software AG

Adabas

トリガとストアドプロシージャ

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 トリガとストアドプロシージャ

構成

このマニュアルでは、トリガとストアドプロシージャを実装および保守する Adabas 機能のインストールと使用に必要なすべての情報について説明します。

マニュアルは、次の項目で構成されています。

はじめに	プロシージャの概要と、プロシージャのプログラミングと処理に関する特徴について説明します。 プロシージャが処理される方法について説明します。
インストールと設定	インストールと設定について説明します。
処理とパフォーマンス	プロシージャの実行時処理について詳細に説明し、処理シーケンスを通じてのパフォーマンスについて説明します。
プログラミングとパフォーマンス	パフォーマンスの問題など、プロシージャの作成に関する問題について説明します。
ストアドプロシージャの呼び出し	ストアドプロシージャのリンクルーチン STPLNKnn とともに PC コマンドを使用して、ストアドプロシージャを呼び出す方法について説明します。
トリガメンテナンス	Adabas Online System（AOS）のメインメニューからアクセスできるオンラインのトリガメンテナンス機能を使用する方法について説明します。 ユーザーと管理者の両方に関する情報があります。
TRGMAIN	ユーザープログラムからトリガを保守するための TRGMAIN API について説明します。
TRGUNLD と TRGLOAD ユーティリティ	トリガ定義をワークファイルにロード解除してからトリガファイルにロードするための TRGUNLD と TRGLOAD ユーティリティについて説明します。
プログラミング例	サンプルプログラムのリストに注釈を付けて説明します。

メッセージおよびコード

トリガとストアドプロシージャに関するメッセージとコードの解釈、およびそれらのメッセージやコードで特定される問題の解決については、『*Adabas メッセージおよびコードマニュアル*』を参照してください。SYSTRG メッセージの説明は、Natural SYSERR ユーティリティにあります。

2 Conventions

このマニュアルでは、変数表記"*vrs*"（ADA *vrs* と同様）は、製品のバージョン、リビジョン、およびシステムメンテナンス（SM）レベルを表します。

Adabas マニュアルでは、DD で始まるデータセット名は、DD 接頭辞を含まない VSE データセット名と区別できるように、DD と残りのデータセット名の間にスラッシュを入れて記載されます。 スラッシュはデータセット名の一部ではありません。

3 はじめに

Adabas トリガとストアドプロシージャ機能は、トリガやストアドプロシージャを定義、処理、および監視するときに使用します。

このchapterでは、両方のタイプのプロシージャおよびそれらの特性について説明します。 この機能のコンポーネントについて紹介し、オンラインサービスやバックグラウンド処理を提供するためにコンポーネントを組み合わせて使用する方法について説明します。

このchapterでは、次のトピックについて説明します。

プロシージャ

*プロシージャ*は、Natural の標準機能を使用して記述され、テストされた Natural サブプログラムです。 トリガとストアドプロシージャの主な違いは、その実行方法にあります。

- *トリガ*は、指定されたイベント（通常は関連テーブルに対するデータアクセスや更新）が発生すると、自動的に実行（起動）されます。イベントは、選択条件が満たされると発生します。選択条件には、ファイル番号の他、コマンドタイプ、フォーマットバッファ内に存在するフィールドの名前、またはそれら両方が含まれることがあります。
- *ストアドプロシージャ*は、データベース管理システム（つまり Adabas）が特殊なユーザーコールを受け取ると実行されます。

トリガであるか、ストアドプロシージャであるかにかかわらず、同じパラメータがサブプログラムに渡されます。

ストアドプロシージャ

ストアドプロシージャはユーザーアプリケーションから呼び出されて、特殊なコールをプロシージャに発行します。1つ以上のパラメータをプロシージャに渡すこともあります。プロシージャは、Adabas によって実行されます。

プロシージャはデータベース（サーバー）にロードされる Natural システムファイル（Adabas ファイル）に格納されるため、サーバーとの間に発生するデータトラフィックの量が減少します。

ストアドプロシージャのサブプログラムがサンプルとして用意されています。ストアドプロシージャのフロントエンドで定義済みのコール構造は修正できます。

次の図は、ストアドプロシージャの使用法を示しています。 STPLNK は、 ストアドプロシージャ要求の呼び出しに使用されるストアドプロシージャのリンクルーチンです。 STPRBE は、プロシージャから呼び出されるレコードバッファ抽出ルーチンのことで、コール元のルーチンから渡されるパラメータの取得に使用されます。

STPLNK の詳細は、「*ストアドプロシージャのリンクルーチン（**STPLNKnn**）*」を参照してください。 STPRBE のプログラミングに関する情報は、「*レコードバッファ抽出ルーチン（**STPRBE**）*」を参照してください。

トリガ

トリガは、トリガするイベントと、トリガされるプロシージャの 2 つの部分から成ります。

トリガするイベントは、Adabas ファイル番号、オプションのコマンドまたはフィールド名（あるいはそれら両方）など一連の選択条件で定義されます。 条件が満たされると、イベントが発生し、それに応じてトリガされるプロシージャが実行されます。 例えば、EMPLOYEES ファイルの SALARY フィールドに対する更新コマンドによって、トリガが起動します。

