Adabas

Adabas 8 の導入計画

バージョン 8.1.3

June 2008

Adabas

This document applies to Adabas Version 8.1.3 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

目次

1 Adabas 8 の導入計画 1
2 全般的な情報 3
リリース時期 4
互換性 4
ドキュメント 4
3 旧バージョンの Adabas からの移行 5
4 アーキテクチャの変更点 9
制限の解除 10
スパンドレコードのサポート 11
ラージオブジェクト（LB）フィールドのサポート 14
ロング英数字（LA）フィールドの変更点 21
FDT の変更点 22
5 ADARUN の変更点 23
6 ユーティリティの変更点 25
7 コマンドの変更点 31
ダイレクトコールの拡張 32
拡張 Adabas コントロールブロック（ACBX）のサポート 32
Adabas バッファ記述（ABD）のサポート 38
フォーマットバッファの変更点 39
LF コマンドの変更点 45
8 ユーザー出口の変更点 47
ユーザー出口 1 48
ユーザー出口 11 48
ハイパー出口の変更点 49
9 Adabas 8 対応のアドオン製品のサポート 51
クライアントベースのアドオン製品の互換性に関する全般的な情報 52
Adabas System Coordinator バージョン 8 のサポート 54
Adabas Fastpath バージョン 8 のサポート 57
Adabas SAF Security バージョン 8 のサポート 58
Adabas Transaction Manager バージョン 8 のサポート 59
Adabas Vista バージョン 8 のサポート 59
目次 63

1 Adabas 8 の導入計画

本書では、Adabas 8 をインストールして使用する場合や旧バージョンの Adabas から移行する場合に役立つ情報について説明します。

Caution: ここに記載されている内容は、製品リリースの前に変更される可能性があります。 Software AG は、リリース前に本製品および本ドキュメントの内容を変更する権利を留保します。また、記載されている機能がすべて実装されているとは限りません。 しかし、この情報はバージョン8へのアップグレードを正しく評価するためには十分に的確なものです。

本書は、Adabas 8 の設定と管理に携わる方を対象としています。 また、最新の Adabas 製品群について熟知していることを前提としています。

このdocumentの構成は次のとおりです。

全般的な情報	Adabas8のリリース時期、互換性、およびマニュアル提供形態について説明しています。
*旧バージョンの**Adabas** からの移行*	旧バージョンの Adabas から Adabas 8 に移行する際の考慮事項について説明しています。
アーキテクチャの変更点	Adabas8になってアーキテクチャがどのように変わったのかについて説明しています。
ユーティリティの変更点	Adabas8になってユーティリティがどのように変わったのかについて説明しています。
***ADARUN** の変更点*	Adabas8になって ADARUN がどのように変わったのかについて説明しています。
コマンドの変更点	Adabas8になってコマンドがどのように変わったのかについて説明しています。
ユーザー出口の変更点	Adabas8になってハイパー出口がどのように変わったのかについて説明しています。

•	*Adabas 8 対応のアドオン製品のサポート*	Adabas 8 の Adabas アドオン製品のサポートについて説明しています。

2 全般的な情報

- リリース時期 4
- 互換性 4
- ドキュメント 4

このchapterでは、次のトピックについて説明します。

リリース時期

Adabas 8 は 2006 年の第 2 四半期または第 3 四半期にご利用いただける予定です。

互換性

Adabas 8 は、現在サポートされているバージョンの Adabas と完全な互換性があります。

ドキュメント

このリリースのドキュメントの配布形態は、HTML 形式と PDF 形式です。 旧バージョンの Adabas では製本版のドキュメントが用意されていましたが、メインフレーム系の Adabas ドキュメントとしては今回から製本版はなくなりました。

この Adabas ドキュメントには、Adabas 8 の技術的な最新情報がすべて網羅されています。 このリリースのすべての拡張機能およびその他の情報については、を参照してください。

3 旧バージョンの Adabas からの移行

旧バージョンの Adabas から Adabas 8 に移行する場合には、ここに記載されている考慮事項に目を通しておいてください。 Adabas アドオンサポートの詳細については、「*Adabas 8 対応のアドオン製品のサポート*」を参照してください。

1. Adabas 8 には、新しいダイレクトコールインターフェイスが追加されましたが、これは現行版の ACB ベースのダイレクトコールインターフェイスと完全な互換性を保持しています。この新しいダイレクトコールインターフェイスは、Adabas 8 の ACBX 構造と ABD 構造に基づいています。

   ACB ベースのダイレクトコールインターフェイスを使用する既存のアプリケーションプログラムは、変更なしで従来と同じ方法で実行できます。さらに、ACBX ベースまたは ACB ベースのどちらのダイレクトコールインターフェイスをアプリケーションプログラムで使用するのかを、コールごとに決めることができます。1 つのプログラムが両方のインターフェイスを使用できます。

   一部の Adabas 8 の新機能では、アプリケーションプログラムが ACBX ベースのダイレクトコールインターフェイスを使用する必要があります。例えば、Adabas 8 でサポートされている、大きな（32K を超える）バッファまたはセグメント化したバッファ（複数のフォーマットバッファとレコードバッファ）をアプリケーションプログラムで操作するには、ACBX ベースのダイレクトコールインターフェイスを使用する必要があります。詳細については、「*コマンドの変更点*」を参照してください。
2. Adabas 8 では MU フィールドと PE フィールドの制限が強化されましたが、これを有効にするには、既存のファイルを再ロードするか再編成する必要があります。この他にも次の作業を行う必要があります。
   - ワークプールを大きくする（ADARUN LWP パラメータ）
   - WORK データセットのプロテクションエリアを大きくする（ADARUN LP パラメータ）
   - 内部フォーマットプールを大きくする（ADARUN LFP パラメータ）
3. 「*ユーティリティの変更点*」に記載されている ADADBS MUPEX 機能などの Adabas 8 の機能を使って、MU または PE の制限を拡張してファイルを作成した場合と、アプリケーション

プログラムが、フォーマットバッファの xxc エレメントを使って MU フィールドまたは PE グループのオカレンスカウントを読み込む場合には、プログラムがオカレンスカウントを 2 バイト以上のレコードバッファフィールド（例えば FB='MUC,2,B.'）に読み込むかどうかを確認してください。プログラムがオカレンスカウントを 1 バイトのフィールド（例えば FB='MUC.'、FB='MUC,1,B.'）に読み込む場合には、2 バイトのオカレンスカウント値を処理できるようにプログラムを修正する必要があります。 拡張 MU/PE が制限されたファイルの MU フィールドまたは PE グループのオカレンスカウント用のレコードバッファに 1 バイトフィールドのみを指定した場合、Adabas はレスポンスコード 55、サブコード 9 を返します。

4. ハイパーディスクリプタに拡張 MU または PE フィールドを使用する場合は、拡張 MU または PE のフィールド、パラメータリスト、および入力パラメータに適合するように、対応するハイパー出口を再アセンブルする必要があります。 詳細は、「*ハイパー出口の変更点*」を参照してください。

5. ラージオブジェクトフィールド（新しい LB オプションで定義されているフィールド）は、通常、LB フィールドを含んでいるファイル（基本ファイル）に関連付けられている LOB ファイルに格納されますが、 LB フィールド値が小さい場合（253 バイト以下）には、基本ファイルに直接格納されます。 LOB ファイルでは LB フィールド値はセグメント単位に分割され、1 つ以上のセグメントレコードに格納されます。最後のレコードを除き、この各セグメントレコードがデータストレージブロック全体を構成します。

   ラージオブジェクトフィールドを使用する場合は、次の項目に従って、Adabas ニュークリアス（ADARUN）のパラメータを見直してください。

   - LB フィールド値に関連する挿入、更新、削除を行うと、LOB ファイル内の関連セグメントレコードがすべてホールド状態になります。 ホールド処理が発生すると、ADARUN NH および NISNHQ パラメータとしてカウントされます。 したがって、同時に実行されるトランザクションで、LOB ファイルのセグメントレコードが最大何件更新される可能性があるのかを推定し、その値に基づいて、これらのパラメータを大きくする必要があります。
   - LB フィールド値の挿入、更新、削除が行われる前と後のイメージは、それぞれ WORK データセットのプロテクションエリアに書き込まれます。 したがって、同時に実行されるトランザクションで更新される可能性がある LB フィールド値がどのくらいのサイズになるのか推定し、その値の 2 倍の WORK ブロック数に応じて LP パラメータを大きくする必要があります。
   - LOB ファイルを構成するセグメントレコードは、それぞれユニークディスクリプタ（UQ オプション）によって識別されます。LB フィールド値を更新または削除するときは、LOB セグメントレコードごとにユニークディスクリプタプールのスペースとして約 30 バイト必要になります。 したがって、同時に実行されるトランザクションで削除または変更される可能性がある LB フィールド値の数と、ラージオブジェクトごとのセグメントレコードの最大件数を乗じ、さらにその数に 30 を乗じた数に応じて、LDEUQ パラメータを大きくする必要があります。
   - データベースから LOB 値を読み込むか、またはデータベースに LOB 値を書き込む Adabas コールが、最大でいくつ同時に発生するか推定し、データベースから読み込むか、またはデータベースに書き込む可能性がある最大 LOB 値と乗じた値のサイズに応じて、NAB パラメータを大きくする必要があります。

- データベースから読み込むか、またはデータベースに書き込む可能性がある最大 LOB 値のサイズになるように、LU パラメータを大きくする必要があります。 LU パラメータで指定する内容は、Adabas が割り当てるメモリエリアのサイズではなく、1 回の Adabas コールで使用可能なアタッチドバッファ（NAB パラメータで定義）のエリアの最大長です。

ADARUN パラメータの詳細については、「*Adabas の初期化（ADARUN ステートメント）*」を参照してください。

6. ラージオブジェクト（LB）フィールドまたはスパンドレコードのいずれかを使用する場合は、ADARUN LWP パラメータが 50,000 バイトとスレッド数（NT パラメータ）を乗じた値以上になるようにしてください。 ADARUN LWP パラメータの詳細については、「*LWP：Adabas ワークプール長*」を参照してください。
7. Adabas 8 でユーティリティを使用する場合、いくつかの制限があります。 Adabas ユーティリティが Ababas 8 でどのように変更され、それを使用するうえでどのような制限と制約事項があるのかについては、「*ユーティリティの変更点*」と「*制限および制約事項*」に記載されているユーティリティの機能拡張と制限の説明をお読みください。

4 アーキテクチャの変更点

- 制限の解除 10
- スパンドレコードのサポート 11
- ラージオブジェクト（LB）フィールドのサポート 14
- ロング英数字（LA）フィールドの変更点 21
- FDT の変更点 22

このchapterでは、Adabas 8 になってアーキテクチャがどのように変わったのかについて説明しています。 このリリースのすべての拡張機能およびその他の情報については、を参照してください。

制限の解除

- 論理エクステントの制限
- 物理エクステントの制限
- MU および PE の上限

論理エクステントの制限

Adabas の各ファイルエクステントのタイプごとの論理ファイルエクステントは 5 個に制限されていましたが、この制限は解除されました。 定義可能になった論理ファイルエクステントの最大数は、アソシエータの第1データセット（DDASSOR1）のブロックサイズから決定されます。エクステントの情報は、FCB の可変セクション内に格納されます。 使用された FCB サイズがアソシエータデータセットのブロックサイズに到達するまで、新しいエクステントを追加できるようになりました。 例えば、標準の 3390 デバイスタイプの場合、ファイルはタイプごとに 40 を超えるエクステントを持つことができます（他のタイプのエクステント数が少ない場合、1つのタイプのエクステント数はより多くすることができます）。

物理エクステントの制限

Adabas データベースのアソシエータおよびデータストレージコンポーネントには、各 5 個以上の物理エクステントを格納できるようになりました。 物理エクステントあたりの最大数は 99 に設定されていますが、実際の最大数はそれよりも少ない可能性があります。これは、ファイルの論理エクステントの最大数と同様に、最初のアソシエータデータセット（ASSOR1）のブロックサイズによって物理エクステントの最大数が変わるためです。 例えば、標準の 3390 デバイスタイプでは、アソシエータ、データストレージ、および DSST の各エクステントの個数は 75 を超えることができます（他のタイプのエクステント数が少ない場合は、1つのタイプのエクステントをより多く持てます）。

MU および PE の上限

MU フィールドまたは PE グループのそれぞれのオカレンス数は、1 レコードあたり、191 から 65,534 に増えましたが、実際の上限は、最大データストレージレコード長（ADALOD MAXRECL パラメータ）によって決まり、デフォルトではデータストレージブロックのサイズから 4 を引いた数に設定されています。

Note: 1 つのレコードに 191 個を超える MU または PE フィールドを使用する場合、ファイルがこれを使用できるように明示的に許可する必要があります（デフォルトでは許可されていません）。 これは、新しい ADADBS MUPEX 機能または、ADACMP COMPRESS `MUPEX` と `MUPECOUNT` パラメータを使用して行います。

すべての MU フィールドおよび PE グループ、およびその他のフィールドは、1 つの圧縮レコードに収まらなければなりません。 スパンドレコード（Adabas 8 で導入）を使用すると、より多くの MU フィールドおよび PE グループを格納できます。

