINDEX)



xdfld-options 構文 (HISAM、SHISAM、HDAM、HIDAM、PHDAM、または PHIDAM)



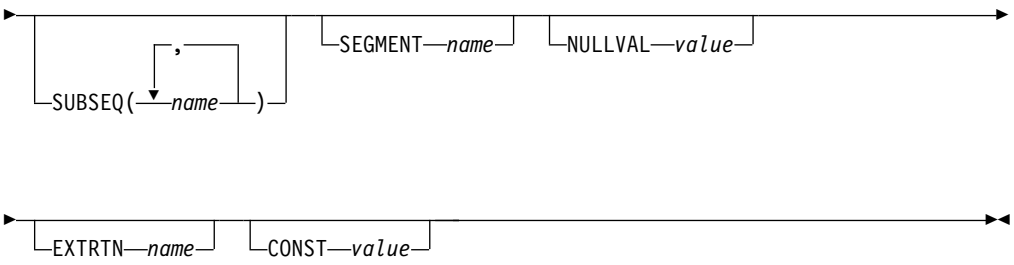


table-options 構文 (PHIDAM または PHDAM)

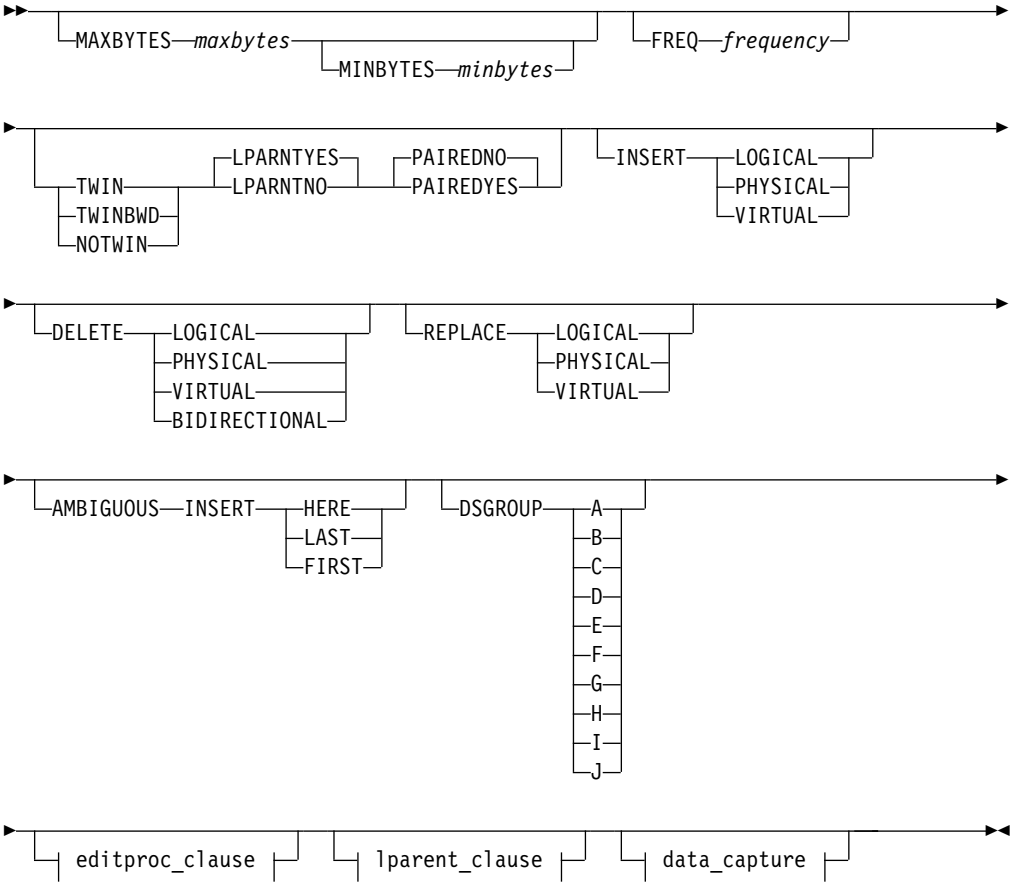
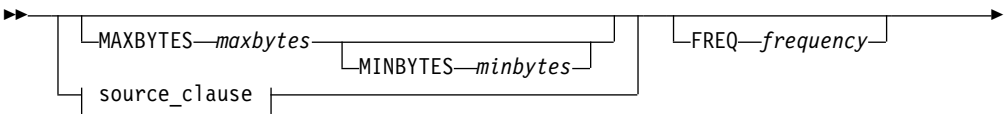


table-options 構文 (HIDAM または HDAM)



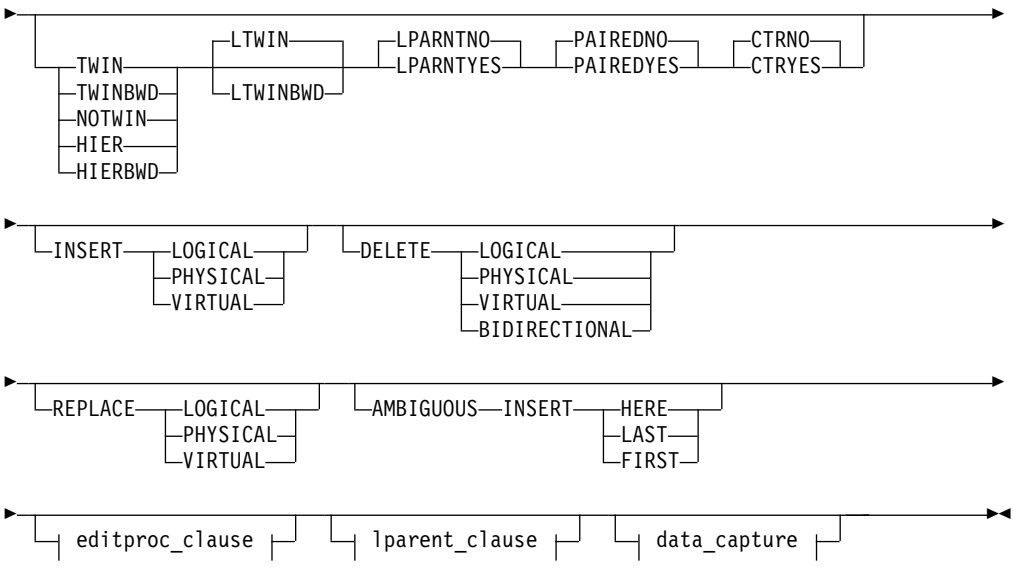


table-options 構文 (DEDB)

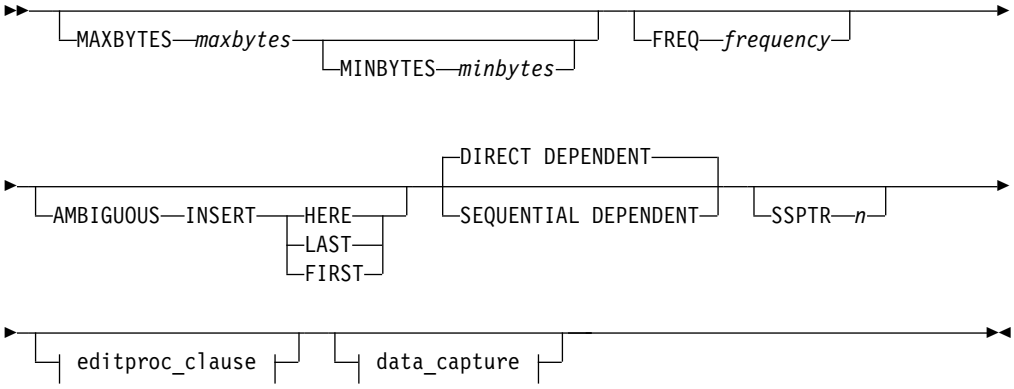
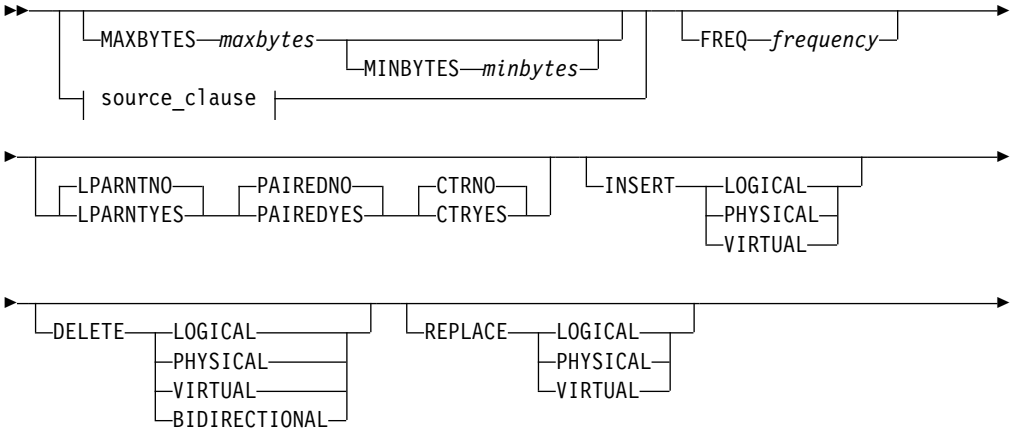


table-options 構文 (HISAM または SHISAM)



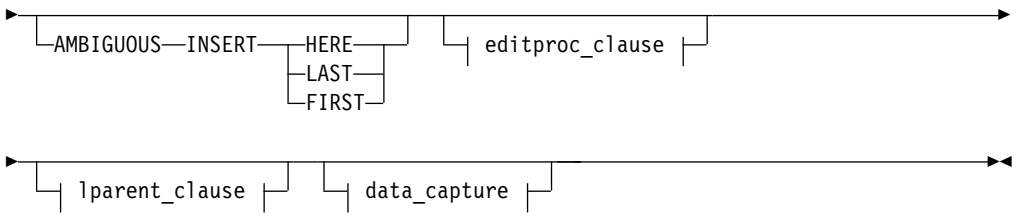


table-options 構文 (HSAM または SHSAM)

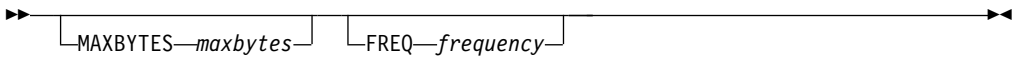


table-options 構文 (INDEX)

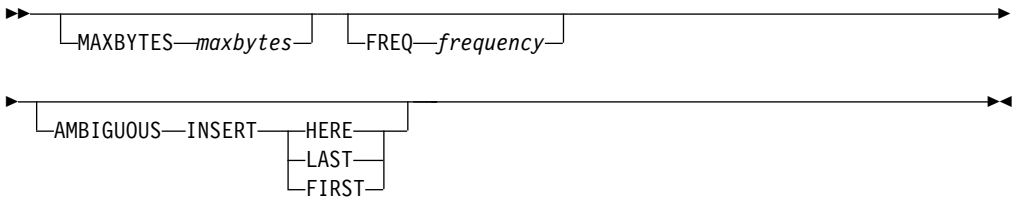
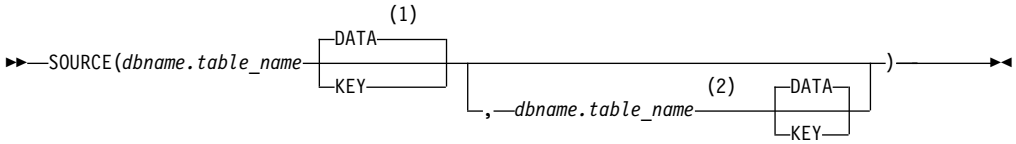


table-options 構文 (LOGICAL)

source-clause 構文

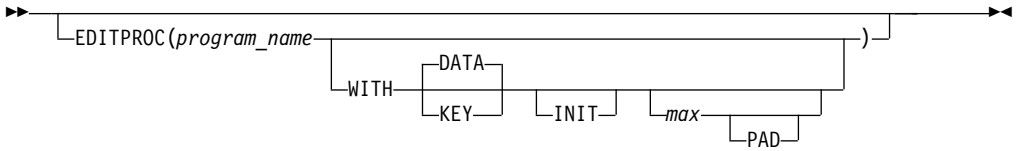


注:

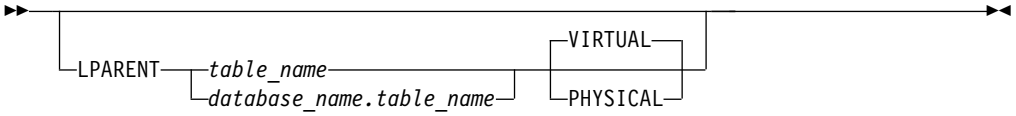
1 KEY は、HISAM および SHISAM では許可されません。

2 2 つ目の部分は、HISAM および SHISAM では許可されません。

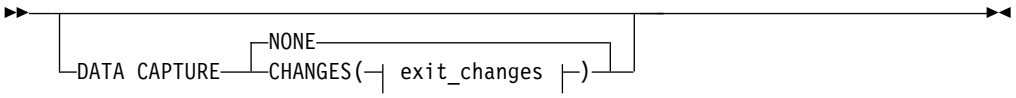
editproc-clause 構文



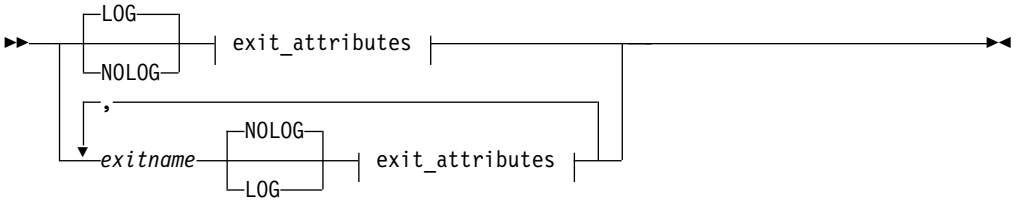
lparent-clause 構文



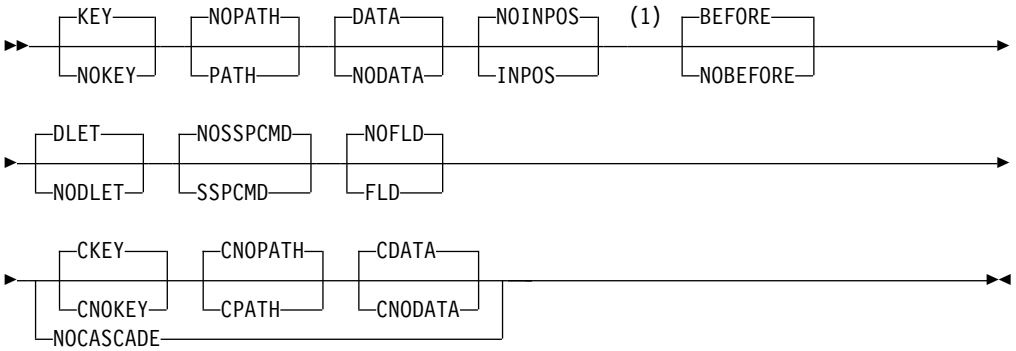
data_capture 構文



exit_changes 構文



exit_attributes 構文



注:

1 BEFORE、NOBEFORE、DLET、NODLET、SSPCMD、NOSSPCMD、FLD、および NOFLD は、DEDB 専用です。

CREATE TABLE のキーワード・パラメーター

CREATE TABLE ステートメントには、以下のキーワード・パラメーターが定義されています。

TABLE *table_name*
外部名は 1 文字から 128 文字の大文字の英数字ストリングとして指定します。表の名前には、下線文字を含めることができます。表の名前は、データベース内で固有でなければなりません。

制約事項: 表の名前は、予約済みの SQL キーワードにすることはできません。また、DFS で開始することもできません。

lchild_definition
前に column-definition がなければなりません。

INTERNALNAME *internalname*
定義するセグメント・タイプの内部名を指定します。指定した名前は、このセグメントへのすべての参照で DL/I およびアプリケーション・プログラムにより使用されます。重複するセグメント名は許可されません。 *internalname* パラメーターは、1 文字から 8 文字の英数字値にする必要があります。各文字は、A から Z または 0 から 9 の範囲、あるいは文字 $、#、または @ でなければなりません。

制約事項: 名前の先頭文字を数値にすることはできません。

制約事項: 定義するセグメント・タイプの内部名を指定した後で、ALTER TABLE ステートメントを使用して内部名を変更することはできません。セグメント・タイプの内部名を変更するには、まず DROP TABLE ステートメントを使用して表を削除する必要があります。次に、CREATE TABLE ステートメントを使用して表を再作成し、新しい内部名を指定する必要があります。

INTERNALNAME パラメーターのデフォルト値は、'TBL' 接頭部から始まり、その後に増分番号が続きます。 IMS はデフォルトの内部名を生成するためにオプションとしてリストされますが、独自の内部名を指定することが強く推奨されます。これにより、IMS およびプログラム・ビュー (PSB) で使用される内部名を制御することができます。

例えば、次のようにします。

```
TBL00001
TBL00255
```

IN *dbname.tablespace_name*
表が属するデータベースおよび表スペースを指定します。

注: この節は、DEDB、LOGICAL、PSINDEX、PHIDAM、および PHDAM データベースには適用されません。代わりに、IN DATABASE を使用します。

IN DATABASE *dbname*
表が属するデータベースを指定します。表スペース名が指定されない場合、表は最後に定義された表スペースに関連付けられます。

CREATE TABLE (table-options) のキーワード・パラメーター

CREATE TABLE (table-options) ステートメントには、以下のキーワード・パラメーターが定義されています。

SOURCE
IMS 内部表の名前であり、以下の 2 つの目的で使用されます。

- 定義する仮想論理子セグメント・タイプで表される実論理子セグメント・タイプを識別するため。
- 論理データベース内に定義されるセグメント・タイプで表される、物理データベース内のセグメント・タイプ (複数も可) を識別するため。

制約事項: PHDAM および PHIDAM データベースでは物理対しかサポートしないため、SOURCE キーワードは使用できません。

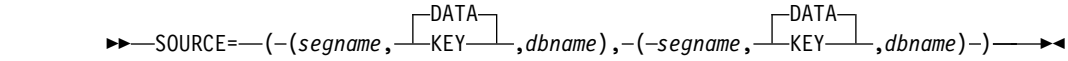
仮想論理子を定義するステートメントは、次のようになります。

segname
実論理子の名前を指定します。

DATA
segname のキーとデータの両方の部分を、セグメントの組み立てに使用することを示します。このパラメーターは必須です。

dbname
実論理子が入っている物理データベースの名前を指定します。

論理データベースのセグメント・タイプを定義するステートメントは、次のようになります。



(*segname*, KEY | DATA,*dbname*)
最初のオカレンスは、論理セグメントとして定義されている物理データベース内のセグメントを指すか、またはこの論理データベースの連結セグメント・タイプの最初の部分として使用する物理データベース内の論理子セグメント・タイプを指します。

segname
物理データベース内の論理子セグメント・タイプを指します。

KEY
segname に指定されているセグメントのキー部分を、キー・フィードバック域に入れることを指定します。*segname* で表されている論理セグメント・タイプを処理するための呼び出しを発行するときは、このセグメントをユーザーの入出力域に入れてはなりません。

DATA
segname で指定したセグメントのキー部分をキー・フィードバック域に入れる必要のあることと、*segname* で表されている論理セグメント・タイプを処理するための呼び出しを発行するときは、このセグメントをユーザーの入出力域に入れる必要があることを指定します。

dbname
segname を含む物理データベースの名前を指定します。(*segname*, KEY|DATA, *dbname*) の 2 番目のオカレンスは、この論理データベース内の連結セグメントの目標親部分に使用される、物理データベース内の論理親セグメント・タイプまたは物理親セグメント・タイプを指し示します。この 2 番目のオカレンスの各パラメーターの説明は、最初のオカレンスの説明と同じです。

(*segname*, KEY | DATA, *dbname*) の最初のオカレンスが仮想論理子を指している場合、2 番目のオカレンス (指定する場合) は、実論理子の物理親を指していなければなりません。

ソース・セグメントを使用して、連結セグメントを表すときは、検索呼び出しの際にこの 2 つのセグメントのいずれを (または両方を) ユーザーの入出力域に入れるかを、KEY と DATA のパラメーターで指定します。DATA を指定すると、セグメントはユーザーの入出力域に入れられます。 KEY を指定すると、セグメントはユーザーの入出力域には入れられず、シーケンス・フィールド・キー (存在している場合) が PCB のキー・フィードバック域に入れられます。連結セグメントのキーは、論理子がどのパスからアクセスされるかによって、論理子のキー、つまり物理兄弟シーケンス・フィールドか論理兄弟シーケンス・フィールドのいずれかになります。KEY および DATA パラメーターは、検索タイプの呼び出しのみに適用されます。

挿入呼び出しでは、ユーザーの入出力域には常に論理子セグメントと、挿入規則が物理でない限り、論理親セグメントが入っていなければなりません。KEY がセグメントについて指定されている場合でも、参照されたセグメントを含んでいる論理データベースに対して呼び出しが出された場合には、そのセグメントを含んでいるデータベースを IMS が使用できなければなりません。SOURCE セグメント指定の最初のオカレンスが論理子を指している場合は、連結セグメントの目標親を指す 2 番目のオカレンスも指定する必要があります。明示的に指定されていなければ、デフォルトによりブロックの作成時に KEY パラメーターで組み込まれます。

論理 DBD 生成で定義するセグメントは、1 つ以上の物理 DBD 生成で前もって定義されているセグメントの物理定義を使わなければなりません。

SEGM ステートメントで INDEX データ・セット内のセグメントを定義する場合は、SOURCE パラメーターは無効です。

MAXBYTES *maxbytes*
MINBYTES *minbytes*

minbytes パラメーターを含む場合には、セグメント・タイプを可変長として定義します。maxbytes フィールドでは、このセグメント・タイプのオカレンスの最大長を指定します。maxbytes パラメーターに許される最大値と最小値は、固定長セグメントに関して説明した値と同じです。

セグメントが圧縮ルーチンによって処理される場合は、異常終了 0799 を回避するために、セグメント長を指定された最大定義より長くできるかどうかを示す制御情報を収容できるよう maxbytes フィールドを設定します。拡張を可能にするには、maxbytes に 10 バイトの任意の値を追加します。

minbytes パラメーターでは、可変長セグメントが使用する最小ストレージ量を指定します。minbytes の最大値は、maxbytes に指定する値です。minbytes の最小値は次のものでなければなりません。

- セグメント・タイプを編集/圧縮ルーチンで処理しない場合、または編集/圧縮ルーチンで処理はするがキー圧縮オプションを指定していない場合は、セグメント・タイプにシーケンス・フィールドを指定してあれば、minbytes にはシーケンス・フィールド全体を入れられる値を指定しなければなりません。

- キー圧縮オプションを使う編集/圧縮ルーチンで処理するセグメント・タイプ、または順序付けられていないセグメントの場合の最小値は、4 です。

HSAM、SHSAM、INDEX、PSINDEX、または SHISAM データベースのセグメントは可変長にすることができないため、これらのデータベースでは minbytes パラメーターは無効です。

高速機能 DEDB では、セグメントは、2 バイト・フィールド (この 2 バイト長フィールドを含むセグメントの長さを定義) で始まり、列で指定するユーザー・データが後に続きます。 minbytes の値には、4 (最小値) から maxbytes (最大値) を指定できますが、minbytes 値は、このセグメントのシーケンス・フィールドを入れるのに十分な大きさでなければなりません (つまり、minbytes ≥ START - 1 + 表に続くシーケンス・フィールドの BYTES)。例えば、シーケンス・フィールドの長さが 20 バイトで START= 7 のセグメントでは、最小 minbyte 値は 26 です。該当する任意の DL/I 呼び出しでは、実際のセグメント長は、シーケンス・フィールドを含んだ長さと maxbytes の値の間になります。maxbytes の値は、制御インターバル・サイズ - 120 を超えてはなりません。

TWINBWD | NOTWIN | TWIN | HIER | HIERBWD
定義するセグメント・タイプのオカレンスの接頭部に、ポインター・フィールドを予約することを指定します。 これらのフィールドは、このセグメントを隣接した親セグメントおよび兄弟セグメントに関連付けるために使用されます。

TWINBWD
定義するセグメントの接頭部に、4 バイトの物理兄弟順方向ポインター・フィールドと 4 バイトの逆方向物理兄弟ポインター・フィールドを予約します。逆方向物理兄弟ポインターを使用すると、削除パフォーマンスが向上します。

推奨事項: このオプションは、HIDAM および PHIDAM データベースのルート・セグメントに使用するようお勧めします。

NOTWIN
定義するセグメント・タイプのオカレンスの接頭部に、物理兄弟順方向ポインターのスペースを予約しないようにします。

NOTWIN は、次の場合に従属セグメント・タイプに対して指定できます。

- 物理親に階層ポインターが指定されていない。
- 物理親セグメント・タイプのオカレンスの物理子として保管されている従属セグメント・タイプのオカレンスが 1 つ以下である。

さらに、NOTWIN は、HDAM および PHIDAM データベースのルート・セグメント・タイプにも指定できますが、それはランダム化モジュールが同義語 (同じブロックおよびアンカー・ポイントを持っている異なる値のキー) を生成しない場合に限られます。

NOTWIN が従属セグメント・タイプについて指定されている場合に、従属セグメントの 2 番目のオカレンスを、ある与えられた物理親セグメントの物理子としてロードまたは挿入しようとすると、次のようになります。

- 初期ロード時に 2 番目のオカレンスを挿入しようとすると、LB 状況コードが戻されます。

- 初期ロード後に 2 番目のオカレンスを挿入しようとすると、II 状況コードが戻されます。

同義語をロードまたは挿入する試みはリジェクトされて、LB 状況コードまたは II 状況コードが出されます。

TWIN
定義するセグメントの接頭部に、4 バイトの物理兄弟順方向ポインター・フィールドを予約します。

HIER
定義するセグメント・タイプのオカレンスの接頭部に、4 バイトの階層順方向ポインター・フィールドを予約します。HALDB では HIER はサポートされません。

HIERBWD
定義するセグメント・タイプのオカレンスの接頭部に、4 バイトの階層順方向ポインター・フィールドと 4 バイトの階層逆方向ポインター・フィールドを予約します。逆方向階層ポインターを用いると、削除のパフォーマンスが上がります。 HALDB では HIERBWD はサポートされません。

LPARNTYES | LPARNTNO
論理親のタイプを指定します。

LPARNTYES
このパラメーターを指定できるのは、定義されるセグメント・タイプが論理子であり、しかも論理親が HDAM、HIDAM、PHDAM、または PHIDAM データベース内に存在する場合に限られます。論理親が HISAM データベース内にある場合は、このパラメーターを省略し、定義するセグメントの PARENT= パラメーターに PHYSICAL を指定してください。

HDAM、HIDAM、および HISAM データベースの場合、 LPARNT は、定義するセグメント・タイプのオカレンスの接頭部に 4 バイトの論理親ポインター・フィールドを予約します。

PHDAM および PHIDAM データベースの場合、LPARNT は、定義するセグメント・タイプのオカレンスの接頭部に 28 バイトの拡張ポインター・セットを予約します。

LPARNTNO
定義するセグメント・タイプが論理子ではなく、論理親が HDAM、HIDAM、PHDAM、および PHIDAM のデータベースのいずれにも存在しないことを指定します。

PAIREDYES | PAIREDNO
このセグメントが双方向論理関係を持つかどうかを指定します。

PAIREDYES
このセグメントが両方向論理関係を持つことを示します。このパラメーターは、以下のタイプに指定します。

- 仮想論理子セグメント・タイプ
- 両方向論理関係にある両方の物理対の論理子セグメント・タイプ

PAIRED を指定した場合、LTWIN および LTWINBWD パラメーターは無効になります。

PAIREDNO
このセグメントが双方向論理関係を持たないことを示します。

CTRNO | CTRYES

CTRNO
定義するセグメント・タイプのオカレンスの接頭部に、4 バイトのカウンター・フィールドを予約しません。

CTRYES

定義するセグメント・タイプのオカレンスの接頭部に、4 バイトのカウンター・フィールドを予約します。カウンターが必要になるのは、HISAM、HDAM、または HIDAM データベースの中の論理親セグメントが、論理子ポインターで自分に接続されていない論理子セグメントを持っている場合です。ユーザーがこのパラメーターを指定しなくても、カウンターは、必要とされるすべてのセグメントの中に、DBD 生成時に自動的に入れられます。しかし、後で DBD 生成を行うのを避けるために、ユーザーは将来のカウンターの必要性を予想して、このパラメーターを使用してセグメント・タイプのオカレンスの接頭部の中にカウンター・フィールドを確保しておくことができます。HALDB では CTR はサポートされません。

INSERT {LOGICAL | PHYSICAL | VIRTUAL}
DELETE {LOGICAL | PHYSICAL | VIRTUAL | BIDIRECTIONAL}
REPLACE {LOGICAL | PHYSICAL | VIRTUAL}
定義するセグメント・タイプのオカレンスの挿入、削除、および置換に使用される規則を指定します。これらのパラメーターは、論理子セグメント、およびそれらの物理親セグメントと論理親セグメントに指定します。論理関係を持たないすべてのセグメント・タイプについては、これらを省略する必要があります。

AMBIGUOUS INSERT {LAST | FIRST | HERE}
この表が定義するセグメント・タイプの新しいオカレンスが、物理データベースに挿入される場所を指定します (物理兄弟順序を確立します)。この値は、シーケンス・フィールドを持たないセグメント、またはシーケンス・フィールドが固有でないセグメントを処理する場合にのみ使用します。固有のシーケンス・フィールドが定義されているセグメント・タイプに指定された場合、この値は無視されます。

HDAM および PHDAM のルートを除いて、FIRST、LAST、または HERE の規則は、データベースの初期ロードには適用されず、セグメントはロード・モードに示された順序でロードされます。初期ロードまたは HD 再ロードで、固有のシーケンス・フィールドが HDAM のルートに関して定義されていない場合は、FIRST、LAST、または HERE の挿入規則によりルートがチェーニングされる順序が決まります。したがって、HDAM または PHDAM データベースの再ロードでは、HERE または FIRST が使用されると、順序付けられていないルートの順序が反転されます。

DEDB セグメント以外は、LAST がデフォルトです。

高速機能順次従属セグメント処理の場合は、FIRST の挿入規則が常に使用され、これを変更することはできません。直接従属セグメント処理の場合は、FIRST、LAST、または HERE を指定できます。デフォルトは HERE です。

FIRST
シーケンス・フィールドが定義されていないセグメントの場合には、既存のすべての物理兄弟の前に新しいオカレンスが挿入されます。固有でないシーケンス・フィールドが定義されているセグメントの場合には、同じシーケンス・フィールド値を持つ既存のすべての物理兄弟の前に新しいオカレンスが挿入されます。

LAST
シーケンス・フィールドが定義されていないセグメントの場合には、既存のすべての物理兄弟の後ろに新しいオカレンスが挿入されます。固有でないシーケンス・フィールドが定義されているセグメントの場合には、同じシーケンス・フィールド値を持つ既存のすべての物理兄弟の後ろに新しいオカレンスが挿入されます。

HERE
シーケンス・フィールドのないセグメントの場合には、位置が確立されている物理兄弟の直前に新しいオカレンスが挿入されます。挿入するセグメントの物理兄弟に位置が確立されていなければ、既存のすべての物理兄弟の前に新しいオカレンスが挿入されます。固有でないシーケンス・フィールドが定義されているセグメントの場合には、同じシーケンス・フィールド値を持ち、位置が確立されている物理兄弟の直前に新しいオカレンスが挿入されます。同じシーケンス・フィールド値を持っている物理兄弟に位置が確立されていなければ、同じシーケンス・フィールド値を持つすべての物理兄弟の前に新しいオカレンスが挿入されます。挿入位置は、直前の DL/I 呼び出しで確立された位置により異なります。