拡張 MU または PE が制限されたファイルが作成された場合、MU フィールドまたは PE グループのオカレンスカウントをレコードバッファの1バイトフィールドに読み込んではいけません。これを行うと、Adabas はレスポンスコード 55、サブコード 9 を返します。 したがってフォーマットバッファで xxC エレメントを使用してオカレンスカウントを読み込む（FB='MUC.' または FB='MUC,1,B.' など）すべてのアプリケーションは、オカレンスカウントを 2 バイト以上のフィールドに読み込ませるように（FB='MUC,2,B.' または FB='MUC,4,B.' など）変更する必要があります。

スパンドレコードのサポート

Adabas では、今回のリリースからスパンドレコードの概念が導入されています。 データベース内では、論理レコードは多くの物理レコードに分割されます。それぞれのレコードは単一のデータストレージ（DS）ブロックに格納されます。 分割された各物理レコードには、それぞれ ISN が割り当てられます。 最初の物理レコードは*プライマリレコード*と呼ばれ、圧縮レコードの先頭が含まれ、*プライマリ ISN* が割り当てられます。 残りの物理レコードは、*セカンダリレコード*と呼ばれ、論理レコードの残りのデータを含みます。 セカンダリレコードには、*セカンダリ ISN* が割り当てられます。 これらの ISN は、N2 コマンドの使用時に割り当てられるユーザー ISN、または L1 コマンドの I オプション使用時に使用される ISN に影響しません。 スパンドレコードでは、セカンダリアドレスコンバータを使用して、セカンダリレコードが格納されているデータストレージブロックの RABN にセカンダリ ISN をマッピングします。

スパンドレコードは、1 つのプライマリレコードと 1 つ以上のセカンダリレコードで構成されます。 ただし、スパンドレコードのセグメント数は制限されています。 Adabas ニュークリアスは、スパンドレコード内に最大 5 個の物理レコード（1 個のプライマリレコードと 4 個のセカンダリレコード）を許可しています。

スパンドレコードはアプリケーションプログラムで直接表示させることはできません。 アプリケーションは、プライマリレコードの ISN を参照してから、スパンドレコードにアクセスします。

スパンドレコードは、拡張 Adabas ファイルおよびマルチクライアントファイルでもサポートされています。

Note: スパンドレコードのサポートは、ファイルに明示的に許可する必要があります。 ADADBS RECORDSPANNING 機能、または **ADACMP COMPRESS** の SPAN パラメータを使用してこれを行うことができます。

このsectionでは、次のトピックについて説明します。

- スパンドレコードの構造

- スパンドレコードの識別
- セカンダリレコードのセグメンテーション
- パディングファクタ
- スパンドレコードの ISN の使用
- 関係する ADARUN パラメータ
- スパンドレコードに関するレポート
- スパンドレコードのセキュリティ保護

スパンドレコードの構造

スパンドレコードは、1つのプライマリレコードと1つ以上のセカンダリレコードで構成されます。 スパンドレコードのプライマリレコードおよびセカンダリレコードは ISN を使用して結合されます。 各物理レコードのヘッダーには、現在のレコードの ISN、プライマリレコードの ISN、また次のセカンダリレコードの ISN が含まれています。 また、ヘッダーには、現在のレコードがプライマリレコードかセカンダリレコードかを示す情報も含まれています。

各物理レコードのヘッダーは、そのレコードの長さ情報も提供します。これはレコードがセグメント化されていても同様です。この場合、レコード長はセグメントの長さになります。

スパンドレコードの識別

ADACMP COMPRESS の SPAN パラメータ、ADADBS の RECORDSPANNING 機能、または Adabas Online System の同等機能で明示的に要求した場合に限り、スパンドレコードはファイルに含めることができます。 ADAREP データベースレポート機能および Adabas Online System レポート機能を使用すると、スパンドレコードを使用できるようにファイルが定義されているかどうかを確認できます。

ファイルの SPAN 属性は、ADAULD UNLOAD 機能内で保持されます。つまり、ファイルがアンロードされたり、削除されたり、再ロードされた場合に、スパンドレコードのサポートは変わらずに維持されます。

191 を超える MU または PE オカレンスが許可されているファイルにも同じ規則が適用されます。 圧縮レコード内の 191 を超える MU および PE オカレンスの識別の詳細については、「*圧縮レコード内の 191 を超える MU および PE オカレンスの特定*」を参照してください。

セカンダリレコードのセグメンテーション

セカンダリレコードは、フィールド単位またはバイト単位のいずれかでセグメント化されます。パフォーマンス上の理由から、セグメンテーションは可能な限りフィールド単位に行われます。ただし、LB（ラージオブジェクト）タイプではないフィールドのサイズがデータストレージブロックサイズよりも大きい場合、このレコードはバイト単位で分割されます。 フィールドのサイズがデータストレージブロックの残りの空きスペースより大きく、データストレージブロックサイズより小さい場合、このレコードはバイト単位ではなくフィールド単位で分割されます。各セカンダリレコードのヘッダーには、セグメントレコードがどのタイプであるかを示す情報が格納されています。

パディングファクタ

パディングファクタは、ブロックの領域をすべて使用するために、一般的にはスパンドレコードでは無視されます。 したがって、大抵の場合はレポート上でゼロが表示されます。 パディングファクタは、スパンドレコードの最後の短いセグメントのみに使用されます。

スパンドレコードの ISN の使用

プライマリレコードおよびセカンダリレコードは、アドレスコンバータ（AC）を使用して Adabas によりアドレスされます。ただし、プライマリアドレスコンバータはプライマリレコードの ISN のみを、対応するデータストレージブロックの RABN にマッピングします。 スパンドレコードでは、セカンダリアドレスコンバータを使用して、セカンダリレコードが格納されているデータストレージブロックの RABN にセカンダリ ISN をマッピングします。 したがって、各スパンドレコードのインデックスは 1 つだけになり、インデックス構造に何も影響を与えません。

ISN の範囲は、プライマリ ISN およびセカンダリ ISN 用に個別に維持されます。 ISN が格納または取り扱われる場合は、必ずそのアクションの対象がプライマリ ISN なのか、セカンダリ ISN なのかが区別されます。

すべてのコマンドは、プライマリレコードの ISN を使用して指定する必要があります。セカンダリレコードの ISN は内部的なもので、ユーザーが使用することはできません。 物理シーケンシャルコマンドは、データストレージ内のセカンダリレコードを自動的にスキップします。 セカンダリ ISN を指定した読み取りコマンドにはエラーが返されます（レスポンスコード 113）。

プライマリレコードの ISN は、TOPISN および MAXISN 値に含まれます。 セカンダリレコードの ISN は含まれません。 代わりに、セカンダリ ISN は、MINSEC および MAXSEC 値に含まれます。 スパンドレコードを含むファイルは MINISN 値を指定してロードできますが、MINISN はプライマリレコードの ISN のみを参照する必要があります（セカンダリレコードの ISN は参照できません）。

関係する ADARUN パラメータ

スパンドレコードを含むファイルをサポートするために、次の ADARUN パラメータ値を大きくする必要がある場合があります。

- 更新するスパンドレコードの数も増加するため、ユーザーごとのホールドキュー内の ISN 数（NISNHQ パラメータ）を大きくする必要がある場合があります。
- スパンドレコードのビフォーイメージとアフターイメージの両方を格納するスペースが必要なため、また、複数の更新スレッドの並行実行をサポートするため、Adabas ワークプール長（LWP）を大きくする必要がある場合もあります。 大きなディスクリプタバリューテーブルを格納するスペースが必要な場合もあります（PE グループのディスクリプタのオカレンス数は、最大 65,534 です）。

スパンドレコードに関するレポート

最大レコード長統計は、スパンされたファイルとは関連がありません。最大レコード長をレポートするユーティリティは、統計結果として "N/A"（該当なし）を表示するようになりました。FCB の最大レコード長フィールドは、スパンドレコードを使用しているファイルの場合には、値が大きくなります。

スパンドレコードのセキュリティ保護

スパンドレコードを含んでいるファイルは、暗号化することができ、セキュリティバイバリューにより保護することができます。 プライマリレコードの ISN が参照されている場合、すべてのセカンダリのセグメントレコードも読み込まれる必要があります。したがって、処理時間が重要な要素となります。

ラージオブジェクト（LB）フィールドのサポート

Adabasファイルに大型のフィールド（ラージオブジェクト（*LB*）フィールド）を含めることができるようになりました。このフィールドには、最大で 2,147,483,643 バイト（約 2 GB）までのデータを格納することができます。 Adabas 8 では、LB フィールド全体を格納および取得することのみが可能です。LB フィールドの一部を取得したり、LB フィールドの最後にデータを追加したり、LB フィールドの最後のデータを削除したりする機能は、今後の Adabas バージョンでサポートされる予定です。

このsectionでは、次のトピックについて説明します。

- Adabas 8 における LB フィールドの処理
- FDT でのラージオブジェクト（LB）フィールドの定義
- LB フィールドでサポートされるフォーマットバッファ
- ロング英数字フィールドとラージオブジェクトフィールドの比較

Adabas 8 における LB フィールドの処理

Adabas 8 では、バイナリ型と文字型の LB フィールドがサポートされています。 バイナリ LB フィールドを変更することはできません。格納されているのと同じLBフィールドのバイナリバイト文字列が、LB フィールドが読み込まれたときに取り出されます。 バイナリ LB フィールドの詳細については、「*バイナリ LB フィールドの定義*」を参照してください。

文字型のラージオブジェクトについては、データベース、ファイル、およびユーザーに対するユニバーサルエンコーディングサポート（UES）関連の定義に従って、文字コードが変換されます。 新しいフィールドオプションである NB（非空白圧縮）が LB フィールド定義に存在するかによって、文字型 LB フィールドの末尾の空白が Adabas により削除されるかどうかが示されます。

FDT でのラージオブジェクト（LB）フィールドの定義

LBフィールドは、新しいLBフィールドオプションを使用して、フィールド定義テーブル（FDT）で定義する必要があります。 フィールド形式は、"A"（英数字）に設定する必要があります。

LB フィールドのデフォルトのフィールド長は、現在、ゼロとして定義されている必要があります。 Adabasの今後のリリースでは、ゼロ以外のデフォルトフィールド長がサポートされ、ロング英数字（LA）フィールドとラージオブジェクト（LB）フィールドに 253 を超える数値を定義できるようになる予定です。

LB フィールドでは、次の示す事項は*許可されません*。

- ディスクリプタ自体、または特殊なディスクリプタ（フォネティック、サブ、スーパー、またはハイパーディスクリプタ）の親に指定する。
- FI または LA オプションとともに定義する。
- サーチバッファ内、またはサーチバッファのフォーマット選択条件として指定する。

LB フィールドでは、次に示す事項が許可されます。

- MU、NB（新しい非空白圧縮オプション）、NC、NN、NU、またはNVオプションを任意の組み合わせで定義する。
- 単一グループまたは PE グループの一部にする。

このsectionでは、次のトピックについて説明します。

- LB フィールドでの NV オプションの指定
- 新しい NB オプション
- バイナリ LB フィールドの定義
- LB フィールドの例

LB フィールドでの NV オプションの指定

NV（変換なし）オプションを指定すると、フィールド値を受け渡すマシンのアーキテクチャがAdabas サーバーとは異なっていても、フィールド値は変換されません。

新しい NB オプション

新しい NB オプションを LA および LB フィールドとともに使用すると、空白圧縮を制御できます。 NB オプションを指定すると、フィールドの末尾の空白が削除されなくなります。 フィールドに NB オプションを指定した場合、そのフィールドに NU オプションまたは NC オプションも指定する必要があります。NB 処理には、NC または NU の使用も同時に必要となります。Adabas の今後のリリースでは、通常の英数字フィールドやワイド文字フィールドにも NB オプションを指定できるようになる予定です。

バイナリ LB フィールドの定義

NV および NB オプションを両方とも指定すると、バイナリ LB フィールドが定義されます。このフィールドがいったん保存されるとフィールド値は変更できなくなります。

Note: バイナリ LB フィールドの定義には、フォーマット B は使用されません。これは、一部の環境ではフォーマット B が、バイト順の異なるバイトスワップを意味するためです。 バイトスワッピングは、バイナリ LB フィールドに適用されません。

LB フィールドの例

次の表は、LB フィールドの FDT 定義の有効な例を示しています。

FDT の指定	説明
`1,L1,0,A,LB,NU`	フィールド L1 は、空値省略の文字列 LB フィールドです。
`1,L2,0,A,LB,NV,NB,NU,MU`	フィールド L2 は、マルチプルバリュー、空値省略のバイナリ LB フィールドです。

LB フィールドでサポートされるフォーマットバッファ

LB フィールドをフォーマットバッファに指定する方法は、通常のフィールドを指定するときとほとんど同じです。 このsectionでは、フォーマットバッファに LB フィールドを指定するうえで、通常と異なる点や特殊な事項について説明します。

- 範囲の表記
- オカレンスインデックス
- LB フィールドフォーマットの指定
- LB フィールドの長さの指定

範囲の表記

マルチプルバリュー LB フィールドおよびピリオディックグループ内の LB フィールドでは、範囲の表記を使用して、フィールドの固定オカレンス数を指定することができます。 例えば、次のようにフォーマットバッファを指定すると、LB フィールド L2 の最初から 10 番目までの値が選択されます。

```
FB='L21-10.'
```

ただし、範囲の終了位置を明示しない 1-N 表記を指定して、フィールドのすべてのオカレンスを選択することはできません。 例えば、次のようにフォーマットバッファを指定することはできません。

```
FB='L21-N.'
```

1-N 表記は、LB フィールドではサポートされません。

オカレンスインデックス

マルチプルバリュー LB フィールドおよびピリオディックグループ内の LB フィールドでは、フィールドの特定のオカレンスを指定する必要があります。 次の例では、マルチプルバリュー LB フィールド L2 の最初の値が選択されます。

```
FB='L21.'
```

同様に、次の例では、LB フィールド L3 の 2 番目の PE グループインスタンス内にある 5 番目の値が選択されます。

```
FB='L32(5).'
```

基本フィールドを指定する場合には、オカレンスインデックスが必要です。 例えば、L2 がマルチプルバリューフィールドの場合、次のようにフォーマットバッファを指定することはできません。

```
FB='L2.'
```

LB フィールドフォーマットの指定

LB フィールドのフォーマットバッファには、A（英数字）を指定することができます。 フォーマットを指定しない場合は、FDT 内の LB フィールドのフォーマット定義が適用されます。

LB フィールドの長さの指定

次に示すように、フォーマットバッファ指定を使用して、LBフィールドの長さを対応するレコードバッファに設定する方法は 3 つあります。

- 明示的な長さ指定
- ゼロの長さ指定
- アスタリスク（*）を使った長さ表記

Note: フォーマットバッファ指定で、LB フィールドの長さを指定しない場合は、FDT に定義されているデフォルトの長さ（ゼロ）が適用されます。 この場合、このsectionで説明しているように、長さ指定がゼロの場合のルールが適用されます。

明示的な長さ指定

フォーマットバッファで明示的に長さを指定することができます。LB フィールドのフォーマット要素でゼロ以外の長さを指定すると、その長さがレコードバッファの LB 値に割り当てられるスペースの大きさになります。 指定できる有効な最大長は 2,147,483,647 です。

次の例では、レコードバッファの LB フィールド L1 に 50,000 バイトが割り当てられ、フィールド AA に 10 バイトが割り当てられます。

```
FB='L1,50000,AA,10,A.'
```

フィールドのフォーマット要素に明示的な長さ設定が含まれている場合は、フィールド全体に必要なスペースをレコードバッファに確保しておく必要があります。 十分なスペースが指定されていない場合は、エラー（レスポンスコード 53）が返されます。

ゼロの長さ指定

フォーマット要素にゼロの長さを指定すると、レコードバッファの LB フィールド値のために確保されるスペースの大きさは可変になり、実際の LB 値の内容によって変わります。 この場合、レコードバッファの LB 値の先頭から 4 バイトには、その 4 バイトの長さも含む LB フィールドの実際の長さ（LB 値の長さ + 4）が格納されます。 有効な最大長は、4 バイト分も含めて、2,147,483,647 です。 LA フィールドの場合は、レコードバッファに 2 バイトのフィールド長が格納されます。

次の例では、LB フィールド L1 のレコードバッファの先頭から 4 バイトに L1 フィールドの長さが格納され、その次のバイトから実際の L1 フィールドの値が格納されます。 次に、L1 フィールドの値の直後に続く 10 バイトのフィールド AA が割り当てられます。

```
FB='L1,0,AA,10,A.'
```

フィールドのフォーマット要素にゼロの長さ設定が含まれている場合は、フィールド全体に必要なスペースをレコードバッファに確保しておく必要があります。 十分なスペースが指定されていない場合は、エラー（レスポンスコード 53）が返されます。

アスタリスク（*）を使った長さ表記

LA および LB フィールドにのみ、フォーマット要素に長さではなくアスタリスク（*）を指定できます。この指定を行うと、レコードバッファに確保される LB フィールド値のスペースの大きさは可変になり、LB フィールドの実際の値に必要な大きさになります。ただし、長さゼロの指定とは異なり、レコードバッファでは LB フィールド値の前に 4 バイト長フィールドは*存在しません*。LB フォーマット要素に対応するレコードバッファエリアには、LB フィールドの値のみが含まれます。実際の LB フィールド値の長さは読み込みコマンド用に*取得される必要があり*、新しいフォーマットバッファ長さインジケータ *L* を使用する更新コマンド用に*指定される必要があります*。長さインジケータの詳細については、このマニュアルの「*長さインジケータ（L）*」を参照してください。

次の例では、LB フィールド L1 のレコードバッファには L1 フィールドの値のみが含まれます。次に 10 バイトが割り当てられた AA フィールドの値が続きます。

```
FB='L1,*,AA,10,A.'
```

次の例では、LB マルチバリューフィールド L2 のレコードバッファには、L2 フィールドの最初の 10 個の値が含まれます。

```
FB='L21-10,*.'
```

レコードバッファは、そのフォーマット要素のアスタリスクの長さ設定が含まれている場合は、フィールド全体が使用する十分な空きスペースが確保されるため、必ずしも必要ではありません。 ただし、読み込みコマンドの処理中に次の*両方*の条件が合致した場合、フィールド値が切り捨てられます。

- レコードバッファの空きスペースがフィールド値に対して不十分である。
- アスタリスク表記付きのフィールドがフォーマットバッファの*最後*に指定されている。

これらの条件に合致する場合、エラーは返されません。 上記の 2 番目の例（`FB='L21-10,*.'`）でこの条件が合致した場合、Adabas は読み取る 10 個の値を対応するレコードバッファセグメントの長さに合わせて切り落とします （右から順番に切り落とされます。つまり、最後の値が最初に切り落とされます。まだ残りの空きスペースが不十分な場合、最後から2番目が切り落とされ、以下同様に続きます）。極端な場合、このフィールド用にまったく空きスペースがない場合は、値はゼロバイトに切り落とされます。

上記の最初の例では（`FB='L1,*,AA,10,A.'`）、レコードバッファセグメントが非常に短い場合、固定長が指定されたフィールドまたは長さがゼロ（0）のフィールドには切り落としが許可されていないため、切り落としは発生しません。 切り落としは発生しませんが、ニュークリアスはレスポンスコード 53（レコードバッファが非常に小さい）を返します。

Adabas ニュークリアスにより実行される読み取りコマンドでのみ、アスタリスク表記が指定された値が切り落とされます。更新コマンドでは切り落としは発生しません。 さらに、ADACMP ユーティリティは、アスタリスク表記が指定された値を切り落としません。

ロング英数字フィールドとラージオブジェクトフィールドの比較

関連のある LA フィールドおよび LB フィールド機能を比較している次の表は、データベースのフィールドを定義する際にどちら使用するかを決定するのに役立ちます。

機能	LA フィールドの動作	LB フィールドの動作
フォーマットバッファ内のゼロフィールド長指定	対応するレコードバッファエリア内の2バイトが、LAフィールドの実際の長さを格納するために使用されます。	対応するレコードバッファエリア内の4バイトが、LB フィールドの実際の長さを格納するために使用されます。
データレコードストレージ	英数字フィールドおよびワイド文字フィールドが、圧縮レコード内に格納されます。 すべての長い値は、同じ圧縮レコード内に収まらなければなりません。シンプルデータレコードまたはスパンドデータレコードの最大長により、格納できる長い値の数および長さが制限されます。複数の長い値がレコードに含まれると、問題となる場合があります。	一部の LB フィールド値（253 バイト超）は、別のラージオブジェクトファイル（LOB ファイル）にオフラインで格納され、LOB ファイル内の LB フィールド値への参照のみがデータレコードに含まれます。これにより、通常のフィールド、または LA フィールドを使用するより、単一のデータレコードに長いオブジェクトを格納できるようになります。ただし、この動作により、LBフィールドの実行時およびファイルメンテナンスのパフォーマンスのオーバーヘッドが増加します。 小さなLBフィールド値（253バイト以下）は圧縮レコード内に直接格納されます。これにより、小さな値に対するパフォーマンスは向上しますが、同一の圧縮レコードに格納可能な小さな LB フィールドのオカレンス数が制限されます。
フォーマットバッファ内のアスタリスク（*）フィールド長表記	すべての長さの LA フィールドがサポートされます。	すべての長さの LB フィールドがサポートされます。
最大長が 16,381 バイト*以下*の格納オブジェクト	英数字またはワイド文字のLAフィールドを使用できます。これにより、LBフィールドのオーバーヘッドを避けることができますが、単一レコードに格納できるフィールドの数が制限されます。	英数字の LB フィールドを使用できます。
最大長が 16,381 バイトを超える格納オブジェクト	サポートされません。	16,381 バイトを超えるオブジェクトがサポートされます。
シンプルデータレコードまたはスパンドデータレコードに収まらない非常に多くのラージオブジェクト	サポートされません。	複数のラージオブジェクトがサポートされます。

ロング英数字（LA）フィールドの変更点

今回のリリースでは、ロング英数字（LA）フィールドで次の更新が行われました。

- LA フィールドの FDT 定義で、新しく、フィールドの空白圧縮を制御する NB オプションを指定できるようになりました。 NB オプションの詳細については、「*FDT の変更点*」を参照してください。
- LA フィールドは、フォーマットバッファ内で 253 個を超える固定フィールド長を指定できるようになりました。 詳細は、「*フォーマットバッファの変更*」を参照してください。
- 新しいアスタリスク（*）フィールド長指定が、フォーマットバッファ内の LA フィールドでサポートされます。 アスタリスクを使ったフィールド長指定の詳細については、「*アスタリスク（*）を使った長さ表記*」を参照してください。
- *長さインジケータ*と呼ばれる新しいフォーマットバッファインジケータ（*L*）を使用して、LA フィールド値の実際の長さを取得または指定できるようになりました。詳細は、「*フォーマットバッファの変更*」を参照してください。
- マルチプルバリュー LA フィールドおよびピリオディックグループ内の LA フィールドに基本フィールドを指定する場合は、オカレンスインデックスまたは "1-N" インデックスが必要です。 特定のインデックスまたはインデックス範囲を使用する必要があります。 例えば、L2 が MU オプション付きの LA フィールドの場合、次のフォーマットバッファ指定は*無効*です。

```
FB='L21-N.'
```

```
FB='L2.'
```

ただし、L2 フィールドの 1 番目から 3 番目の値を要求している次のフォーマットバッファ指定は有効です。

```
FB='L21-3.'
```

LA フィールドと LB フィールドの違いについては、「*ロング英数字フィールドとラージオブジェクトフィールドの比較*」を参照してください。

FDT の変更点

FDT のフィールド定義の変更点は次のとおりです。

- バージョン 8 では内部 FDT 構造が拡張され、これらの大きな FDT は 4 個以上のアソシエータブロックを使用することができます。 大きな FDT に必要な追加のブロックは、アソシエータのフリースペースから自動的に割り当てられます。 下位互換性および変換に対応できるように、アソシエータ内の FDT 用の固定スペースが確保されます。
- 新しい LB フィールドオプション（ラージオブジェクトフィールドオプション）を使用して、フィールドを LB フィールドとして定義できるようになりました。 詳細は、「*FDT でのラージオブジェクト（LB）フィールドの定義*」を参照してください。
- 新しい NB オプション（非空白圧縮オプション）を指定すると、LA フィールドと LB フィールドから末尾の空白が削除されなくなります。詳細は、「*FDT でのラージオブジェクト（LB）フィールドの定義*」を参照してください。
- LB フィールドには、従来と同じように NV オプションを指定することもできます。 このオプションを指定すると、LB フィールドの文字コードが変換されなくなり、A フォーマットと W フォーマット間の変換ができなくなります。詳細は、「*FDT でのラージオブジェクト（LB）フィールドの定義*」を参照してください。

5 ADARUN の変更点

ADARUN CLOGLAYOUT パラメータに新しく 8 を指定できるようになりました。この値を指定すると、Adabas コマンドログ（CLOG）の形式が Adabas 8 形式になります。 ADARUN CLOGLAYOUT=4 設定は実行されません。また設定の 4 はサポートされません。 代わりに CLOGLAYOUT=5 をお使いください。

旧バージョンの Adabas では、ロードライブラリを APF 認可してから実行している場合、EXCP と EXCPVR のどちらかを選択することができませんでした。 APF 認可なしで実行している場合は常に EXCP が使用され、APF 認可ありで実行している場合には EXCPVR が常に使用されます。 APF 認可を実行している場合に EXCP を使用するには、特別な A$- zap または AY- zap の適用が必要です。

今回のリリースでは、ADARUN パラメータとして新しく EXCPVR を使用できるようになりました。 このパラメータを使用すると、APF 認可の実行時に EXCP または EXCPVR のどちらを使用するかを指定できます。 このパラメータの詳細については、「*EXCPVR：EXCP または EXCPVR の使用の制御*」を参照してください。

zap はもう必要ないため、古い A$-zap や AY-zap に対する更新は提供されません。

6 ユーティリティの変更点

Adabasユーティリティはすべて、Adabas 8の新しい拡張機能をサポートするように更新されています。 このサポートの一部は、新規、または修正されたユーティリティパラメータの形で現れます。 それ以外の場合は、サポートは内部的に追加され、ユーティリティの使用に影響はありません。

Note: Adabas 8のユーティリティ使用上の制限および制約事項については、「*制限および制約事項*」に記載されているユーティリティの制約事項を参照してください。

次の表では、このリリースでユーザーインターフェイスが変更されたAdabasユーティリティについて説明します。 Adabasユーティリティの機能の詳細については、「*ユーティリティ*」を参照してください。