物理パスに対して出される挿入呼び出しでは、指定した挿入規則よりコマンド・コードの L (last) が優先されるため、新しいオカレンスは LAST の挿入規則に従って挿入されます。

DSGROUP
PHDAM および PHIDAM データベースについて、複数データ・セット・グループを指定します。形式は DSGROUP c です。ここで、c は A から J の文字です。これによって、PHDAM および PHIDAM データベースを最大 10 個のデータ・セット・グループに分割できます。すべてのセグメントで、デフォルトは A (1 区分当たり 1 個のデータ・セット) です。ルート・セグメントに指定する場合、DSGROUP A でなければなりません。

制約事項: A から J のシーケンスにギャップがあってはなりません。例えば、DSGROUP C を CREATE TABLE ステートメントで指定した場合、DSGROUP B を指定した CREATE TABLE ステートメントが少なくとも 1 個必要で、各 HALDB 区画は A、B、および C データ・セットを持つことになります。

FREQ *frequency*
このセグメントが物理親の各オカレンスごとに何回現れる可能性があるかを示す見積回数を指定します。frequency パラメーターは、0.01 から 2^{24}-1 の符号なしの 10 進数でなければなりません。これがルート・セグメントの場合、「frequency」は、定義されるデータベース内に現れるデータベース・レコードの最大数の見積もりです。従属セグメントに適用される場合の FREQ パラメーターの値は、データベースの各データ・セット・グループの論理レコード長および物理ストレージ・ブロック・サイズを判別するのに使用されます。

CCSID *encoding*

セグメント内の文字データのエンコードを指定する、1 文字から 25 文字のオプション・フィールド。

CCSID パラメーターに指定する値には、以下の文字を含めることはできません。

- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

表内の **CCSID** パラメーターの値は、このセグメントのデータベース内の **CCSID** パラメーターの値をオーバーライドします。表で **CCSID** パラメーターが指定されていない場合、デフォルト値は、データベースの **CCSID** パラメーターの値、または (データベースで **CCSID** が指定されていない場合は) 値 `Cp1047` (EBCDIC エンコードを指定) のいずれかです。

この値は、列定義の **CCSID** パラメーターによって個々のフィールドでオーバーライドすることができます。

DIRECT DEPENDENT | SEQUENTIAL DEPENDENT

DEDB アクセス・タイプによって定義されたデータベース専用。DEDB 従属セグメントのタイプを記述します。 ルート・セグメントに指定してはなりません。 1 つの DEDB につき 1 つの順次従属セグメントしか許されていません。指定する場合には、これが最初の従属セグメント・タイプでなければなりません。直接従属セグメント・タイプがデフォルトです。

制約事項: DIRECT DEPENDENT または SEQUENTIAL DEPENDENT を指定して、DEDB 従属セグメントのタイプを定義した後で、ALTER TABLE ステートメントを使用してセグメント・タイプを変更することはできません。DEDB 従属セグメントのタイプを変更するには、まず DROP TABLE ステートメントを使用して表を削除する必要があります。次に、CREATE TABLE ステートメントを使用して表を再作成し、目的のセグメント・タイプのキーワードを指定する必要があります。

SSPTR *n*

DEDB アクセス・タイプによって定義されたデータベース専用。サブセット・ポインターの数を指定します。0 から 8 の値を指定できます。0 を指定するか、あるいは SSPTR を指定しない場合は、サブセット・ポインターを使用しないことになります。

EDITPROC *routinename*

DEDB または全機能データベースにセグメント編集/圧縮の出口ルーチンを選択します。

全機能データベースのセグメント編集/圧縮の場合

SOURCE キーワードを指定している場合は、このキーワードを指定してはなりません。DL/I EDITPROC キーワードは、
HSAM、SHSAM、SHISAM、INDEX のデータベース、および論理データベースには無効です。これは、すべてのデータベースの論理子セグメントについても無効です。HISAM データベースに使用する場合は、HISAM ルート・セグメントのシーケンス・フィールド・オフセットを変更してはなりません。さらに、セ

グメント編集/圧縮オプションが指定されているセグメント・タイプに指定できる最小セグメント長は、4 バイトです。

要確認**:** セグメント編集/圧縮出口ルーチンを使用しており、しかもセグメントを可変長として定義した場合は、可変長セグメントの圧縮時には、DBD で定義されたセグメントの最小の長さまでヌル・バイトが埋め込まれることに注意してください。最小のセグメントの長さが、圧縮長をオーバーライドします。これにより、過度に圧縮されたセグメントのロード時に、追加スペースが提供されます。

routinename
ユーザー提供の編集/圧縮出口ルーチンの名前を指定します。この名前は、1 文字から 8 文字の英数字値でなければならず、IMS.SDFSRESL 内の他の名前と同じであってはならず、また、データベース名と同じであってもなりません。

DATA
示された出口ルーチンがデータ・フィールドのみを圧縮または修正することを指定します。シーケンス・フィールドを修正してはならないだけでなく、セグメントの開始点に対してのシーケンス・フィールドの位置を変更するデータ・フィールドも修正してはなりません。圧縮ルーチンの名前が指定されていても、パラメーターが選択されていない場合は、DATA がデフォルトです。

KEY
指名されたセグメント内のどのフィールドも、出口ルーチンによって圧縮または修正が可能であることを指定します。このパラメーターは、HISAM データベースのルート・セグメントについては無効です。

INIT
セグメント出口ルーチンが初期設定と終了処理制御を必要とすることを示します。このパラメーターを指定すると、データベースのオープン後およびデータベースのクローズ後に編集/圧縮ルーチンに制御が渡されます。

max
圧縮出口ルーチンの間に固定長セグメントの長さを増やすことのできる最大バイト数を指定します。 1 バイトから 32 767 バイトを指定できます。 *max* のデフォルトは 10 です。

PAD
MAX で指定した数値は埋め込み用に使用し、MAX としては使用しないことを示します。1 から 32 767 までの数値は、挿入されたセグメントの圧縮後の長さが PAD 値よりも短いときに、そのサイズまで挿入セグメントが埋め込まれることを示します。

DEDB のセグメント編集/圧縮の場合

routinename
ユーザー提供のセグメント編集/圧縮出口ルーチンの z/OS ロード・モジュール名を指定します。ルーチン名が必要です。

DATA
セグメントのユーザー・データ部分のみが圧縮されることを指定します。 DATA がデフォルトです。

制約事項: KEY パラメーターは、DEDB についてはサポートされていません。KEY パラメーターを指定すると、エラー・メッセージが発行されます。

INIT
この指定により、データベースの最初の区域がオープンされた直後にセグメント圧縮出口ルーチンが制御を獲得し、データベースの最後の区域がクローズされる直前に制御を戻すことができます。指定された値の範囲内にセグメント長がある限り、セグメントの圧縮または拡張についてのフィールド修飾の検査中にエラーは起こりません。

制約事項: EDITPROC 節は、表の末尾に固有キー・フィールドが含まれる DEDB 表では禁止されています。

LPARENT *table_name* {VIRTUAL | PHYSICAL}
定義する表の論理親を指定します。

table_name
IMS 内部表の名前、および定義する表の論理親の名前を指定します。論理親が同じデータベース内に存在する場合、表の名前のみを指定するだけで構いません。論理親が別のデータベース内に存在する場合、データベースと表の名前の両方 (「database_name.tablename」など) を指定する必要があります。

VIRTUAL | PHYSICAL
論理親の連結キー (LPCK) を論理子セグメントの一部として保管するかどうかを指定します。論理子セグメントに対してのみパラメーターを指定してください。PHYSICAL を指定した場合は、LPCK は、各論理子セグメントと一緒に保管されます。VIRTUAL を指定した場合は、LPCK は論理子セグメントに保管されません。論理親が HISAM データベース内にある論理子セグメントに対しては、PHYSICAL を指定する必要があります。また、論理親の連結キーの任意の部分を使用して物理兄弟にチェーンに配列されている論理子セグメントに対しても指定する必要があります。

- PHDAM および PHIDAM
  - PHDAM および PHIDAM に対しては PHYSICAL がデフォルトです。
  - VIRTUAL を PHDAM または PHIDAM に対して指定した場合、それは無視され、PHYSICAL が使用されます。
- HDAM および HIDAM
  - HDAM および HIDAM に対しては VIRTUAL がデフォルトです。
  - HDAM および HIDAM データベース内のシンボリック・ポインターは、LPCK を使用し、PHYSICAL の指定を必要とします。

CREATE TABLE (column-definition) のキーワード・パラメーター

CREATE TABLE (column-definition) ステートメントには、以下のキーワード・パラメーターが定義されています。

column_name
column_name は、IMS カタログのみに保管され、定義するデータベースには保管されていない外部名を表します。外部名は 1 文字から 128 文字の大文字の英

数字ストリングとして指定します。外部名には下線文字を含めることができます。列名はセグメント内で固有でなければなりません。

制約事項**:** 列名は、予約済み SQL キーワードにすることも、DFS で開始することもできません。

IMS Universal ドライバーによって制限されている予約済み SQL キーワードのリストについては、IMS JDBC ドライバーにより制限されるポータブル SQL キーワード (アプリケーション・プログラミング)を参照してください。

INTERNALNAME *internalname*

セグメント・タイプ内のこのフィールドの名前を指定します。この名前は、アプリケーション・プログラムによって DL/I 呼び出し SSA で参照することができます。フィールド名はセグメント定義内で固有でなければなりません。fldname1 値は、1 文字から 8 文字の英数字値にする必要があります。以下のタイプのフィールドでは INTERNALNAME パラメーターは必須です。

- SEQ パラメーターを指定するキー順フィールド・タイプ
- セグメント検索引数 (SSA) によって参照されるフィールド・タイプ
- センシティブ・フィールドとして PSB が参照するフィールド・タイプ。
- XDFLD によって参照されるフィールド・タイプ

その他のフィールド・タイプでは、INTERNALNAME パラメーターを省略することができます。 INTERNALNAME パラメーターを省略すると、データベースのデータ管理ブロック (DMB) 内のストレージを節約できます。ただし、このフィールドでの検索を可能にするには INTERNALNAME パラメーターを指定する必要があります。以下のタイプのフィールドでは INTERNALNAME パラメーターを指定できません。

- 配列として定義されているフィールド。配列として定義されているフィールドには、フィールド定義に ARRAY が含まれています。
- 配列エレメントとして定義されているフィールド。配列エレメントであるフィールドは、列内の IN キーワードに配列フィールドの名前を指定します。
- 1 つ以上のネストされた動的配列を含む構造として定義されたフィールド。構造として定義されているフィールドには、列に STRUCT が含まれています。
- 動的配列を含む構造内に一緒に含まれているフィールド。構造内に含まれているフィールドは、列内の IN キーワードに構造フィールドの名前を指定します。
- セグメント内で動的配列の後に続くフィールド。動的配列の後に続くフィールドは、STARTAFTER パラメーターを指定します。
- 別のフィールドの開始位置に相対させた開始位置を指定する RELSTART パラメーターを含むフィールド。
- XML によって定義されたフィールド。

/CK 列および /SX 列の場合、INTERNALNAME パラメーターが指定されている必要があります。 /CK 名または /SK 名を指定する場合、それらの名前は二重引用符 (") で囲む必要があります。

- HSAM、SHSAM、INDEX、PSINDEX、および DEDB は、/CK 列も /SX 列も許可しません。
- HISAM および SHISAM は、/CK 列のみを許可します。

- HDAM、HIDAM、PHDAM、および PHIDAM は、/CK 列と /SX 列を許可します。

datatype のキーワード・パラメーター

CREATE TABLE (datatype) ステートメントには、以下のキーワード・パラメーターが定義されています。

ARRAY | BINARY | ...

DATATYPE パラメーターに DECIMAL が指定されている場合、デフォルトの INTERNAL TYPECONVERTER は符号付き PACKEDDECIMAL です。

DATE、TIME、または TIMESTAMP が指定されている場合は、INTERNAL TYPECONVERTER キーワードで LONG または CHAR を指定するか、USER TYPECONVERTER を指定することができます。 INTERNAL TYPECONVERTER LONG がデフォルトです。LONG が使用されると、その値は 1970 年 1 月 1 日からのミリ秒数として DASD に保管されます。

XML を指定すると、デフォルトの INTERNAL TYPECONVERTER は XML_CLOB です。これは、XML が指定されている場合に有効な唯一の値です。

STRUCT または ARRAY が指定されている場合、デフォルトの INTERNAL TYPECONVERTER はそれぞれ STRUCT または ARRAY です。これらは、このいずれかが指定されている場合に、それぞれ有効な唯一の値です。

その他のデータ・タイプの値ではすべて、その値がデフォルトの INTERNAL TYPECONVERTER として使用されます。

有効値は以下のとおりです。

ARRAY

ARRAY が指定された場合は、以下のようになります。

- INTERNALNAME パラメーターはサポートされません。
- BYTES または MAXBYTES パラメーターに指定するバイト値は、その配列に含まれるすべてのフィールドのバイト数の合計以上でなければなりません。

ARRAY として定義されたフィールド、または ARRAY を含むフィールドは再定義できません。

配列として定義されているフィールドには、フィールド定義に ARRAY が含まれています。

配列エレメントであるフィールドは、列内の IN キーワードに配列フィールドの名前を指定します。

BINARY

BINARY は、TYPE P または TYPE X で指定することができます。デフォルトで列サイズは 1 バイトになりますが、MAXBYTES キーワードを使用して独自のサイズを指定することができます。

BIT

BIT を指定する場合、列サイズを 1 バイトに設定します。MAXBYTES を指定する場合、指定できる値は 1 のみです。

BYTE
BYTES を指定する場合、列サイズを 1 バイトに設定します。MAXBYTES を指定する場合、指定できる値は 1 のみです。

UBYTE
UBYTE を指定する場合、列サイズを 1 バイトに設定します。MAXBYTES を指定する場合、指定できる値は 1 のみです。

CHAR
CHAR を指定する場合、デフォルトの列サイズは 1 バイトです。CHAR に続く括弧内または MAXBYTES キーワードで値を指定することで、実際の列サイズを指定することができます。例えば、CHAR(8) のようにします。

日付
DATE が指定されている場合、INTERNAL TYPECONVERTER CHAR または USER TYPECONVERTER *convertername* のいずれかが含まれている列定義も指定しない限りは、MAXBYTES 8 のみを指定することができます。

DECIMAL(*pp*,*ss*)

pp 精度。0 より大きい 1 バイトから 2 バイトの数値フィールド。

ss 位取り。0 以上の、1 バイトから 2 バイトの数値フィールド。 *ss* に指定する値は、*pp* の値より大きくてはなりません。

BYTES パラメーターには、使用されている 10 進数フォーマットと一致する値を指定する必要があります。

デフォルトの 10 進数フォーマットは符号付きパック 10 進数です。符号付きパック 10 進数フォーマットの場合に BYTES パラメーターとして必要な値を計算するには、*length* = ceiling ((*pp* + 1) / 2) という式を使用します。

デフォルトの 10 進数フォーマットは、INTERNAL TYPECONVERTER パラメーターを指定して変更できます。

INTERNAL TYPECONVERTER と指定してゾーン 10 進フォーマットを使用する場合は、*length* = *pp* という式を使用して、BYTES パラメーターの値を計算します。

DOUBLE
DOUBLE を指定する場合、MAXBYTES 8 のみを指定することができます。

FLOAT
FLOAT を指定する場合、MAXBYTES 4 のみを指定することができます。

INT
INT を指定する場合、MAXBYTES 4 のみを指定することができます。

UINT
UINT を指定する場合、MAXBYTES 4 のみを指定することができます。

LONG
LONG を指定する場合、MAXBYTES 8 のみを指定することができます。

ULONG
ULONG を指定する場合、MAXBYTES 8 のみを指定することができます。

OTHER
ユーザー定義のデータ型を使用することを指定します。OTHER を指定する場合は、ユーザー提供のタイプ・コンバーターを指定する USER TYPECONVERTER パラメーターを含む列定義も指定する必要があります。

SHORT
SHORT を指定する場合、MAXBYTES 2 のみを指定することができます。

USHORT
USHORT を指定する場合、MAXBYTES 2 のみを指定することができます。

STRUCT
STRUCT を指定する場合、構造に動的配列が含まれていると、この列を 1 次キーとして定義することはできません。動的配列は、データ・タイプ ARRAY と、DEPENDSON キーワードおよび MAXBYTES キーワードによって定義されます。

また、BYTES または MAXBYTES パラメーターに指定するバイト値は、その構造に含まれるすべてのフィールドのバイト数の合計以上でなければなりません。

TIME
TIME が指定された場合、INTERNAL TYPECONVERTER CHAR または USER TYPECONVERTER *convertername* が含まれている列定義も指定しない限り、MAXBYTES 8 のみを指定することができます。

TIMESTAMP
TIMESTAMP が指定された場合、**INTERNAL TYPECONVERTER CHAR** または **USER TYPECONVERTER** *convertername* が含まれている列定義も指定しない限り、MAXBYTES 8 のみを指定することができます。

XML

制約事項: INTERNALNAME キーワードが指定されている場合、あるいは 1 次キーとして定義されている列の場合は、XML はサポートされません。

CREATE TABLE (ims-column-syntax) のキーワード・パラメーター

CREATE TABLE (ims-column-syntax) ステートメントには、以下のキーワード・パラメーターが定義されています。

BYTES *bytes*
定義するフィールドの長さをバイトで指定します。システム関連フィールド以外のフィールドでは、BYTES は、値が 255 を超えない有効な自己定義項でなければなりません。

索引ソース・セグメント・タイプの連結キーまたはその連結キーの一部分をシステム関連フィールドとして定義する場合には、指定する値を 255 より大きくすることができますが、索引ソース・セグメントの連結キーの長さを超えてはなりません。

バイト長が 255 を超える可能性があるのは、IMS が検索できないように列が定義されている場合です。これらの列は、1 次キーとして定義することができず、INTERNALNAME キーワードを指定することができません。

/SX システム関連フィールドの長さは常に 4 バイトです。したがって、BYTES パラメーターは指定しても無視されます。

このフィールドが STRUCT または ARRAY によって構造または配列として定義されている場合、BYTES に指定する値は、その構造または配列に含まれるすべてのフィールドのバイト数の合計以上でなければなりません。

XML の場合、BYTES パラメーターはオプションであり、BYTES の有効値の範囲は 0 からそのセグメントの最大サイズまでです。XML の場合に BYTES パラメーターを省略すると、BYTES および MAXBYTES は許可されません。

CCSID *encoding*

列内の文字データのエンコードを指定する、単一引用符に囲まれた 1 文字から 25 文字のオプション・フィールド。これは、INTERNAL TYPECONVERTER が CHAR である場合にのみ有効です。

指定する値には、以下の文字を含めることができません。

- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

列に指定されていない場合、デフォルト値は、表で指定された値、あるいは (表で指定されない場合は) データベースで指定された値によって決まります。パラメーターが表でもデータベースでも指定されない場合、デフォルト値は Cp1047 であり、これは EBCDIC エンコードを指定します。

DEPENDSON

動的配列内のエレメントの数を定義するフィールドの名前を指定します。参照されるフィールドの列は、DEPENDSON パラメーターを指定する FIELD ステートメントの前に配置する必要があります。

DEPENDSON パラメーターは、ARRAY も指定されている場合にのみ有効です。MINOCCURS の値と MAXOCCURS の値が異なる場合、DEPENDSON は必須です。

DEPENDSON パラメーターによって参照されるフィールドは、以下のいずれかのデータ・タイプ値を指定して定義する必要があります。

- INT
- SHORT
- LONG
- (pp) または (pp,ss) が指定された DECIMAL。ss は 0 または 00。

TYPE {C | X | P}

このフィールド内のデータのマスクまたは埋め込みに IMS が使用する、文字のタイプを決定します。

C 英数字データまたはデータのタイプの組み合わせを指定します。 C を指定すると、IMS がフィールドの未使用バイトを充てんする必要がある場合、IMS は値を左寄せし、値の右側の未使用バイトを X'40' によって

充てんします。例えば、5 バイト・フィールド内の 3 バイト値 X'F5F4F3' は、X'F5F4F34040' として書き出されます。

X 16 進データを指定します。X を指定すると、IMS がフィールドの未使用バイトを充てんする必要がある場合、IMS は値を右寄せし、値の左側の未使用バイトを X'00' によって充てんします。例えば、5 バイト・フィールド内の 3 バイト値 X'543210' は、X'0000543210' として書き出されます。

P パック 10 進数データ。P を指定すると、IMS がフィールドの未使用バイトを充てんする必要がある場合、IMS は値を右寄せし、値の左側の未使用バイトを X'00' によって充てんします。例えば、5 バイト・フィールド内の 3 バイト値 X'54321C' は、X'000054321C' として書き出されます。

MAXBYTES ***max_array_bytes***

フィールド・インスタンスのバイト長が動的配列内のエレメント数に応じて異なる可能性がある場合に、フィールドの最大サイズをバイト単位で指定します。MAXBYTES と BYTES は同時には指定できません。

MAXBYTES の値は、このフィールドの下にネストされている、すべてのフィールドのバイト値の最大合計値以上でなければなりません。

MAXBYTES パラメーターが必須かつ有効であるのは、以下の場合のみです。

- フィールドが動的配列として定義されている場合。フィールドが動的配列であるのは、配列内のエレメント数がそのフィールドのインスタンスごとに異なる可能性がある場合です。動的配列の定義内で DEPENDSON パラメーターは、動的配列のインスタンスの配列エレメント数を定義する、セグメント定義内の別のフィールドを参照します。
- 動的配列として定義済みのネストされたフィールドを含む、静的配列または構造として定義されたフィールドの場合。

IN ***column_name***

このフィールドを含む構造または配列として定義されているフィールドの名前を指定します。参照されるフィールドは、DATATYPE=ARRAY または DATATYPE=STRUCT のいずれかを指定して定義されている必要があります。

INTERNAL TYPECONVERTER

IMS データをアプリケーション・プログラムが要求するデータ型に変換するために IMS が使用する、内部変換ルーチンを指定します。

INTERNAL TYPECONVERTER または **USER TYPECONVERTER** のいずれかを指定することができますが、両方を指定することはできません。 **INTERNAL TYPECONVERTER** と **USER TYPECONVERTER** は同時には指定できません。

INTERNAL TYPECONVERTER パラメーターの有効な値は、以下のとおりです。

ARRAY | BINARY | ...

DECIMAL データ・タイプの場合、デフォルトの **INTERNAL TYPECONVERTER** は符号付き PACKEDDECIMAL です。

DATE、TIME、または TIMESTAMP データ・タイプの場合、**INTERNAL TYPECONVERTER** キーワードで LONG または CHAR を指定するか、**USER TYPECONVERTER** を指定する必要があります。INTERNAL

TYPECONVERTER LONG がデフォルトです。LONG が使用されると、その値は 1970 年 1 月 1 日からのミリ秒数として DASD に保管されます。

XML データ・タイプの場合、デフォルトの **INTERNAL TYPECONVERTER** は XML_CLOB です。これは、XML に対して有効な唯一の値です。

STRUCT または ARRAY データ・タイプの場合、デフォルトの **INTERNAL TYPECONVERTER** はそれぞれ STRUCT または ARRAY です。これらは、それぞれ有効な唯一の値です。

その他のすべてのデータ・タイプの場合、その値がデフォルトの **INTERNAL TYPECONVERTER** として使用されます。

有効値は以下のとおりです。

ARRAY
ARRAY が指定された場合は、以下のようになります。

- **INTERNALNAME** パラメーターはサポートされません。
- **BYTES** または **MAXBYTES** パラメーターに指定するバイト値は、その配列に含まれるすべてのフィールドのバイト数の合計以上でなければなりません。

ARRAY として定義されたフィールド、または ARRAY を含むフィールドは再定義できません。

配列エレメントは、列の IN キーワードに配列の名前を指定します。

BIT
BIT を指定する場合、MAXBYTES 1 のみを指定することができます。

BYTE
BYTE を指定する場合、MAXBYTES 1 のみを指定することができます。

UBYTE
UBYTE を指定する場合、MAXBYTES 1 のみを指定することができます。

CLOB
文字ラージ・オブジェクトは、データベース管理システム内の文字データの集合です。

DOUBLE
DOUBLE を指定する場合、MAXBYTES 8 のみを指定することができます。

FLOAT
FLOAT を指定する場合、MAXBYTES 4 のみを指定することができます。

INT
INT を指定する場合、MAXBYTES 4 のみを指定することができます。

UINT
UINT を指定する場合、MAXBYTES 4 のみを指定することができます。

LONG
LONG を指定する場合、MAXBYTES 8 のみを指定することができます。

ULONG
ULONG を指定する場合、MAXBYTES 8 のみを指定することができます。

SHORT
SHORT を指定する場合、MAXBYTES 2 のみを指定することができます。

USHORT
USHORT を指定する場合、MAXBYTES 2 のみを指定することができます。

XML_CLOB

制約事項: **NAME** パラメーターが指定されている場合、データ・タイプ XML はサポートされません。

ZONEDDECIMAL
ZONEDDECIMAL は、IMS Universal JDBC ドライバーおよび IMS Universal DL/I ドライバーのデータ・タイプ拡張子です。データ・タイプ XML を指定する必要があります。

INTERNAL TYPECONVERTER パラメーターで指定する値は、列のデータ・タイプとして指定する値と整合している必要があります。ほとんどの場合、**INTERNAL TYPECONVERTER** には、データ・タイプとして指定した値と同じ値を指定する必要があります。

ISSIGNEDYES | ISSIGNEDNO
このパラメーターは、DECIMAL データ・タイプにのみ有効です。デフォルトは ISSIGNEDYES です。

MINOCCURS ***min_array_elements***
DECIMAL データ・タイプの場合、デフォルトの INTERNAL TYPECONVERTER は符号付き PACKEDDECIMAL です。

MAXOCCURS ***max_array_elements***
ARRAY の場合のみに、ARRAY 内のエレメントの最大数を指定する必須の数値。MAXOCCURS は必ず MINOCCURS 以上とし、ゼロであってはなりません。

OVERFLOW ***table_name***
XML 文書の保持用に定義された列に収まらない XML 文書の一部の保管に使用できる、従属表の 1 文字から 8 文字の内部名。従属表の親は、XML データ列を含む表です。 OVERFLOW は、XML を指定した列のみに適用されます。

PATTERN
日付、時刻、およびタイム・スタンプの Java データ・タイプ用のパターンを指定する、単一引用符で囲まれた 1 文字から 50 文字のオプション・フィールド。

PATTERN は、データ・タイプとして DATE、TIME、または TIMESTAMP が指定され、INTERNAL TYPECONVERTER キーワードで CHAR が指定されている場合にのみ適用されます。 PATTERN はこれ以外のデータ型では無効です。

パターンは大/小文字が区別され、必ず単一引用符で囲む必要があります。

キーワード値の区切り文字として使用される単一引用符を除き、PATTERN キーワードに指定する値に以下の文字を含めることはできません。
- 単一引用符および二重引用符

- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

指定できるパターンは、Java クラス java.text.SimpleDateFormat によって定義されます。DDL では、PATTERN に入力した値が Java で定義されたパターンに従っているかどうかはチェックされません。

例えば、yyyy.MM.dd という Java 形式を入力すると、結果として得られる時刻形式は「2013.01.01」になります。

PROPERTIES *name value*

USER TYPECONVERTER パラメーターに指定されたユーザー・タイプ・コンバーターに対してプロパティーを指定します。これらのプロパティーが、ユーザー・タイプ・コンバーターに渡されます。

PROPERTIES パラメーターは、USER TYPECONVERTER が指定されている場合にのみ有効です。

PROPERTIES キーワードに指定される名前とプロパティーは、大/小文字を区別します。また、単一引用符で囲む必要があります。

PROPERTIES キーワードでは以下の文字はサポートされません。

- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

プロパティー名の最大長は 128 文字です。プロパティー値の最大長も 128 文字です。

形式は次のとおりです。

```
PROPERTIES ('name1' = 'value1' , 'name2' = 'value2')
```

例えば、次のとおりです。

```
PROPERTIES ('DOG' = 'BUTCH' , 'CAT' = 'LUCY')
```

RELSTART *relative_start_position*

配列 (または状況によっては構造) のエレメントとして定義されているフィールドの開始位置を指定します。有効な値は 1 から 32767 です。

RELSTART に指定する値は、配列または構造の開始位置に相対させた、そのフィールドの開始バイト・オフセットです。例えば、配列内の最初のフィールドは、そのフィールドを含む配列がセグメントのバイト 50 で開始する場合でも、通常は RELSTART 1 を指定します。

配列フィールドを親として指定するフィールドでは、RELSTART は必須です。

構造を親として指定するフィールドでは、その構造フィールドが RELSTART または STARTAFTER を使用して定義されている場合、RELSTART は必須です。

次の例では、フィールド DYNARRAY は動的配列です。フィールド STRUCT01 は構造です。フィールド FLD03 および FLD04 は両方とも STRUCT01 を親として指定しています。セグメント内で動的配列が STRUCT01 の前に配置されているため、FLD03 と FLD04 の開始オフセットは STRUCT01 の開始位置との相対によってのみ指定できます。

```
FIELD EXTERNALNAME=ARRAYNUM,DATATYPE=DECIMAL(7,0),START=1,BYTES=4
FIELD EXTERNALNAME=DYNARRAY,DATATYPE=ARRAY,START=5,MAXBYTES=100
      MINOCCURS=10,MAXOCCURS=50,DEPENDSON=ARRAYNUM
FIELD EXTERNALNAME=FLD01,RELSTART=1,BYTES=2,PARENT=DYNARRAY
FIELD EXTERNALNAME=FLD02,STARTAFTER=DYNARRAY,BYTES=10
FIELD EXTERNALNAME=STRUCT01,DATATYPE=STRUCT,STARTAFTER=FLD02,BYTES=10
FIELD EXTERNALNAME=FLD03,RELSTART=1,BYTES=5,PARENT=STRUCT01
FIELD EXTERNALNAME=FLD04,RELSTART=6,BYTES=5,PARENT=STRUCT01
```

START、STARTAFTER、および RELSTART は同時には指定できません。

START *start_position*

定義するフィールドの開始位置をセグメントの先頭からの相対的なバイトで指定します。 START の値は、値が 32767 を超えない数値項でなければなりません。セグメントの最初のバイトの開始位置は 1 です。可変長セグメントの場合、最初の 2 バイトはセグメントの長さを含みます。したがって、実際、ユーザー・データ・フィールドが始まるのは 3 バイトからです。フィールドをオーバーラップすることが許されています。論理子セグメントを定義する場合、セグメント・タイプの最初の *n* バイトは、論理親または物理親の連結キーです。位置 1 から開始するフィールドは、そのフィールドの全部または一部を定義します。位置 n+1 から開始するフィールドは、交差データで始まります。

START をシステム関連フィールドに使用すると、連結キーの一部を索引ソース・セグメント・タイプ内のフィールドとして記述できます。この方法で使用すると、START は、連結キーの関連部分の開始位置を、連結キーの先頭から相対的に指定します。この場合、連結キーの最初のバイトは、位置 1 であると見なされます。これは、値が、連結キーの長さ + 1 を超えない数値項でなければなりません。**BYTES** パラメーターに指定された値を引いてください。/SX システム関連フィールドの開始位置パラメーターは無視されます。

START、STARTAFTER、および RELSTART は同時には指定できません。

XML の場合、START パラメーターはオプションであり、START 0 の指定が可能です。XML の場合に START パラメーターを省略すると、START 0 がデフォルトになります。

STARTAFTER *field_name*

フィールドが動的配列の後に開始するために、そのフィールドの開始バイト・オフセットを計算できない場合に、セグメント内でこのフィールドの直前に配置されるフィールドの名前を指定します。この名前は、INTERNALNAME キーワードで指定した名前にすることはできません。

STARTAFTER が必須かつ有効なのは、動的配列として定義されたフィールドが前のオフセットでそのフィールドの前に配置されているために、そのフィールドの開始位置を計算できない場合のみです。

動的配列があると、セグメント内の後続のフィールドの開始オフセットの計算が不可能になります。これは、動的配列のバイト長がセグメントのインスタンスごとに異なる場合があるためです。動的配列フィールドの列は、DEPENDSON および MAXBYTES パラメーターを含めることによって識別できます。

STARTAFTER パラメーターは、配列フィールドを親として定義するフィールドでは指定できません。 RELSTART パラメーターを代わりに指定してください。

START、STARTAFTER、および RELSTART は同時には指定できません。

URL *xml_schema_url*
このフィールドを記述する XML スキーマを参照する URL の、単一引用符で囲まれた 1 文字から 256 文字のオプション・フィールド。

以下に例を示します。

```
URL 'MySchema.xsd'
```

URL キーワードに指定する値には、以下の文字を含めることはできません。
- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

URL パラメーターは、XML_CLOB データに対して XML が指定されている場合にのみ適用されます。

USER TYPECONVERTER *typeconverter*
型変換に使用されるユーザー提供 Java クラスの、単一引用符で囲まれた 1 文字から 256 文字の完全修飾 Java クラス名を指定します。

例えば、次のとおりです。

```
USER TYPECONVERTER 'class://com.ibm.ims.dli.types.PackedDateConverter'
```

USER TYPECONVERTER キーワードに指定する値には、以下の文字を含めることはできません。
- 単一引用符および二重引用符
- ブランク
- より小 (<) およびより大 (>) 記号
- アンパーサンド (&)

USER TYPECONVERTER は、INTERNAL TYPECONVERTER と相互に排他的です。

CREATE TABLE (inline-constraints) のキーワード・パラメーター

CREATE TABLE (inline-constraints) ステートメントには、以下のキーワード・パラメーターが定義されています。

CONSTRAINT *constraint_name*
制約の名前を指定します。制約名を指定しないと、ユニーク制約名が生成されます。名前を指定する場合には、以前に表で指定した制約のいずれの名前とも異なっていなければなりません。

PRIMARY KEY NON UNIQUE
このフィールドが、セグメント・タイプ内のシーケンス・フィールドであることを表します。

NON UNIQUE
セグメント・タイプのオカレンスのシーケンス・フィールドで重複値を使用できることを示すオプション・キーワード。ルート・セグメント・タイプでは、各オカレンスのシーケンス・フィールドは、HDAM の場合を除いて、固有値を含んでいる必要があります。HDAM データベースのルート・セグメント・タイプはキー・フィールドを必要としません。キー・フィールドが定義されても、それが固有である必要はありません。

指定されない場合、セグメント・タイプのオカレンスのシーケンス・フィールドでは固有値のみを使用できます。従属セグメント・タイプでは、ある物理親セグメントのもとにある各オカレンスのシーケンス・フィールドは、固有値を含んでいなければなりません。

CREATE TABLE (constraint) のキーワード・パラメーター

CREATE TABLE (constraint) ステートメントには、以下のキーワード・パラメーターが定義されています。

CONSTRAINT *constraint_name*
制約の名前を指定します。制約名を指定しないと、ユニーク制約名が生成されます。名前を指定する場合には、以前に表で指定した制約のいずれの名前とも異なっていなければなりません。

PRIMARY KEY(*column_name*) NON UNIQUE
このフィールドが、セグメント・タイプ内のシーケンス・フィールドであることを表します。

NON UNIQUE
セグメント・タイプのオカレンスのシーケンス・フィールドで重複値を使用できることを示すオプション・キーワード。ルート・セグメント・タイプでは、各オカレンスのシーケンス・フィールドは、HDAM の場合を除いて、固有値を含んでいる必要があります。HDAM データベースのルート・セグメント・タイプはキー・フィールドを必要としません。キー・フィールドが定義されても、それが固有である必要はありません。

指定されない場合、セグメント・タイプのオカレンスのシーケンス・フィールドでは固有値のみを使用できます。従属セグメント・タイプでは、ある物理親セグメントのもとにある各オカレンスのシーケンス・フィールドは、固有値を含んでいなければなりません。

CREATE TABLE (references-clause) のキーワード・パラメーター

CREATE TABLE (references-clause) ステートメントには、以下のキーワード・パラメーターが定義されています。

FOREIGN KEY REFERENCES
従属セグメント・タイプの場合、このセグメントの物理親の名前を指定します。

REFERENCES *table_name*
従属セグメントの親セグメントを指定するもので、IMS 外部表の名前です。

CREATE TABLE (map-definition) のキーワード・パラメーター

CREATE TABLE (map-definition) ステートメントには、以下のキーワード・パラメーターが定義されます。

MAP
マップ定義の前には列定義がなければなりません。MAP ステートメントを使用すると、表内の列の代替マッピングが可能になります。制御列に関連する 1 つ以上の CASE ステートメントのグループが表内でネストされます。制御列は、表インスタンスで使用されているケースを識別します。

column_name
セグメント・インスタンスに使用されるマップ・ケースを判別する値が含まれる、この表内の列の外部名。列にこのマップの CASE ステートメントの *caseid* 値に対応する値が含まれていない場合、このマップは、この表インスタンスに使用されていません。

AS *map_name*
このマップの名前を定義する、1 文字から 128 文字のオプションの英数字フィールド。指定されない場合、IMS は、この表内で固有の名前を自動的に生成します。 DFS 接頭部は、IMS によって予約されており、ユーザーが作成する名前の一部にすることはできません。

CREATE TABLE (case-definition) のキーワード・パラメーター

CREATE TABLE (case-definition) ステートメントには、以下のキーワード・パラメーターが定義されています。

CASE
CASE ステートメントは、マップ・ケースを定義します。マップ・ケースは、表内の特定のバイト範囲のオプションの代替フィールド・レイアウトを定義する列のセットです。

1 つのセグメント内の同じバイト範囲をマップするマップ・ケースは、MAP ステートメントによってグループ化されます。また MAP ステートメントは、マップ・ケースを、表定義で別個に定義された制御フィールドにリンクします。

各マップ・ケースには固有の ID があります。表のインスタンスでは、有効なマップ・ケースの ID が、セグメント作成時に制御フィールドに保管されます。

IMS Universal ドライバーが使用されていない限り、マップ・ケースによって定義されたフィールド・レイアウトを、このバイト範囲にアクセスするアプリケーション・プログラムに対し、COBOL コピーブックまたはその他のプログラミング成果物を使用して定義する必要があります。表インスタンスへのアクセス時に、アプリケーション・プログラムは制御フィールドの値を確認して、どのコピーブックを使用するかを決定します。

アプリケーション・プログラムが IMS Universal ドライバーを使用して IMS にアクセスする場合は、アプリケーション・プログラムに対してフィールド・レイアウトを定義するための追加のプログラミング成果物は必要ありません。

caseid
固有の文字または16 進ストリングを定義する 1 バイトから 128 バイトのフィールド。セグメントのフィールド構造の一部または全部がこのケースによってマップされている場合、表インスタンスはユーザー定義制御フィールドにこの *caseid* 値を指定します。

この値を文字ストリングとして指定する場合は、単一引用符内に指定する必要があります (例えば、`'name01'`)。この値を 16 進ストリングとして指定する場合は、単一引用符内に指定し、その後に 16 進インディケーターを指定する必要があります (例えば、`'00000001'x`)。

caseid 値には、英数字、下線 (_)、@、$、および # を含めることができます。あるいは、16 進ストリングで指定することができます。値の長さは、

ユーザー定義の制御フィールドの長さでサポートされているものでなければなりません。英数字の場合、値の長さは、必ず制御フィールドの BYTES パラメーターに指定された値以下とします。 16 進ストリングの場合、CASEID 値の長さは、必ず制御フィールドの BYTES パラメーターに指定された値の正確に 2 倍とします。

ケース ID は、そのケースの所属先マップ内で固有でなければなりません。

AS *case_name*
このケースの名前を定義する、1 文字から 128 文字のオプションの英数字フィールド。ケース名は表内で固有でなければなりません。指定されない場合、IMS は、この表内で固有の名前を自動的に生成します。 DFS 接頭部は、IMS によって予約されており、ユーザーが作成する名前の一部にすることはできません。

CREATE TABLE (lchild) のキーワード・パラメーター

CREATE TABLE (lchild 構文) ステートメントには、以下のキーワード・パラメーターが定義されています。

lchild_definition
前に column-definition がなければなりません。

database_name.table_name
table_name パラメーターは、論理子の IMS 内部名、索引ポインター、索引ターゲット、HIDAM または PHIDAM ルート・セグメント・タイプ (先行 TABLE によって定義したセグメント・タイプに関連付けられる) を指定します。 **database_name** パラメーターは、**table_name** パラメーターで指定されたセグメント・タイプを含むデータベースの名前です。このデータベースで **table_name** パラメーターが定義されている場合は、**database_name** パラメーターを省略することができます。

database_name パラメーターは、1 文字から 8 文字の英数字値にする必要があります。 **table_name** パラメーターは、1 文字から 128 文字の英数字値にする必要があります。

SINGLE|DOUBLE|NONE|INDEX|SYMBOL
論理関係または索引関係で使用するポインターを指定します。索引データベース生成で省略された場合、SINGLE がデフォルトです。索引ターゲット・セグメント・タイプに続く LCHILD ステートメントには、INDEX または SYMBOL を指定する必要があります。索引関係のこの部分にはデフォルトは用意されていません。単一方向論理関係または物理対の双方向論理関係を設定する LCHILD ステートメントで省略された場合、NONE がデフォルトです。仮想対の双方向論理関係を設定する LCHILD ステートメントで省略されたか NONE として指定された場合、SINGLE がデフォルトです。

制約事項:

- PHDAM および PHIDAM データベースでは、オペランド INDEX と NONE のみがサポートされています。その他のオペランドは、エラーとして処理されます。
- DEDB 副次索引データベースの場合は、SYMBOL オペランドのみがサポートされます。

SINGLE
論理関係、または直接アドレス・ポインターが作成する索引関係に使用します。先行 TABLE で定義されたセグメント・タイプの各オカレンスに、論理子先頭ポインター・フィールドが確保されることを指定します。先行 TABLE で論理親を定義している場合には、このポインター・フィールドに論理子セグメント・タイプの最初のオカレンスを指す直接アドレス・ポインターが入ります。先行 TABLE で HIDAM 1 次索引データベース・セグメント・タイプを定義している場合には、このポインター・フィールドに、HIDAM データベース・ルート・セグメントを指す直接アドレス・ポインターが入ります。先行 TABLE で副次索引データベース内の索引ポインター・セグメント・タイプを定義している場合には、このポインター・フィールドに、索引ターゲット・セグメントを指す直接アドレス・ポインターが入ります。

DOUBLE
論理親セグメントに予約する 2 個の 4 バイト・ポインター・フィールド (論理子先頭および論理子最終) を指定するのに使用します。この 2 つのポインターは、論理親のもとにある論理子セグメント・タイプの最初と最後のオカレンスを指します。論理子最終ポインターは、論理子が順序付けられておらず、rules パラメーターが LAST の場合に役立ちます。

NONE
論理親から論理子セグメントへの論理関係をインプリメントしないか、あるいは論理子直接アドレス・ポインターでインプリメントしない場合、これを使用します。この場合は、論理親から論理子への関係が存在しないか、あるいは物理対のセグメントを使用して関係が維持されます。論理親セグメントにポインター・フィールドは予約されません。

INDEX
HIDAM データベースの LCHILD ステートメントに指定します。これは HIDAM データベースの DBD 生成の過程で、HIDAM ルート・セグメント・タイプと HIDAM 1 次索引の間の索引関係を確立するのに使用します。索引ターゲット・セグメント・タイプと副次索引の間の索引関係を確立するターゲット・データベースについての DBD の LCHILD ステートメントにも、INDX を指定できます。これらの場合には、1 次索引 DBD または副次索引 DBD の LCHILD ステートメントでパラメーターを省略するか、SINGLE を指定してください。 HIDAM 1 次索引の LCHILD ステートメントは、副次索引の LCHILD ステートメントの前に置く必要があります。

要件**:** ターゲット・データベースが HALDB の場合、DBD ステートメントの ACCESS パラメーターで PSINDEX パラメーターを使用して、索引データベースを HALDB 索引として定義する必要があります。

SYMBOL
副次索引のターゲット・データベースの DBD 生成に使用します。これは、索引ターゲット・セグメントの連結キーを、直接ポインターではなく索引ポインター・セグメントに入れることを指定します。索引ターゲット・セグメント・タイプが HISAM データベースにある場合には、SYMBOL を指定する必要があります。索引ターゲット・セグメントが HDAM または HIDAM データベースにある場合は、SYMBOL はオプションです。

さらに、INDEX DBD で SYMBOL パラメーターを使用すると、索引ポインター・セグメントの接頭部の中に 4 バイトの直接アドレス索引ターゲット・セグメント・ポインター (これは、索引ポインターがシンボリックのときには使用されない) 用のスペースが予約されません。

PAIR
双方向論理関係の場合にのみ指定します。指定された名前は、LCHILD ステートメントで指定された論理子セグメント/表と物理的または仮想的に対にされている論理子セグメント/表です。名前は、1 文字から 8 文字の英数字値にする必要があります。

制約事項: PHDAM および PHIDAM データベースでは物理対しかサポートしないため、これらのデータベースを使用するときには仮想対に対してこのパラメーターを使用できません。

INDEXFIELD
索引 DBD 生成の場合にのみ、LCHILD ステートメントに指定します。これは、HIDAM データベースの 1 次索引の DBD 生成時に指定した HIDAM ルート・セグメント・タイプのシーケンス・フィールドの名前、または副次索引データベースの DBD 生成時に指定した索引ターゲット・セグメント・タイプの索引フィールド (XDFLD ステートメントにより定義) の名前を指定します。このパラメーターは、PHIDAM データベースの 1 次索引には必要ありません。

RKSIZE
ターゲット・データベースのルート・キー・サイズを指定します。このパラメーターは、区分副次索引 (PSINDEX) データベースのみに使用されるものであり、他のタイプのデータベースでは無効です。 (DBD ソースでは必須、DDL ではオプション)。

FIRST|LAST|HERE
仮想論理子にシーケンス・フィールドが定義されていないか、固有でないシーケンス・フィールドが定義されている場合、論理関係に使用します。これらの条件下では、FIRST、LAST、または HERE の規則により、順序が制御されます。すなわち、論理関係にある実論理子のオカレンスは、論理子ポインターおよび論理兄弟ポインターにより、論理親から順序付けられます (これにより、論理兄弟順序が確立されます)。

制約事項: PHDAM および PHIDAM データベースでは物理対しかサポートしないため、これらのデータベースを使用するときには仮想対に対してこのパラメーターを使用できません。

FIRST
論理子にシーケンス・フィールドが指定されていない場合に、新しいオカレンスが論理子の既存の最初のオカレンスの前に挿入されることを示します。固有でないシーケンス・フィールドが論理子に指定されている場合、新しいオカレンスは、同じキーを持つ既存のすべてのオカレンスの前に挿入されます。

LAST
論理子にシーケンス・フィールドが指定されていない場合に、新しいオカレンスが論理子の既存の最後のオカレンスの後ろに挿入されることを示します。固有でないシーケンス・フィールドが論理子に指定されている場合に

は、新しいオカレンスは、同じキーを持つ既存のすべてのオカレンスの後ろに挿入されます。 LAST はデフォルト・オプションです。

HERE
挿入は、直前の DL/I 呼び出しで確立された位置によって異なることを示します。シーケンス・フィールドが定義されていない場合には、セグメントは、直前の呼び出しで位置が確立された論理兄弟の前に挿入されます。直前の呼び出しで位置が確立されていない場合には、新しい兄弟は、既存のすべての論理兄弟の前に挿入されます。固有でないシーケンス・フィールドが定義されている場合は、同じシーケンス・フィールド値を持つ直前の呼び出しで位置が確立された論理兄弟の前にセグメントが挿入されます。同じシーケンス・フィールド値を持っている論理兄弟に位置が確立されていなければ、同じシーケンス・フィールド値を持つすべての兄弟の前にセグメントが挿入されます。

論理子の新しいオカレンスが、その物理親から挿入される場合には、論理兄弟チェーン上の論理子には前の位置は存在しません。したがって、シーケンス・フィールドが定義されていなければ、新しいオカレンスは、論理兄弟チェーン上の既存のすべてのオカレンスの前に置かれるか、あるいは、固有でないシーケンス・フィールドが定義されている場合には、同じシーケンス・フィールド値を持つ既存のすべてのオカレンスの前に置かれます。

論理パスに対して出される挿入呼び出しでは、指定した挿入規則よりコマンド・コードの L (LAST) が優先されるため、新しいオカレンスは LAST の挿入規則に従って挿入されます。

CREATE TABLE (xdfld) のキーワード・パラメーター

CREATE TABLE (xdfld 構文) ステートメントには、以下のキーワード・パラメーターが定義されています。

xdfld_definition
前に lchild-definition がなければなりません。

xdfld_name
索引ターゲット・セグメントの索引データ・フィールドの名前を指定します。指定した名前は、索引ポインター・セグメント・タイプの検索フィールドを、索引ターゲット・セグメント・タイプ内のフィールドとして実際に表します。 指定した名前を使用すると、索引ポインター・セグメントの検索フィールド・キーで索引ターゲット・セグメント・タイプを呼び出す SSA を修飾できます。これにより、副次索引内のデータに基づいた 1 次または 2 次の処理シーケンスで、索引ターゲット・セグメント・タイプのオカレンスにアクセスできます。
fldname は、1 文字から 26 文字の英数字値にする必要があります。

指定した名前は、副次索引の内容に基づいて索引ターゲット・セグメントのオカレンスにアクセスするのに使用されるため、その名前は、索引ターゲット・セグメント・タイプに指定されるすべてのフィールド名の間で固有でなければなりません。

INTERNALNAME
オプションの IMS 内部名。1 文字から 8 文字の英数字値にする必要があります。

SEGMENT
この副次索引関係に索引ソース・セグメント・タイプを指定します。索引ターゲット・セグメント・タイプより階層の下にある、その後で定義されるセグメント・タイプの名前でなければなりません。あるいは、索引ターゲット・セグメント・タイプ自身の名前にすることができます。指定するセグメント名が、論理子セグメントであってはなりません。このパラメーターを省略すると、索引ターゲット・セグメント・タイプは索引ソース・セグメントと見なされます。

CONST
特定の副次索引の索引ポインター・セグメントのすべてを示すために使用する 1 つの文字を指定します。このパラメーターはオプションです。このパラメーターの目的は、複数の副次索引が同じ副次索引データベース内に存在している場合に、それぞれの副次索引に関連したすべての索引ポインター・セグメントを識別することです。 1 バイトの16 進ストリングの用語 (X'F9' など) として指定する必要があります。

制約事項: CONST は HALDB または DEDB データベースに対してサポートされません。

SRCH
副次索引の検索フィールドに使用する、索引ソース・セグメントのフィールドを指定します。 list1 は、列定義で定義された、索引ソース・セグメント・タイプ内の 1 個から 5 個のフィールド名のリストでなければなりません。複数の名前が含まれている場合には、それらをコンマで区切って、括弧で囲む必要があります。リスト内の名前の順序は、索引ポインター・セグメントの検索フィールド内で連結されているフィールド値の順序です。関係しているフィールドの長さの合計は、索引ターゲット・セグメントの索引付きフィールドの長さになります (これは、セグメント検索引数に反映されていなければなりません)。

SUBSEQ
副次索引のサブシーケンス・フィールドとして使用する索引ソース・セグメントのフィールド (ある場合) を指定します。 list2 は、列定義で定義された、索引ソース・セグメント内の 1 個から 5 個のフィールド名のリストでなければなりません。複数の名前が含まれている場合には、それらをコンマで区切って、括弧で囲む必要があります。リスト内の名前の順序は、索引ポインター・セグメントのサブシーケンス・フィールド内で連結されているフィールド値の順序です。このパラメーターはオプションです。

DDATA
副次索引の重複データ・フィールドに使用する索引ソース・セグメントのフィールド (ある場合) を指定します。 list3 は、列定義で定義された、索引ソース・セグメント内の 1 個から 5 個のフィールド名のリストでなければなりません。複数の名前が含まれている場合には、それらをコンマで区切って、括弧で囲む必要があります。リスト内の名前の順序は、索引ポインター・セグメントの重複データ・フィールド内で連結されているフィールド値の順序です。このパラメーターはオプションです。

NULLVAL
索引ポインター・セグメントの検索フィールドで使用する索引ソース・セグメントのデータに、指定の値が入っているときに、索引ポインター・セグメントが作成されないようにします。

この値は、1 バイトの 16 進ストリングの用語である必要があります。例えば、X'10'、X'40' (ブランクの場合)、X'00' (ゼロの場合) などです。パック 10 進値が必要な場合には、有効な数値およびゾーン数字または符号数字を持つ 16 進項 (パックの正の 3 の場合 X'3F' または負の 9 の場合 X'9D') として指定する必要があります。

SRCH パラメーターに指定された索引ソース・セグメントの各フィールドが、すべてのバイトにこのパラメーターの値を持っている場合は、索引付けは実行されません。例えば、NULLVAL X'F9' が指定されていると、関連する索引には、値 C'9999...9' で索引付けされる項目はありません。

パック 10 進数フィールドの場合は、多少異なります。パック 10 進数フィールドでは、検索フィールドを構成する各フィールドは、別々のパック値であると見なされます。例えば、検索フィールドが 3 つの 2 バイト・パック・ソース・フィールドで構成されている場合に NULLVAL X'9F' が指定されているとすると、X'9F' を含む索引項目はすべて抑制されるため、検索フィールド値 X'999F999F999F' を持つ索引項目は存在しないことになります。

さらに、同じ NULLVAL X'9F' であっても、検索フィールドが 6 バイト・フィールド 1 つで、値が X'99999999999F' であるときには、索引付けは行われません。

検査される符号の形式は、指定された形式のみです。例えば、X'9C' を指定すると、X'9F' 抑制は行われません。 NULLVAL と EXTRTN の両方のオペランドを指定するときには、そのいずれでも抑制されない限り、セグメントの索引付けが実行されます。

EXTRTN
選択した索引ポインター・セグメントの作成を抑制するときに使用する、ユーザー提供の索引保守出口ルーチンの名前を指定します。パラメーター (name1) には、索引付きデータベースで行われる変更のために、DL/I が索引項目の挿入、削除、または置換を行おうとするときにいつも制御を受け取るユーザー提供のルーチンの名前を指示する必要があります。この出口ルーチンは、影響を受ける索引ソース・セグメントを検査し、索引ポインター・セグメントを生成すべきかどうかを判別できます。NULLVAL と EXTRTN の両方のオペランドを指定するときには、そのいずれでも抑制されない限り、セグメントの索引付けが実行されます。

使用上の注意

このステートメントは、IMS DBD 生成ユーティリティーの AREA ステートメントと同等です。

例: COGDBD

```
DBD   NAME=COGDBD,                                          C
            ENCODING=Cp1047,                                  C
            ACCESS=(HDAM,OSAM),                               C
            RMNAME=(DFSHDC40,3,3,25),                         C
            PASSWD=NO
      DATASET  DD1=COGDATA,                                   C
            DEVICE=3390,                                      C
            SIZE=(8192)
      SEGM  NAME=ROOT,                                        C
            PARENT=0,                                         C
            BYTES=(20),                                       C
```

```
|                         RULES=(LLL,HERE)
|               FIELD NAME=(ROOTKEY,SEQ,U),                                  C
|                         BYTES=12,                                          C
|                         START=1,                                           C
|                         TYPE=C,                                            C
|                         DATATYPE=CHAR
|               FIELD NAME=TABTYPE,                                          C
|                         BYTES=8,                                           C
|                         START=13,                                          C
|                         TYPE=C,                                            C
|                         DATATYPE=CHAR
|               SEGM      NAME=TSINT,                                        C
|                         PARENT=ROOT,                                       C
|                         BYTES=(8,6),                                       C
|                         REMARKS='This describes table TSINT.',             C
|                         RULES=(LLL,HERE)
|               FIELD     NAME=RNUM,                                         C
|                         BYTES=4,                                           C
|                         START=3,                                           C
|                         DATATYPE=INT
|               FIELD     NAME=LL,                                           C
|                         BYTES=2,                                           C
|                         START=1,                                           C
|                         DATATYPE=SHORT
|               FIELD     NAME=CSINT,                                        C
|                         EXTERNALNAME=CSINT,                                C
|                         BYTES=2,                                           C
|                         START=7,                                           C
|                         DATATYPE=SHORT
|               SEGM      NAME=TINT,                                         C
|                         EXTERNALNAME=TESTINTEGER,                          C
|                         PARENT=ROOT,                                       C
|                         BYTES=(10,6),                                      C
|                         REMARKS='This describes table TINT.',              C
|                         RULES=(LLL,HERE)
|               FIELD     NAME=RNUM,                                         C
|                         BYTES=4,                                           C
|                         START=3,                                           C
|                         DATATYPE=INT
|               FIELD     NAME=LL,                                           C
|                         BYTES=2,                                           C
|                         START=1,                                           C
|                         DATATYPE=SHORT
|               FIELD     NAME=CINT,                                         C
|                         EXTERNALNAME=CINTEGER,                             C
|                         BYTES=4,                                           C
|                         START=7,                                           C
|                         DATATYPE=INT
|
|           CREATE DATABASE COGDBD
|           ACCESS HDAM OSAM
|           RMNAME(DFSHDC40 RMANCH 3 RMRBN 3 RMBYTES 25)
|           CCSID 'Cp1047';
|
|           CREATE TABLESPACE COGDATA
|             SIZE PRIMARY 8192;
|
|           CREATE TABLE TEST_ROOT (
|               ROOT_KEY CHAR(12) INTERNALNAME ROOTKEY PRIMARY KEY ,
|               TABLE_TYPE CHAR(8) INTERNALNAME TABTYPE
|           ) IN COGDBD.COGDATA
|             INTERNALNAME ROOT
|             MAXBYTES 20
|             AMBIGUOUS INSERT HERE;
|
|           CREATE TABLE TEST_SHORT_INTEGER (
|               TABLE_LENGTH SHORT INTERNALNAME LL,
```

```
    R_NUMBER INT INTERNALNAME RNUM,
    C_SHORT_INTEGER SHORT INTERNALNAME CSINT,
    FOREIGN KEY REFERENCES TEST_ROOT
) IN COGDBD.COGDATA
  INTERNALNAME TSINT
  MAXBYTES 8
  MINBYTES 6
  AMBIGUOUS INSERT HERE;

COMMENT ON TABLE TEST_SHORT_INTEGER IN COGDBD IS 'This describes table TSINT.';

CREATE TABLE TESTINTEGER (
TABLE_LENGTH SHORT INTERNALNAME LL,
R_NUMBER INT INTERNALNAME RNUM,
CINTEGER INT INTERNALNAME CINT,
FOREIGN KEY REFERENCES TEST_ROOT
) IN COGDBD.COGDATA
  INTERNALNAME TINT
  MAXBYTES 10
  MINBYTES 6
  AMBIGUOUS INSERT HERE;

COMMENT ON TABLE TESTINTEGER IN COGDBD IS 'This describes table TINT.';
```

例: DECIMAL を指定した COGDBD の続き

```
SEGM     NAME=TDEC,                                          C
               PARENT=ROOT,                                      C
               BYTES=(10,6),                                     C
               REMARKS='This describes table TDEC.',             C
               RULES=(LLL,HERE)
         FIELD    NAME=RNUM,                                     C
                  BYTES=4,                                       C
                  START=3,                                       C
                  DATATYPE=INT
         FIELD    NAME=LL,                                       C
                  BYTES=2,                                       C
                  START=1,                                       C
                  DATATYPE=SHORT
         FIELD    NAME=CDEC,                                     C
                  EXTERNALNAME=CDECIMAL,                         C
                  BYTES=4,                                       C
                  START=7,                                       C
                  DATATYPE=DECIMAL(7,2)

CREATE TABLE TEST_DECIMAL (
    TABLE_LENGTH SHORT INTERNALNAME LL,
    R_NUMBER INT INTERNALNAME RNUM,
    CDECIMAL DECIMAL(7,2) INTERNALNAME CDEC,
    FOREIGN KEY REFERENCES TEST_ROOT
) IN COGDBD.COGDATA
  INTERNALNAME TDEC
  MAXBYTES 10
  MINBYTES 6
  AMBIGUOUS INSERT HERE;

COMMENT ON TABLE TEST_DECIMAL IN COGDBD IS 'This describes table TDEC.';
```

例: DFSMARSH を指定した COGDBD の続き

```
SEGM     NAME=TNCHAR,                                        C
               PARENT=ROOT,                                      C
               BYTES=(38,6),                                     C
               REMARKS='This describes table TNCHAR.',           C
               RULES=(LLL,HERE)
         FIELD    NAME=RNUM,                                     C
                  BYTES=4,                                       C
```

```
                START=3,                                                C
                DATATYPE=INT
      FIELD     NAME=LL,                                                C
                BYTES=2,                                                C
                START=1,                                                C
                DATATYPE=SHORT
      FIELD     NAME=CNCHAR,                                            C
                EXTERNALNAME=CNCHAREXT,                                 C
                BYTES=32,                                               C
                START=7,                                                C
                DATATYPE=CHAR
   DFSMARSH     ENCODING=UTF-8,                                         C
                INTERNALTYPECONVERTER=CHAR

CREATE TABLE TEST_NEW_CHAR (
    TABLE_LENGTH SHORT INTERNALNAME LL,
    R_NUMBER INT INTERNALNAME RNUM,
    CNCHAREXT CHAR(32) INTERNALNAME CNCHAR CCSID 'UTF-8',
    FOREIGN KEY REFERENCES TEST_ROOT
) IN COGDBD.COGDATA
  INTERNALNAME TNCHAR
  MAXBYTES 38
  MINBYTES 6
  AMBIGUOUS INSERT HERE;

COMMENT ON TABLE TEST_NEW_CHAR IN COGDBD IS 'This describes table TNCHAR.';
```

例: PATTERN を指定した COGDBD の続き

```
SEGM     NAME=TTS,                                                  C
                PARENT=ROOT,                                            C
                BYTES=(35,6),                                           C
                REMARKS='This describes table TTS.',                    C
                RULES=(LLL,HERE)
      FIELD     NAME=RNUM,                                              C
                BYTES=4,                                                C
                START=3,                                                C
                DATATYPE=INT
      FIELD     NAME=LL,                                                C
                BYTES=2,                                                C
                START=1,                                                C
                DATATYPE=SHORT
      FIELD     NAME=CTS,                                               C
                EXTERNALNAME=CTSNAME,                                   C
                BYTES=29,                                               C
                START=7,                                                C
                DATATYPE=TIMESTAMP
      DFSMARSH ENCODING=Cp1047,                                         C
                INTERNALTYPECONVERTER=CHAR,                             C
                PATTERN='yyyy-MM-dd HH:mm:ss.fffffffff'

CREATE TABLE TEST_TIMESTAMP (
    TABLE_LENGTH SHORT INTERNALNAME LL,
    R_NUMBER INT INTERNALNAME RNUM,
    CTSNAME TIMESTAMP INTERNALNAME CTS CCSID 'Cp1047'
        PATTERN 'yyyy-MM-dd HH:mm:ss.fffffffff',
    FOREIGN KEY REFERENCES TEST_ROOT
) IN DATABASE COGDBD
  INTERNALNAME TTS
  MAXBYTES 35
  MINBYTES 6
  AMBIGUOUS INSERT HERE;

COMMENT ON TABLE TEST_TIMESTAMP IN COGDBD IS 'This describes table TTS.';
```