ユーティリティ	変更の概要
ADAACK	ADAACK ユーティリティでは、スパンドレコードのサポートが提供されます。 ただしADAACKは、渡されたISNをプライマリISN、またはレコードの唯一のISNと見なします。ISN がスパンドレコードのプライマリ ISN の場合、スパンドレコードの関連するすべてのセグメントレコードは、自動的にセカンダリアドレスコンバータ内でチェックされます。 特定の ISN についてのエラー情報を出力する場合、ADAACK ユーティリティでは、レコードがスパンされている場合に、問題がプライマリ ISN とセカンダリ ISN のどちらかなのかを示すようになりました。
ADACDC	現時点では、スパンドレコードは ADACDC ユーティリティでサポートされていません。 ただし、ADACDC 実行で `IGNORESPANNED` パラメータが指定されると、ADACDC 処理はスパンドレコードを無視して警告メッセージを発行し、処理を続行します。 リターンコード "4" が返されます。
ADACMP	次の*新しい*パラメータが ADACMP COMPRESS ユーティリティに追加され、MU/PE 拡張、スパンドレコードおよび LB フィールドがサポートされました。 ■ `DATADEVICE` パラメータは、スパンドレコードのセグメンテーションに使用されるデータストレージデバイスタイプを指定します。 SPANパラメータが指定された場合、ADACMP

ユーティリティ	変更の概要
	はスパンされた長い圧縮レコードをセグメントに分割します。このセグメントは、DATADEVICE パラメータで暗示的に指定されるデータストレージのブロックサイズよりわずかに小さくなります。 SPAN パラメータなしで DATADEVICE が指定された場合、このパラメータは許容される最大の圧縮レコード長を得るために使用されます。 長すぎる圧縮レコードは、エラーと見なされ、DD/FEHL データセットに書き出されます。 ■ HEADER パラメータは、ADACMP 圧縮ロジックが、非圧縮入力レコードの中にセグメント化された ADACMP レコードヘッダーが存在するかどうかを示します。 デフォルトは NO です。 （これは、このリリースで ADACMP DECOMPRESS 用に導入された HEADER パラメータとは逆です） ■ LOBDEVICE パラメータは、COMPRESS 機能で生成される *LOB ファイル*のロードに使用されるデバイスタイプを指定します。 ■ LOBVALUES パラメータは、非圧縮入力データが長い LB 値（253 バイトよりも大きい値）を含むことができるかどうかを示します。 ■ MAXLOGRECLEN パラメータは、物理的にセグメント化された非圧縮レコードを論理的に圧縮レコードに編成するために ADACMP により使用されるバッファのサイズ（バイト単位）の指定に使用されます。 このバッファは、HEADER=YES が指定された場合のみ割り当てられ、使用されます。 ■ MUPEX パラメータは、ファイルの拡張 MU/PE の制限が可能かどうかを示します。 このオプションが指定されない場合、指定できる MU フィールドの最大値および PE フィールドの最大値は 191 です。 ■ MUPECOUNT パラメータは、COMPRESS 機能用の入力レコードのバリューカウントフィールドのサイズを指定します。 有効な値は、"1" または "2" です。 "1" が指定された場合、入力データで MU または PE 値に先行する各バリューカウントフィールドは 1 バイトである必要があり、"191" を超えることはできません。 "2" が指定された場合、入力データで MU または PE 値に先行する各バリューカウントフィールドは 2 バイトである必要があります。 MUPEX パラメータが指定されている場合のみ、バリューカウントは 191 を超えることができます。 ■ SPAN パラメータにより、レコードの圧縮後にその圧縮レコードがデバイスのデータストレージのブロックサイズを超過した場合に、レコードをスパンすることが可能になります。 既存の ADACMP COMPRESS パラメータに次の*変更*が行われました。 ■ DEVICE パラメータにより生成されるレポートに、ファイルに MUPEX パラメータが設定されたかどうかを表す項目が追加されました。 ■ FNDEF パラメータの構文が変更され、MU および PE オプションのオカレンス数を指定できるようになりました。 また、NB および LB フィールドオプションを指定することもできます。 ■ Adabas 8 の ADACMP COMPRESS 機能では、MAXPE191 パラメータはサポートされなくなりました。 このパラメータが指定された場合、警告メッセージが発行され、処理が続行されます。 ■ USERISN が、HEADER=YES とともに指定された場合、ADAH ヘッダーの直後に ISN が論理レコードの一部として続きます。

ユーティリティ	変更の概要
	LB フィールドを含むファイルに ADACMP COMPRESS を実行すると、*LOB ファイル*に LB フィールド値が格納されるように、DDAUSB1 JCL DD コントロールステートメントに指定されているシーケンシャル出力データセットがもう 1 つ作成されます。 ADACMP DECOMPRESS 処理では、拡張 MU および PE の制限が可能です。また、スパンドレコードを入力とすることも可能です。 各非圧縮出力レコードの MU または PE 値に先行するバリューカウントのサイズは、拡張 MU および拡張 PE がファイルでサポートされているかによって異なります。 拡張 MU および拡張 PE がファイルでサポートされている場合、バリューカウントは 2 バイトになります。 拡張 MU および拡張 PE がファイルでサポートされていない場合、バリューカウントは 1 バイトになります。 さらに、次の機能がサポートされています。 ■ LB フィールドの圧縮解除。 この機能を使用できるように、LOBVALUES パラメータが新しく追加されました。 ■ 新しい HEADER パラメータが ADACMP DECOMPRESS ユーティリティに追加され、ADACMP 非圧縮ロジックがグメント化された ADACMP レコードヘッダー（ADAH および ADAC）を非圧縮出力の一部として生成すべきかどうかを示します。 デフォルトは NO です。 ■ 新しい `MAXLOGRECLEN` パラメータを使用して、非圧縮出力データの物理レコードを 1 つ以上スパンしている論理レコードを ADACMP が編成するのに使用するバッファのサイズ（バイト単位）を指定できます。 このバッファは、HEADER=YES が指定された場合のみ割り当てられ、使用されます。 ■ ISN パラメータが変更され、HEADER=YES が指定されている場合には、ISN は論理レコードの一部として ADAH ヘッダーの直後に続きます。 従来、ADACMP エラーに生成される DD/FEHL エラーデータセットは、FEHL 物理レコード長を超える拒否レコードを切り捨てていました。 バージョン 8 では、拒否されたレコードは切り捨てられずにセグメント化されます。 この変更により、DD/FEHL LRECL 設定には、最低 500 バイトが必要となります。
ADADBS	ADADBS ユーティリティに、4 つのデータベースサービスが追加され、MU/PE フィールドの数の増加とレコードのスパンがサポートされました。 ■ MUPEX 機能により、ファイルの MU または PE フィールドに許可される最大オカレンス数を指定できます。 ■ RECORDSPANNING 機能は、ファイルでスパンドレコードの使用を有効にします。 ■ RESETPPT 機能は、ASSO データセット上の PPT ブロックをリセットします。 ■ SPANCOUNT 機能は、ファイルに含まれるスパンドレコードの件数をカウントします。
ADADCK	ADADCK ユーティリティは、次のようにスパンドレコードのヘッダーの妥当性をチェックします。 ■ ヘッダーに含まれている ISN が検証されます。 ヘッダーには、スパンドレコードチェーン内のプライマリレコードの ISN、チェーン内の直前のスパンドレコード、およびチェーン内の次のスパンドレコードの ISN が含まれています。

ユーティリティ	変更の概要
	■ スパンドレコードのIDビットがチェックされます。スパンドレコードのヘッダーには、このレコードがプライマリスパンドレコードか、セカンダリスパンドレコードであるかを示すビットが含まれます。 すべてのスパンドレコードでは、これらのビットの 1 つのみをオンにすることができます。 新しい MAXPISN パラメータが導入され、データストレージファイルがスパンされているかどうかのチェックが行われるプライマリ ISN の最大数を設定できるようになりました。 デフォルトは 1000 です。
ADAFRM	ADAFRM ユーティリティを使用して、PLOG 全体の再フォーマットをすることなく、PLOG から複数の PLOG ヘッダーを消去できるようになりました。 これを行うには、FROMRABN パラメータと同時に NUMBER パラメータも指定する必要があります。また、SIZE パラメータを "1" に指定する必要があります。 さらに、このユーティリティは、Adabas 8 で可能なより多くの物理アソシエータおよびデータストレージエクステント（99 個）を扱えるようになりました。
ADAICK	ADAICK DSCHECK 機能を実行すると、プライマリおよびセカンダリ ISN が出力で識別されるようになりました。
ADALOD	ADALOD LOAD および UPDATE 機能で、スパンドレコードおよび関連するセカンダリアドレスコンバータをサポートする次の新しいパラメータが導入されました。 ■ AC2RABN パラメータにより、セカンダリアドレスコンバータのスペース割り当てを指定または更新できます。 セカンダリアドレスコンバータは、セカンダリスパンドレコードのセカンダリ ISN を、セカンダリレコードが格納されているデータストレージブロックの RABN にマッピングするために使用されます。 ■ オプションのMAXISN2パラメータにより、ISN内のセカンダリアドレスコンバータ（AC2）を目的のサイズに設定できます。 セカンダリアドレスコンバータは、セカンダリスパンドレコードのセカンダリ ISN を、セカンダリレコードが格納されているデータストレージブロックの RABN にマッピングするために使用されます。 ラージオブジェクト（LB）フィールドおよび関連する *LOB ファイル*をサポートできるように、次の新しいパラメータおよびパラメータ値が ADALOD LOAD 機能に導入されました。 ■ オプションの LOBFILE パラメータにより、*基本ファイル*に関連付けられた *LOB ファイル*のファイル番号を指定できます。 このパラメータは、基本ファイルのロード時に使用されます。 ■ オプションの BASEFILE パラメータにより、*LOB ファイル*に関連付けられた*基本ファイル*のファイル番号を指定できます。 このパラメータは、LOB ファイルのロード時に使用されます。 ■ 新しいファイルタイプである LOB を FILE パラメータで指定すると、Adabas LOB ファイルを定義済みの FDT とともにロードしているかどうかが示されます。
ADAORD	スパンドレコードをサポートするため、次の 2 つの新しいパラメータが、ADAORD の REORASSO、REORDB、REORFASSO、REORFILE、および STORE 機能に追加されました。 ■ AC2RABN パラメータにより、ファイルの 2 次的なアドレスコンバータエクステントの開始 RABN を指定できます。

ユーティリティ	変更の概要
	■ オプションのMAXISN2パラメータにより、ISN内のセカンダリアドレスコンバータ（AC2）を目的のサイズに設定できます。 セカンダリアドレスコンバータは、セカンダリスパンドレコードのセカンダリ ISN を、セカンダリレコードが格納されているデータストレージブロックの RABN にマッピングするために使用されます。 ADAORD STORE を使用すると、Adabas 5.1～5.3、6.1、6.2、7.1、7.2、または 7.4 のデータベースやファイルを再構築して、バージョン 8 形式で保存できるようになります。 この他、Adabas 8 データベースやファイルを再構築して、Adabas の以前のリリースのいずれかの形式で保存することもできますが、その場合には、データベースまたはファイルが、ファイルの再構築に使用したバージョン以降に導入された新機能を使っていないことが前提になります。
ADAREP	ADAREP により生成されるレポートに、MUPEX およびスパンドレコードオプションがデータベースに設定されたかどうかを表す項目が追加されました。 レポートの［Contents of Database］セクションでは、次の変更が行われました。 ■ この表からパディングファクタが削除され、大きなエクステント値が収まるようになりました。 ただし、従来どおりレポートで提供される個々のファイル詳細にもこの値を表示させることができます。これは、LAYOUT=1 が指定されている場合に表示されます。 ■ レポートの作成中に LAYOUT=1 が指定された場合、Adabas 8 でサポートされる大きなエクステント値がファイルごとに表示されます。 ■ ファイルが 10 個以上のファイルエクステントをさらに構築することができない場合、ADAREP は、そのファイルの右にアスタリスク（*）マークを付けます。 レポートの［File Options］セクションでは、"T" は、2 バイトの MU/PE インデックスがファイルでアクティブであることを示し、"S" は、ファイルでスパンドレコードの使用が有効になっていることを示します。 さらに、［Contains LOB Fields］列は、ファイルが 1 つまたは複数のLBフィールドを含んでいるかどうかを示し（含まれている場合は"L"を表示）、［LOB File］列（最後の列）は、ファイルが LOB であるかどうかを示します（LOB ファイルの場合は、"L" を表示）。 これらの 2 つの LB フィールド列は互いに排他的であり、どちらか一方しかマークされません。 レポートの［Physical Layout of the Database］セクションでは、2 次的なアドレスコンバータエクステント（スパンドレコード用）は［Table File Type］列で［AC2］と表示されます。 レポートの［File Information］セクションでは、［Two Byte MU/PE］という新しいフィールドが 2 バイトの MU/PE インデックスがファイルでアクティブかどうかを示します。 同じセクションでは、最大、予想される最大、および最小のセカンダリ ISN が表示され、また、新しい［Spanned Rec Supp］フィールドには、スパンドレコードがファイルでアクティブかどうかが示されます。 さらに、［Contain LOB Fields］フィールドは、ファイルに 1 つ以上のLBフィールドが含まれているかどうかを示し、［LOB File］フィールドはファイルがLOBファイルであるかどうかを示します。 レポートの［Space Allocation］セクションでは、2 次的なアドレスコンバータエクステント（スパンドレコード用）は［List Type］列で［AC2］と表示されます。