例: 配列

```
SEGM    NAME=HOSPITAL,
                EXTERNALNAME=HOSPITAL,
                ENCODING=Cp1047,
                PARENT=0,
                BYTES=(900),
                RULES=(LLL,HERE)
...

      FIELD     EXTERNALNAME=TABLEARRAY,
                BYTES=14,
                START=224,
                MINOCCURS=1,
                MAXOCCURS=1,
                DATATYPE=ARRAY
      FIELD     EXTERNALNAME=TABLEARRAY1,
                BYTES=2,
                START=224,
                TYPE=X,
                PARENT=TABLEARRAY,
                DATATYPE=CHAR
      DFSMARSH INTERNALTYPECONVERTER=CHAR
      FIELD     EXTERNALNAME=TABLEARRAY2,
                BYTES=4,
                START=226,
                TYPE=X,
                PARENT=TABLEARRAY,
                DATATYPE=CHAR
      DFSMARSH INTERNALTYPECONVERTER=CHAR
      FIELD     EXTERNALNAME=TABLEARRAY3,
                BYTES=8,
                START=230,
                TYPE=X,
                PARENT=TABLEARRAY,
                DATATYPE=CHAR
      DFSMARSH INTERNALTYPECONVERTER=CHAR
...


CREATE TABLE hospital (
    ...
    tablearray ARRAY MAXBYTES 14 START 224 MINOCCURS 1 MAXOCCURS 1,
    tablearray1 CHAR(2) IN tablearray,
    tablearray2 CHAR(4) IN tablearray,
    tablearray3 CHAR(8) IN tablearray,
    ...
) IN DATABASE dedbjn21
  INTERNALNAME hospital
  MAXBYTES 900
  AMBIGUOUS INSERT HERE
```

例: 動的配列

```
FIELD EXTERNALNAME=ARRAYNUM,DATATYPE=DECIMAL(7,0),START=1,BYTES=4
FIELD EXTERNALNAME=DYNARRAY,DATATYPE=ARRAY,START=5,MAXBYTES=100
      MINOCCURS=10,MAXOCCURS=50,DEPENDSON=ARRAYNUM
FIELD EXTERNALNAME=FLD01,RELSTART=1,BYTES=2,PARENT=DYNARRAY
FIELD EXTERNALNAME=FLD02,STARTAFTER=DYNARRAY,BYTES=10
FIELD EXTERNALNAME=STRUCT01,DATATYPE=STRUCT,STARTAFTER=FLD02,BYTES=10

CREATE TABLE dynamic_array_table (
    arraynum DECIMAL(7,0)
    dynarray ARRAY MAXBYTES 100 MINOCCURS 10 MAXOCCURS 50 DEPENDSON arraynum,
    fld01 SHORT RELSTART 1 IN dynarray,
```

```
    fld02 CHAR(10) STARTAFTER dynarray,
    struct01 STRUCT BYTES 10 STARTAFTER fld02
) IN DATABASE dynarrdb
  INTERNALNAME dynarrs;
```

例: マップ

```
DFSMAP   NAME=MAP1,                                              C
               DEPENDINGON=CASENUM
~~~~~~~~DFSCASE NAME=CASE1 redefines the schema for bytes 791 to 831.~~~~
      DFSCASE  NAME=CASE1,                                       C
               CASEID=CASEONE,                                   C
               CASEIDTYPE=C,                                     C
               MAPNAME=MAP1
      FIELD    EXTERNALNAME=FIELDB,                              C
               CASENAME=CASE1,                                   C
               BYTES=20,                                         C
               START=791,                                        C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=FIELDC,                              C
               CASENAME=CASE1,                                   C
               BYTES=20,                                         C
               START=811,                                        C
               DATATYPE=CHAR
~~~~~~~~DFSCASE NAME=CASE2 redefines the schema for bytes 831 to 855~~~~
      DFSCASE  NAME=CASE2,                                       C
               CASEID=CASETWO,                                   C
               CASEIDTYPE=C,                                     C
               MAPNAME=MAP1
      FIELD    EXTERNALNAME=FIELDD,                              C
               CASENAME=CASE2,                                   C
               BYTES=20,                                         C
               START=831,                                        C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=CARDTYPE,                            C
               BYTES=4,                                          C
               START=851,                                        C
               DATATYPE=CHAR

CREATE TABLE customer (
    id INT PRIMARY KEY,
    ...
    casenum CHAR(12),
    cardtype CHAR(4),
    ...
    MAP casenum AS MAP1 (
      CASE caseone AS case1 (
        fieldb CHAR(20) START 792 ,
        fieldc CHAR(20) START 811
      ) ,
      CASE casetwo AS case2 (
        fieldd CHAR(20) START 831,
        cardtype CHAR(4) START 851)
      )
    )
) IN DATABASE dedbjn21
  INTERNALNAME customer
  MAXBYTES 900
```

例: 構造

```
FIELD NAME=(NAME),                                               C
               EXTERNALNAME=PAYEE_NAME,                          C
               BYTES=20,                                         C
               START=11,                                         C
               DATATYPE=CHAR
      DFSMARSH ENCODING=Cp1047,                                  C
```

```
                    INTERNALTYPECONVERTER=CHAR

          FIELD EXTERNALNAME=INDIVIDUALNAME,                              C
                    BYTES=20,                                             C
                    START=11,                                             C
                    DATATYPE=STRUCT,                                      C
                    REDEFINES=PAYEE_NAME,                                 C
                    REMARKS='This is a STRUCT with lastname and firstname'

          FIELD EXTERNALNAME=LASTNAME,                                    C
                    BYTES=10,                                             C
                    START=11,                                             C
                    DATATYPE=CHAR,                                        C
                    PARENT=INDIVIDUALNAME
          DFSMARSH ENCODING=Cp1047,                                       C
                    INTERNALTYPECONVERTER=CHAR

          FIELD EXTERNALNAME=FIRSTNAME,                                   C
                    BYTES=10,                                             C
                    START=21,                                             C
                    DATATYPE=CHAR,                                        C
                    PARENT=INDIVIDUALNAME
          DFSMARSH ENCODING=Cp1047,                                       C
                    INTERNALTYPECONVERTER=CHAR


CREATE TABLE customer (
    ...
    payee_name CHAR(20) START 11 CCSID 'Cp1047',
    individualname STRUCT BYTES 20 START 11,
    lastname CHAR(10) IN individualname,
    firstname CHAR(10 IN individualname
) IN DATABASE dedbjn21
  INTERNALNAME customer
  MAXBYTES 900

COMMENT ON COLUMN customer.individualname IN dedbjn21 IS 'This is
     a STRUCT with lastname and firstname'
```

例: 構造の続き

```
FIELD    EXTERNALNAME=PERSONAL_INFO,                                C
                    BYTES=184,                                            C
                    START=156,                                            C
                    DATATYPE=STRUCT
          DFSMARSH INTERNALTYPECONVERTER=STRUCT

          FIELD    EXTERNALNAME=ADDRESS,                                  C
                    BYTES=184,                                            C
                    START=156,                                            C
                    MINOCCURS=2,                                          C
                    MAXOCCURS=2,                                          C
                    PARENT=PERSONAL_INFO,                                 C
                    DATATYPE=ARRAY

          FIELD    EXTERNALNAME=NAME_TYPE,                                C
                    BYTES=1,                                              C
                    RELSTART=1,                                           C
                    PARENT=ADDRESS,                                       C
                    DATATYPE=CHAR

          FIELD    EXTERNALNAME=INDIVIDUAL_NAME,                          C
                    BYTES=20,                                             C
                    RELSTART=2,                                           C
                    PARENT=ADDRESS,                                       C
                    DATATYPE=STRUCT
```

```
|                         FIELD    EXTERNALNAME=LAST_NAME,                          C
|                                  BYTES=12,                                        C
|                                  RELSTART=1,                                      C
|                                  PARENT=INDIVIDUAL_NAME,                          C
|                                  DATATYPE=CHAR
|
|                         FIELD    EXTERNALNAME=FIRST_NAME,                         C
|                                  BYTES=8,                                         C
|                                  RELSTART=13,                                     C
|                                  PARENT=INDIVIDUAL_NAME,                          C
|                                  DATATYPE=CHAR
|
|                         FIELD    EXTERNALNAME=ADDRESS_LINE2,                      C
|                                  BYTES=40,                                        C
|                                  RELSTART=22,                                     C
|                                  PARENT=ADDRESS,                                  C
|                                  DATATYPE=CHAR
|
|                         FIELD    EXTERNALNAME=CITY,                               C
|                                  BYTES=20,                                        C
|                                  RELSTART=62,                                     C
|                                  PARENT=ADDRESS,                                  C
|                                  DATATYPE=CHAR
|
|                         FIELD    EXTERNALNAME=STATE,                              C
|                                  BYTES=2,                                         C
|                                  RELSTART=82,                                     C
|                                  PARENT=ADDRESS,                                  C
|                                  DATATYPE=CHAR
|
|                         FIELD    EXTERNALNAME=ZIP,                                C
|                                  BYTES=9,                                         C
|                                  RELSTART=84,                                     C
|                                  PARENT=ADDRESS,                                  C
|                                  DATATYPE=CHAR
|                   ...
|                         FIELD    EXTERNALNAME=COMPANY_CARDS,                      C
|                                  BYTES=200,                                       C
|                                  START=584,                                       C
|                                  DATATYPE=STRUCT
|
|                         FIELD    EXTERNALNAME=CARDS,                              C
|                                  BYTES=200,                                       C
|                                  START=584,                                       C
|                                  MINOCCURS=5,                                     C
|                                  MAXOCCURS=5,                                     C
|                                  PARENT=COMPANY_CARDS,                            C
|                                  DATATYPE=ARRAY
|
|                         FIELD    EXTERNALNAME=COMPANY_NAME,                       C
|                                  BYTES=20,                                        C
|                                  RELSTART=1,                                      C
|                                  PARENT=CARDS,                                    C
|                                  DATATYPE=STRUCT
|
|                         FIELD    EXTERNALNAME=CO_TYPE,                            C
|                                  BYTES=12,                                        C
|                                  RELSTART=1,                                      C
|                                  PARENT=COMPANY_NAME,                             C
|                                  DATATYPE=CHAR
|
|                         FIELD    EXTERNALNAME=NEW_TYPE,                           C
|                                  BYTES=12,                                        C
|                                  RELSTART=1,                                      C
|                                  REDEFINES=CO_TYPE,                               C
|                                  PARENT=COMPANY_NAME,                             C
|                                  DATATYPE=CHAR
```

```
      FIELD    EXTERNALNAME=CO_NAME,                                  C
               BYTES=8,                                               C
               RELSTART=13,                                           C
               PARENT=COMPANY_NAME,                                   C
               DATATYPE=CHAR

      FIELD    EXTERNALNAME=EMPLOYEE_NAME,                            C
               BYTES=20,                                              C
               RELSTART=21,                                           C
               PARENT=CARDS,                                          C
               DATATYPE=STRUCT

      FIELD    EXTERNALNAME=EMPLOYEE_LAST_NAME,                       C
               BYTES=12,                                              C
               RELSTART=1,                                            C
               PARENT=EMPLOYEE_NAME,                                  C
               DATATYPE=CHAR

      FIELD    EXTERNALNAME=EMPLOYEE_MAIDEN_NAME,                     C
               BYTES=12,                                              C
               RELSTART=1,                                            C
               REDEFINES=EMPLOYEE_LAST_NAME,                          C
               PARENT=EMPLOYEE_NAME,                                  C
               DATATYPE=CHAR

      FIELD    EXTERNALNAME=EMPLOYEE_FIRST_NAME,                      C
               BYTES=8,                                               C
               RELSTART=13,                                           C
               PARENT=EMPLOYEE_NAME,                                  C
               DATATYPE=CHAR


CREATE TABLE employee (
    ...
    personal_info STRUCT BYTES 184 START 156,
    address ARRAY BYTES 184 START 156 MINOCCURS 2 MAXOCCURS 2 IN personal_info,
    name_type CHAR RELSTART 1 IN address,
    individual_name STRUCT BYTES 20 RELSTART 2 IN address,
    last_name CHAR(12) RELSTART 1 IN individual_name,
    first_name CHAR(8) RELSTART 13 IN individual_name,
    address_line2 CHAR(40) RELSTART 22 IN address,
    city CHAR(20) RELSTART 62 IN address,
    state CHAR(20) RELSTART 82 IN address,
    zip CHAR(9) RELSTART 84 IN address,
    company_cards STRUCT BYTES 200 START 584,
    cards ARRAY BYTES 200 START 584 MINOCCURS 5 MAXOCCURS 5 IN company_cards,
    company_name STRUCT BYTES 20 RELSTART 1 IN cards,
    co_type CHAR(12) RELSTART 1 IN company_name,
    new_type CHAR(12) RELSTART 1 IN company_name,
    co_name CHAR(8) RELSTART 13 IN company_name,
    employee_name STRUCT BYTES 20 RELSTART 21 IN cards,
    employee_last_name CHAR(12) RELSTART 1 IN employee_name,
    employee_maiden_name CHAR(12) RELSTART 1 IN employee_name,
    employee_first_name CHAR(8) RELSTART 13 IN employee_name
) IN dedbjn21
  INTERNALNAME employee
  MAXBYTES 900
```

例: 論理関係 EMPDB2

```
DBD    NAME=EMPDB2,ACCESS=(HDAM,OSAM),                          X
               RMNAME=(DFSHDC40,1,5,200)
        DATASET  DD1=DFSEMPL
        SEGM   NAME=EMPL,PARENT=0,BYTES=56
        LCHILD NAME=(SALESPER,AUTODB),PAIR=EMPSAL,POINTER=DBLE
        FIELD  NAME=(EMPNO,SEQ,U),BYTES=6,START=1,TYPE=C
```

```
         FIELD  NAME=LASTNME,BYTES=25,START=7,TYPE=C
         FIELD  NAME=FIRSTNME,BYTES=25,START=32,TYPE=C
         SEGM   NAME=EMPSAL,PARENT=EMPL,PTR=PAIRED,                    X
                SOURCE=((SALESPER,DATA,AUTODB))
         FIELD  NAME=(DLRNO,SEQ,U),BYTES=4,START=1,TYPE=C    (LPK)
         SEGM   NAME=EMPLINFO,PARENT=EMPL,BYTES=61
         FIELD  NAME=(STATE,SEQ,M),BYTES=2,START=51,TYPE=C
         FIELD  NAME=ADDRESS,BYTES=61,START=1,TYPE=C
         FIELD  NAME=STREET,BYTES=25,START=1,TYPE=C
         FIELD  NAME=CITY,BYTES=25,START=26,TYPE=C
         FIELD  NAME=ZIP,BYTES=9,START=53,TYPE=C
        DBDGEN
        FINISH
        END

CREATE DATABASE empdb2
  ACCESS HDAM OSAM
  RMNAME(DFSHDC40 RMANCH 1 RMRBN 5 RMBYTES 200);

CREATE TABLESPACE dfsempl
  IN empdb2;

CREATE TABLE employee (
    empno CHAR(6) INTERNALNAME empno PRIMARY KEY,
    lastnme CHAR(25) INTERNALNAME lastnme,
    firstnme CHAR(25) INTERNALNAME firstnme,
    LCHILD autodb.sales_person PAIR empyee_salary DOUBLE
) IN empdb2.DFSEMP1
  INTERNALNAME empl
  MAXBYTES 56;

CREATE TABLE employee_salary (
    dealer_number CHAR(4) INTERNALNAME dlrno PRIMARY KEY,
    FOREIGN KEY REFERENCES employee
) IN empdb2.DFSEMP1
  INTERNALNAME empsal
  SOURCE (autodb.sales_person);

CREATE TABLE employee_information (
    address CHAR(61) START 1 INTERNALNAME address,
    street CHAR(25) START 1 INTERNALNAME street,
    city CHAR(25) START 26 INTERNALNAME city,
    state CHAR(2) START 51 INTERNALNAME state PRIMARY KEY NON UNIQUE,
    zip CHAR(9) START 53 INTERNALNAME zip,
    FOREIGN KEY REFERENCES employee
) IN empdb2.dfsemp1
  INTERNALNAME emplinfo
  MAXBYTES 61;
```

例: 論理データベース

```
DBD    NAME=EMPLDB2,ACCESS=LOGICAL
       DATASET LOGICAL
       SEGM    NAME=EMPL,PARENT=0,SOURCE=((EMPL,,EMPDB2))
       SEGM    NAME=DEALER,PARENT=EMPL,                               X
               SOURCE=((EMPSAL,KEY,EMPDB2),(DEALER,DATA,AUTODB))
       SEGM    NAME=SALESINF,PARENT=DEALER,                           X
               SOURCE=((SALESINF,,AUTODB))
       SEGM    NAME=EMPLINFO,PARENT=EMPL,                             X
               SOURCE=((EMPLINFO,,EMPDB2))

CREATE DATABASE empldb2
 ACCESS LOGICAL;

CREATE TABLE empl
IN empldb2
SOURCE(empdb2.empl);
```

```
CREATE TABLE dealer (
    FOREIGN KEY REFERENCES empl
) IN empldb2
  SOURCE(empdb2.empsal KEY, autodb.dealer DATA);

CREATE TABLE salesinf (
    FOREIGN KEY REFERENCES empl
) IN empldb2
  SOURCE(autodb.salesinf);

CREATE TABLE emplinfo (
    FOREIGN KEY REFERENCES empl
) IN empldb2
  SOURCE(empdb2.emplinfo);
```

例: 副次索引データベース

```
DBD    NAME=SINDEX22,ACCESS=(INDEX,VSAM)
        DATASET  DD1=SINDX2P
        SEGM   NAME=SINDXB,PARENT=0,BYTES=34
        FIELD  NAME=(XFLDB,SEQ,U),BYTES=28,START=1,TYPE=C   SEARCH
        FIELD  NAME=COUNT,BYTES=2,START=25,TYPE=C           DUP DATA
        FIELD  NAME=ENQUIRS,BYTES=4,START=25,TYPE=P         USER DATA
        LCHILD NAME=(DEALER,AUTODB),INDEX=XFLD2
        DBDGEN
        FINISH
        END

CREATE DATABASE sindex22
  ACCESS INDEX;

CREATE TABLESPACE sindx2p
IN sindex22;

CREATE TABLE secondary_indexb (
    xfldb CHAR(28) INTERNALNAME xfldb PRIMARY KEY,
    count CHAR(2) INTERNALNAME count start(25),
    enquirs BINARY(4) START(25) INTERNALNAME enquirs,
    LCHILD autodb.dealer LCINDEX xfld2
) IN sindex22
  INTERNALNAME sindxb
  MAXBYTES 34;
```

例: データ・キャプチャー出口

```
********************************************************************
*        DBD DHVNTZ02 FROM CMVC (CDCI19-3.DBDGEN)                  *
********************************************************************
  DBD       NAME=DHVNTZ02,                                         C
               ACCESS=(HIDAM,VSAM),PASSWD=NO,VERSION=CDCTEST
********************************************************************
*        DATASET GROUP NUMBER 1
********************************************************************
DSG001 DATASET DD1=HIDAM,DEVICE=3300,SIZE=(2048),SCAN=3
********************************************************************
*        SEGMENT NUMBER 1                                          *
********************************************************************
    SEGM    NAME=K1,                                               C
               PARENT=0,BYTES=10,RULES=(LLL,LAST),PTR=(NOTWIN,,,,)     C
               EXIT=(*,LOG,PATH,KEY,DATA)
      FIELD NAME=(K1,SEQ,U),                                       C
               START=1,BYTES=5,TYPE=C
      FIELD NAME=(ID),                                             C
               START=6,BYTES=4,TYPE=C
       LCHILD  NAME=(INDEX,DXVNTZ02),                              C
               PTR=INDX,RUTLES=LAST,TYPE=C
```

```
. . . .
. . . .
*********************************************************************
*        SEGMENT NUMBER 5                                           *
*********************************************************************
    SEGM    NAME=K5,PARENT=((K1,SNGL)),                              C
               BYTES=10,RULES=(LLL,LAST),PTR=(TWIN,,,,)              C
               EXIT=(*,LOG,PATH,KEY,DATA)
      FIELD NAME=(K5,SEQ,U),                                        C
               START=1,BYTES=5,TYPE=C
      FIELD NAME=(ID),                                              C
               START=6,BYTES=4,TYPE=C
       LCHILD  NAME=(K3),                                           C
               PTR=SNGL,PAIR=K5X,RULES=LAST
       LCHILD  NAME=(K8),                                           C
               PTR=DBLE,PAIR=K5Y,RULES=HERE
. . . .
. . . .
    DBDGEN
    FINISH
    END
```

例: データ・キャプチャー出口

```
*********************************************************************
*        DBD DIVNTZ02 FROM CMVC (CDCI29-3 DBDGEN)                   *
*********************************************************************
 DBD NAME=DIVNTZ02,ACCESS=(HISAM,VSAM),VERSION=DIVNTZ02,             X
               EXIT=(COBXSQL,LOG,PATH,KEY,DATA,NOCASCADE)
*
DSG01 DATASET  DD1=DBHVSAM1,DEVICE=3330,OVFLW=DBHVSAM2,             X
               BLOCK=(00004,00002),RECORD=(00200,00200)
*
 SEGM          NAME=J1,                                             X
               PARENT=0,                                            X
               BYTES=10,                                            X
               FREQ=1,                                              X
               POINTER=NONE,                                        X
               RULES=(PPP,LAST)
 FIELD  NAME=(J1,SEQ,U),BYTES=005,START=00001,TYPE=C
 FIELD  NAME=ID,BYTES=4,START=6,TYPE=C
*******************************************************
* DLI Change Data Capture - Change delete rule to virtual
*******************************************************
 SEGM          NAME=J2,                                             X
               PARENT=((J1,SNGL),(J6,PHYSICAL,DIVNTZ02)),           X
               BYTES=37,                                            X
               FREQ=0000000001.00,                                  X
               POINTER=NONE,                                        X
               RULES=(PVP,LAST)
 FIELD  NAME=(J2,SEQ,U),BYTES=005,START=00016,TYPE=C
 FIELD  NAME=ID,BYTES=4,START=21,TYPE=C
 FIELD  NAME=(1J2),BYTES=004,START=00026,TYPE=C
 FIELD  NAME=(2J2),BYTES=004,START=00030,TYPE=C
 FIELD  NAME=(3J2),BYTES=004,START=00034,TYPE=C
. . . .
. . . .
 SEGM          NAME=J15,EXIT=NONE,                                  X
               PARENT=J12,                                          X
               POINTER=PAIRED,                                      X
               SOURCE=((K10,DATA,DHVNTZ02))
 FIELD  NAME=(J15,SEQ,U),BYTES=5,START=16,TYPE=C
 FIELD  NAME=ID,BYTES=4,START=22,TYPE=C
               DBDGEN
               FINISH
               END
```

関連資料:

各データベース・タイプの DBD 生成 (システム・ユーティリティー)

CREATE TABLESPACE

CREATE TABLESPACE ステートメントは、全機能データベースのデータ・セット・グループ、または高速機能高速処理データベース (DEDB) のデータベース・エリアを IMS に対して定義します。

呼び出し

このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

- 863 ページの『CREATE TABLESPACE 構文』
- 863 ページの『GSAM 構文』
- 863 ページの『HDAM または HIDAM 構文』
- 863 ページの『HSAM または SHSAM 構文』
- 863 ページの『HISAM または INDEX 構文』
- 864 ページの『SHISAM 構文』
- 864 ページの『DEDB 構文』

以下のデータベース・タイプには、1 つの TABLESPACE ステートメントしか定義できません。

- HSAM
- SHSAM
- GSAM
- HISAM
- SHISAM
- INDEX

以下のデータベース・タイプでは、TABLESPACE ステートメントは許可されません。

- PSINDEX
- LOGICAL
- PHDAM
- PHIDAM

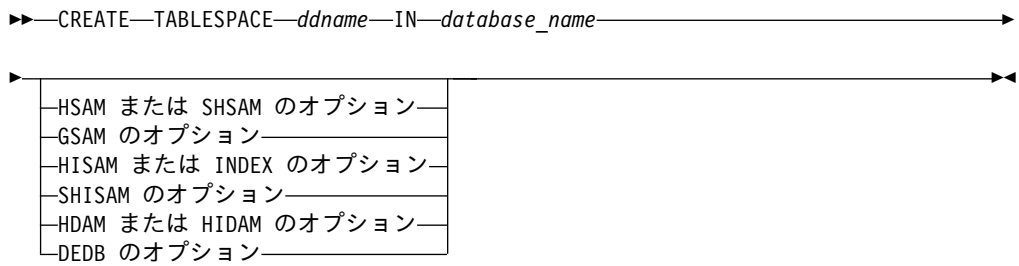
以下のデータベース・タイプには、1 個から 10 個の TABLESPACE ステートメントを定義することができます。

- HDAM
- HIDAM

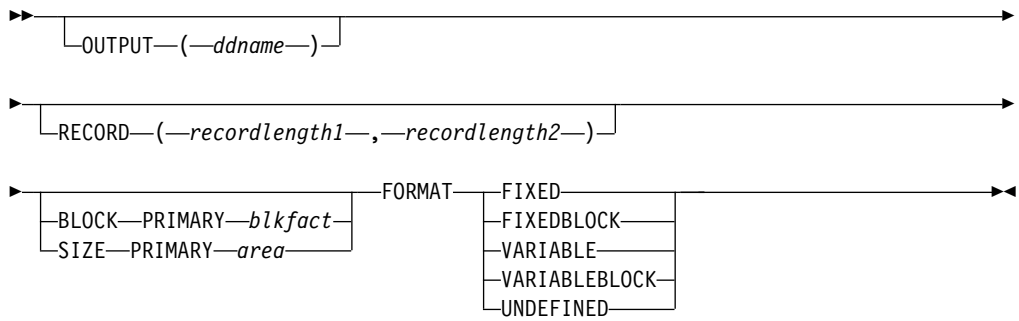
以下のデータベース・タイプには、1 個から 2048個の TABLESPACE ステートメントを定義することができます。

- DEDB

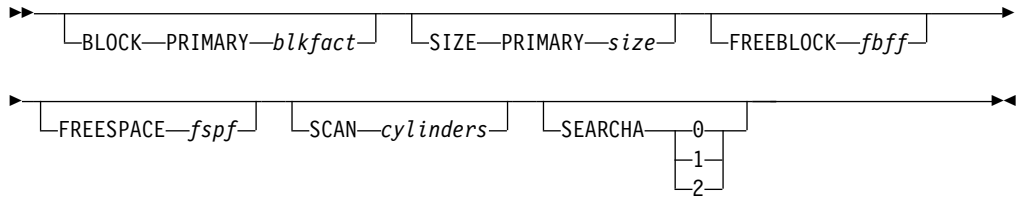
CREATE TABLESPACE 構文



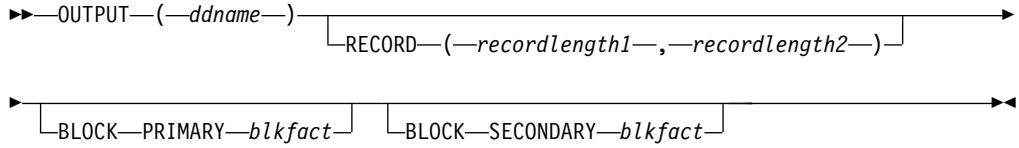
GSAM 構文



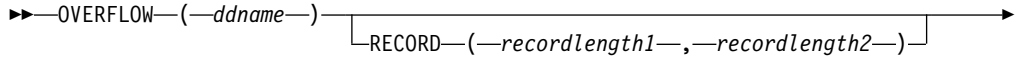
HDAM または HIDAM 構文

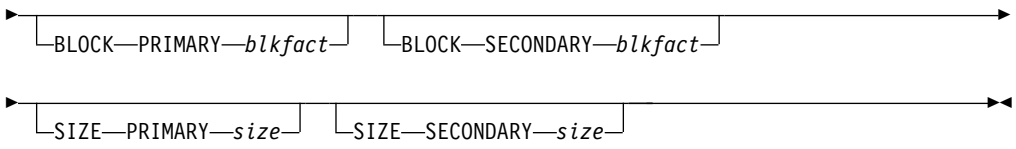


HSAM または SHSAM 構文

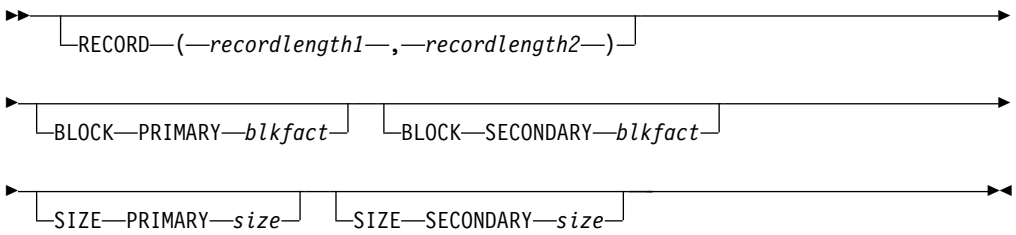


HISAM または INDEX 構文

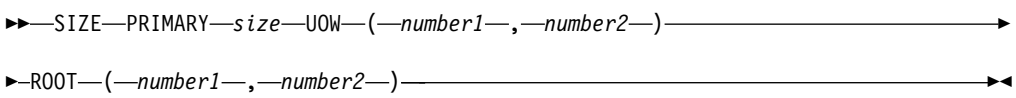




SHISAM 構文



DEDB 構文



説明

CREATE TABLESPACE ステートメントで使用される DD 名は、IMS システムまたはアカウントの中で固有である必要があります。複数の DBD に固有でない DD 名があると、データベースが破損する可能性があります。データベースの破損につながる可能性のある 1 つの状態は、両方の DD 名が同時に誤って使用される場合です (両方が、データ通信システムの 2 つの異なるメッセージ領域で使用されるか、あるいはデータベース専用システムのバッチ DL/I 領域で使用される 1 つの PSB の 2 つの PCB で使用される場合)。

CREATE TABLESPACE ステートメントには、以下のキーワード・パラメーターが定義されています。

CREATE TABLESPACE *ddname*
このデータ・セット・グループ内の 1 次データ・セット、あるいは DEDB のエリアを定義する DD 名を指定します。この名前は、1 文字から 8 文字の英数字で指定する必要があります。このパラメーターで示されるデータ・セットを IMS が使用する方法は、次のリストに示されているように、定義されるデータベースのタイプによって異なります。

HSAM | SHSAM | GSAM
入力データ・セットの DD 名。入力データ・セットは、アプリケーション・プログラムがデータベースからデータを取得する際に使用されます。

HISAM | SHISAM | INDEX
データ・セット・グループ内の 1 次データ・セットの DD 名。

HIDAM | HDAM
データ・セット・グループ内のデータ・セットの DD 名。

DEDB
単一エリア・データ・セットの場合は区域名または DD 名ですが、多重エリア・データ・セットの場合は区域名のみが可能です。データベースが DBRC に登録されている場合は、このパラメーターに区域名を指定してください。