ユーティリティ	変更の概要
	最終的に、Adabas ニュークリアス 75、76 および 77 により、3 個の新しいチェックポイントが書き込まれる可能性があります。
ADASAV	ADASAV RESTONL FMOVE および ADASAV RESTORE FMOVE 機能で、スパンドレコードおよび関連するセカンダリアドレスコンバータをサポートする次の新しいパラメータが導入されました。 ■ AC2RABN パラメータにより、FMOVE により指定されたファイルごとに、セカンダリアドレスコンバータ開始 RABN を指定できます。 ■ MAXISN2 パラメータにより、FMOVE により指定された各ファイルに割り当てられる、セカンダリ ISN の新しい数を指定できます。
ADASEL	新しいパラメータはありませんが、ADASEL は更新され、MU/PE の拡張がサポートされました。 特に、ADASEL SELECT IF ステートメントで MU または PE フィールドのインデックスを指定する場合、インデックスは "1" から "65,534" の範囲で指定できるようになりました。以前のリリースの Adabas では、これらのインデックス値の範囲は、"1" から "191" に制限されていました。 ADASEL は、その処理中にスパンドレコードを認識しますが、スパンドレコードが含まれたファイルを処理することができません。
ADAULD	新しいパラメータはありませんが、ADAULD が更新され、MU/PE の拡張とスパンドレコードがサポートされました。 特に、実行中に読み書きされるレコードセグメント数を表示する 2 つの新しい統計が ADAULD ユーティリティにより生成されます。

7 コマンドの変更点

- ダイレクトコールの拡張 32
- 拡張 Adabas コントロールブロック（ACBX）のサポート 32
- Adabas バッファ記述（ABD）のサポート 38
- フォーマットバッファの変更点 39
- LF コマンドの変更点 45

このchapterでは、Adabas 8 になって、コマンド、ACB、および Adabas ダイレクトコールインターフェイスがどのように変更されたかについて説明します。 本書には、次のトピックが含まれています。

ダイレクトコールの拡張

Adabas 8 では、新しい Adabas 8 拡張 Adabas コントロールブロック（ACBX）構造と Adabas バッファ記述（ABD）構造に基づくダイレクトコールインターフェイスが導入されました。

Notes:

1. この機能拡張はすべて、従来の Adabas ACB ベースのダイレクトコールインターフェイスと互換性があります。
2. ACB ベースのダイレクトコールインターフェイスを使用する既存のアプリケーションプログラムは、変更なしで従来と同じ方法で実行できます。
3. さらに、ACBX ベースまたは ACB ベースのどちらのダイレクトコールインターフェイスをアプリケーションプログラムで使用するのかを、コールごとに決めることができます。 1 つのプログラムが両方のインターフェイスを使用できます。

一部の Adabas 8 の新機能では、アプリケーションプログラムが ACBX ベースのダイレクトコールインターフェイスを使用する必要があります。 例えば、アプリケーションプログラムで大きな（32K を超える）バッファを使用すること、またはセグメント化したバッファ（複数のフォーマットバッファとレコードバッファ）を使用することは、Adabas 8 でサポートされている機能であり、ACBX ベースのダイレクトコールインターフェイスを使用する必要があります。

新しい ACBX ダイレクトコールの詳細については、「*Adabas の呼び出し*」を参照してください。 ACBX 構造および ABD の詳細については、「*Adabas コントロールブロックの構造（ACB および ACBX）*」および「*Adabas バッファ記述（ABD）*」を参照してください。

拡張 Adabas コントロールブロック（ACBX）のサポート

新しく拡張 Adabas コントロールブロック（ACBX）が導入され、Adabas コマンドのバッファサイズを大きくできるようになりました。 従来の拡張型でない Adabas コントロールブロック（ACB）も今までどおりサポートされており、現行のアプリケーションもそのままで動作しますが、Adabas 8 の拡張機能を使用する場合には、ACBX に切り替える必要があります。 特に、Adabas 8 の長い（32K を超える）バッファまたはセグメント化されたバッファ（フォーマットバッファとレコードバッファの複数ペア、またはフォーマット、レコードおよびマルチフェッチの3つのバッファの複数セット）機能を使用している場合、ACBX を使用する必要があります。

ACBXを使用しない場合は、お使いのアプリケーションプログラムは既存のダイレクトコールインターフェイス（ACB）を使用したAdabasコールと新しいインターフェイス（ACBX）を使用したコールを自由に切り替えることができます。

このsectionでは、次のトピックについて説明します。

- Adabas 8 が ACB と ACBX のどちらを使用するかを区別する方法
- ACB と ACBX の相違点

Adabas でサポートされているコントロールブロック（構造と DSECT も含む）の詳細については、「*Adabas コントロールブロックの構造（ACB および ACBX）*」を参照してください。

Adabas 8 が ACB と ACBX のどちらを使用するかを区別する方法

すべてのアプリケーションプログラムは、ACB および ACBX のどちらのダイレクトコールも行えます。 コントロールブロック（ACB または ACBX）は、ACB または ACBX インターフェイスのどちらを使用しても、Adabasコールの最初のパラメータです。 Adabas 8では、コントロールブロックのオフセット2に "F" という文字で始まる値があるかどうかによって、コール時にどちらのコントロールブロックが使用されるかが決まります。 ACB のオフセット 2 は、コマンドコードフィールド（ACBCMD）ですが、F という文字で始まる Adabas コマンドはないため、ACBを使用するダイレクトコールでは、オフセット2に "F" で始まる値は含まれません。 ACBX内のオフセット2は、新しいACBXを識別する新しいバージョンのフィールド（ACBXVER）です。

オフセット 2 上の "F" の有無により、Adabas 8 がダイレクトコールを解釈する方法が異なります。 オフセット 2 に "F" が指定されている場合、Adabas は、コントロールブロックと残りのダイレクトコールパラメータを ACBX コールとして解釈します。オフセット 2 に "F" が指定されていない場合、Adabasは、コントロールブロックと残りのダイレクトコールパラメータをACBコールとして解釈します。 何らかの理由で残りのコントロールブロックフィールドおよびダイレクトコールパラメータが、オフセット2の "F" の有無により示されたコールのタイプに正しく指定されない場合は（ACB パラメータが ACBX コールに指定された場合など）、エラーとなるかコールの結果が予想どおりにならない場合があります。 新しい ACBX を使用してダイレクトコールを指定する方法については、「*ダイレクトコールの変更点*」を参照してください。

ACB と ACBX の相違点

ACBX は多くの点で ACB と異なります。 ACBX に含まれているフィールドには ACB にない新しいものもあり、また一部の ACBX フィールドのサイズは ACB の該当フィールドよりも大きくなっています。 ACBX におけるこれらの拡張は、基本的な構造を長期間にわたって変更することなく将来行われる予定のAdabasへの機能強化に対応できるように、ACBX構造に十分な柔軟性を持たせるために行われました。

このsectionでは、ACB と ACBX の相違点について説明します。

- コントロールブロック長
- バッファ長フィールド

- コマンドオプションフィールド、アディションフィールド、および予約フィールド
- 単位の相違
- フィールド長の相違
- ACBX の新規フィールド
- ACB 多目的フィールドの変化
- 構造およびオフセットの相違点

コントロールブロック長

新しい ACBX は長さが 192（X'C0'）バイトですが、ACB の長さは 80 バイトです。

バッファ長フィールド

バッファ長フィールドは ACB に含まれていますが ACBX には含まれていません。 Adabas 8 では、バッファ長フィールドを各 Adabas バッファ記述（ABD）に指定します。 したがって ACBX には、ACB に含まれる ACBFBL、ACBIBL、ACBRBL、ACBSBL および ACBVBL に対応するバッファフィールドが含まれません。コールに関連付けられた ABD が代わりに使用されます。 1 つの ABD は、単独の Adabas バッファセグメントを表します。 詳細は、「*Adabas バッファ記述*」で説明しています。

コマンドオプションフィールド、アディションフィールド、および予約フィールド

ACBX では、コントロールブロックのコマンドオプションフィールド、アディションフィールド、および予約フィールドの数が次のように増えています。

- ACBX には、8 個のコマンドオプションフィールドが含まれていますが、ACB では 2 個です。
- ACBX には、6 個のアディションフィールドが含まれていますが、ACB では 5 個です。
- ACBX には、4 個の予約フィールドが含まれていますが、ACB では 1 個です。

  予約された ACBX フィールドはバイナリの 0 に設定する必要があります。予約された 4 フィールド（ACBXRSV4）は、バイナリの 0 に初期化する必要があり、常にその状態を保つようにします。

単位の相違

コマンドタイム（スレッドタイム）を計測する単位は ACB と ACBX で異なります。 ACB では、コマンドタイム（ACBCMDT）を 16 マイクロ秒単位で計測します。ACBX では、コマンドタイム（ACBXCMDT）を 1/4096 マイクロ秒単位で計測します。

フィールド長の相違

ACBX では、多数のコントロールブロックフィールドが従来よりも長くなっています。 次の表は、これらの変更点を要約したものです。

フィールドタイトル	長さ	
	ACB	ACBX
ファイル番号	2	4
データベース ID	2	4
ISN	4	4
ISN 下限	4	4
ISN 数	4	4
圧縮レコード長	4	8
非圧縮レコード長	4	8
コマンドタイム	4	8
ユーザーエリア	4	16
フォーマットバッファ長	2	4（ABD 内）
レコードバッファ長	2	4（ABD 内）
サーチバッファ長	2	4（ABD 内）
バリューバッファ長	2	4（ABD 内）

ACBX の新規フィールド

ACBX に次のフィールドが新しく導入されました。

ACBX DSECT 名	説明
ACBXADD6	アディション 6
ACBXCOP3	コマンドオプション 3
ACBXCOP4	コマンドオプション 4
ACBXCOP5	コマンドオプション 5
ACBXCOP6	コマンドオプション 6
ACBXCOP7	コマンドオプション 7
ACBXCOP8	コマンドオプション 8
ACBXDBID	データベース ID です。 ACB では、データベース ID は X'30' コールではレスポンスコードフィールド（ACBRSP）内、その他の論理コールでは ACBFNR の 1 バイト目に格納されます。
ACBXERRA	バッファ内へのエラーオフセット（32 ビット）です。
ACBXERRB	エラー文字フィールド（フィールド名）です。
ACBXERRC	エラーサブコードです。

ACBX DSECT 名	説明
ACBXERRD	複数のバッファが関連する場合のエラーバッファ ID です。
ACBXERRE	複数のバッファが関連する場合のエラーバッファシーケンス番号です。
ACBXERRG	バッファ内へのエラーオフセット（64 ビット）です。このフィールドは現在サポートされていません。
ACBXLCMP	圧縮レコード長（レコード全体が読み込まれていない場合はレコードの一部）です。ACB では、圧縮レコード長はアディション 2 フィールド（ACBADD2）に格納されます。
ACBXLDEC	非圧縮レコード長です。 ACB では、非圧縮レコード長はアディション 2 フィールド（ACBADD2）に格納されます。
ACBXLEN	ACBX の長さ（現在 192）です。
ACBXRSV2	予約済みです。 このフィールドの値はゼロに設定されている必要があります。
ACBXRSV3	予約済みです。 このフィールドの値はゼロに設定されている必要があります。
ACBXRSV4	Adabas が使用するように予約済みです。
ACBXSUBR	Adabas アドオン製品により使用されるサブコンポーネントレスポンスコードです。
ACBXSUBS	Adabas アドオン製品により使用されるサブコンポーネントレスポンスサブコードです。
ACBXSUBT	Adabas アドオン製品により使用されるサブコンポーネントエラーテキストです。
ACBXVER	C'F2' に設定した場合、このフィールドは Adabas に新しい拡張 ACB（ACBX）が使用されていることを示します。

ACB 多目的フィールドの変化

ACB フィールドはいろいろな用途で使用できましたが、次に示すように、ACBX では用途に応じて新しいフィールドに分割されました。

- ACB では、X'30' コールのデータベース ID を格納するためにレスポンスコードフィールド（ACBRSP）が使用されます。 その他の論理コールの場合、1 バイトのデータベース ID は、ファイル番号フィールド（ACBFNR）の 1 バイト目に格納されていました。 ACBX では、この目的のためにデータベース ID フィールド（ACBXDBID）が提供されています。
- ACB では、特定の Adabas レスポンスコードのエラー情報を保持するために ACBADD2 フィールドが使用されます。 ACBX では、エラー情報を保持するために、エラー情報フィールド（ACBXERR で始まる一連のフィールド）が新しく作成されました。
- ACB では正常にコールを行うために、処理されたデータの圧縮および非圧縮レコード長を返すために ACBADD2 フィールドが使用されます。 ACBX では正常にコールを行うために、圧縮レコードフィールド（ACBXLCMP）に、Adabas により処理された圧縮データの長さが含まれます。また非圧縮レコードフィールド（ACBXLDEC）には、非圧縮データの長さが含まれます。

構造およびオフセットの相違点

次の表に示すように、ACBXフィールドのオフセットおよびシーケンスは、対応するACBフィールドとは全般的に異なります。 ACBX 構造の詳細については、「*Adabas コントロールブロックの構造（ACB および ACBX）*」を参照してください。

オフセット	ACB DSECT フィールド名	ACBX DSECT フィールド名
00	ACBTYPE（コールタイプ）	ACBXTYPE（コールタイプ）
01	予約	ACBXRSV1（予約 1）
02	ACBCMD（コマンドコード）	ACBXVER（ACBX バージョンインジケータ）
04	ACBCID（コマンド ID）	ACBXLEN（ACBX 長）
06	*（ACBCID の続き）*	ACBXCMD（コマンドコード）
08	ACBFNR（ファイル番号）	ACBXRSV2（予約 2）
0A	ACBRSP（レスポンスコード -- X'30' コールでデータベース ID に使用される）	ACBXRSP（レスポンスコード）
0C	ACBISN（ISN）	ACBXCID（コマンド ID）
10	ACBISL（ISN 下限）	ACBXDBID（データベース ID）
14	ACBISQ（ISN 数）	ACBXFNR（ファイル番号）
18	ACBFBL（フォーマットバッファ長）	ACBXISNG（8 バイト ISN）
1A	ACBRBL（レコードバッファ長）	*（ACBXISNG の続き）*
1C	ACBSBL（サーチバッファ長）	ACBXISN（ISN -- *ACBXISNG に含まれる*）
1E	ACBVBL（バリューバッファ長）	*（ACBXISN および ACBXISNG の続き）*
20	ACBIBL（ISN バッファ長）	ACBXISLG（8 バイト ISN 下限）
22	ACBCOP1（コマンドオプション 1）	*（ACBXISLG の続き）*
23	ACBCOP2（コマンドオプション 2）	*（ACBXISLG の続き）*
24	ACBADD1（アディション 1）	ACBXISL（ISN 下限 -- *ACBXISLG に含まれる*）
28	*（ACBADD1 の続き）*	ACBXISQG（8 バイト ISN 数）
2C	ACBADD2（アディション 2）	ACBXISQ（ISN 数 -- *ACBXISQG に含まれる*）
30	ACBADD3（アディション 3）	ACBXCOP1（コマンドオプション 1）
31	*（ACBADD3 の続き）*	ACBXCOP2（コマンドオプション 2）
32	*（ACBADD3 の続き）*	ACBXCOP3（コマンドオプション 3）
33	*（ACBADD3 の続き）*	ACBXCOP4（コマンドオプション 4）
34	*（ACBADD3 の続き）*	ACBXCOP5（コマンドオプション 5）
35	*（ACBADD3 の続き）*	ACBXCOP6（コマンドオプション 6）
36	*（ACBADD3 の続き）*	ACBXCOP7（コマンドオプション 7）
37	*（ACBADD3 の続き）*	ACBXCOP8（コマンドオプション 8）
38	ACBADD4（アディション 4）	ACBXADD1（アディション 1）

オフセット	ACB DSECT フィールド名	ACBX DSECT フィールド名
40	ACBADD5（アディション 5）	ACBXADD2（アディション 2）
44	*（ACBADD5 の続き）*	ACBXADD3（アディション 3）
48	ACBCMDT（コマンドタイム）	*（ACBXADD3 の続き）*
4C	ACBUSER（ユーザーエリア）	ACBXADD4（アディション 4）
54	---	ACBXADD5（アディション 5）
5C	---	ACBXADD6（アディション 6）
64	---	ACBXRSV3（予約 3）
68	---	ACBXERRG（バッファ内のエラーオフセット、64 ビット -- 現在サポートされていません）
6C	---	ACBXERRA（バッファ内のエラーオフセット、32 ビット）
70	---	ACBXERRB（エラー文字フィールド）
72	---	ACBXERRC（エラーサブコード）
74	---	ACBXERRD（エラーバッファ ID）
75	---	ACBXERRE（エラーバッファシーケンス番号）
78	---	ACBXSUBR（サブコンポーネントレスポンスコード）
7A	---	ACBXSUBS（サブコンポーネントレスポンスサブコード）
7C	---	ACBXSUBT（サブコンポーネントエラーテキスト）
80	---	ACBXLCMP（圧縮レコード長）
88	---	ACBXLDEC（非圧縮レコード長）
90	---	ACBXCMDT（コマンドタイム）
98	---	ACBXUSER（ユーザーエリア）
A8	---	ACBXRSV4（予約 4）

Adabas バッファ記述（ABD）のサポート

Adabas 8 では ACBX インターフェイスを使用すると、バッファのセグメント化（フォーマットバッファとレコードバッファの複数ペア、またはフォーマット、レコードおよびマルチフェッチの3つのバッファの複数セット）できるため、バッファの総数は固定ではなくなり、制限もなくなりました。 個別のバッファは ACBX のフィールド自身により記述されなくなりました（ACB ではバッファ長は ACB で定義されます）。その代わり、各バッファは独自の Adabas バッファ記述（ABD）構造を持ちます。ABD には、バッファの種類、場所、サイズ、およびその他の関連情報が記述されます。

ACBXを使用してABDのアドレスをAdabasへのダイレクトコールに指定することができます。このsectionでは、ABD の構造について説明します。

Adabas 8 では、次の 8 種類のバッファについて ABD を定義できます。

- フォーマットバッファ
- レコードバッファ
- マルチフェッチバッファ
- サーチバッファ
- バリューバッファ
- ISN バッファ
- パフォーマンスバッファ（Adabas Review で使用）
- ユーザーバッファ

各 Adabas バッファセグメントは単一の ABD で表されますが、同一のプログラムで 1 つの種類に対して複数の ADB を定義できます。 各 ABD のオフセット 4（ABDID）は、ABD で定義されたバッファの種類を識別します。

ABD（構造も含む）の詳細については、「*Adabas バッファ記述（ABD）*」を参照してください。

フォーマットバッファの変更点

Adabas 8 では、フォーマットバッファとフォーマットバッファ指定が次のように変更されました。

- 新しいフォーマットバッファインジータ（*L*）を使用して、LB フィールド値またはロング英数字（LA）フィールド値の実際の長さを取得したり、指定したりできるようになりました。 このフォーマットバッファエレメントは、*長さインジケータ*と呼ばれます。
- バッファのデータが 32K より大きくても、Adabas コマンドに指定できるようになりました。
- 英数字フィールドとワイド文字フィールドに対してフォーマットバッファ内で指定できるフィールド長は、253 バイトから 2,147,483,647 バイトまで拡張されました。
- ACBX インターフェイスを使用して Adabas のフォーマットバッファ、レコードバッファ、およびマルチフェッチバッファを Adabas ダイレクトコールに指定すると、これらのバッファは複数のセグメントに分割されるため、ストレージ内に確保される領域が連続している必要はなくなりました。
- フォーマットバッファでラージオブジェクト（LB）フィールドがサポートされるようになりました。 これに伴い、フィールド長にゼロを指定したり、アスタリスク（*）を使ったフィールド長表記を指定したりできるようになりました。 LB フィールドに対するフォーマットバッファのサポートについては、「*LB フィールドでサポートされるフォーマットバッファ*」を参照してください。

- Adabas 8 の拡張 MU/PE 制限を使ったファイルについては、2 バイトのオカレンスカウント値（例えば FB='MUC,2,B.'）を処理できるように、オカレンスカウントのフォーマットバッファ要素を調整（例えば FB='MUC.'）する必要があります。

このsectionでは、次のトピックについて説明します。

- 長さインジケータ（L）
- 複数の Adabas バッファセグメントを指定する際のルール

長さインジケータ（L）

新しいフォーマットバッファインジータ（*L*）を使用して、LA または LB フィールド値の実際の長さを取得したり、指定したりできるようになりました。 このフォーマットバッファエレメントは、*長さインジケータ*と呼ばれます。

Note: 現在、長さインジケータは、LA または LB フィールドのフォーマットバッファ指定にのみ使用できます。 他のフィールドにおける長さインジケータの使用のサポートは、Adabas の今後のリリースで導入される予定です。

長さインジケータは、フィールド名に続いて文字 *L* を使用して指定します。例えば、`FB='ACL,4,B.'` と指定すると、AC フィールドの長さが返されます。 フィールドがマルチプルバリューフィールドであるか、ピリオディックグループ内にある場合は、フィールド名および文字 L の後に関連するオカレンスインデックスが続きます。例えば、`FB='ACL2,4,B.'` と指定すると、マルチプルバリューフィールド AC の 2 番目の値が返されます。 圧縮フィールド長は、4 バイトのバイナリ形式で返されます。 他の長さおよび形式は指定できません。

このsectionでは、次のトピックについて説明します。

- MU/PE フィールドにおける長さインジケータの使用
- 読み込みコマンドにおける長さインジケータの使用
- 更新コマンドにおける長さインジケータの使用

MU/PE フィールドにおける長さインジケータの使用

長さインジケータを MU または PE フィールドで使用する場合、オカレンスインデックスを指定する必要があります。オカレンスインデックスの範囲を指定することもできます。 ただし、1-N オカレンスインデックスは指定できません。 以下に例を示します。

1. 次の例では、ピリオディックグループフィールド AC の 2 番目のオカレンスの 5 個目の値の長さが返されます。

```
FB='ACL2(5).'
```

2. 次の例では、マルチプルバリューフィールド AC の最初の 10 個の値の長さが返されます。

```
FB='ACL1-10.'
```

3. 1-N 表記は MU または PE フィールドの長さインジケータでは許可されていないため、次の例は正しくありません。

```
FB='ACL1-N.'
```

4. 長さインジケータはオカレンスインデックスを持たない MU フィールドをサポートしないため、次の例も正しくありません。

```
FB='MCL.'
```

また、MU または PE フィールドの同一のフォーマットバッファ内で、長さインジケータとアスタリスク長さ表記による値の要求を組み合わせる場合、値の要求には長さの要求に対応する範囲を使用する必要があります。 長さの要求と値の要求は同一のフォーマットバッファセグメントでも、異なるフォーマットバッファセグメントのどちらで指定しても構いません。 例として、XX が、MU オプション付きの LA または LB フィールドである場合を以下に示します。

1. 次の有効な例では、XX フィールドの最初の 2 個の値の長さと実際の値を要求しています。

```
FB='XXL1-2,XX1-2,*.'
```

```
FB='XXL1,XXL2,XX1,*,XX2,*.'
```

2. 次の*無効*な例では、XX フィールドの最初の 2 個の値の長さと実際の値を要求しようとしています。 ただしこの例では、MU フィールドに対して長さの要求に指定した範囲と、値の要求に指定した範囲が、対応する形式で指定されて異なため、無効となります。

```
FB='XXL1,XXL2,XX1-2,*.'
```

```
FB='XXL1-2,XX1,*,XX2,*.'
```

3. 次の 2 つのフォーマットバッファは、XX フィールドの 3 番目および 4 番目の値の長さとその実際の値を要求しています。

```
FB='XXL3,XXL4.'
```

```
FB='XX3,*,XX4,*.'
```

4. 次の無効なフォーマットバッファは、XX フィールドの 3 番目および 4 番目の値の長さとその実際の値を要求していますが、長さの要求に指定した範囲と、対応する形式で指定されていないため、エラーになります。

```
FB='XXL3,XXL4.'
```

```
FB='XX3-4,*.'
```

読み込みコマンドにおける長さインジケータの使用

読み込みコマンドのフォーマットバッファ内のフィールドに長さインジケータを指定する場合、レコードバッファにフィールド値を格納するために必要なバイト数（パディングがなく、長さに関する情報を含まないバイト数）が、レコードバッファ内の対応するフィールド位置に返されます。 レコードバッファに必要なスペースは、フィールドフォーマット、データベース、ファイルおよびユーザーの UES に関連する定義により異なります。

同一のフォーマットバッファ内で、LA または LB フィールドの長さを取得する長さインジケータを指定する際に、そのフィールドの基本フォーマットと異なる（変換された）フォーマットで、実際のフィールド値を要求することはできません。 例えば、文字の LB フィールド L1 がフォーマット A で格納されている場合、フォーマットバッファ指定 `FB='L1L,4,B,L1,*,W.'` は、L1 フィールドの長さの要求に加え、Unicode（フォーマット W）に変換された L1 フィールドの値を要求しているため、正しくありません。 長さ要素はフィールドの長さをフィールド固有のフォーマットで指定しますが、要求されたフォーマット（Unicode）で返される値の長さが変換により異なるため、このような制限があります。

文字 LB フィールド L1（フォーマット A）が 40,000 バイトの EBCDIC 値を含んでいる場合を次の例に示します。

1. L1 のフォーマットバッファ指定が次のようであると仮定します。

```
FB='L1L,4,B.'
```

レコードバッファには、4 バイトバイナリ長の L1 フィールドの値が含まれます。

```
X'00009C40'
```

2. L1 のフォーマットバッファ指定が次のようであると仮定します。

```
FB='L1L,4,B,L1,*,A.'
```

レコードバッファには 4 バイトバイナリ長の L1 フィールドの値がレコードバッファエリアの先頭に含まれ、次に、40,000 文字の実際の L1 データが含まれます。

読み込み操作、長さインジケータ、および **NB**（空白圧縮なし）オプション

フォーマットバッファ内のフィールドが、対応する長さインジケータを使用して指定されていて（例えば `FB='L1L, 4,B,L1.'` など）、フィールドが空白圧縮の*対象外*（NB オプションが FDT 内のフィールドに*指定されている*）の場合、返される長さは、値が格納されたときに指定されたバイト数となります。 ただし、フィールドが空白圧縮の対象となっている場合、返される長さは左端から意味を持つバイト数のみが返され、それ以上の値は空白で埋められます。

読み込み操作、長さインジケータ、および **NU**（空値省略）オプション

フォーマットバッファ内のフィールドが、対応する長さインジケータを使用して指定されていて（例えば `FB='L1L, 4,B,L1.'` など）、フィールドが空値省略（FDT 内でこのフィールドに NB オプションが指定されている）でフィールド値が完全に空白の場合、返されるフィールド値の長さはゼロになります。フィールドが空値省略ではない場合、返されるフィールド値の長さは、1 つの空白の長さ（英数字フィールドで 1 バイト、ワイド文字フィールドで 2 バイト）になります。