IN *database_name*
データベースに属するこのデータ・セット・グループを示します。

database_name
このデータ・セット・グループが属するデータベース。

データベースが動的に割り振られるデータベースの DBD 名を指定します。この名前は、このデータベース・パラメーター・リストを識別するための IMS.SDFSRESL 内のメンバー名として使用されます。この名前が決して IMS.SDFSRESL 内の既存メンバーと競合しないように注意する必要があります。それには IMS モジュールおよびユーザー提供の出口ルーチンが含まれます (ただし、これらに限定されません)。

BLOCK PRIMARY
BLOCK SECONDARY
ブロック化因数を指定して、HSAM、SHSAM、GSAM、HISAM、SHISAM、および索引データベースのデータ・セット・グループのデータ・セットで使用できるようにするか、またはオーバーヘッドを含まない制御インターバル・サイズまたはブロック・サイズを指定して HDAM および HIDAM データベースのデータ・セット・グループのデータ・セットを使用できるようにします。

アクセス方式として VSAM を使用する HISAM、SHISAM、および索引データベースの場合は、BLOCK パラメーターの代わりに SIZE パラメーターを使用して制御インターバル・サイズを指定してください。HISAM、SHISAM、または索引データベースに SIZE キーワードを指定した場合は、BLOCK キーワードは無効です。

RECORD および BLOCK オペランドを使用する場合は、結果として得られる制御インターバル・サイズは、8192 バイトより小さい場合は 512 の倍数でなければなりません。指定したレコード長と指定したブロック化因数の積に VSAM オーバーヘッドを加えたものが 512 の倍数でなく、8192 バイトより小さい場合には、制御インターバル・サイズは、それより大きな最も近い 512 の倍数の値に切り上げて得られます。8192 から 30720 バイト (最大許容サイズ) の制御インターバル・サイズは、2048 バイトの倍数でなければなりません。RECORD オペランドと BLOCK オペランドの積に VSAM オーバーヘッドを加えたものが、8192 から 30720 バイトの値であっても、2048 の倍数ではない場合は、制御インターバル・サイズは、それより大きな最も近い 2048 の倍数の値に切り上げて得られます。

VSAM オーバーヘッドは、ブロック化因数が 1 の場合は 7 バイトであり、そうでない場合は 10 バイトです。OSAM データ・セットの最大ブロック・サイズは 32KB です。

HDAM および HIDAM データベースの場合は、BLOCK PRIMARY パラメーターを使用して、IMS の制御インターバルやブロック・サイズの計算をオーバーライドすることができます。ただし、BLOCK PRIMARY パラメーターで指定された値のほかに、IMS は、ルート・アンカー・ポイント、フリー・スペー

ス・アンカー・ポイント、およびアクセス方式オーバーヘッドのためのスペースを追加します。結果として得られるブロックまたは制御インターバルのサイズは、SIZE PRIMARY パラメーターの説明の中の式を参照するか、IMS の出力を調べることで判別することができます。 SIZE パラメーターが指定されておらず、アクセス方式が VSAM である場合、IMS は、CI 内の未使用スペースを CI 内の各論理レコードに均等に分配することで、最適な VSAM LRECL 値を計算します。 SIZE PRIMARY パラメーターが指定されている場合、これは実行されません。

以下の表は、BLOCK および RECORD オペランドの使用について説明します。

表 *167. BLOCK* および *RECORD* オペランド

データベース・タイプ	**BLOCK** および **RECORD** オペランドの使い方
HSAM/SHSAM	**BLOCK** BLOCK PRIMARY は、入力データ・セットに適用され、常に 1。 BLOCK SECONDARY は、出力データ・セットに適用され、常に 1。 **RECORD** *recordlength1* は、入力レコードの長さ。 *recordlength2* は、出力レコードの長さ。 HSAM/SHSAM は、常に非ブロック化される。LRECL と BLKSIZE は等しい。
GSAM	**BLOCK** BLOCK PRIMARY は、入出力データ・セットに適用される。 BLOCK SECONDARY は、無効なサブパラメーター。 **RECORD** *recordlength1* は、LRECL の長さであるか、可変長レコードの最大サイズ。 *recordlength2* は、可変長レコードの最小サイズ。 **SIZE** SIZE PRIMARY は、入出力データ・セットの BLKSIZE。 SIZE SECONDARY は、無効なサブパラメーター。
HISAM/SHISAM	**BLOCK** BLOCK PRIMARY は、1 次データ・セットのブロック化因数。 BLOCK SECONDARY は、オーバーフロー・データ・セットのブロック化因数。 **RECORD** *recordlength1* は、データ・セットの論理レコード長。 *recordlength2* は、オーバーフロー・データ・セットの論理レコード長。

表 *167. BLOCK* および *RECORD* オペランド *(*続き*)*

データベース・タイプ	**BLOCK** および **RECORD** オペランドの使い方
HIDAM、HDAM	**BLOCK** size0 は、OSAM または VSAM データ・セット・グループのオーバーヘッドを含まないサイズ。 **RECORD** 無視されます。
DEDB	BLOCK および RECORD オペランドは無効。
INDEX	**BLOCK** BLOCK PRIMARY は、1 次データ・セットのブロック化因数。 BLOCK SECONDARY は、オーバーフロー・データ・セットのブロック化因数。 **RECORD** *recordlength1* は、1 次データ・セットの論理レコード長。 *recordlength2* は、オーバーフロー・データ・セットの論理レコード長。

注**:** TABLESPACE ステートメントで *recordlength1* と *recordlength2* の両方を指定するときは、GSAM の場合を除き、*recordlength2* は *recordlength1* と等しいかそれ以上でなければなりません。

FORMAT
データ・セット内のレコードのフォーマットを指定します。有効なレコード・フォーマットは以下のとおりです。

FIXED
固定長。

FIXEDBLOCK
固定長およびブロック化。

VARIABLE
可変長。

VARIABLEBLOCK
可変長およびブロック化。

UNDEFINED
未定義長。

このキーワードは必須で、GSAM データベースのみに有効です。

FREEBLOCK
フリー・ブロック頻度因子を指定します。データベースのロードまたは再編成中に、このデータ・セット・グループ内の *n* 番目の制御インターバルまたはブロックごとに、それがフリー・スペースとして残されます。有効な範囲は 0 から 100 (1 を除く) です。デフォルトは 0 です。

値を小さくすると、データベース内のフリー・スペースの頻度が増えます。例えば、値を 2 にすると、各データの後にフリー・スペース・ブロックが生じることになります。この場合には、再編成ユーティリティーまたはロード・ユーティ

リティーの実行時に、フリー・スペース・ブロックのために余分の処理が必要になるので、システム・パフォーマンスが低下することになります。

FREEBLOCK は、IMS keyword FRSPC=(*fbff*,)) と同等です。

このキーワードはオプションで、HDAM または HIDAM にのみ有効です。

FREESPACE
フリー・スペース・パーセント係数を指定します。これは、フリー・スペースとしてこのデータ・セット・グループに残す各制御インターバルまたはブロックの最小パーセントです。有効な範囲は 0 から 99 です。デフォルトは 0 です。

このキーワードはオプションで、HDAM または HIDAM にのみ有効です。

FREESPACE は、IMS keyword FRSPC=(,*fspf*)) と同等です。

OUTPUT (*ddname*)
出力データ・セットの DD 名 (1 文字から 8 文字の英数字) を指定します。HSAM または SHSAM データベースには必須ですが、GSAM データベースにはオプションです。この出力データ・セットは、データベースのロード時に IMS が使用します。このキーワードは、他のデータベース・アクセス・タイプには無効です。

OUTPUT は、IMS キーワード DD2= と同等です。

OVERFLOW (*ddname*)
このデータ・セット・グループ内のオーバーフロー・データ・セットの DD 名 (1 文字から 8 文字の英数字) を指定します。次のものについてはこのパラメーターを指定する必要があります。

- 固有でないキーが付いている索引ポインター・セグメントが入っている INDEX データベース。
- HISAM データベースのすべてのデータ・セット・グループ。ただし、HISAM データベース内に 1 つのセグメント・タイプしか定義されていない場合は除きます。

以下の条件が適用されます。

- 単純 HISAM (SHISAM) データベースには無効です。
- 1 つのセグメント・タイプのみが含まれる HISAM データベースには必須ではありません。
- すべての索引セグメントが索引のキー順データ・セット内に挿入されるため、索引 DBD には必須ではありません。
- アクセス・タイプ SHISAM で定義された INDEX データベースには無効です。
- HISAM および INDEX データベース・アクセス・タイプにのみ有効です。

RECORD(*recordlength1*,*recordlength2*)
このデータ・セット・グループに使用するデータ管理論理レコード長を指定します。このキーワードはオプションであり、
HSAM、SHSAM、GSAM、HISAM、SHISAM、INDEX にのみ有効です。

SCAN *cylinders*
セグメント挿入操作時に使用可能ストレージ・スペースを検索するときに、スキャンする直接アクセス装置のシリンダーの数を指定します。このパラメーターはオプションであり、HIDAM または HDAM データベースにのみ有効です。指

定する場合、この値は、255 を超えない 10 進整数でなければなりません。一般的な値は 0 から 5 です。デフォルトは 3 です。0 を指定した場合、現行シリンダーのスペースのみがスキャンされます。

スキャンは、現行シリンダー位置から両方向に実行されます。スキャン限界値が原因で、現行エクステントの外側の区域がスキャンに含まれる場合には、IMS は、スキャンが現行エクステントの境界を超えないようにスキャン限界を調整します。このパラメーターで定義されたシリンダー範囲内にセグメント挿入のためのスペースが見つからない場合は、データベースのデータ・セット・グループの現在の終わりのスペースが使用されます。

SEARCHA 0 | 1 | 2
IMS が HD データベースにセグメントを挿入するときに使用する、HD スペース検索アルゴリズムのタイプを指定します。

0 IMS が使用する HD スペース検索アルゴリズムを選択することを指定します。 0 がデフォルトです。

1 IMS が使用する HD スペース検索アルゴリズムは、2 番目に望ましいブロックまたは CI 内のスペースを検索しないという指定です。

2 IMS が使用する HD スペース検索アルゴリズムは、2 番目に望ましいブロックまたは CI 内のスペースを検索するという指定です。

このキーワードはオプションで、HDAM または HIDAM データベースにのみ有効です。

SIZE PRIMARY *size1*
HISAM、SHISAM、INDEX の場合、このキーワードは、データ・セット・グループ内の 1 次データ・セットの制御インターバルまたはブロック・サイズを指定します。

HDAM、HIDAM の場合、このキーワードは、データ・セット・グループ内のデータ・セットの制御インターバルまたはブロック・サイズを指定します。
GSAM の場合、このキーワードは、入出力データ・セットのブロック・サイズを指定します。

DEDB の場合、このキーワードは必須で、制御インターバルを指定します。

このキーワードは、他のすべてのデータベース・タイプには無効です。

SIZE PRIMARY は、IMS keyword SIZE=(*size1,*)) と同等です。

SIZE SECONDARY *size2*
HISAM、SHISAM、INDEX の場合、このキーワードは、オーバーフロー・データ・セットの制御インターバルまたはブロック・サイズを指定します。

このキーワードは、HISAM、SHISAM、および INDEX にのみ有効です。

SIZE SECONDARY は、IMS keyword SIZE=(*,size2*)) と同等です。

ROOT(*number1*,*number2*)
区域のルート・アドレス可能部分と独立オーバーフロー用の予約区域に割り振られる合計スペースを指定します。

number1
区域のルート・アドレス可能部分に割り振られる合計スペースを指定しま

す。これは、UOW で表されます。VSAM データ・セットの残りの部分は、順次従属データ用に予約されます。

有効な範囲は 2 から 32767 です。VSAM データ・セットのスペース量より大きくすることはできません。

number2
独立オーバーフロー用に予約するスペース (UOW) を指定します。これは、1 以上で、かつ *number1* で指定された値未満でなければなりません。独立オーバーフローは UOW を含んでいませんが、スペース割り振りの単位として UOW サイズが使用されます。

再編成 UOW は、DEDB 初期設定ユーティリティーにより自動的に割り振られます。VSAM スペース定義には、この追加の UOW を含めてください。つまり、必要な合計スペースは、ルート・アドレス可能域、独立オーバーフロー、および再編成用の 1 つの追加 UOW になります。再編成 UOW は、高速 DEDB 直接再編成ユーティリティーでは使用されませんが、IMS の他の機能で使用される場合があります。

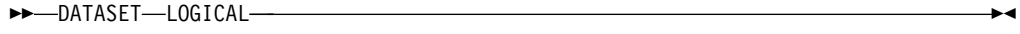
ROOT キーワードは必須で、DEDB にのみ有効です。

UOW(*number1*,*number2*)
必須で、DEDB にのみ有効です。*number1* は、作業単位の制御インターバルの数を指定します。有効な範囲は 2 から 32767 です。 *number2* は、オーバーフロー・セクション中の制御インターバルの数を指定します。 1 以上だが、*number1* より少なくとも 1 小さい任意の値。

使用上の注意

CREATE TABLESPACE ステートメントは、IMS DBD 生成ユーティリティーの DATASET または AREA ステートメントと同等です。

IMS では、DATASET ステートメントは、LOGICAL データベース・アクセス・タイプでも使用されますが、DDL では必須ではありません。



例: 基本 HDAM データベース

DBD 生成ユーティリティーに対して以下を入力すると、HDAM データベースが定義されます。

```
*************************************************************************
*        DBD COGDBD FROM CATU02-F                                       *
*************************************************************************
      DBD   NAME=COGDBD,                                                C
               ENCODING=Cp1047,                                         C
               ACCESS=(HDAM,OSAM),                                      C
               RMNAME=(DFSHDC40,3,3,25),                                C
               PASSWD=NO
      DATASET  DD1=COGDATA,                                             C
               DEVICE=3390,                                             C
               SIZE=(8192),                                             C
               REMARKS='Dataset Group 1'
      SEGM  NAME=ROOT,                                                  C
               PARENT=0,                                                C
               BYTES=(20),                                              C
```

```
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT SMALLINT
*************************************************************************
      SEGM      NAME=TSINT,                                             C
                PARENT=ROOT,                                            C
                BYTES=(8,6),                                            C
                REMARKS='This describes table TSINT.',                  C
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT INT
*************************************************************************
      SEGM      NAME=TINT,                                              C
                PARENT=ROOT,                                            C
                BYTES=(10,6),                                           C
                REMARKS='This describes table TINT.',                   C
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT BIGINT
*************************************************************************
      SEGM      NAME=TBINT,                                             C
                PARENT=ROOT,                                            C
                BYTES=(14,6),                                           C
                REMARKS='This describes table TBINT.',                  C
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT DECIMAL(7,2)
*************************************************************************
      SEGM      NAME=TDEC,                                              C
                PARENT=ROOT,                                            C
                BYTES=(10,6),                                           C
                REMARKS='This describes table TDEC.',                   C
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT FLOAT
*************************************************************************
      SEGM      NAME=TFLT,                                              C
                PARENT=ROOT,                                            C
                BYTES=(10,6),                                           C
                REMARKS='This describes table TFLT.',                   C
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT REAL
*************************************************************************
      SEGM      NAME=TRL,                                               C
                PARENT=ROOT,                                            C
                BYTES=(10,6),                                           C
                REMARKS='This describes table TRL.',                    C
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT DOUBLE
*************************************************************************
      SEGM      NAME=TDBL,                                              C
                PARENT=ROOT,                                            C
                BYTES=(14,6),                                           C
                REMARKS='This describes table TDBL.',                   C
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT CHAR(32)
*************************************************************************
      SEGM      NAME=TCHAR,                                             C
                PARENT=ROOT,                                            C
                BYTES=(38,6),                                           C
                REMARKS='This describes table TCHAR.',                  C
                RULES=(LLL,HERE)
*************************************************************************
*        SEGMENT NCHAR(32)
```

```
*********************************************************************
      SEGM     NAME=TNCHAR,                                          C
               PARENT=ROOT,                                          C
               BYTES=(38,6),                                         C
               REMARKS='This describes table TNCHAR.',               C
               RULES=(LLL,HERE)
. . . .
. . . .
      DBDGEN
      FINISH
      END
```

DDL では、CREATE TABLESPACE ステートメントによって同じデータ・セット・グループ (cogdata) が定義され、各表が割り当てられます。

```
CREATE TABLESPACE cogdata
 IN COGDBD
 SIZE PRIMARY 8192;
COMMENT ON TABLESPACE cogdata IN cogdbd IS 'Dataset Group 1';

CREATE TABLE tsinit
 IN cogdbd.cogdata
 ...

CREATE TABLE tinit
 IN cogdbd.cogdata
 ...

CREATE TABLE tbinit
 IN cogdbd.cogdata
 ...

CREATE TABLE tdec
 IN cogdbd.cogdata
 ...

CREATE TABLE tflt
 IN cogdbd.cogdata
 ...

CREATE TABLE trl
 IN cogdbd.cogdata
 ...

CREATE TABLE tdbl
 IN cogdbd.cogdata
 ...

CREATE TABLE tchar
 IN cogdbd.cogdata
 ...

CREATE TABLE tnchar
 IN cogdbd.cogdata
 ...
```

例: 複数のデータ・セット・グループを持つ HIDAM データベース

DBD 生成ユーティリティーに対して以下を入力すると、2 つのデータ・セット・グループ (DSG001 と DSG002) を持つ HIDAM データベースが作成されます。セグメント・タイプ K1、K2、K3、K4、K5、K6、および K8 が DSG001 に割り当てられます。セグメント・タイプ K5X、K5Y、K9、K10、K11、K12、K13、および K14 が DSG002 に割り当てられます。

```
*********************************************************************
*        DBD DHVNTZ02 FROM CMVC (CDCI19-3.DBDGEN)                   *
*********************************************************************
  DBD       NAME=DHVNTZ02,                                          C
               ACCESS=(HIDAM,VSAM),                                 C
               PASSWD=NO,                                           C
               VERSION=CDCTEST
*********************************************************************
*        DATASET GROUP NUMBER 1
*********************************************************************
DSG001 DATASET DD1=HIDAM,                                          C
               DEVICE=3330,                                        C
               SIZE=(2048),                                        C
               SCAN=3,                                             C
               REMARKS='Dataset Group 1'
*********************************************************************
*        SEGMENT NUMBER 1                                           *
*********************************************************************
    SEGM    NAME=K1,                                               C
               PARENT=0,                                           C
               BYTES=10,                                           C
               EXIT=(*,LOG,PATH,KEY,DATA),                         X
               RULES=(LLL,LAST),                                   C
               PTR=(NOTWIN,,,,)
. . . .
. . . .
*********************************************************************
*        SEGMENT NUMBER 2                                           *
*********************************************************************
    SEGM    NAME=K2,                                               C
               PARENT=((K1)),                                      C
               BYTES=10,                                           C
               RULES=(LLL,LAST),                                   C
               PTR=(TWIN,,,CTR,)
. . . .
. . . .
*********************************************************************
*        SEGMENT NUMBER 3                                           *
*********************************************************************
    SEGM    NAME=K3,                                               C
               PARENT=((K2,SNGL),                                  C
               (K5,PHYSICAL)),                                     C
               BYTES=34,                                           C
               RULES=(LVL,LAST),                                   C
               PTR=(TWIN,LTWIN,LPARNT,,)
. . . .
. . . .
*********************************************************************
*        SEGMENT NUMBER 4                                           *
*********************************************************************
    SEGM    NAME=K4,                                               C
               PARENT=((K3,SNGL)),                                 C
               BYTES=10,                                           C
               RULES=(LLL,LAST),                                   C
               PTR=(TWIN,,,,)
. . . .
. . . .
*********************************************************************
*        SEGMENT NUMBER 5                                           *
*********************************************************************
    SEGM    NAME=K5,                                               C
               PARENT=((K1,SNGL)),                                 C
               BYTES=10,                                           C
               EXIT=(*,LOG,PATH,KEY,DATA),                         X
               RULES=(LLL,LAST),                                   C
               PTR=(TWIN,,,,)
. . . .
```

```
. . . .
*************************************************************************
*        SEGMENT NUMBER 6                                               *
*************************************************************************
    SEGM    NAME=K6,                                                    C
               PARENT=((K5,SNGL)),                                      C
               BYTES=10,                                                C
               RULES=(LLL,LAST),                                        C
               PTR=(TWIN,,,,)
. . . .
. . . .
*************************************************************************
*        DATASET GROUP NUMBER 2
*************************************************************************
DSG002 DATASET DD1=HIDAM2,                                              C
               DEVICE=3330,                                             C
               SIZE=(512),                                              C
               SCAN=3,                                                  C
               REMARKS='Dataset Group 2'
*************************************************************************
*        SEGMENT NUMBER 7                                               *
*************************************************************************
    SEGM    NAME=K5X,                                                   C
               PARENT=((K5)),                                           C
               PTR=PAIRED,                                              C
               SOURCE=((K3,DATA,DHVNTZ02))
. . . .
. . . .
*************************************************************************
*        SEGMENT NUMBER 8                                               *
*************************************************************************
    SEGM    NAME=K5Y,                                                   C
               PARENT=((K5)),                                           C
               PTR=PAIRED,                                              C
               SOURCE=((K8,DATA,DHVNTZ02))
. . . .
. . . .
*************************************************************************
*        DATASET GROUP NUMBER 1
*************************************************************************
DSG001 DATASET
*************************************************************************
*        SEGMENT NUMBER 9                                               *
*************************************************************************
    SEGM    NAME=K8,                                                    C
               PARENT=((K1,SNGL),                                       C
               (K5,PHYSICAL)),                                          C
               BYTES=32,                                                C
               RULES=(LVL,LAST),                                        C
               PTR=(TWIN,LTWINBWD,LPARNT,,)
. . . .
. . . .
*************************************************************************
*        DATASET GROUP NUMBER 2
*************************************************************************
DSG002 DATASET
*************************************************************************
*        SEGMENT NUMBER 10                                              *
*************************************************************************
    SEGM    NAME=K9,                                                    C
               PARENT=K1,                                               C
               BYTES=29,                                                C
               EXIT=(*,LOG,PATH,KEY,DATA),                              X
               RULES=(VLV,LAST),                                        C
               PTR=(TWIN,,,CTR,)
. . . .
. . . .
```

```
************************************************************************
*        SEGMENT NUMBER 11                                             *
************************************************************************
    SEGM    NAME=K10,                                                  C
               PARENT=((K9,SNGL),                                      C
               (J12,PHYSICAL,DIVNTZ02)),                               C
               BYTES=26,                                               C
               RULES=(VVV,LAST),                                       C
               PTR=(TWIN,LTWINBWD,,,)
. . . .
. . . .
************************************************************************
*        SEGMENT NUMBER 12                                             *
************************************************************************
    SEGM    NAME=K11,                                                  C
               PARENT=((K9,DBLE)),                                     C
               BYTES=29,                                               C
               RULES=(LLL,LAST),                                       C
               PTR=(TWIN,,,,)
. . . .
. . . .
************************************************************************
*        SEGMENT NUMBER 13                                             *
************************************************************************
    SEGM    NAME=K12,                                                  C
               PARENT=((K11,DBLE)),                                    C
               BYTES=20,                                               C
               RULES=(LLL,LAST),                                       C
               PTR=(TWIN,,,,)
. . . .
. . . .
************************************************************************
*        SEGMENT NUMBER 14                                             *
************************************************************************
    SEGM    NAME=K13,                                                  C
               PARENT=((K11,DBLE)),                                    C
               BYTES=20,                                               C
               RULES=(LLL,HERE),                                       C
               PTR=(TWIN,,,,)
. . . .
. . . .
************************************************************************
*        SEGMENT NUMBER 15                                             *
************************************************************************
    SEGM    NAME=K14,                                                  C
               PARENT=((K9,SNGL),                                      C
               (J12,PHYSICAL,DIVNTZ02)),                               C
               BYTES=24,                                               C
               RULES=(LVV,LAST),                                       C
               PTR=(TWIN,,,,PAIRED)
. . . .
. . . .
    DBDGEN
    FINISH
    END
```

以下の DDL は、同じデータ・セット・グループおよび同等の表割り当てを作成します。

```
CREATE TABLESPACE hidam
 IN DHVNTZ02
 SIZE PRIMARY 2048
 SCAN 3;
COMMENT ON TABLESPACE hidam IN dhvntz02 IS 'Dataset Group 1';

CREATE TABLESPACE hidam2
 IN DHVNTZ02
```

```
 SIZE PRIMARY 512
 SCAN 3;
COMMENT ON TABLESPACE hidam2 IN dhvntz02 IS 'Dataset Group 2';

CREATE TABLE k1
 IN dhvntz02.hidam
 ...

CREATE TABLE k2
 IN dhvntz02.hidam
 ...

CREATE TABLE k3
 IN dhvntz02.hidam
 ...

CREATE TABLE k4
 IN dhvntz02.hidam
 ...

CREATE TABLE k5
 IN dhvntz02.hidam
 ...

CREATE TABLE k6
 IN dhvntz02.hidam
 ...

CREATE TABLE k5x
 IN dhvntz02.hidam2
...

CREATE TABLE k5y
 IN dhvntz02.hidam2
...

CREATE TABLE k8
 IN dhvntz02.hidam
...

CREATE TABLE k9
 IN dhvntz02.hidam2
...

CREATE TABLE k10
 IN dhvntz02.hidam2
...

CREATE TABLE k11
 IN dhvntz02.hidam2
...

CREATE TABLE k12
 IN dhvntz02.hidam2
...

CREATE TABLE k13
 IN dhvntz02.hidam2
...

CREATE TABLE k14
 IN dhvntz02.hidam2
...
```

例: 複数のデータ域を持つ高速機能 DEDB

以下の DBD 生成ユーティリティー入力は、7 個の区域を持つ DEDB を作成します。

```
  DBD       NAME=DEDBJN21,                                          C
               ENCODING=Cp1047,                                     C
               ACCESS=(DEDB),                                       C
               RMNAME=(RMOD3),                                      C
               PASSWD=NO
*************************************************************************
*        AREA NUMBER 1
*************************************************************************
       AREA    DD1=HOSPAR0,                                         C
               DEVICE=3330,                                         C
               SIZE=(2048),                                         C
               UOW=(15,10),                                         C
               ROOT=(10,5)
       AREA    DD1=HOSPAR1,                                         C
               DEVICE=3330,                                         C
               SIZE=(2048),                                         C
               UOW=(15,10),                                         C
               ROOT=(10,5)
       AREA    DD1=HOSPAR2,                                         C
               DEVICE=3330,                                         C
               SIZE=(2048),                                         C
               UOW=(15,10),                                         C
               ROOT=(10,5)
       AREA    DD1=HOSPAR3,                                         C
               DEVICE=3330,                                         C
               SIZE=(2048),                                         C
               UOW=(15,10),                                         C
               ROOT=(10,5)
       AREA    DD1=HOSPAR4,                                         C
               DEVICE=3330,                                         C
               SIZE=(2048),                                         C
               UOW=(15,10),                                         C
               ROOT=(10,5)
       AREA    DD1=HOSPAR5,                                         C
               DEVICE=3330,                                         C
               SIZE=(2048),                                         C
               UOW=(15,10),                                         C
               ROOT=(10,5)
       AREA    DD1=HOSPAR6,                                         C
               DEVICE=3330,                                         C
               SIZE=(2048),                                         C
               UOW=(15,10),                                         C
               ROOT=(10,5)
```

以下の DDL は、同等のデータ域を定義します。

```
CREATE TABLESPACE hospar0
 IN dedbjn21
 SIZE PRIMARY 2048
 UOW(15, 10)
 ROOT(10, 5);

CREATE TABLESPACE hospar1
 IN dedbjn21
 SIZE PRIMARY 2048
 UOW(15, 10)
 ROOT(10, 5);

CREATE TABLESPACE hospar2
 IN dedbjn21
 SIZE PRIMARY 2048
 UOW(15, 10)
```

```
 ROOT(10, 5);

CREATE TABLESPACE hospar3
 IN dedbjn21
 SIZE PRIMARY 2048
 UOW(15, 10)
 ROOT(10, 5);

CREATE TABLESPACE hospar4
 IN dedbjn21
 SIZE PRIMARY 2048
 UOW(15, 10)
 ROOT(10, 5);

CREATE TABLESPACE hospar5
 IN dedbjn21
 SIZE PRIMARY 2048
 UOW(15, 10)
 ROOT(10, 5);

CREATE TABLESPACE hospar6
 IN dedbjn21
 SIZE PRIMARY 2048
 UOW(15, 10)
 ROOT(10, 5);
```

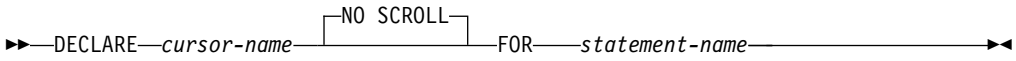
DECLARE CURSOR

DECLARE CURSOR ステートメントは、カーソルを定義します。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。 これは、実行可能ステートメントではありません。

構文



説明

DECLARE CURSOR ステートメントには、以下のキーワード・パラメーターが定義されています。

cursor-name
カーソルの名前を指定します。この名前は、ソース・プログラム内で既に宣言されているカーソルを示すものであってはなりません。

statement-name
カーソルがオープンされたときに必ずカーソルの結果表を指定する準備済み *select-statement* を指定します。*statement-name* は、ソース・プログラムの別の DECLARE CURSOR ステートメントで指定されたステートメント名と同じであってはなりません。準備済み SELECT ステートメントについては、 896 ページの『PREPARE』を参照してください。

注

COBOL プログラム内のカーソル**:** COBOL のソース・プログラムにおいては、DECLARE CURSOR ステートメントが、名前を使ってカーソルを明示参照するすべてのステートメントより前になければなりません。

例

この例では、DYSQL という名前のステートメントに対して C1 という名前のカーソルを宣言しています。

```
EXEC SQLIMS
DECLARE C1 CURSOR FOR DYSQL
END-EXEC.

EXEC SQLIMS
PREPARE DYSQL FROM :SELECT-STATEMENT
END-EXEC

EXEC SQLIMS OPEN C1 END-EXEC.

EXEC SQLIMS
FETCH C1 INTO :HOSPCODE, :HOSPNAME, :WARDNAME, :PATNAME
END-EXEC.

IF SQLIMSCODE = 100
PERFORM DATA-NOT-FOUND
ELSE
PERFORM GET-REST-OF-HOSP
UNTIL SQLIMSCODE IS NOT EQUAL TO ZERO.

EXEC SQLIMS CLOSE C1 END-EXEC.
```

DECLARE STATEMENT

DECLARE STATEMENT ステートメントは、アプリケーション・プログラムの文書化に使います。このステートメントは、準備済み SQL ステートメントを識別するのに使う名前を宣言します。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。 これは、実行可能ステートメントではありません。

構文

説明

DECLARE ステートメントには、以下のキーワード・パラメーターが定義されています。

statement-name **STATEMENT**
準備済み SQL ステートメントを指定するためにアプリケーション・プログラムで使用する 1 つ以上の名前をリストします。

例

この例は、COBOL プログラムにおける DECLARE STATEMENT ステートメントの使い方を示したものです。UPD という名前のステートメントを宣言しています。

```
EXEC SQLIMS
  DECLARE UPD STATEMENT
END-EXEC.

EXEC SQLIMS
  PREPARE UPD FROM :SQLSTMT
END-EXEC.
IF SQLIMSCODE < 0
  MOVE '**** PREPARE ERROR ****' TO ERR-MSG1
  PERFORM 100-ERROR
ELSE
  PERFORM EXECUTE-STMT
END-IF
```

DELETE

DELETE ステートメントは、表から行を削除します。

検索 DELETE 形式は、検索条件により任意に決定される 1 つ以上の行を削除します。

呼び出し

このステートメントは、COBOL あるいは Java アプリケーション・プログラムに組み込むことも、対話式に発行することもできます。

- 『DELETE 構文』
- 『table 構文』

DELETE 構文

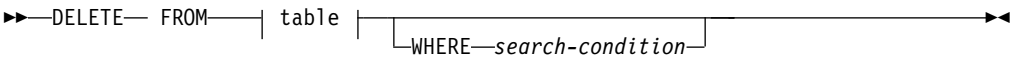
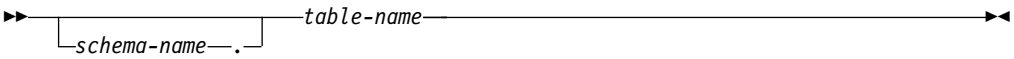


table 構文



説明

DELETE ステートメントには、以下のキーワード・パラメーターが定義されています。

DELETE FROM
削除する行がある表を指定します。

table-name
table-name は、SQL 照会のテーブルの名前を定義します。この名前は IMS 内のセグメントを識別する必要があります。

schema-name

schema-name は、SQL 照会のスキーマを定義します。 IMS では、スキーマ名は PCB 名です。

WHERE

削除する行を指定します。この文節を省略することも、検索条件を指定することもできます。この文節が省略されると、表のすべての行が削除されます。

search-condition

708 ページの『検索条件』で述べた検索条件です。検索条件内の各 *column-name* は、表の列を示すものでなければなりません。

表の各行に検索条件が適用され、検索条件の結果が真となった行が削除されます。

例

表 PCB01.HOSPITAL から、病院 Alexandria および Santa Teresa に関するすべての行を削除します。

```
DELETE FROM PCB01.HOSPITAL WHERE HOSPNAME = 'Alexandria' OR HOSPNAME = 'Santa Teresa';
```

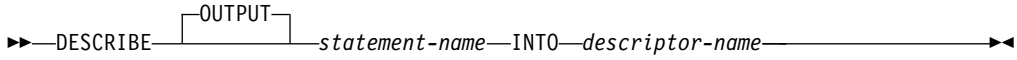
DESCRIBE OUTPUT

DESCRIBE OUTPUT ステートメントは、準備済みステートメントに関する情報を入手します。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。これは、動的に準備できる実行可能ステートメントです。

構文



説明

DESCRIBE OUTPUT ステートメントには、以下のキーワード・パラメーターが定義されています。

OUTPUT

statement-name を指定する場合に、準備済み SELECT ステートメントの選択リスト列に関する情報を戻すように DESCRIBE に指示するオプションのキーワード。

statement-name

準備済みステートメントを指定します。この名前は、DESCRIBE ステートメントの実行時に、準備されているステートメントを示すものでなければなりません。

INTO *descriptor-name*

SQL 記述子域 (SQLIMSDA) を指定します。これについては、 917 ページの

『SQL 記述子域 (SQLIMSDA)』で説明しています。アプリケーション内で SQLIMSDA を宣言するには、INCLUDE SQLIMSDA ステートメントを使用します。

DESCRIBE ステートメントの実行後に、SQLIMSDA 内の SQLN を除く全フィールドが IMS によって設定されるかまたは無視されます。

例

組み込み SQLIMSDA を使用して DESCRIBE ステートメントを実行します。DESCRIBE の後、SQLIMSD は返される結果フィールドの数を指定します。SQLIMSD が 0 の場合、ステートメントは非 SELECT ステートメント (INSERT、UPDATE、または DELTEE など) です。 SQLIMSD がゼロより大きい場合、ステートメントは SELECT ステートメントで、各結果フィールド用にストレージを割り振り、SQLIMSDA の SQLIMSDATA フィールドに対してそのアドレスを指定します。最後に、結果データ・セットを SQLIMSDA にフェッチします。

```
EXEC SQLIMS
   INCLUDE SQLIMSDA
END-EXEC

EXEC SQLIMS
  DECLARE C1 CURSOR FOR DYSQL
END-EXEC.

EXEC SQLIMS
   PREPARE DYSQL FROM :SELECT-STATEMENT
END-EXEC

EXEC SQLIMS
   DESCRIBE DYSQL INTO :SQLIMSDA
END-EXEC

IF SQLIMSD > 0
   EXEC SQLIMS OPEN C1 END-EXEC.
   .... /* Code to allocate the storage for each result field */
   .... /* Set the storage address to each SQLIMSDATA variable */
   EXEC SQLIMS FETCH C1 INTO :SQLIMSDA END-EXEC.

   IF SQLIMSCODE = 100
      PERFORM DATA-NOT-FOUND
   ELSE
      PERFORM GET-REST-OF-HOSP
      UNTIL SQLIMSCODE IS NOT EQUAL TO ZERO.

   EXEC SQLIMS CLOSE C1 END-EXEC.
```

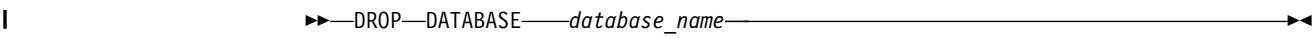
DROP DATABASE

DROP DATABASE ステートメントは、IMS から データベースを削除します。データベースが削除されると、その記述が現行 IMS のカタログから削除されます。

呼び出し

このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

構文



説明

DROP DATABASE ステートメントには、以下のキーワード・パラメーターが定義されています。

DATABASE *database_name*
ドロップするデータベースを指定します。この名前は IMS 内のデータベースを識別する必要があります。データベースがドロップされると、そのデータベースのすべての表、索引もドロップされます。

使用上の注意

データベースのすべてのオブジェクト (表スペース、表、および列など)、および保留中の変更もドロップされます。

例

以下の例を使用して既にデータベースを作成しており (CREATE DATABASE (アプリケーション・プログラミング API)を参照)、そのデータベースをドロップしようとしていると仮定します。

```
DROP DATABASE hospdbd1
```

DROP PROGRAMVIEW

DROP PROGRAMVIEW ステートメントは、アプリケーション PROGRAMVIEW (PSB) を IMS から削除します。アプリケーション PROGRAMVIEW が削除されると、その記述が現行 IMS のカタログから削除されます。

呼び出し

このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

構文

説明

DROP PROGRAMVIEW ステートメントには、以下のキーワード・パラメーターが定義されています。

PROGRAMVIEW *psb_name*
ドロップする PSB を指定します。この名前は IMS 内の PSB を識別する必要があります。

使用上の注意

プログラム・ビューおよびスキーマの定義に対する保留中の変更もすべてドロップされます。

例

```
DROP PROGRAMVIEW PSB123
```

DROP TABLE

DROP TABLE ステートメントは、IMS 内のデータベースから既存の表を削除します。そのテーブルに直接または間接的に従属するリソースも削除されます。表が削除されると、その記述が現行 IMS のカタログから削除されます。

呼び出し

このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

構文

説明

DROP TABLE ステートメントには、以下のキーワード・パラメーターが定義されています。

TABLE *table_name*
ドロップする表の 1 文字から 128 文字の大文字英数字の名前を識別します。表の名前には、下線文字を含めることができます。名前は、IMS 内に存在する表を識別する必要があります。表がドロップされると、その子として定義されているすべての表もドロップされます。

IN *database_name*
表が定義されているデータベースの DBD 名を指定します。 DROP TABLE を指定することは、表がデータベースから削除されることを示していました。これにより、このデータベースの変更がトリガーされます。

使用上の注意

表が直接または間接的にドロップされると、次の項目もドロップされます。

- 表に関連付けられたすべての特権。
- FOREIGN KEY 節を使用してドロップする表に関連付けられた子として定義されているすべての表。

例

以下の例を使用して既に表を作成しており (CREATE TABLE (アプリケーション・プログラミング API)を参照)、その表をドロップしようとしていると仮定します。

```
DROP TABLE testinteger IN COGDBD
```

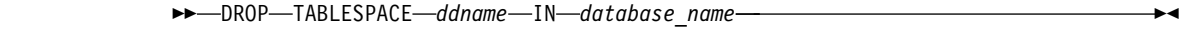
DROP TABLESPACE

DROP TABLESPACE ステートメントは、データベース内のデータ・セット・グループ、または DEDB のエリアを削除します。TABLESPACE のドロップは、データベース・リソースに対する変更です。

呼び出し

このステートメントは、IMS Universal JDBC ドライバーを使用した IMS への接続が確立されている Java アプリケーション・プログラムから実行することができます。これは実行可能ステートメントですが、動的に準備することはできません。

構文



説明

DROP TABLESPACE ステートメントには、以下のキーワード・パラメーターが定義されています。

TABLESPACE *ddname*
ドロップする表スペースの 1 文字から 8 文字の英数字 DD 名を識別します。DD 名は、IMS 内に存在する表スペースを識別する必要があります。

IN *database_name*
表スペースを削除するデータベースを指定します。

使用上の注意

表スペースがドロップされると、同じコミット有効範囲内に新規の表スペースを作成することができます。表スペースの定義に対する保留中の変更もすべてドロップされます。表スペースが直接または間接的にドロップされると、必ずその表スペース内のすべての表が次の使用可能な表スペースに移動されます。ドロップされた表スペースが最後の表スペースである場合、-9000 (メッセージおよびコード) エラー・メッセージを受信します。

表スペースは、以下のアクセス・タイプのデータベースからドロップすることができます。

- DEDB
- GSAM
- HDAM
- HIDAM
- HISAM
- HSAM
- INDEX
- SHISAM
- SHSAM

例

以下の例を使用して既に表スペースを作成しており (CREATE TABLESPACE (アプリケーション・プログラミング API)を参照)、その表スペースをドロップしようとしていると仮定します。

```
DROP TABLESPACE hidam2 IN DHVNTZ02
```

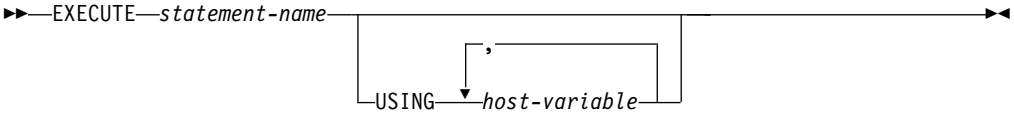
EXECUTE

EXECUTE ステートメントは、準備済み SQL ステートメントを実行します。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。 これは実行可能ステートメントですが、動的に準備することはできません。

構文



説明

EXECUTE ステートメントには、以下のキーワード・パラメーターが定義されています。

statement-name
実行する準備済みステートメントを指定します。*statement-name* は、作業単位内で前もって準備されたステートメントを示すものでなければなりません。また、準備済みステートメントは SELECT ステートメントであってはなりません。

USING
準備済みステートメント内のパラメーター・マーカー (疑問符) に置き換わる値を持つ変数を、この後にリストします。(パラメーター・マーカーについては、896 ページの『PREPARE』を参照してください。) 準備済みステートメントにパラメーター・マーカーが含まれている場合は、 EXECUTE ステートメントに USING を入れなければなりません。パラメーター・マーカーがなければ、USING は無視されます。

パラメーター・マーカーの値の置換について詳しくは、パラメーター・マーカーの置換を参照してください。

host-variable
構造または変数を指定します。これは、アプリケーション・プログラム内でホスト構造および変数を宣言する際の規則に従って記述しておかなければなりません。構造に対する参照は、構造のそれぞれの変数に対する参照に置き換えられます。変数の数は、準備済みステートメント内のパラメーター・マーカーの数と同じでなければなりません。*n* 番目の変数が、準備済みステートメント内の *n* 番目のパラメーター・マーカーの値を提供します。

注

パラメーター・マーカーの置換:

準備済みステートメントが実行される前に、そのステートメント内の各パラメーター・マーカーは実際に、それに対応するホスト変数に置き換えられます。置換は、ソースがホスト変数の値であり、ターゲットが変数である割り当て演算です。割り当て規則は、割り当てと比較 (アプリケーション・プログラミング API)で列への割り当てに関して説明した規則です。

例

この例では、UPDATE ステートメントは、変数 SQLSTMT から準備されて実行されます。

```
EXEC SQLIMS
  DELCARE UPD STATEMENT
END-EXEC.

EXEC SQLIMS
  PREPARE UPD FROM :SQLSTMT
END-EXEC.
IF SQLIMSCODE < 0
  MOVE '**** PREPARE ERROR ****' TO ERR-MSG1
  PERFORM 100-ERROR
ELSE
  EXEC SQLIMS
    EXECUTE UPD
  END-EXEC
END-IF.
```

FETCH

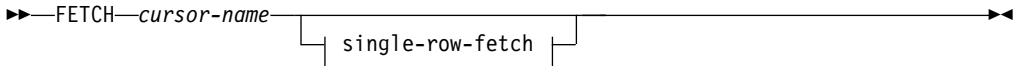
FETCH ステートメントは、結果表の行にカーソルを置きます。このステートメントはゼロまたは 1 を返すことができ、ターゲットの指定があれば、行の値をホスト変数に割り当てます。

呼び出し

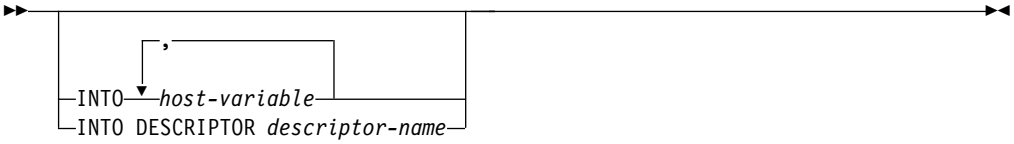
このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。 これは実行可能ステートメントですが、動的に準備することはできません。

- 『FETCH 構文』
- 888 ページの『single-row-fetch 構文』

FETCH 構文



single-row-fetch 構文



説明

FETCH ステートメントには、以下のキーワード・パラメーターが定義されています。

INTO *host-variable*
ホスト変数をリストします。*host-variable* はそれぞれ、アプリケーション・プログラム内でホスト構造および変数を宣言する際の規則に従って記述された構造または変数を示すものでなければなりません。構造に対する参照は、構造のそれぞれの変数に対する参照に置き換えられます。結果行の最初の値が最初のホスト変数に割り当てられ、2 番目の値が 2 番目のホスト変数に割り当てられるといった形で、以下同様に続いていきます。

INTO DESCRIPTOR *descriptor-name*
SQLIMSDA を指定します。これは、ホスト出力変数の有効な記述を含みます。関連する SELECT ステートメントからの結果値は、出力ホスト変数形式でアプリケーション・プログラムに戻されます。

FETCH ステートメントが処理される前に、SQLIMSDA 内の次のフィールドを設定しておく必要があります。

- SQIMSLDA での SQLIMSVAR オカレンスの数を示す SQLIMSN
- SQLIMSDA 内で割り振られるストレージのバイト数を示す SQLIMSABC
- ステートメントの処理時に使われる SQLIMSDA 内の変数の数を示す SQLIMSD
- 変数の属性を示す SQLIMSVAR オカレンス

SQLIMSDA は、すべての SQLIMSVAR オカレンスを入れることができる大きさにしなければなりません。各 SQLIMSVAR オカレンスは、結果表内の値が割り当てられるホスト変数またはバッファーを記述します。SQLIMSVAR の説明、SQLIMSVAR オカレンスの回数を判別する方法など、SQLIMSDA について詳しくは、 917 ページの『SQL 記述子域 (SQLIMSDA)』を参照してください。

SQLIMSD は、ゼロ以上で SQLIMSN 以下の値に設定しなければなりません。

cursor-name
フェッチ操作で使用するカーソルを指定します。カーソル名は、宣言済みカーソルまたは割り当てカーソルを示している必要があります。 FETCH ステートメントを実行する場合、カーソルはオープン状態になければなりません。

例

例 1: FETCH ステートメントは、SELECT ステートメントの結果をフェッチして、アプリケーション・プログラム変数 HOSPCODE および HOSPNAME に入れます。フェッチする行が残っていない場合には、検出不能条件が戻されます。

```
EXEC SQLIMS
  DECLARE C1 CURSOR FOR DYSQL
END-EXEC.

EXEC SQLIMS
   PREPARE DYSQL FROM :SELECT-STATEMENT
END-EXEC

EXEC SQLIMS OPEN C1 END-EXEC.

EXEC SQLIMS FETCH C1 INTO :HOSPCODE, :HOSPNAME END-EXEC.

   IF SQLIMSCODE = 100
      PERFORM DATA-NOT-FOUND
   ELSE
      PERFORM GET-REST-OF-HOSP
      UNTIL SQLIMSCODE IS NOT EQUAL TO ZERO.

   EXEC SQLIMS CLOSE C1 END-EXEC.
```

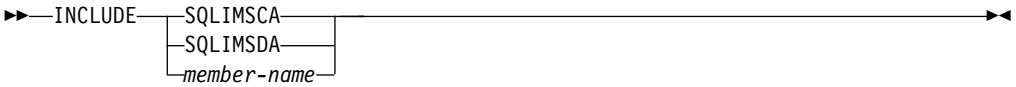
INCLUDE

INCLUDE ステートメントは、アプリケーション・コード (宣言およびステートメントを含む) を、ソース・プログラムに挿入します。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。 これは、実行可能ステートメントではありません。

構文



説明

INCLUDE ステートメントには、以下のキーワード・パラメーターが定義されています。

SQLIMSCA

SQL 連絡域 (SQLIMSCA) の記述を含めることを指示します。INCLUDE SQLIMSCA は、同じアプリケーション・プログラムの中で複数回指定してはなりません。COBOL の場合、INCLUDE SQLIMSCA は、作動ストレージ・セクションまたはリンケージ・セクションで指定しなければなりません。

SQLIMSCA の説明については、 914 ページの『SQL 連絡域 (SQLIMSCA)』を参照してください。

SQLIMSDA
SQL 記述子域 (SQLIMSDA) の記述を含めることを示します。SQLIMSDA の説明については、 917 ページの『SQL 記述子域 (SQLIMSDA)』を参照してください。

member-name
アプリケーション・プログラムを (IMS コプロセッサーを使用して) 準備する場合は、ライブラリー入力データとする区分データ・セットのメンバーを指定します。 SQL ID を指定する必要があります。

メンバーには、ホスト言語ソース・ステートメントおよび INCLUDE ステートメント以外の SQL ステートメントを含めることができます。COBOL の場合は、データ部または手続き部以外のところに INCLUDE *member-name* を指定してはいけません。

注

アプリケーション・プログラムを (IMS コプロセッサーを使用して) 準備する場合、INCLUDE ステートメントはソース・ステートメントに置き換えられます。したがって、INCLUDE ステートメントは、アプリケーション・プログラムの中で、その結果作成されたソース・ステートメントがコンパイラーで受け入れ可能となる場所に指定しなければなりません。

INCLUDE ステートメントでは、INCLUDE ステートメントを含むソース・ステートメントの参照はできません。

例

COBOL プログラムに SQL 連絡域を含めます。

```
EXEC SQLIMS INCLUDE SQLIMSCA END-EXEC.
```

INSERT

INSERT ステートメントは行を表に挿入します。

VALUES を使った INSERT の形式は、与えられた値または参照された値を使って、表に単一行を挿入します。

呼び出し

このステートメントは、COBOL あるいは Java アプリケーション・プログラムに組み込むことも、対話式に発行することもできます。 INSERT は、アプリケーション・プログラムに組み込むことができます。これは、動的に準備できる実行可能ステートメントです。

- 『COBOL の構文』
- 891 ページの『Java の構文』

COBOL の構文

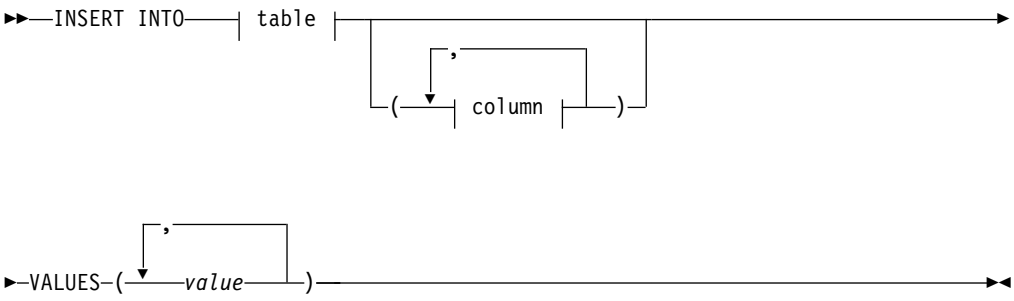
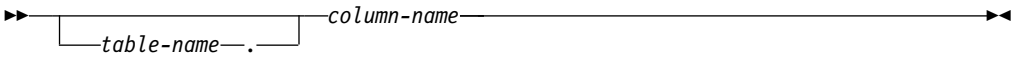
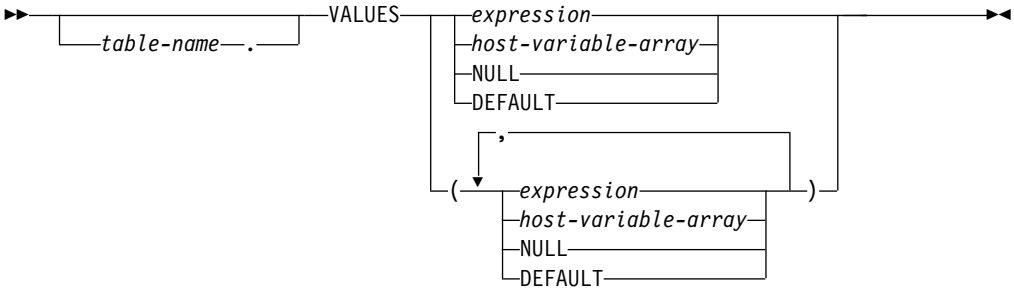


table 構文

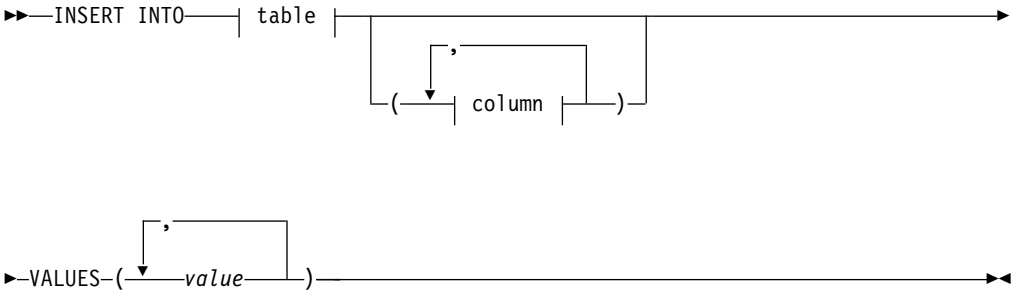
column 構文



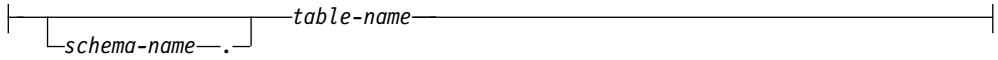
multi-row-insert 構文



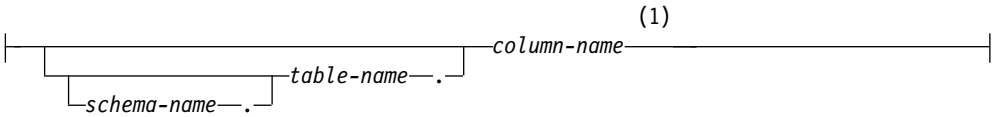
Java の構文



テーブル:



列:



注:

1 複数のテーブルに同じ列名を使用できますが、テーブルが修飾されていない場合、各テーブルで列を検索する必要があります。

説明

INSERT ステートメントには、以下のキーワード・パラメーターが定義されています。

INSERT INTO
INSERT ステートメントのオブジェクトを指定します。

table-name
table-name は、SQL 照会のテーブルの名前を定義します。この名前は IMS 内のセグメントを識別する必要があります。

schema-name
schema-name は、SQL 照会のスキーマを定義します。 IMS では、スキーマ名は PCB 名です。

column-name
挿入値が用意される列を指定します。それぞれの名前は、セグメントのフィールドを示す必要があります。列はどのような順序で指定しても構いませんが、同じ列を複数回指定してはいけません。

列リストを省略すると、表内のすべての列をメタデータによって識別される順序で識別するようにリストしたものを暗黙的に指定したことになります。

VALUES
値をリストした形式で、新しい行を 1 行指定します。VALUES 文節内の値の数は、列リスト内の名前の数と同じでなければなりません。最初の値がリストの 1 列目に挿入され、2 番目の値が 2 列目に挿入されるといった形で、以下同様に続いていきます。値のリストは、括弧で囲む必要があります。

注

挿入規則:

挿入値は、以下の規則を満たさなければなりません。以下の規則に従わない場合、または INSERT ステートメントの実行中にその他のエラーが発生した場合には、行は挿入されず、カーソルの位置は変更されません。

- 長さ。列の挿入値が数値の場合、その列は、数値の整数部分を表せるだけの容量を持つ数値列でなければなりません。列の挿入値がストリングの場合、その列は、少なくともそのストリングの長さと同じ長さ属性を持つストリング列、または、ストリングが日付、時刻、あるいはタイム・スタンプを表している場合は日時列でなければなりません。

- 参照制約。非ルート・レベルで表にレコードを挿入する場合は、表のすべての外部キー・フィールドの値を指定する必要があります。外部キー・フィールドにより、標準の SQL 処理を使用して階層パス内に挿入する新規レコード (またはセグメント・インスタンス) が正しく配置されます。これはリレーショナル・データベースの外部キーと似ています。
- 列リストの省略。列リストを省略する場合には、INSERT ステートメントがバインドまたは (動的実行の場合) 準備されたときに表に存在していた列ごとに、値を指定しなければなりません。

挿入された行数:

COBOL の場合、SQLIMSCA 内の SQLIMSERRD(3) の値は、INSERT ステートメントの実行が完了した後に挿入された行数です。 SQLIMSCA の完全な説明については、 914 ページの『SQL 連絡域 (SQLIMSCA)』を参照してください。

バイナリー・フィールドの挿入:

COBOL の場合、バイナリー・フィールドの挿入時には、パラメーター・マーカーを使用してバイナリー値を指定する必要があります。パラメーター・マーカーを使用しないと、408 (データ・タイプが非互換) エラーが発生します。

例

ルートにデータを挿入

次のステートメントは、新規 HOSPITAL レコードを挿入します。

```
INSERT INTO PCB01.HOSPITAL (HOSPCODE, HOSPNAME)
VALUES ('R1210050000A', 'O''MALLEY CLINIC')
```

階層パス内の指定された表にデータを挿入

非ルート・レベルで表にレコードを挿入する場合は、表のすべての仮想外部キー・フィールドの値を指定する必要があります。次のステートメントは、新規 ILLNESS レコードを特定の HOSPITAL、WARD、および PATIENT 表に挿入します。この例では、ILLNESS 表には、HOSPITAL_HOSPCODE、WARD_WARDNO、および PATIENT_PATNUM の 3 つの仮想外部キーがあります。新規レコードは、HOSPITAL 表に値が 'H5140070000H' の HOSPCODE が存在し、WARD 表に値 '01' が存在し、および PATIENT 表に 'R1210050000A' の PATNUM 値が存在する場合にのみ挿入されます。

```
INSERT INTO PCB01.ILLNESS (HOSPITAL_HOSPCODE, WARD_WARDNO,
   ILLNAME, PATIENT_PATNUM)
VALUES ('H5140070000H', '01', 'COLD', 'R1210050000A')
```

次のステートメントは、新規 WARD レコードを特定の HOSPITAL 表に挿入します。この例では、WARD 表には仮想外部キー HOSPITAL_HOSPCODE があります。新規レコードは、HOSPITAL 表に、値が 'H5140070000H' の HOSPCODE が存在する場合にのみ挿入されます。

```
INSERT INTO PCB01.WARD (WARDNO, HOSPITAL_HOSPCODE, WARDNAME)
VALUES ('0001', 'H5140070000H', 'EMGY')
```

サブフィールドがある検索可能フィールドにデータを挿入
検索可能フィールドがサブフィールドで構成されている場合、すべてのサブフィールドの値を設定してデータを挿入することで、検索可能フィールドに完全に値を入力できます。

仮想外部キー・フィールドを指定しないで非ルート・レベルでレコードを挿入
このステートメントでは、WARD_WARDNO 仮想外部キー・フィールドが欠落しています。この照会は、すべての外部キーに有効な値を指定しなければならない参照整合性の制約に違反しているため、失敗します。

```
INSERT INTO PCB01.PATIENT (HOSPITAL_HOSPCODE, PATNAME, PATNUM)
VALUES ('HW3201', 'JOHN O''CONNER', 'Z800')
```

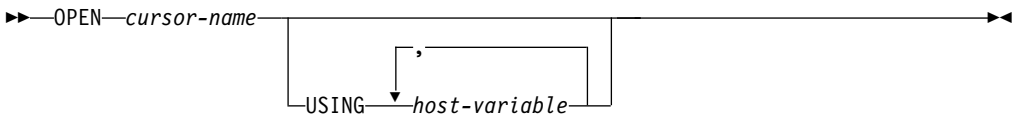
OPEN

OPEN ステートメントは、カーソルをオープンして、結果表の行の処理に使用できるようにします。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。 これは実行可能ステートメントですが、動的に準備することはできません。

構文



説明

OPEN ステートメントには、以下のキーワード・パラメーターが定義されています。

cursor-name
オープンするカーソルを指定します。*cursor-name* は、 878 ページの『DECLARE CURSOR』で説明したような宣言済みカーソルを示している必要があります。OPEN ステートメントの実行時には、カーソルはクローズ状態でなければなりません。

カーソルの SELECT ステートメントは、以下のいずれかのタイプの SELECT ステートメントです。

- DECLARE CURSOR ステートメントで指定する *statement-name* で示される準備済み SELECT ステートメント

このステートメントが正しく準備されていないか、SELECT ステートメントでない場合、カーソルは正しくオープンできません。

カーソルの結果表は、SELECT ステートメントを評価することによって導き出されます。評価には、OPEN ステートメントの USING 文節で指定されるホス

ト変数の現行値が使われます。結果表の行は、OPEN ステートメントの実行時に導き出すことができます。カーソルはオープン状態で、その結果表の最初の行の前に置かれます。

USING
カーソルのステートメント内で、カーソルの宣言に応じてパラメーター・マーカー (疑問符)を指定します。

- DECLARE CURSOR ステートメントに *statement-name* が含まれていた場合、ステートメントは PREPARE ステートメントによって準備済みです。OPEN ステートメントの USING 文節に指定したホスト変数は、準備済みステートメント内のパラメーター・マーカーを置き換えます。これは、OPEN ステートメントの USING 文節の典型的な使用法です。パラメーター・マーカーの置換については、 896 ページの『PREPARE』を参照してください。

  準備済みステートメントにパラメーター・マーカーが含まれている場合は、USING を使用する必要があります。準備済みステートメントにパラメーター・マーカーが含まれていなければ、 USING は無視されます。

host-variable
ホスト構造または変数を指定します。これはホスト構造および変数の宣言規則に従って、アプリケーション・プログラムに記述されていなければなりません。このステートメントが実行されるときに、構造に対する参照はその各変数に対する参照に置き換えられます。変数の数は、準備済みステートメント内のパラメーター・マーカーの数と同じでなければなりません。*n* 番目の変数が、準備済みステートメント内の *n* 番目のパラメーター・マーカーに対応します。適切な場所に、パラメーター・マーカーの値のソースとしてロケーター変数を指定することができます。