更新コマンドにおける長さインジケータの使用

更新コマンドのフォーマットバッファに長さインジケータが指定されている場合、レコードバッファの対応する値がレコードバッファ内のフィールドの実際の値の長さを表します。基本フィールドの長さインジケータ 1 個のみが指定でき、このインジケータはフォーマットバッファでアスタリスク（*）表記を伴う必要があります。

長さインジケータは、レコードバッファに可変長を指定するどのフォーマット要素（アスタリスク表記または長さゼロ表記による）より先にフォーマットバッファセグメント内で出現する必要があります。 つまり、長さインジケータの場所は常に同じで、このフォーマットで指定されるどのフィールドの値からも影響を受けません。

複数の Adabas バッファセグメントを指定する際のルール

ACBXインターフェイスを使用してAdabasのフォーマットバッファ、レコードバッファ、およびマルチフェッチバッファをAdabasダイレクトコールに指定すると、これらのバッファは複数のセグメントに分割されるため、ストレージ内に確保される領域が連続している必要はなくなりました。 サーチバッファ、バリューバッファ、ISN バッファ、モニタバッファ、ユーザー情報バッファはそれぞれ、旧リリースと同様に個別のセグメントに指定する必要があります。

ACBXのAdabasダイレクトコールでは、複数セグメントのフォーマットバッファに次のルールが適用されます。

- 複数のフォーマットバッファとレコードバッファは、通常、対になります。レコードバッファ内のデータを、フォーマットバッファ内のフォーマット指定では*記述していない*コマンドの場合は、フォーマットバッファセグメントの指定は必要ありません。指定した場合は無視されます。
- マルチフェッチ機能を使用すると、1 回の Adabas コールで複数のレコードを読み込むことができます。 マルチフェッチ機能を使用する場合、フォーマット、レコード、およびマルチフェッチの各バッファセグメントを 3 つ 1 組で指定する必要があります。
- 各フォーマットバッファセグメントは、以前のリリースと同様に、ピリオドで終了し、単独のフォーマットバッファとして完結した有効なものである必要があります。

  フォーマット ID をコールに指定した場合は、そのコールは該当するすべてのフォーマットバッファセグメントに関連付けられます。 後続のコールでも類似したフォーマットバッファ（バッファセグメントのシーケンスを含む）を指定する場合、そのコールに別のフォーマットIDを指定するか、またはそのフォーマットIDをいったん解放してから以前に使用したフォーマット ID を再定義する必要があります。
- Adabas ダイレクトコールにバッファまたはバッファセグメントのシーケンスを指定する方法は、次に示すように、コマンドのタイプによって異なります。

  1. ACBXの後には、ゼロ個以上のフォーマットバッファセグメント（通常はレコードバッファセグメントと同じ数）、同じ数のマルチフェッチバッファセグメント（マルチフェッチ機能を使ったコールの場合）が続きます。

     レコードバッファ内のデータを、フォーマットバッファ内のフォーマット指定では*記述していない*コマンドの場合は、フォーマットバッファセグメントの指定は必要ありません。指定した場合は無視されます。
  2. Adabasコマンドのタイプによっては、サーチバッファセグメント、バリューバッファセグメント、または ISN バッファセグメントが 1 つ、その後に続きます。
  3. コールの最後には、0～1 個のパフォーマンスバッファセグメント、または 0～1 個のユーザー情報バッファセグメントを 1 つ付加することができます。

LF コマンドの変更点

コマンドオプション 2 に "S" を設定して LF コマンドを実行したときに、ラージオブジェクト（LB）フィールドが検出されると、LB フィールドの状態が F タイプのフィールドエレメントに返されます。 第 2 フォーマットバイト（エレメントのオフセット 7 またはバイト 8）のビット 6 が、このフィールドに LB（ラージオブジェクト）オプションが設定されたことを示すために設定されました。 さらに、第2フォーマットバイトのビット1は、LBフィールドが、NBオプション付きで定義されているかどうかを示します。 詳細については、『*Adabas コマンドリファレンス*』の LF コマンドに関する記述を参照してください。

8 ユーザー出口の変更点

- ユーザー出口 1 48
- ユーザー出口 11 48
- ハイパー出口の変更点 49

このchapterでは、Adabas 8 のユーザー出口の変更点について説明します。

ユーザー出口 1

ユーザー出口 11 の導入に伴い、ユーザー出口 1 はサポートされなくなりました。 ただし、移行を簡単にするため、サンプルユーザー出口 UEX11UX1 が提供され、既存のユーザー出口 1 の前に挿入してユーザー出口 11 として起動させることができます。 このサンプルは、ACB ダイレクトコールインターフェイスを使用して行われるダイレクトコールでのみ動作します。Adabas 8 ACBX ダイレクトコールインターフェイスを使用したダイレクトコールでは動作しません。「*ユーザー出口**11***」の説明にあるように、このユーザー出口は、ユーザー出口 11 の制約を受けます。 特に、変更可能なフィールドは、ファイル番号（CQEFNR）、アディション 2（ACBADD2）、アディション 3（ACBADD3）、およびユーザーエリア（ACBUSER）の各フィールドに制限されます。 他の ACB フィールドを変更しても、CQE に他の変更を加えても、ニュークリアスからは無視されます。 詳細は、サンプルユーザー出口内のコメントを参照してください。既存のユーザー出口 1 とのリンクの方法などが記述されています。 Adabas ニュークリアスの、この移行支援サポートは将来の Adabas リリースでは廃止される予定です。

ユーザー出口 11

新しく導入されたユーザー出口は、Adabasニュークリアスがコマンドを受け取った直後に Adabas から制御を受け取ります。 コマンドのタイプ（シンプルアクセス、コンプレックスアクセス、更新）の判別には使用されますが、コマンド自体は処理されません。

このユーザー出口の最も一般的な使用法としては、ACBX へのセキュリティパスワードやサイファコードの挿入があります。

このユーザー出口の機能は、従来のユーザー出口 1 とほぼ同じですが、CQE 構造と ACBX データ構造のコピーがユーザー出口 11 の処理中に使用される点が異なります。ユーザー出口 1 で使用される実際の構造は、ユーザー出口 11 では使用されません。 ACBX の特定フィールドのみが出口により変更される場合があります。この特定フィールドとしては、ACBXFNR（ファイル番号）、ACBXADD2（アディション 2）、ACBXADD3（アディション 3）および ACBXUSER（ユーザーエリア）が該当します。 他の ACBX フィールドを変更しても、CQE を変更しても、ニュークリアスからは無視されます。 DSECT EX11PARM は、ユーザー出口 11 のパラメータリストをマッピングします。

ハイパー出口の変更点

今回のリリースでは、ハイパー出口のロジックが変更され、ハイパーディスクリプタ指定に拡張MU/PE フィールドを含むことができるようになりました。 Adabas 8 ニュークリアスは、従来のハイパー出口が指定されているかどうかを確認し、従来のパラメータリストを検出するとレスポンスコードを返します。

さらに、Adabas 8 にはハイパー出口スタブが含まれており、既存のハイパー出口は Adabas 8 パラメータリストを変更せずに使用することができます。ハイパー出口スタブは、新しい Adabas 8 パラメータエリアを使用するために既存のハイパー出口をすぐに更新したくないお客様に向けた一時的な解決方法として用意されています。 Adabas 8 におけるすべてのハイパー出口のサポートについては、「*ハイパーディスクリプタ出口 01 - 31*」を参照してください。

9 Adabas 8 対応のアドオン製品のサポート

- クライアントベースのアドオン製品の互換性に関する全般的な情報 52
- Adabas System Coordinator バージョン 8 のサポート 54
- Adabas Fastpath バージョン 8 のサポート 57
- Adabas SAF Security バージョン 8 のサポート 58
- Adabas Transaction Manager バージョン 8 のサポート 59
- Adabas Vista バージョン 8 のサポート 59

このchapterでは、バージョン 8 でアドオン製品がどのようにサポートされるのかについて説明します。

Caution: ここに記載されている内容は、製品リリースの前に変更される可能性があります。 Software AG は、リリース前に本製品および本ドキュメントの内容を変更する権利を留保します。また、記載されている機能がすべて実装されているとは限りません。 しかし、この情報はバージョン8へのアップグレードを正しく評価するためには十分に的確なものです。

クライアントベースのアドオン製品の互換性に関する全般的な情報

クライアントベースのアドオン製品としては、次のものがあります。

- Adabas Fastpath
- Adabas SAF Security
- Adabas Transaction Manager
- Adabas Vista
- Adabas System Coordinator

Adabas 8 の基本的な拡張機能はほとんど、これらのアドオン製品には直接、影響しません。 つまり、アップグレードの際に、Adabas 8 拡張機能が障害になることはありません。 例えば、スパンドレコードのような内部的な Adabas 8 の特性や、またはユーティリティの内部仕様の変更は、これらの製品には影響しません。

これらの製品では、必要に応じて、Adabas の新機能や変更された機能が次のようにサポートされています。

*特定の **Adabas 8** 機能のサポート*	特定の Adabas 8 機能が現行のアドオン製品の機能に直接関係する場合、該当するアドオン製品は、Adabas 8 の機能拡張をサポートするように機能が追加されます。
***Adabas 8** 機能の対応状況*	特定の Adabas 8 機能が現行のアドオン製品の機能に直接的には*関係しない*場合、その Adabas 8 機能は制限なく使用できますが、アドオン製品からは使用できないことがあります。

このsectionでは、次のトピックについて説明します。

- 特定の Adabas 8 機能のサポート
- Adabas 8 機能の対応状況
- Adabas 8 のサポートに必要な作業

- Adabas 7.4 の互換性

特定の Adabas 8 機能のサポート

クライアントベースのアドオン製品では、次の Adabas 8 機能がサポートされています。

- 新しい ACBX 呼び出しプロトコル
- 新しい FDT レイアウト
- ラージオブジェクト（LB）フィールド（必要な場合）
- MU/PE 上限の拡大（必要な場合）

Adabas 8 機能の対応状況

その他の Adabas 8 機能は、クライアントベースのアドオン製品には特に影響がないため、これらの製品とともに使用できます。

Adabas 8 のサポートに必要な作業

必要に応じて、Adabas のバージョンレベルが自動的に検知されるため、クライアントベースのアドオン製品側では、Adabas 8 をサポートするための操作は特に必要ありませんが、 これとは別に変換作業が必要になります。

具体的な例としては、クライアントベースのアドオン製品で共有されるシステムコンフィグレーションファイルがあります。システムコンフィグレーションファイルには、実行時にソフトウェアで効率的な処理を行うために必要な情報がすべて格納されています。システムコンフィグレーションファイルは、次の手順に従って、以前のリリースから変換する必要があります。

- バージョン 8 レベルに応じた新しいファイルを使用する必要があります。
- 変換ユーティリティを使用すると、バージョン 7.4 ファイル形式の内容を新しいバージョン 8 ファイル形式に変換することができます。
- 変換処理に失敗した場合は、新しいファイルを空の状態にしてから、変換処理をやり直す必要があります。

Adabas 7.4 の互換性

次の条件に合致する場合、バージョン 8.1 クライアントベースのアドオン製品は、Adabas 7.4 と Adabas 8 のデータベースと併用することができます。

- 処理対象が Adabas 7.4 データベースの場合は、Adabas 8 機能は*使用できません*。
- バージョン 8.1 クライアントベースのアドオンを、Adabas 7.4 やバージョン 7.4 互換の Adabas アドオンと連動させるためには、修正版が必要になることがあります。この修正版は、クライアントベースのアドオン製品の販売に合わせて公開されます。

- クライアントベースのアドオン製品は、同じバージョンレベルに揃える必要があります。 例えば、Adabas Fastpath 7.4 と Adabas Vista 8.1 の機能を併用することはできません。

Adabas System Coordinator バージョン 8 のサポート

Adabas System Coordinator は、Adabas に対応したクライアントベースのアドオン製品です。 Adabas System Coordinator およびその他のクライアントベースのアドオン製品のバージョン 8 への移行についての一般的な情報は、「*クライアントベースのアドオン製品の互換性に関する一般情報*」を参照してください。

Adabas System Coordinator は、Adabas Fastpath、Adabas SAF Security、Adabas Transaction Manager、および Adabas Vista に基本テクノロジを提供します。 このため、これらのその他の製品が高度に統合化されるように、いくつかの機能拡張が Adabas System Coordinator に実装されました。

このsectionでは、次のトピックについて説明します。

- **クライアントのランタイムコンフィグレーション**
- **機能向上したダイナミッククライアントランタイム設定**
- **システムコンフィグレーションファイルの変更**
- **可変長データのサポート**
- **バージョン管理機能**
- **他のクライアントベースのアドオン製品を使用せずに Adabas System Coordinator をクライアントジョブに適用する**

クライアントのランタイムコンフィグレーション

同じジョブの中で Adabas クライアントセッションを別々に処理する必要性が高くなってきました。 例としては、次のような場合があります。

- CICSXYZ 内のクライアントのうち、クライアント "ABC" だけは、特殊なトレースコントロールを使用する必要がある。
- CICSXYZ のトランザクション "D412" は、他のトランザクションよりも短いタイムアウト設定で処理できるようにする必要がある。
- ジョブ PRODA032 のステップ名 "S0010" は、Adabas System Coordinator を使用しないようにする必要がある。