注

カーソルのクローズ状態*:* アプリケーション・プロセス内のすべてのカーソルがクローズ状態となるのは、以下の場合です。

- アプリケーション・プロセスが開始されるとき。
- アプリケーション・プロセスで新しい作業単位が開始されるとき。

カーソルは、次のような場合にもクローズ状態となります。

- CLOSE ステートメントが実行された場合。
- エラーが検出されたために、カーソルの位置が予測不能になった場合。

カーソルの結果表から行を取り出すためには、カーソルがオープンしているときに FETCH ステートメントを実行しなければなりません。カーソルをクローズ状態からオープン状態に変える手段としては、 OPEN ステートメントを実行する方法しかありません。

パラメーター・マーカーの置換*:* OPEN ステートメントが実行される前に、その照会内の各パラメーター・マーカーは実際に、それに対応するホスト変数に置き換えられます。この置換は、ソースがホスト変数の値であり、ターゲットが IMS 内の変数である割り当て演算です。割り当て規則は、 697 ページの『割り当てと比較』で列への割り当てに関して説明した規則です。

カーソルの SELECT ステートメントを評価する時点で、ステートメント内の各パラメーター・マーカーは実際に、それに対応するホスト変数の値に置き換えられます。置換処理の詳細については、パラメーター・マーカーの置換を参照してください。

例

例 *1:* フェッチされる行の先頭にカーソルを位置付ける OPEN ステートメントを実行します。

```
EXEC SQLIMS
  DECLARE C1 CURSOR FOR DYSQL
END-EXEC.

EXEC SQLIMS
   PREPARE DYSQL FROM :SELECT-STATEMENT
END-EXEC

EXEC SQLIMS OPEN C1 END-EXEC.

EXEC SQLIMS FETCH C1 INTO :HOSPCODE, :HOSPNAME, :WARDNAME, :PATNAME END-EXEC.

   IF SQLIMSCODE = 100
      PERFORM DATA-NOT-FOUND
   ELSE
      PERFORM GET-REST-OF-HOSP
      UNTIL SQLIMSCODE IS NOT EQUAL TO ZERO.

   EXEC SQLIMS CLOSE C1 END-EXEC.
```

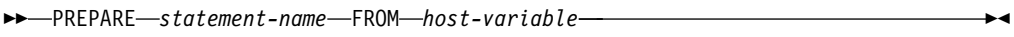
PREPARE

PREPARE ステートメントは、ストリング形式のステートメントから、実行可能な SQL ステートメントを作成します。文字ストリング形式は、ステートメント・ストリング と呼ばれます。実行可能形式は準備済みステートメント と呼ばれます。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。これは、動的に準備できる実行可能ステートメントです。

構文



説明

PREPARE ステートメントには、以下のキーワード・パラメーターが定義されています。

statement-name
準備済みステートメントを指定します。この名前が既存の準備済みステートメントを示す場合、その準備済みステートメントは破棄されます。この名前は、オープン・カーソルの SELECT ステートメントである準備済みステートメントを示すものであってはなりません。

FROM

ステートメント・ストリングを指定します。ステートメント・ストリングは、指定された *host-variable* の値です。

host-variable

アプリケーション・プログラム内でステートメント・ストリングの可変長ストリング変数を宣言する際の規則に従って記述されているホスト変数を示すものでなければなりません。 SQL ステートメントの長さは、32767 を超えてはなりません。

注

ステートメント・ストリングに対する規則*:* 指定された *statement-name* の値をステートメント・ストリング と呼びます。ステートメント・ストリングは、可変長文字ホスト変数を使用して宣言する必要があります。最初の 2 バイトには、SQL ステートメントの長さが含まれなければなりません。 SQL ステートメントの最大長は、32,767 です。例えば、次のようになります。

```
01  STMTSTR.
    49  STMTSTR-LEN  PIC S9(4) COMP VALUE +180.
    49  STMTSTR-TXT  PIC X(180) VALUE SPACES.
```

ステートメント・ストリングは、以下のいずれかの SQL ステートメントでなければなりません。

- DELETE
- INSERT
- SELECT
- UPDATE

ステートメント・ストリングは、次のことをしてはなりません。

- EXEC SQLIMS で始める。
- END-EXEC またはセミコロンで終了する。
- ホスト変数に対する参照を含む。

パラメーター・マーカー*:* ステートメント・ストリングには、ホスト変数に対する参照を入れることはできませんが、パラメーター・マーカー を入れることはできます。パラメーター・マーカーは、準備済みステートメントの実行時に、ホスト変数の値によって置き換えられます。パラメーター・マーカーは疑問符 (?) であり、 ステートメント・ストリングが静的 SQL ステートメントであればホスト変数が現れる位置に、疑問符を入れることができます。パラメーター・マーカーがどのように値に置き換えられるかについては、 886 ページの『EXECUTE』 および 894 ページの『OPEN』を参照してください。

エラーの検査*:* PREPARE ステートメントを実行すると、ステートメント・ストリングが解析され、エラーの検査を受けます。ステートメント・ストリングが無効であれば、準備済みステートメントは作成されず、その作成を妨げることになったエラー状態が SQLIMSCA に報告されます。

参照および実行規則*:* 準備済みステートメントは、次のような種類のステートメントで参照できますが、制約事項があります。

DESCRIBE
制限なし

DECLARE CURSOR
カーソルをオープンするときは、SELECT でなければなりません。

EXECUTE
SELECT であってはなりません。

ステートメント名の有効範囲*:* *statement-name* の有効範囲は、*cursor-name* の有効範囲と同じです。*cursor-name* の有効範囲について詳しくは、 878 ページの『DECLARE CURSOR』を参照してください。

例

例 *1:* PREPARE ステートメント上のホスト変数を使用して、動的 SELECT ステートメントを準備します。 SELECT ステートメントの本文は、SELECT-STATEMENT という名前の変数内にあります。

この例では、ホスト変数 SELECT-STATEMENT 内にあるステートメントの本文は、SELECT HOSPCODE、HOSPNAME、WARDNAME、PATNAME FROM PCB01.HOSPITAL、WARD、および PATIENT です。

```
EXEC SQLIMS
DECLARE C1 CURSOR FOR DYSQL
END-EXEC.

EXEC SQLIMS
PREPARE DYSQL FROM :SELECT-STATEMENT
END-EXEC

EXEC SQLIMS OPEN C1 END-EXEC.

EXEC SQLIMS FETCH C1 INTO :HOSPCODE, :HOSPNAME, :WARDNAME, :PATNAME END-EXEC.

IF SQLIMSCODE = 100
PERFORM DATA-NOT-FOUND
ELSE
PERFORM GET-REST-OF-HOSP
UNTIL SQLIMSCODE IS NOT EQUAL TO ZERO.

EXEC SQLIMS CLOSE C1 END-EXEC.
```

例 *2:* パラメーター・マーカーを使用して動的 INSERT ステートメントを準備し、実行します。

INSERT ステートメントの場合:

```
INSERT INTO PCB01.HOSPITAL HOSPCODE, HOSPNAME VALUES(?,?)
```

以下のステートメントは、パラメーター・マーカーを使用して INSERT ステートメントを準備および実行します。実行に先立ち、パラメーター・マーカーの値が、ホスト変数 S1、S2 に読み込まれます。

```
EXEC SQLIMS
PREPARE DYSQL FROM :INSERT-STATEMENT
END-EXEC

EXEC SQLIMS
EXECUTE USING :S1, :S2
END-EXEC.
```

SELECT

SELECT ステートメントは、1 つ以上の表からデータをリトリーブするために使用します。結果は、表形式の結果セットで返されます。

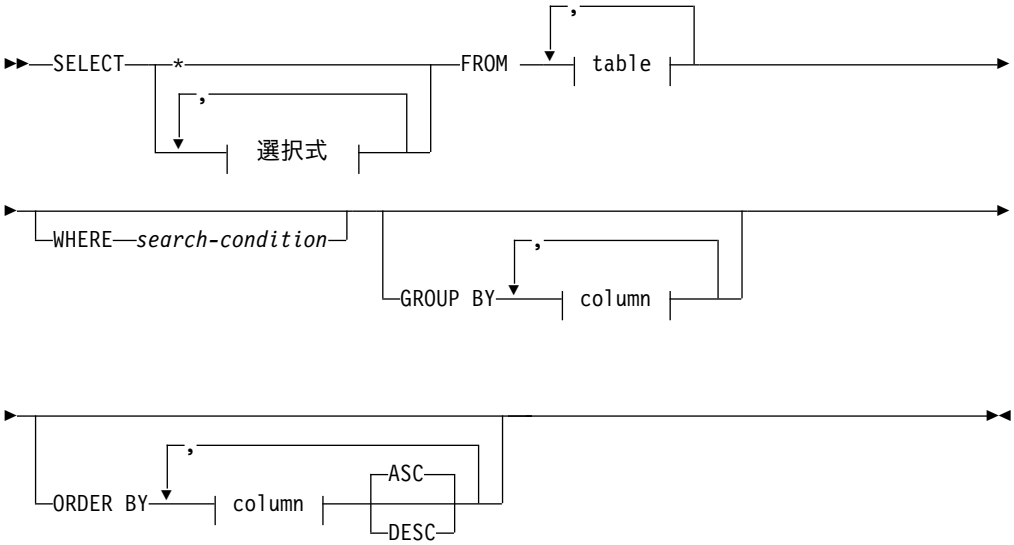
呼び出し

このステートメントは、COBOL または Java アプリケーション・プログラムで使用することができますが、構文は異なります。

COBOL アプリケーション・プログラムの場合、これは実行可能ステートメントですが、動的に準備することはできません。

- COBOL の構文
- Java の構文

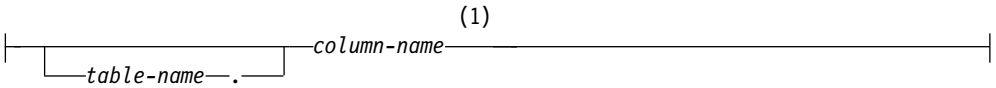
COBOL の構文



選択式 :

テーブル:

列:

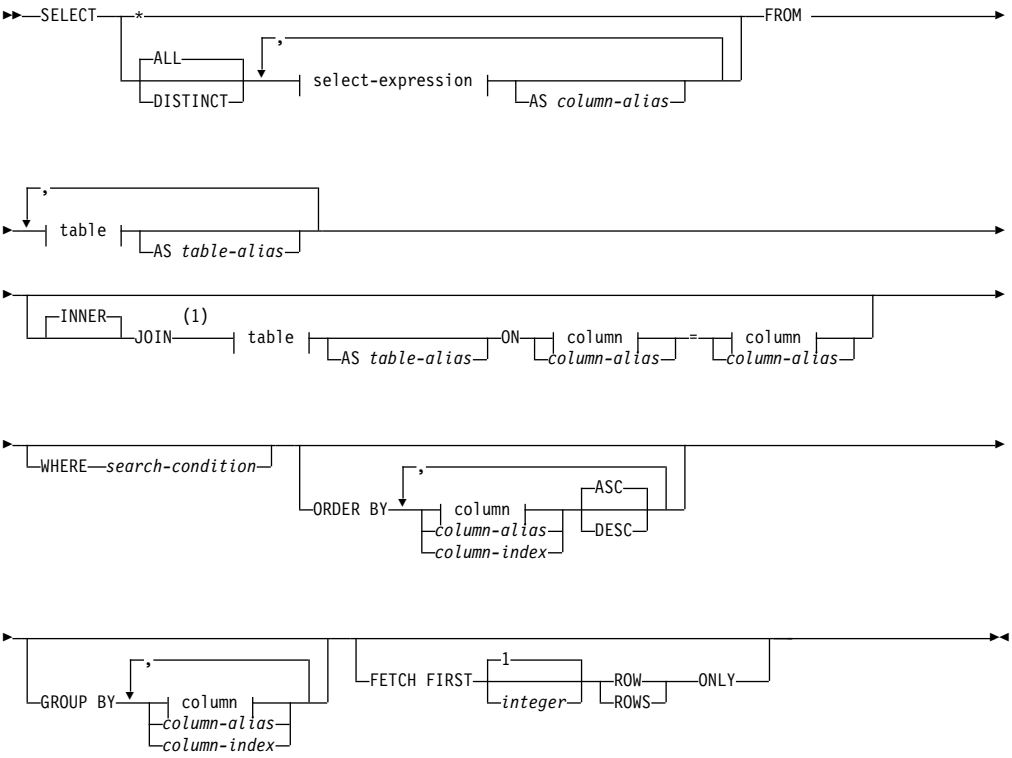


aggregate-function:

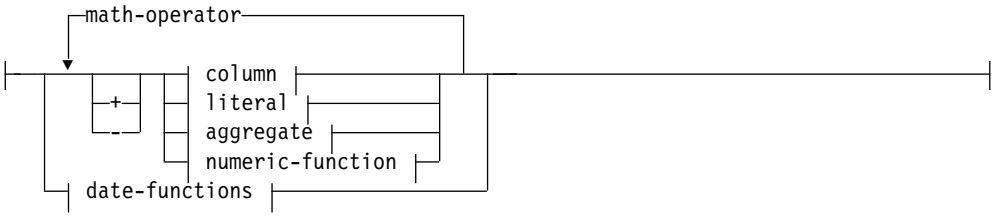
注:

1 複数の表で同じ列名を使用することはできますが、表が修飾されていない場合、列が属している表を判別するためにあいまいさ検査が実行されます。

Java の構文

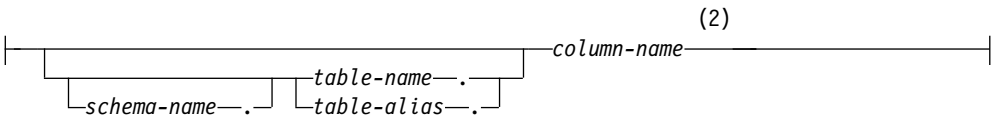


選択式 :



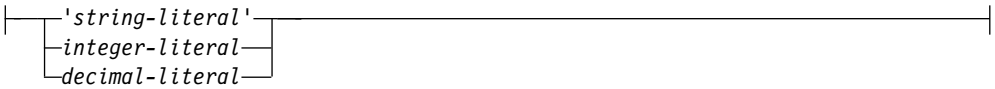
テーブル:

列:

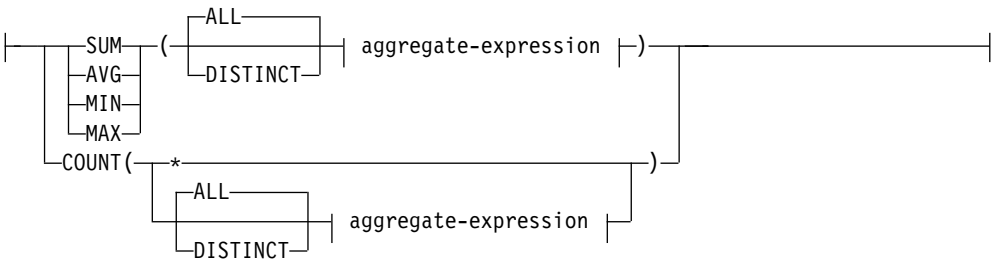


計算演算子 :

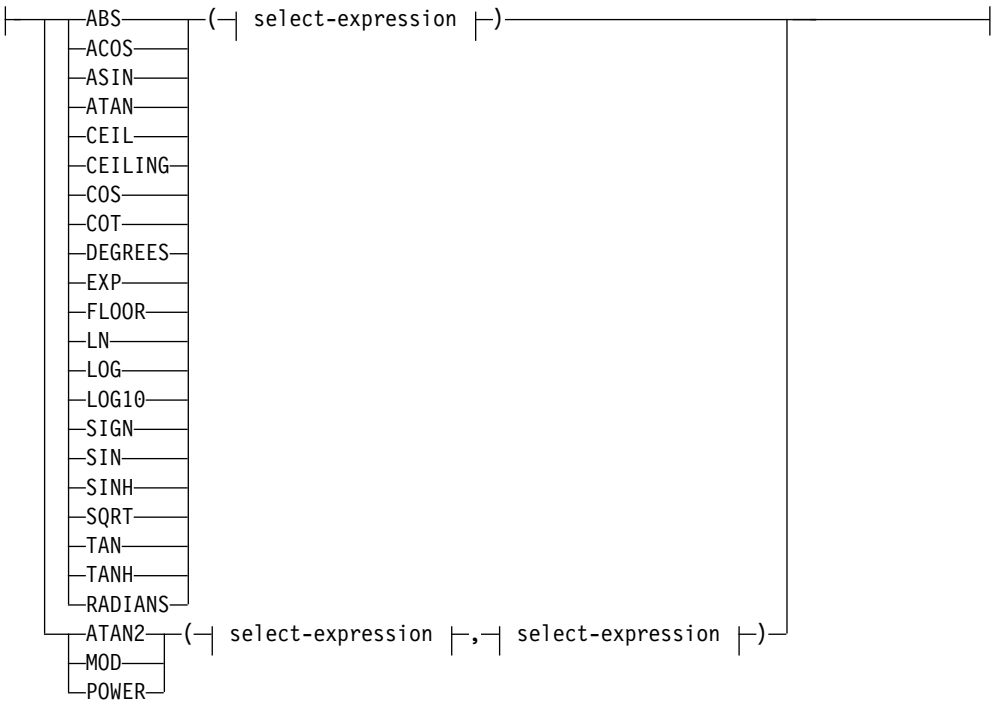
literal:



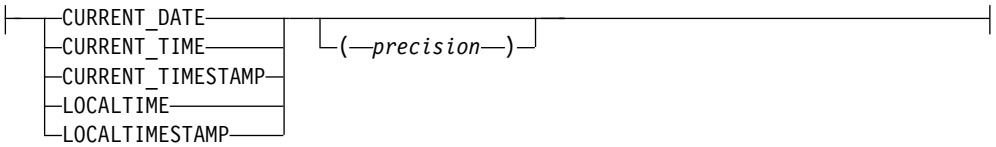
集合:



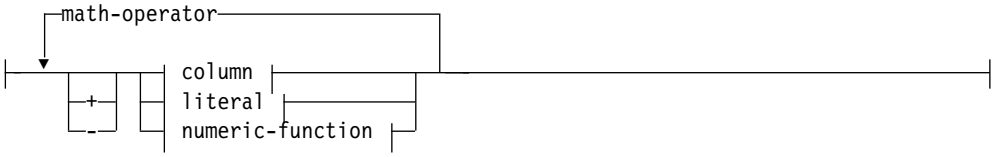
数字関数 :



日時関数 :



集合式 :



注:

1 JOIN 表には参照整合性がなければなりません。これは、親セグメントのキー・フィールドおよび従属セグメントの仮想外部キーによって表されます。表のコンマ区切りリストと JOIN ステートメントの両方を指定することはできません。

2 複数のテーブルに同じ列名を使用できますが、テーブルが修飾されていない場合、各テーブルで列を検索する必要があります。

説明

SELECT ステートメントには、以下のキーワード・パラメーターが定義されています。

ALL
最終結果表のすべての行を保存します。余分な重複行を取り除くことはしません。これはデフォルトです。

DISTINCT
最終結果表の重複行の各セットのうち 1 つを残して、すべての行を取り除きます。 このキーワードは、Java アプリケーション・プログラムにのみ有効です。

2 つの行が互いに重複するのは、最初の行のそれぞれの値が 2 番目の行の対応する値に等しい場合だけです。重複行の判別にあたっては、2 つの NULL 値は等しいものと見なされます。

AS *column-alias*
結果列の名前を付けるか、または名前変更します。名前は固有でなければなりません。AS 節は、COBOL アプリケーション・プログラムでも .NETアプリケーション・プログラムでもサポートされません。

FROM *table-name*
行をリトリーブする元のテーブルを指定します。この名前は IMS 内のセグメントを識別する必要があります。

AS *table-alias*
テーブルの名前を変更します。名前は固有でなければなりません。AS 節は、COBOL アプリケーション・プログラムでも .NET アプリケーション・プログラムでもサポートされません。

INNER JOIN
JOIN
結合演算子を指定しないと、INNER がデフォルトになります。INNER JOIN キーワードは、両方のテーブルに一致している列があれば、両方のテーブルからすべての行を選択します。

WHERE
リトリーブする行を指定します。この文節を省略することも、検索条件を指定することもできます。この文節が省略されると、表のすべての行がリトリーブされます。

search-condition
708 ページの『検索条件』で述べた検索条件です。検索条件内の各 *column-name* は、表の列を示すものでなければなりません。

表の各行に検索条件が適用され、検索条件の結果が真となった行がリトリーブされます。

ORDER BY
ORDER BY 文節は、結果表の行の順序を指定します。

column、*column-alias* または *column-index* で指定した値を使用して、表の結果の行の順序付けを行います。

column-index n は、結果表の *n* 番目の列を示します。

ASC
column、*column-alias* または *column-index* の値を使用して昇順で順序付けます。ASC がデフォルトです。

DESC
column、*column-alias* または *column-index* の値を使用して降順で順序付けます。

GROUP BY
GROUP BY 節は、前の節の結果である中間結果表の行をグループ化することで構成される結果表を指定します。

GROUP BY の結果は、行のグループの集合です。複数の行からなる各グループでは、各 column、*column-alias* または *column-index* のすべての値が等しく、column、column-alias、または column-index の値のセットが同じであるすべての行が同じグループに属します。グループ化を行う場合、各 column、*column-alias*、または *column-index* のすべての NULL 値が等しいと見なされます。

SELECT ステートメントに集合選択式と非集合選択式の両方が含まれている場合、すべての非集合選択式を GROUP BY 節に入れる必要があります。

schema-name
schema-name は、SQL 照会のスキーマを定義します。 IMS では、スキーマ名は PCB 名です。

table-name
table-name は、SQL 照会のテーブルの名前を定義します。この名前は IMS 内のセグメントを識別する必要があります。

table-alias
table-alias は、SQL 照会で定義されている別名を定義し、*table-name* の代わりに使用することができます。

column-name
column-name は、SQL 照会の列の名前を定義します。

column-name
column-name は、SQL 照会の列の名前を定義します。

'string-literal'
'*string-literal*' は、UTF-8 でエンコードされた静的文字ストリングを定義します。

integer-literal
integer-literal は、–2,147,483,648 から 2,147,483,647 の範囲内で整数値を定義します。

decimal-literal
decimal-literal は、倍精度の 10 進数値を定義します。

SUM
SUM 関数は、1 組の数値の合計を戻します。

AVG
AVG 関数は、1 組の数値の平均を戻します。

MIN
MIN 関数は、1 組の値の中の最小値を戻します。

MAX
MAX 関数は、1 組の値内の最大値を戻します。

COUNT
COUNT 関数は、1 組の行または値の中の行数または値の数を戻します。

数値関数:

ABS
ABS 関数は、数値の絶対値を戻します。

ACOS
ACOS 関数は、ラジアンで表した角度としての引数の逆余弦を戻します。ACOS と COS 関数は逆の演算を行います。

ASIN
ASIN 関数は、ラジアンで表した角度としての引数の逆正弦を戻します。ASIN と SIN 関数は逆の演算を行います。

ATAN
ATAN 関数は、ラジアンで表した角度としての引数の逆正接を戻します。ATAN と TAN 関数は逆の演算を行います。

CEIL
CEILING
CEILING 関数は、引数以上の最小の整数を戻します。

COS
COS 関数は、ラジアンで表した角度としての引数の余弦を戻します。COS と ACOS 関数は逆の演算を行います。

COT
COT 関数は、ラジアンで表した角度としての引数の余接を戻します。COT と TAN 関数は相互に逆の演算を行います。

DEGREES
DEGREES 関数は、ラジアンで表した角度としての引数の度数を戻します。

EXP
EXP 関数は、自然対数 (e) の基数を引数によって指定された数で累乗した値を戻します。EXP と LN 関数は逆の演算を行います。

FLOOR
FLOOR 関数は、引数以下の最大の整数を戻します。

LN
LOG
LN 関数および LOG 関数は、引数の自然対数を戻します。LN と EXP 関数は逆の操作を行います。

LOG10
LOG10 関数は、数値の常用対数 (底 10) を戻します。

SIGN
SIGN 関数は、引数の符号の標識を戻します。

SIN
SIN 関数は、ラジアンで表した角度としての引数の正弦を戻します。

SINH
SINH 関数は、ラジアンで表した角度としての引数の双曲線正弦を戻します。

SQRT
SQRT 関数は引数の平方根を戻します。

TAN
TAN 関数は、ラジアンで表した角度としての引数の正接を戻します。

TANH
TANH 関数は、ラジアンで表した角度としての引数の双曲線正接を戻します。

RADIANS
RADIANS 関数は、度数で表された引数のラジアン数を戻します。

ATAN2
ATAN2 関数はラジアンで表した角度としての x と y 座標の逆正接を戻します。

MOD
MOD 関数は、最初の引数を 2 番目の引数で除算し、剰余を戻します。

POWER
POWER 関数は、最初の引数を 2 番目の引数で累乗した値を戻します。

CURRENT_DATE
CURRENT_DATE 特殊レジスターは、SQL ステートメントがアプリケーションで実行されるときの時刻機構の読み取り時点に基づいた日付を指定します。

CURRENT_TIME
CURRENT_TIME 特殊レジスターは、SQL ステートメントがアプリケーションで実行されるときの時刻機構の読み取り時点に基づいた時刻を指定します。

precision
precision は、秒の小数部を指定します。 *precision* の範囲は 0 から 12 までです。デフォルトの精度 (precision) は 3 です。

CURRENT_TIMESTAMP
CURRENT_TIMESTAMP 特殊レジスターは、SQL ステートメントがアプリケーションで実行されるときの時刻機構の読み取り時点に基づいたタイム・スタンプを指定します。

precision
precision は、秒の小数部を指定します。 *precision* の範囲は 0 から 12 までです。デフォルトの精度 (precision) は 6 です。

LOCALTIME
LOCALTIME 特殊レジスターは、SQL ステートメントがアプリケーションで実行されるときの時刻機構の読み取り時点に基づいた時間を指定します。

precision
precision は、秒の小数部を指定します。 *precision* の範囲は 0 から 12 までです。デフォルトの精度 (precision) は 3 です。

LOCALTIMESTAMP

LOCALTIMESTAMP 特殊レジスターは、SQL ステートメントがアプリケーションで実行されるときの時刻機構の読み取り時点に基づいたタイム・スタンプを指定します。

precision

precision は、秒の小数部を指定します。 *precision* の範囲は 0 から 12 までです。デフォルトの精度 (precision) は 6 です。

注記

- 複数の表から選択を実行する場合に、これらの表の 1 つ以上に同じ列名が存在する場合は、列を表名で修飾する必要があります。そうしないと、あいまいさのエラーが発生します。
- FROM 節では、データを選択するすべての表をリストする必要があります。FROM 節でリストされた表は、IMS データベースの同じ階層パスに存在している必要があります。
- PSB 内には複数のデータベース PCB があるため、照会では PSB 内のどの PCB を使用するかを指定する必要があります。使用する PCB を指定するには、SQL ステートメントの FROM 節で参照されるセグメントを、セグメント名の前に PCB 名を付けて必ず修飾します。 PSB に 1 つしか PCB が含まれていない場合にのみ、PCB 名を省略することができます。

例

* 記号を指定してすべてのフィールドを選択

次のステートメントは、PATIENT 表のすべてのフィールドをリトリーブします。

```
SELECT *
FROM PCB01.PATIENT
```

次のステートメントは、HOSPITAL 表から病院名、および WARD 表からすべてのフィールドをリトリーブします。

```
SELECT HOSPITAL.HOSPNAME, WARD.*
FROM PCB01.HOSPITAL, PCB01.WARD
```

指定した列を選択

次のステートメントは、WARD 表と PATIENT 表から、それぞれ病棟名と患者名をリトリーブします。

```
SELECT WARD.WARDNAME,PATIENT.PATNAME
FROM PCB01.WARD, PATIENT
```

ORDER BY を指定して選択

ORDER BY 節は、行をソートするために使用します。デフォルトでは、結果は昇順の番号順またはアルファベット順でソートされます。次のステートメントは、すべての個別の病院名をアルファベット順でソートしてリトリーブします。

```
SELECT DISTINCT HOSPNAME FROM PCB01.HOSPITAL
   ORDER BY HOSPNAME
```

次のステートメントは、すべての病棟名をアルファベット順でソートし、各病棟の患者番号を昇順の番号順でソートしてリトリーブします。 2 つの WARDNAME 値が ORDER BY 比較で同じである場合、対応する

PATCOUNT 値によって同等比較を行います (この場合は、対応する PATCOUNT 値が小さい数値である行が先に表示されます)。

```
SELECT WARDNAME, PATCOUNT FROM PCB01.WARD
   ORDER BY WARDNAME, PATCOUNT
```

照会結果を降順の番号順または逆のアルファベット順でソートする場合は、DESC 修飾子を使用します。次のステートメントは、すべての患者名を逆のアルファベット順でリトリーブします。

```
SELECT PATNAME FROM PCB01.PATIENT
   ORDER BY PATNAME DESC
```

照会結果を昇順の番号順または逆のアルファベット順で明示的にソートする場合は、ASC 修飾子を使用します。次のステートメントは、すべての病棟名を昇順のアルファベット順でソートし、各病棟の患者番号を降順の番号順でソートして取り出します。

```
SELECT WARDNAME, PATCOUNT FROM PCB01.WARD
   ORDER BY WARDNAME ASC, PATCOUNT DESC
```

GROUP BY を指定して選択

GROUP BY 節は、列値によってグループ化される、集約関数の結果セットを返すために使用します。次のステートメントは、個別の病棟名によってグループ化された、病院内の各病棟のすべての医師の集合の合計を返します。

```
SELECT WARDNAME, SUM(DOCCOUNT)
FROM PCB01.WARD
WHERE HOSPITAL_HOSPCODE = 'H5140070000H
   GROUP BY WARDNAME
```

次のステートメントは、病院名、病棟名、および各病院の各病棟のすべての患者数を、個別の病院名でグループ化し、病棟名でサブグループ化して返します。

```
SELECT HOSPNAME, WARDNAME, COUNT(PATNAME)
FROM PCB01.HOSPITAL, WARD, PATIENT
   GROUP BY HOSPNAME, WARDNAME
```

DISTINCT を指定して選択

IMS Universal JDBC ドライバー用の SQL の場合、DISTINCT キーワードがサポートされます。次のステートメントは、SQL の PATIENT 表からすべての個別の患者名を取得します。

```
SELECT DISTINCT PATNAME
FROM PCB01.PATIENT
```

AS 節の使用

IMS Universal JDBC ドライバー用の SQL の場合、結果セットの集約関数列または SELECT ステートメントの他のフィールドを名前変更するには、AS 節を使用します。 COBOL アプリケーション・プログラム用の SQL では、AS 節はサポートされません。

IMS Universal JDBC ドライバー用の SQL の場合、DISTINCT キーワードがサポートされます。次のステートメントは、PATIENT 表の個別の患者の集約数に、別名「PATIENTCOUNT」を付けて返します。

```
SELECT COUNT(DISTINCT PATNAME)
   AS PATIENTCOUNT
FROM PCB01.PATIENT
```

次のステートメントは、すべての病院の個別の病棟の集約数に別名「WARDCOUNT」を付け、病院名をアルファベット順でソートし、さらに個別の病院名 (名前変更した列の別名「HOSPITALNAME」) でグループ化して返します。

```
SELECT HOSPNAME AS HOSPITALNAME, COUNT(DISTINCT WARDNAME)
   AS WARDCOUNT
FROM PCB01.HOSPITAL, WARD
   GROUP BY HOSPNAME
   ORDER BY HOSPNAME
```

パラメーター・マーカーを使用する **SELECT** の例:

以下のステートメントは、HOSPNAME のパラメーターに対して提供された値に基づいてデータをリトリーブします。

```
SELECT * FROM PCB01.HOSPITAL WHERE HOSPNAME = ?
```

FETCH FIRST 節の使用例:

以下のステートメントは、返された行のうち最初の *n* 行をフェッチします。

```
SELECT HOSPNAME FROM PCB01.HOSPITAL FETCH FIRST 3 ROWS ONLY
```

無効な **SELECT** 照会の例:

次のステートメントは、FROM 節に WARD 表が欠落しているため無効です。

```
SELECT WARD.WARDNAME,PATIENT.PATNAME
FROM PCB01.PATIENT
```

関連資料:

IMS JDBC ドライバーでサポートされている SQL 集約関数 (アプリケーション・プログラミング)

COBOL でのサポートされている SQL 集約関数 (アプリケーション・プログラミング)

UPDATE

UPDATE ステートメントは、表の行の中の指定した列の値を更新します。

検索 UPDATE 形式は、検索条件により任意に決定される 1 つ以上の行を更新します。

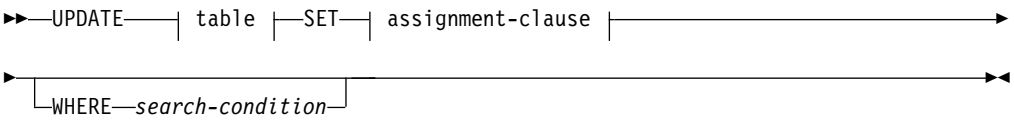
呼び出し

このステートメントは、COBOL あるいは Java アプリケーション・プログラムに組み込むことも、対話式に発行することもできます。 UPDATE は、アプリケーション・プログラムに組み込むことができます。これは、動的に準備できる実行可能ステートメントです。

- 910 ページの『COBOL の構文』
- 910 ページの『Java の構文』

COBOL の構文

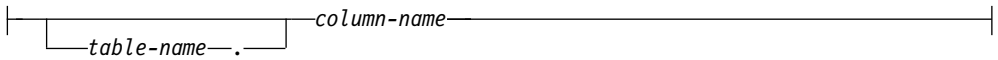
更新



テーブル:

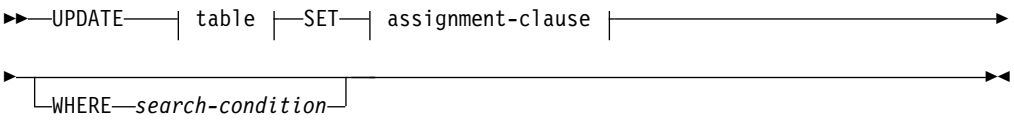
assignment clause:

列:

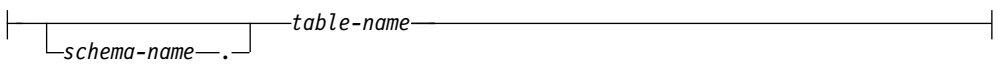


Java の構文

更新

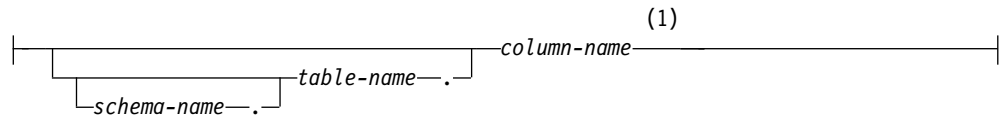


テーブル:



assignment clause:

列:



注:

1 複数のテーブルに同じ列名を使用できますが、テーブルが修飾されていない場合、各テーブルで列を検索する必要があります。

説明

UPDATE ステートメントには、以下のキーワード・パラメーターが定義されています。

UPDATE
UPDATE ステートメントのオブジェクトを指定します。

table-name
table-name は、SQL 照会のテーブルの名前を定義します。この名前は IMS 内のセグメントを識別する必要があります。

schema-name
schema-name は、SQL 照会のスキーマを定義します。 IMS では、スキーマ名は PCB 名です。

SET
列名への値の割り当てを設定します。

column-name
更新する列を指定します。*column-name* は、指定されたセグメントのフィールドを示す必要があります。

WHERE
更新する行を指定します。この文節を省略することも、検索条件を指定することもできます。この文節が省略されると、表のすべての行が更新されます。

search-condition
708 ページの『検索条件』で述べた検索条件です。検索条件内の各 *column-name* は、表の列を示すものでなければなりません。

表の各行に検索条件が適用され、検索条件の結果が真となった行が更新されます。

value
列の新しい値を示します。

schema-name
schema-name は、SQL 照会のスキーマを定義します。 IMS では、スキーマ名は PCB 名です。

table-name
table-name は、SQL 照会のテーブルの名前を定義します。

column-name
column-name は、SQL 照会の列の名前を定義します。

注

更新規則**:**

更新値は、以下の規則を満たしている必要があります。以下の規則に従わない場合、または UPDATE ステートメントの実行中にその他のエラーが発生した場合には、行は更新されず、カーソルの位置は変更されません。

- 割り当て。更新値は、 691 ページの『言語エレメント』で述べた割り当て規則を使用して列に割り当てられます。
- 非ルート・レベルで表内のレコードを更新する場合、更新する正確なレコード (またはセグメント・インスタンス) を識別するために、表のすべての外部キー・フィールドの値を指定する必要があります。
- 外部キー・フィールドでの UPDATE の実行は無効です。

更新される行数**:**

COBOL の場合、SQLIMSCA 内の SQLIMSERRD(3) の値は、UPDATE ステートメントの実行が完了した後に更新された行数です。ここで述べたことの例外など、SQLIMSCA について詳しくは、 914 ページの『SQL 連絡域 (SQLIMSCA)』を参照してください。

例

レコードの **1** つの列を更新

次のステートメントは、ルートを更新します。

```
UPDATE HOSPITAL SET HOSPNAME = 'MISSION CREEK'
WHERE HOSPITAL.HOSPCODE = 'H001007'
```

階層パスの指定されたレコードの複数のフィールドを更新

外部キーを使用し、更新するレコード (またはセグメント・インスタンス) を明確に識別することで、参照整合性が維持されます。次のステートメントは、特定の HOSPITAL の下の WARD レコードを更新します。この例では、WARD 表には仮想外部キー HOSPITAL_HOSPCODE があります。レコードは、HOSPITAL 表に、値が 'H5140070000H' の HOSPCODE が存在する場合にのみ更新されます。

```
UPDATE WARD SET WARDNAME = 'EMGY',
   DOCCOUNT = '2', NURCOUNT = '4'
WHERE HOSPITAL_HOSPCODE = 'H5140070000H'
   AND WARDNO = '01'
```

無効な **UPDATE** 照会の例

仮想外部キー・フィールド (HOSPITAL_HOSPCODE) に有効な値を指定する正しい構文を使用していないため、このステートメントは無効です。

```
UPDATE WARD SET WARDNAME = 'EMGY',
   DOCCOUNT = '2', NURCOUNT = '4'
WHERE HOSPITAL.HOSPCODE = 'H5140070000H'
   AND WARDNO = '01'
```

無効な外部キー・フィールド **UPDATE** 照会の例

外部キー・フィールドに対して UPDATE 照会を行うことは無効です。例えば、次の UPDATE 照会は失敗します。

```
UPDATE WARD SET WARDNAME = 'EMGY',
   HOSPITAL_HOSPCODE = 'H5140070000H'
WHERE WARDNO = '01'
```

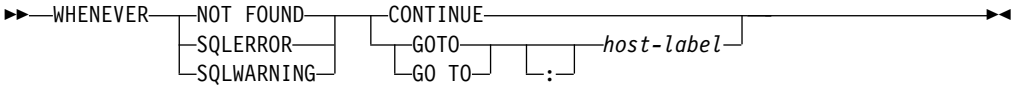
WHENEVER

WHENEVER ステートメントは、指定された例外条件が発生した場合に実行される、ホスト言語ステートメントを指定します。

呼び出し

このステートメントは COBOL アプリケーション・プログラムにのみ組み込むことができます。 これは、実行可能ステートメントではありません。

構文



説明

WHENEVER ステートメントには、以下のキーワード・パラメーターが定義されています。

NOT FOUND、**SQLERROR**、または **SQLWARNING** 文節を使って、例外条件のタイプを指定します。

NOT FOUND
SQLIMSCODE が +100 (SQLIMSSTATE コードの「02000」に相当) になる条件を指定します。

SQLERROR
SQLIMSCODE が負になる条件を指定します。

SQLWARNING
警告条件 (SQLIMSWARN0 が W) になる条件、または SQLIMSCODE が +100 以外の正数になる条件を指定します。

CONTINUE 文節または GO TO 文節は、指定したタイプの例外条件が存在する場合に次に実行するステートメントを指定します。

CONTINUE
ソース・プログラムの次の順次ステートメントを指定します。

GOTO または **GO TO** *host-label*
ホスト・ラベル によって識別されるステートメントを指定します。 *host-label* は単一のトークンで置き換えます。オプションとして、トークンの前にはコロンを付けることができます。トークンの形式はホスト言語によって異なります。例えば、COBOL では、セクション名 か、または非修飾の段落名 となります。

注

WHENEVER ステートメントには、以下の 3 つのタイプがあります。

- WHENEVER NOT FOUND
- WHENEVER SQLERROR
- WHENEVER SQLWARNING

アプリケーション・プログラム内の実行可能な SQL ステートメントはすべて、各タイプの暗黙または明示の WHENEVER ステートメントの有効範囲内にあります。WHENEVER ステートメントの有効範囲は、アプリケーション・プログラム内のステートメントの実行順序ではなく、それらのステートメントがリストされている順序に関連します。

SQL ステートメントは、ソース・プログラム内でその SQL ステートメントの前に指定されている各タイプごとの最後の WHENEVER ステートメントの有効範囲内にあります。あるタイプの WHENEVER ステートメントが SQL ステートメントの前に指定されていない場合、その SQL ステートメントは、そのタイプの暗黙の WHENEVER ステートメントの有効範囲内にあります。すなわち、CONTINUE が指定されていることになります。

例

以下のステートメントは、COBOL プログラムに組み込むことができます。

例 *1:* エラーが生じたステートメントの場合は、ラベル HANDLER へ進みます。

```
EXEC SQLIMS WHENEVER SQLERROR GOTO HANDLER END-EXEC.
```

例 *2:* 警告が生じたステートメントの場合は、処理を継続します。

```
EXEC SQLIMS WHENEVER SQLWARNING CONTINUE END-EXEC.
```

例 *3:* 結果が戻らないステートメントの場合は、ラベル ENDDATA へ進みます。

```
EXEC SQLIMS WHENEVER NOT FOUND GO TO ENDDATA END-EXEC.
```

SQL 連絡域 (SQLIMSCA)

SQLIMSCA は、各 SQL ステートメントの実行後に更新される変数の構造またはコレクションです。実行可能 SQL ステートメントを含むアプリケーション・プログラムでは、SQLIMSCA を必ず 1 つ用意しなければなりません。

COBOL では、INCLUDE ステートメントを使用して SQLIMSCA の宣言を行うことができます。

SQLIMSCA のフィールドの説明

SQLIMS INCLUDE ステートメントには SQLIMSCA フィールドがあります。

下表の名前は、SQLIMS INCLUDE ステートメントによって提供される名前です。

表 *168. SQLIMSCA* のフィールド

COBOL の名前	データ・タイプ	目的
SQLIMSCAID	CHAR(8)	ストレージ・ダンプの「目印」。テキスト「SQLIMSCA」を含む。
SQLIMSCABC	INTEGER	SQLIMSCA の長さ 224 が入る。

表 *168. SQLIMSCA* のフィールド (続き)

COBOL の名前	データ・タイプ	目的
SQLIMSCODE	INTEGER	SQL 戻りコードが入る。 (注 2 (916 ページ) を参照) コード 意味 **0** 正常な実行 (ただし、警告メッセージが出ている可能性もある)。 正の値 正常に実行されたが、警告条件または他の情報が出される。 負の値 エラー状態。
SQLIMSERRML (注 1 (916 ページ) を参照)	SMALLINT	SQLIMSERRMC の長さ標識 (0 から 255 まで)。0 は SQLIMSERRMC の値が妥当でないことを意味する。
SQLIMSERRMC (注 1 (916 ページ) を参照)	VARCHAR(158)	エラー・メッセージが入ります。
SQLIMSERRP	CHAR(8)	プロダクト・シグニチャー、およびエラーの場合は、エラーを検出したモジュール名などの診断情報を提供する。いずれの場合も、IMS では最初の 3 文字は「DQF」または「DFS」である。
SQLIMSERRD(1)	INTEGER	予約済み。
SQLIMSERRD(2)	INTEGER	予約済み。
SQLIMSERRD(3)	INTEGER	DELETE ステートメント、INSERT ステートメント、または UPDATE ステートメントの後で、削除、挿入、更新、またはマージの対象となった行数が入る。
SQLIMSERRD(4)	INTEGER	予約済み。
SQLIMSERRD(5)	INTEGER	予約済み。
SQLIMSERRD(6)	INTEGER	予約済み。
SQLIMSWARN0	CHAR(1)	警告状態に他の標識が設定されていない (つまり、他の標識に W または Z が含まれていない) 場合、ブランクが入る。少なくとも他に 1 つの標識に W または Z が含まれている場合には、W が入る。
SQLIMSWARN1	CHAR(1)	ホスト変数へストリング列を割り当てるときにその値が切り捨てられた場合に、W が入る。
SQLIMSWARN2	CHAR(1)	予約済み。
SQLIMSWARN3	CHAR(1)	結果列の数がホスト変数の数より多い場合に、W が入る。
SQLIMSWARN4	CHAR(1)	準備済みの UPDATE ステートメントまたは DELETE ステートメントに WHERE 文節が含まれていない場合に、W が入る。
SQLIMSWARN5	CHAR(1)	SQL ステートメントが有効な IMS SQL ステートメントでないために実行されなかった場合に、W が入る。

表 *168. SQLIMSCA* のフィールド *(*続き*)*

COBOL の名前	データ・タイプ	目的
SQLIMSWARN6	CHAR(1)	フィールドは異なるタイプの別のフィールドでオーバーレイされるため、そのフィールドが INSERT ステートメントに適切なフォーマットで初期化されない場合、W が入る。ZONEDDECIMAL フィールドと PACKEDDECIMAL フィールドは、INSERT ステートメントの処理中に初期化される。フィールドが別のフィールドによってオーバーレイされ、フィールドを初期化できない場合、EXECUTE 呼び出し中にステートメントに W が設定される。
SQLIMSWARN7	CHAR(1)	予約済み。
SQLIMSWARN8	CHAR(1)	予約済み。
SQLIMSWARN9	CHAR(1)	予約済み。
SQLIMSWARNA	CHAR(1)	予約済み。
SQLIMSSTATE	CHAR(5)	最も新しく実行された SQL ステートメントの結果を表す戻りコードが入る (注 2 を参照)。

注**:**

1. COBOL の場合は、SQLIMSERRM に SQLIMSERRML および SQLIMSERRMC が含まれます。 『組み込まれる SQLIMSCA』 にある各種ホスト言語の例を参照してください。
2. SQLIMSSTATE 値の説明については、SQL コード (メッセージおよびコード)を参照してください。

組み込まれる SQLIMSCA

INCLUDE SQLIMSCA によって提供される SQLIMSCA の記述を、COBOL について示しています。

COBOL:

```
01 SQLIMSCA GLOBAL.
    05 SQLIMSCAID PIC X(8).
    05 SQLIMSCABC PIC S9(9) COMP-5.
    05 SQLIMSCODE PIC S9(9) COMP-5.
    05 SQLIMSERRM.
      49 SQLIMSERRML PIC S9(4) COMP-5.
      49 SQLIMSERRMC PIC X(158).
    05 SQLIMSERRP PIC X(8).
    05 SQLIMSERRD PIC S9(9) COMP-5
    05 SQLIMSWARN.
      10 SQLIMSWARN0 PIC X.
      10 SQLIMSWARN1 PIC X.
      10 SQLIMSWARN2 PIC X.
      10 SQLIMSWARN3 PIC X.
      10 SQLIMSWARN4 PIC X.
      10 SQLIMSWARN5 PIC X.
      10 SQLIMSWARN6 PIC X.
      10 SQLIMSWARN7 PIC X.
    05 SQLIMSEXT.
      10 SQLIMSWARN8 PIC X.
      10 SQLIMSWARN9 PIC X.
      10 SQLIMSWARNA PIC X.
      10 SQLIMSSTATE PIC X(5).
```

SQL 記述子域 (SQLIMSDA)

SQLIMSDA は、変数の集合であり、SQLIMS DESCRIBE ステートメントの実行には必須ですが、FETCHステートメントではオプションとして使用できます。1 つの SQLIMSDA を DESCRIBE ステートメントで使用し、ホスト変数のアドレスで変更し、その後 FETCH ステートメントで再利用することができます。

SQLIMSDA 内の情報の意味は、使用されるコンテキストによって異なります。DESCRIBE の場合、IMS は SQLIMSDA 内のフィールドを設定し、アプリケーション・プログラムへの情報を提供します。FETCH では、アプリケーション・プログラムが SQLIMSDA 内のフィールドを設定し、IMS に情報を提供します。

DESCRIBE ***statement-name***

SQLIMSN は例外ですが、IMS は SQLIMSDA のフィールドを設定し、アプリケーション・プログラムに準備済みステートメントの情報を提供します。各 SQLIMSVAR オカレンスにより、結果表の列を記述します。

FETCH

アプリケーション・プログラムは SQLIMSDA のフィールドを設定し、ホスト変数またはアプリケーション・プログラム内の出力バッファーの情報を IMS に提供します。各 SQLIMSVAR オカレンスにより、ホスト変数または出力バッファーを記述します。

- FETCH の場合は、各 SQLIMSVAR によりホスト変数または、結果の行からの出力値を入れるために使用されるアプリケーション・プログラム内のバッファーを記述します。

SQLIMSDA フィールドの説明

SQLIMSDA は、4 つの変数、ヘッダー、および集合的に SQLIMSVAR と呼ばれる一連の変数のオカレンスを任意の数だけ続けた形で構成されます。

DESCRIBE では、SQLIMSVAR の各オカレンスは表の列を記述します。FETCH では、各オカレンスはホスト変数を記述します。

SQLIMSDA ヘッダー

SQLIMSDA ヘッダー内のフィールドの使用法は、SQLIMSDA が DESCRIBE ステートメントまたは FETCHステートメントで使用されているのかによって異なります。

次の表は、SQLIMSDA ヘッダーのフィールドについて説明しています。

表 *169. SQLIMSDA* ヘッダーのフィールド

COBOL の名前	データ・タイプ	**DESCRIBE** での用途	**FETCH** での用途
sqlimsdaid SQLIMSDAID	CHAR(8)	ストレージ・ダンプの「目印」。テキスト「SQLIMSDA」を含む。	SQLIMSDAID は使用されない。
sqlimsdabc SQLIMSDABC	INTEGER	SQLIMSDA の長さ、`SQLIMSN`*x* `* 44+16` に等しい。	SQLIMSDA の長さ、`SQLIMSN`*x* `* 44+16` 以上。

表 *169. SQLIMSDA* ヘッダーのフィールド *(*続き*)*

COBOL の名前	データ・タイプ	**DESCRIBE** での用途	**FETCH** での用途
sqlimsn SQLIMSN	SMALLINT	このフィールドは、ステートメントの実行前にゼロ以上の値に設定しなければならない。このフィールドは、SQLIMSVAR のオカレンスの総数を示す。 COBOL では、組み込み SQLIMSDA には、SQLIMSVAR のオカレンスが最大 750 含まれる。	SQLIMSDA で提供された SQLIMSVAR のオカレンスの総数。SQLIMSN は、ゼロ以上の値に設定しなければならない。COBOL では、組み込み SQLIMSDA には、SQLIMSVAR のオカレンスが最大 750 含まれる。
sqlimsd SQLIMSD	SMALLINT	SQLIMSVAR のオカレンスによって記述される列の数。	SQLIMSVAR のオカレンスによって記述されるホスト変数の数。

SQLIMSVAR 項目

SQLIMSDA によって記述される各列または各ホスト変数の場合、SQLIMSVAR 項目を使用して記述されます。

この項目のフィールドには、データ・タイプ・コード、長さ属性 (LOB 以外)、列名、ホスト変数アドレス、および標識変数アドレスなどの、列またはホスト変数に関する基本情報が入っています。

必要な **SQLIMSVAR** オカレンスの数の判別**:**

必要な SQLIMSVAR オカレンスの数は、提供された SQLIMSDA の対象であるステートメント、および記述される列またはパラメーターのデータ・タイプによって異なります。

組み込み SQLIMSDA は、SQLIMSVAR のオカレンスを最大 750 提供します。SQLIMSD は、結果の列の数に設定され、必要な SQLIMSVAR オカレンスの数を表します。提供された SQLIMSVAR オカレンスの数が十分ではない場合、IMS は SQLIMSCODE で +239 の警告を返します。

SQLIMSD は結果の列の数に設定されます。

基本 **SQLIMSVAR** のオカレンス内のフィールドの説明**:**

基本 SQLIMSVAR のフィールドの使用は、SQL ステートメントによって異なります。

下表は、基本 SQLIMSVAR のフィールドの内容を説明しています。

表 *170.* 基本 *SQLIMSVAR* のオカレンス内のフィールド

COBOL の名前	データ・タイプ	**DESCRIBE** での使用法	**FETCH** での使用法
sqlimstype SQLIMSTYPE	SMALLINT	列のデータ・タイプ、およびそれに NULL 値を入れられるかどうかを示す。タイプ・コードについては、919 ページの表 171を参照。	ホスト変数のデータ・タイプ、および標識変数が与えられているかどうかを示す。タイプ・コードについては、919 ページの『SQLIMSTYPE および SQLIMSLEN』を参照。

表 170. 基本 SQLIMSVAR のオカレンス内のフィールド (続き)

COBOL の名前	データ・タイプ	DESCRIBE での使用法	FETCH での使用法
sqlimslen SQLIMSLEN	SMALLINT	列の長さ属性。使える値については、『SQLIMSTYPE および SQLIMSLEN』を参照。	ホスト変数の長さ属性。使える値については、『SQLIMSTYPE および SQLIMSLEN』を参照。
sqlimsdata SQLIMSDATA	ポインター	予約済み。	ホスト変数のアドレスが入る。
sqlimsind SQLIMSIND	ポインター	予約済み	SQLIMSTYPE が奇数の場合、関連する標識変数のアドレスが入る。それ以外の場合、このフィールドは使用されない。
sqlimsname SQLIMSNAME	VARCHAR(30)	列の非修飾名またはラベルが入る。名前またはラベルが存在しない場合には、長さ 0 のストリングが入る。名前が 30 バイトを超える場合、名前はバイト境界で切り捨てられる。	列の非修飾名またはラベルが入る。名前またはラベルが存在しない場合には、長さ 0 のストリングが入る。名前が 30 バイトを超える場合、名前はバイト境界で切り捨てられる。

SQLIMSTYPE および SQLIMSLEN:

SQLDA の SQLIMSTYPE フィールドと SQLIMSLEN フィールドの内容は、値を戻している SQL ステートメントにより異なります。

以下の表は、SQLIMSDA の SQLIMSTYPE フィールドおよび SQLIMSLEN フィールドに入る値を示したものです。DESCRIBE では、偶数値の SQLIMSTYPE は NULL にできない列であることを意味します。また、奇数値は NULL にできる列であることを意味します。FETCH では、偶数値の SQLIMSTYPE は、標識変数が与えられていないことを意味します。また、奇数値は、SQLIMSIND に標識変数のアドレスが入っていることを意味します。

表 171. DESCRIBE、FETCH、OPEN、および EXECUTE の場合の SQLIMSTYPE と SQLIMSLEN 値

	DESCRIBE の場合		FETCH の場合	
SQLIMSTYPE	列またはパラメーターのデータ・タイプ	SQLIMSLEN	ホスト変数のデータ・タイプ	SQLIMSLEN
384/385	日付	10 [1]	日付の固定長文字ストリング表現	ホスト変数の長さ属性
388/389	time	8 [2]	時刻の固定長文字ストリング表現	ホスト変数の長さ属性
392/393	タイム・スタンプ	26	タイム・スタンプの固定長文字ストリング表現	ホスト変数の長さ属性
452/453	固定長文字ストリング	列の長さ属性	固定長文字ストリング	ホスト変数の長さ属性
480/481	浮動小数点	単精度の場合は 4 倍精度の場合は 8	浮動小数点	単精度の場合は 4 倍精度の場合は 8
484/485	パック 10 進数	バイト 1 に精度、バイト 2 に位取り	パック 10 進数	バイト 1 に精度、バイト 2 に位取り
492/493	64 ビット整数[4]	8	64 ビット整数	8
496/497	長精度整数	4	長精度整数	4
500/501	短精度整数	2	短精度整数	2

表 171. *DESCRIBE*、*FETCH*、*OPEN*、および *EXECUTE* の場合の *SQLIMSTYPE* と *SQLIMSLEN* 値 (続き)

	DESCRIBE の場合		**FETCH** の場合	
SQLIMSTYPE	列またはパラメーターのデータ・タイプ	**SQLIMSLEN**	ホスト変数のデータ・タイプ	**SQLIMSLEN**
504/505	N/A	N/A	DISPLAY SIGN LEADING SEPARATE, NATIONAL SIGN LEADING SEPARATE	バイト 1 に精度、バイト 2 に位取り
912/913	固定長バイナリー・ストリング	列の長さ属性	固定長バイナリー・ストリング	ホスト変数の長さ属性

組み込まれる SQLIMSDA

INCLUDE SQLIMSDA によって提供される SQLIMSDA では、COBOL のみがサポートされます。

COBOL (IBM COBOL のみ):

```
01 SQLIMSDA GLOBAL.
  02 SQLIMSDAID         PIC X(8).
  02 SQLIMSDABC         PIC S9(9) COMP-5.
  02 SQLIMSN            PIC S9(4) COMP-5.
  02 SQLIMSD            PIC S9(4) COMP-5.
  02 SQLIMSVAR OCCURS 0 TO 750 TIMES DEPENDING ON SQLIMSN.
    03 SQLIMSVAR1.
      04 SQLIMSTYPE     PIC S9(4) COMP-5.
      04 SQLIMSLEN      PIC S9(4) COMP-5.
      04 FILLER REDEFINES SQLIMSLEN.
        05 SQLIMSPRECISION PIC X.
        05 SQLIMSSCALE     PIC X.
      04 SQLIMSDATA     POINTER.
      04 SQLIMSIND      POINTER.
      04 SQLIMSNAME.
        49 SQLIMSNAMEL  PIC S9(4) COMP-5.
        49 SQLIMSNAMEC  PIC X(30).
    03 SQLIMSVAR2 REDEFINES SQLIMSVAR1.
      04 SQLIMSVAR2-RESERVED-1
                        PIC S9(9) COMP-5.
      04 SQLIMSLONGLEN REDEFINES
             SQLIMSVAR2-RESERVED-1
                       PIC S9(9) COMP-5.
      04 SQLIMSVAR2-RESERVED-2
                       PIC S9(9) COMP-5.
      04 SQLIMSDATALEN     POINTER.
      04 SQLIMSDATATYPE-NAME.
        05 SQLIMSDATATYPE-NAMEL
                       PIC S9(4) COMP-5.
        05 SQLIMSDATATYPE-NAMEC PIC X(30).
```

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。本書の他言語版を IBM から入手できる場合があります。ただし、ご利用にはその言語版の製品もしくは製品のコピーを所有していることが必要な場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒*103-8510*
東京都中央区日本橋箱崎町*19*番*21*号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、 IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。実際の結果は特定の構成や稼働条件によって異なります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名前はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。
© (お客様の会社名) (年).
このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。
© Copyright IBM Corp. _年を入れる_.

プログラミング・インターフェース情報

本書では、IMS によって提供されるプロダクト・センシティブ・プログラミング・インターフェースとそれに関連する情報を記述しています。

プロダクト・センシティブ・プログラミング・インターフェースにより、お客様のインストール済み環境で、このソフトウェア製品の診断、修正、モニター、修復、調整、またはチューニングなどの作業を実行することができます。これらのインターフェースを使用すると、IBM のソフトウェア製品の詳細設計やその実現方法に対する依存関係が発生します。このためプロダクト・センシティブ・プログラミング・インターフェースは上記の特別な目的にだけ使用してください。詳細設計やその実現方法に依存しているので、このようなインターフェースに合わせて作成したプログラムは、新しい製品のリリース、バージョンで実行するとき、または保守サービスの結果として、変更が必要になることがあります。プロダクト・センシティブ・プログラミング・インターフェースとそれに関連する情報は、セクションやトピックの単位の場合はその冒頭で識別され、それ以外の場合は「プロダクト・センシティブ・プログラミング・インターフェース」というマーキングで識別されます。IBM では、上記の冒頭部での識別の記述、およびその記述を参照する本書内のすべての記述を、そのような記述によって示される全体コピーまたは部分コピーに含めるよう求めています。

商標

IBM、IBM ロゴおよび ibm.com® は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe、Adobe ロゴ、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

製品資料に関するご使用条件

これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。

適用される条件

このご使用条件は、IBM Web サイトのすべてのご利用条件に追加して適用されます。

個人使用

これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用

これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

権利

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入 関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

IBM オンライン・プライバシー・ステートメント

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。

IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項をご確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、 お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

参考文献

この参考文献のリストには、IMS 14 ライブラリーのすべての資料が記載されています。

表題	頭字語	資料番号
IMS V14 アプリケーション・プログラミング	APG	SC43-3856
IMS V14 アプリケーション・プログラミング *API*	APR	SC43-3857
IMS V14 コマンド 第 *1* 巻*: IMS* コマンド *A-M*	CR1	SC43-3859
IMS V14 コマンド 第 *2* 巻*: IMS* コマンド *N-V*	CR2	SC43-3861
IMS V14 コマンド 第 *3* 巻*: IMS* コンポーネントおよび *z/OS* コマンド	CR3	SC43-3862
IMS V14 コミュニケーションおよびコネクション	CCG	SC43-3855
IMS V14 データベース管理	DAG	SC43-3853
IMS V14 データベース・ユーティリティー	DUR	SC43-3849
IMS Version 14 Diagnosis	DGR	GC19-4216
IMS V14 出口ルーチン	ERR	SC43-3850
IMS V14 インストール	INS	GC43-3851
IMS Version 14 Licensed Program Specifications	LPS	GC19-4231
IMS V14 メッセージおよびコード 第 *1* 巻*: DFS* メッセージ	MC1	GC43-3858
IMS V14 メッセージおよびコード 第 *2* 巻*: DFS* 以外メッセージ	MC2	GC43-3860
IMS V14 メッセージおよびコード 第 *3* 巻*: IMS* 異常終了コード	MC3	GC18-4221
IMS V14 メッセージおよびコード 第 *4* 巻*: IMS* コンポーネント・コード	MC4	GC18-4222
IMS V14 オペレーションおよびオートメーション	OAG	SC43-3852
IMS V14 リリース計画	RPG	GC43-3847
IMS V14 システム管理	SAG	SC43-3854
IMS V14 システム定義	SDG	GC43-3845
IMS V14 システム・プログラミング *API*	SPR	SC43-3846
IMS V14 システム・ユーティリティー	SUR	SC43-3848
Program Directory for Information Management System Transaction and Database Servers V14.01.00		GI10-8988
Program Directory for Information Management System Database Value Unit Edition V14.01.00		GI13-4602
Program Directory for Information Management System Transaction Manager Value Unit Edition V14.01.00		GI13-4601

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

[数字]

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

［特殊文字］

プログラム番号: 5635-A05
5655-DSE
5655-TM3

Printed in Japan

SC43-3857-03

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IMS バージョン **14**

アプリケーション・プログラミング **API**