このようなランタイム制御を実行できることが、非常に重要になってきました。例えば、トレースオプションの対象は、全体ではなく、ごく少数のセッションに限定することができます。 これにより、メモリ消費量全体を最小に抑えられ、同時に調査対象のセッションのみをチェックして問題を集中的に追求できるようになります。

Adabas System Coordinator バージョン 8.1 では、本来の基本ジョブレベル制御に、任意指定の上書き制御を追加することで、これらの設定をあらかじめ指定することができます。 バージョン 8.1 では、次の事項について、上書き制御を事前に設定することができます。

ジョブタイプ	上書き制御
バッチジョブ	1. ステップ名 2. LOGIN（例えば RACF LOGIN ユーザー ID） 3. 特別な API
TSO、CMS、TIAM など	特別な API
COM-PLETE、CICS、IMS、UTM	1. 特別な API 2. LOGIN 3. トランザクションコード

端末オペレータが 1 つのトランザクションから別のトランザクションに移動するときは、コンフィグレーションファイルに指定した内容に従って、実行時の処理を動的に変更することができます。

機能向上したダイナミッククライアントランタイム設定

異なる設定を事前指定して実行時に選択できるだけでなく、現在のセッションのランタイム制御を動的に変更できるようになりました。 例えば、コンフィグレーションファイルの指定内容に関係なく、トレースのオン／オフを切り替えることができます。

Natural では、"SETCOR *xxx*" 機能を使用して、このような動的変更が行われます。*xxx* の部分は、AFP、AVI、ATM または COR という製品コードで置き換えられます。

システムコンフィグレーションファイルの変更

システムコンフィグレーションファイルは、ランタイム処理で非常に重要な役割を担います。したがって、このファイルに問題があるとシステム全体に影響が波及します。 このため、Adabas System Coordinator 8.1 では、代替のコンフィグレーションファイルを定義できるようになりました。

Note: 2 つのコンフィグレーションファイルの内容が常に同じになるように、ユーザー側でファイルを維持する必要があります。

各セッションは、通常、プライマリのシステムコンフィグレーションファイルを使用しますが、そのファイルが使用できない場合は、代替のシステムコンフィグレーションファイルを使用します（定義されている場合）。セッションでコンフィグレーションファイルが認識されると、セッションが継続している間、そのファイルはプライマリのコンフィグレーションファイルとして扱われますが、そのファイルが使用できなくなると、代替のコンフィグレーションファイルが使用

されます。 長期的に見ると、別々のセッションが同時に異なるファイルを使用する可能性があります。どちらかのコンフィグレーションファイルを長期間使用できないようにして、すべてのセッションが使用しているコンフィグレーションファイルを強制的に切り替えるまで、この状態が続きます。

代替のコンフィグレーションファイルを使用する場合、プライマリと代替のコンフィグレーションファイルを両方とも、Coordinatorデーモンの起動時と終了時に使用できるようにする必要があります。このファイルにはリカバリ情報と再開情報が保存されるため、同じ情報が両方のファイルに保存されないと、ファイル間の整合性が取れなくなるからです。

可変長データのサポート

コンフィグレーションファイルに独自の可変長データを保存できるようになりました。このデータは、基本設定または上書き設定と同時に入力することができます。 可変長データを使用して説明を入力できるだけでなく、ランタイムAPIコールを使用すれば、入力したデータ（最大256バイト）を取得することもできます。 したがって、使用する設定に応じて、目的に合わせて実行時のセッションを簡単に区別することができます。

バージョン管理機能

バージョン管理機能を使用すると、バージョン8.1に完全に移行するまで段階的にバージョンをアップグレードすることができます。 アップグレードの対象は、このソフトウェアが組み込まれているクライアントとデータベースです。 TP システムには、ADALNK テクノロジに対するフロントエンドが組み込まれており、例えば ADALNK74 と ADALNK81 を同じクライアントジョブで使用することが可能です。 ADALNKパスは、デフォルトでトランザクションコードに従って選択されます。 この方式を使用すると、段階的な変換が可能です。

Adabas ターゲットデータベースのソフトウェアにも同様の方式が採用されています。 例えば、Adabas System Coordinator および Adabas Fastpath 7.4 を Adabas System Coordinator および Adabas Fastpath 8.1. と併用することができます。これにより、クライアントは7.4から8.1まで一度に移行することが可能です。

他のクライアントベースのアドオン製品を使用せずにAdabas System Coordinatorをクライアントジョブに適用する

Adabas System Coordinatorの通常機能の中には、他のクライアントベースのアドオン製品を使用しなくても、ジョブに適用できる機能があります。 例えば、コマンド再試行機能がジョブに必要なことがありますが、Adabas Fastpathは必要ありません。 Adabas System Coordinator 8.1は、他の製品がなくても稼動できます。

Adabas Fastpath バージョン 8 のサポート

Adabas Fastpath は、Adabas に対応したクライアントベースのアドオン製品です。 Adabas System Coordinator およびその他のクライアントベースのアドオン製品のバージョン 8 への移行についての一般的な情報は、「*クライアントベースのアドオン製品の互換性に関する一般情報*」を参照してください。

Adabas System Coordinator は、Adabas Fastpath、Adabas SAF Security、Adabas Transaction Manager、および Adabas Vista に基本テクノロジを提供します。 このため、これらのその他の製品が高度に統合化されるように、いくつかの機能拡張が Adabas System Coordinator に実装されました。 詳細は、「***Adabas System Coordinator** サポート*」を参照してください。 Adabas Fastpath は、Adabas System Coordinator 8.1 で導入されたダイナミッククライアントランタイム設定に対応しています。

このsectionでは、次のトピックについて説明します。

- **バッファ自動再スタート機能の拡張**
- **保護ファイルに対するダイレクトアクセスの最適化**
- **AFPLOOK の変更点**

バッファ自動再スタート機能の拡張

バッファ自動再スタート機能が拡張され、"n 時間おきに再スタートするだけでなく、時刻が n 時になると再スタート" できるようになりました。

Adabas Fastpath のバッファが 24 時間無休で常時稼動している環境では、コマンドの統計データが現行の 4G の制限を超えてしまい、出力しても意味のないデータになってしまうため、これを回避できるように機能が拡張されました。 この機能を使用すると、制限の超過が発生する標準の時間帯になる前に、再スタートを実行することができます。 また、実際の再スタート時間をシステムの稼働率が低い時間帯に設定できるため、ピーク時を避けて再スタートを実行することもできます。

今後のリリースでは、4G を超える統計データが作成できるようになる予定です。

保護ファイルに対するダイレクトアクセスの最適化

Adabas Fastpath のデフォルトの動作では、保護ファイルはキャッシュされません。これは、セキュリティ上、データがどのセッションで使用されているかを区別できるようにする方が望ましいという前提に立っています。 しかし、環境によっては、このようにデータを保護せずに、保護ファイルをキャッシュする方が要件に合っていることがあります。 その場合でも、この機能は注意して使用してください。

AFPLOOK の変更点

AFPLOOKサンプラによる予測は、現行レベルのAdabas Fastpathの最適化に適合するように変更されました。また、オンラインの管理ツールを使用しなくても、Adabasに付属のAFPLOOKバージョンを事前設定することができるようになります。

Adabas SAF Security バージョン 8 のサポート

Adabas SAF Security は、Adabas に対応したクライアントベースのアドオン製品です。 Adabas System Coordinator およびその他のクライアントベースのアドオン製品のバージョン 8 への移行についての一般的な情報は、「*クライアントベースのアドオン製品の互換性に関する一般情報*」を参照してください。

このsectionでは、次のトピックについて説明します。

- 可変長リソースの名前変更
- ストアドプロシージャの保護
- ホールドベースのコマンド
- Adabas Security
- Adabas SAF Security の非アクティブ化

可変長リソースの名前変更

Adabas SAF Security 8.1 では、保護されているリソースの名前として、番号ではなく、論理名を使用できるようになります。 したがって、リソースの集合を作成しやすくなります。

ストアドプロシージャの保護

PC コールの実行を保護するオプションが導入されます。

ホールドベースのコマンド

アクセスコマンドではなく更新コマンドとして、ホールドベースのコマンド（L4、L5 など）を利用できるオプションが導入される予定です。

Adabas Security

ADASCRパスワードとしてSAFベースのGROUP IDを使用できるオプションが導入されます。

Adabas SAF Security の非アクティブ化

条件によっては、Adabas SAF Security がデータベースの初期化に失敗する可能性があります。この場合、データベースごとにAdabasニュークリアスを終了させるかどうかを設定できる方が便利なこともあります。 これを実現する機能が、Adabas SAF Security バージョン 8.1 に導入される予定です。

Adabas Transaction Manager バージョン 8 のサポート

Adabas Transaction Manager は、Adabas に対応したクライアントベースのアドオン製品です。Adabas System Coordinator およびその他のクライアントベースのアドオン製品のバージョン 8 への移行についての一般的な情報は、「*クライアントベースのアドオン製品の互換性に関する一般情報*」を参照してください。

Adabas System Coordinator は、Adabas Fastpath、Adabas SAF Security、Adabas Transaction Manager、および Adabas Vista に基本テクノロジを提供します。 このため、これらのその他の製品が高度に統合化されるように、いくつかの機能拡張がAdabas System Coordinatorに実装されました。 詳細は、「***Adabas System Coordinator** サポート*」を参照してください。 Adabas Transaction Manager は、Adabas System Coordinator 8.1 で導入されたダイナミッククライアントランタイム設定に対応しています。

このsectionでは、次のトピックについて説明します。

- **非アクティビティタイムアウト処理**

非アクティビティタイムアウト処理

この処理は、Adabas Transaction Manager 固有の機能ではなく、Adabas System Coordinator に用意されている同等の通常機能によって実行されます。

Adabas Vista バージョン 8 のサポート

Adabas Vista は、Adabas に対応したクライアントベースのアドオン製品です。 Adabas System Coordinator およびその他のクライアントベースのアドオン製品のバージョン 8 への移行についての一般的な情報は、「*クライアントベースのアドオン製品の互換性に関する一般情報*」を参照してください。

Adabas System Coordinator は、Adabas Fastpath、Adabas SAF Security、Adabas Transaction Manager、および Adabas Vista に基本テクノロジを提供します。 このため、これらのその他の製品が高度に統合化されるように、いくつかの機能拡張が Adabas System Coordinator に実装されました。 詳細は、「*Adabas System Coordinator サポート*」を参照してください。 Adabas Vista は、Adabas System Coordinator 8.1 で導入されたダイナミッククライアントランタイム設定に対応しています。

このsectionでは、次のトピックについて説明します。

- 正式版とドラフト版に代わる世代管理
- ファイル変換ページ
- ソース名の廃止
- パーティション分割ファイルのターゲットカテゴリの廃止

正式版とドラフト版に代わる世代管理

旧バージョンの Adabas Vista では、個別ファイルのルールとして、正式版とドラフト版の作成が可能でした。 したがって、変更が加えられても、そのランタイムルールがアクティブなまま存在している状態でしたが、 実行可能なランタイムルールとして多数のドラフトルールを一度にアクティブにすることはできませんでした。

ルールを*世代*ごとに管理できれば、完全な新しいルールの組み合わせをバックグラウンドで作成し、同時にアクティブ化することが可能になります。 これは、ランタイム処理に変更を適用する際に、非常に優れた制御方式です。

複数世代のルールを同時に作成できるだけではなく、特定の世代を、アクティブな世代として指定することもできます。 また、いつでもアクティブな世代として別の世代を定義することができます。 この場合、元のアクティブな世代は自動的にアクティブではない状態に設定されます。

世代の基本ルールは直接、変更することはできません。 世代を変更する場合は、*差分修正*を定義する必要があります。 1 つの世代に対して複数の差分を定義することもできます。 差分修正を定義し終わったら、その差分修正を*適用*して、基本世代を更新します。

ファイル変換ページ

Adabas Vista 8.1 では、複数ページの変換ルールを定義できるようになりました。 データベース管理者は、ページごとに名前を付けることができます。 各ページは、1 つ以上のファイル変換ルールで構成されます。 Adabas Vista を使用するように定義されているジョブには、そのジョブの実行時に使用する名前の付いたページを 8 つまで含めることができます。 Adabas Vista は、これらのページをすべて 1 つの変換ルールにまとめ、実行時に使用できるようにします。

ルールは必須のものとして定義することができます。また、ルールに優先順位の番号を付けることもできます。 マージ処理では、この必須設定と優先設定が保持されます。

上記に加えて、必須ページ自体を設定することも可能です。その場合、そのページに含まれる変換ルールが必須である必要はありません。 必須ページは、存在する場合、必ず使用されます。

Adabas Vista のジョブ制御では、必須ページに名前を付ける必要はありません。名前を付けない方が、旧バージョンとの互換性が高くなります。 Adabas Vista 8.1 変換ユーティリティでは、既存の 7.4 ルールに従って変換できるように処理されます。

ソース名の廃止

ソース名フィールドは、変換ルールから削除されました。 Adabas Vista 8.1 変換ユーティリティでは、これらの変換ルールが自動的に除外されます。

パーティション分割ファイルのターゲットカテゴリの廃止

ターゲットカテゴリは、パーティション分割ファイルの定義から除外されました。 ターゲットカテゴリは、変換ルールにのみ使用されます。

索引

A

Adabas
    全般的な情報, 3

す

スパンドレコード
    識別, 12

せ

全般的な情報, 3

ち

チェックポイント
    Adabas ニュークリアス／ユーティリティによる書き込み, 30

ふ

フォーマットバッファ
    構文
        長さインジケータ（L）, 40