Note: トリガを別のトリガで起動させることはできません。

トリガの実行は、開始 Adabas コマンドが Adabas ニュークリアスによって実行される前にするか後にするかを定義できます。 トリガの動作は、トリガと Adabas コマンドが同期しているかどうか、同期しているならばトリガがコマンドのトランザクションロジックに関与しているかどうかによって一部変化します。

トリガには、主な特性が 3 つあります。

- プレコマンドまたは ポストコマンドの実行。

  トリガは、開始 Adabas コマンドの前または後に実行できます。
- 非同期または 同期での実行。

トリガは、Adabas コマンドとは無関係に実行することも、Adabas コマンド処理の中断を必要とする（つまり、トリガされたプロシージャの結果が出るまでコマンド処理を待機させる）こともできます。

- 関与または 非関与。

  同期トリガは、ユーザーのトランザクション（つまり ET）ロジックに関与することも関与しないことも可能です。

プレコマンドまたはポストコマンド
: *プレコマンド*（または "プレ"）トリガは、Adabas ニュークリアスによる開始 Adabas コマンドの処理前に実行されます。 Adabas コマンドの実行前にチェックが行われ、トリガを起動すべきかどうかが確認されます。

  例えば、指定されたファイルに対して `N1` コマンドが発行されるたびにトリガが起動するように定義されている場合、`N1` は開始 Adabas コマンドです。 `N1` の実行前に行われるチェックで、プレコマンドトリガを起動するかどうかが決定されます。トリガされたプロシージャが正常に実行された後で、`N1` が処理されます。

  ポストコマンド（または "ポスト"）トリガは、Adabas ニュークリアスによる開始 Adabas コマンドの処理後に実行されます。 トリガされたプロシージャは、Adabas ニュークリアスからのコマンドのリターンコードがゼロの場合にのみ実行されます。リターンコードがゼロ以外の場合は、トリガがあるかどうかはチェックされず、処理は通常どおりに継続します。正常に実行されたコマンドについては、コマンドの処理の完了前に、トリガがあるかどうかがチェックされます。

  例えば、指定されたファイルに対して `L3` コマンドが発行されるたびにトリガが起動するように定義されている場合、`L3` は開始Adabasコマンドです。ゼロのリターンコードが返された後に行われるチェックで、ポストコマンドトリガを起動するかどうかが決定されます。コマンドが正常に実行されてからユーザーに通知される前に、トリガされたプロシージャが実行されます。

非同期または同期
: *非同期*トリガは、Adabas コマンドの開始とは無関係に実行されます。 ユーザーに対する Adabas コマンドの処理は、トリガされたプロシージャが別の処理として実行される間に、中断されずに続行されます。トリガされたプロシージャが実行されるのは、開始するコマンドからの応答を受信した後です。

  トリガ条件を満たすコマンドが発行されると、トリガが起動され、Adabas コマンドの処理が再開されます。 Adabas コマンドとトリガされるプロシージャが、お互いに直接影響し合いません。

  非同期トリガは、プロシージャと実際の Adabas コマンドの間に依存関係がないときに使用されます。

  *同期トリガ*では、該当ユーザーの Adabas コマンド処理が中断されます。 開始 Adabas コマンドは、トリガされたプロシージャの実行が完了するまで中断されます。プロシージャの結果が Adabas コマンドに影響する可能性があります。

ポストコマンドトリガの場合、トリガされるプロシージャの前に Adabas コマンドが実行された後中断します。コマンドの結果は、トリガされたプロシージャの実行が完了するまでユーザーに返されません。

関与または非関与

同期トリガは、開始 Adabas コマンドのロジックに関与する場合も関与しない場合もあります。

*関与*トリガが起動すると、関与トリガを起動した Adabas コマンドと同じユーザー（コミュニケーション）IDが割り当てられているプロシージャが実行されます。したがって、このトリガはトランザクションのロジックに全面的に関与することになります。

- 開始コマンドの処理で発行された ET（エンドトランザクション）またはBT（バックアウトトランザクション）は、トリガから実行中のすべてのトランザクションに影響を与えます。
- トリガされたプロシージャが発行した ET または BT は、開始コマンドで実行中のすべてのトランザクションに影響を与えます。

*非関与*トリガのユーザー ID は、開始コマンドを識別する Adabas ユーザーキューエレメント（UQE）のユーザー（コミュニケーション）ID とは異なります。したがって、このトリガは、開始コマンドのトランザクションロジックには関与しません。

- 開始コマンドのプロセスで発行された ET または BT は、トリガされるプロシージャに影響を与えません。
- トリガされるプロシージャで発行された ET または BT は、開始コマンドのプロセスに影響を与えません。

コンポーネント

Adabas では、トリガとストアドプロシージャを実装および処理するために、3 つの主要なコンポーネントを使用します。

- Adabasトリガドライバは、Adabasニュークリアスの一部であり、トリガとストアドプロシージャを全般的に制御します。プロシージャの要求を検出すると、Natural トリガドライバを初期化して要求を実行します。
- Naturalトリガドライバは、実際にはトリガとストアドプロシージャの両方を実行するバッチ Natural ニュークリアスとして実行されます。 Adabas トリガドライバとともに動作して、Adabas ニュークリアスと、プロシージャを実行する Natural サブシステムとの処理間コミュニケーションを管理します。
- トリガメンテナンスは、Natural アプリケーションで、 Adabas Online System（AOS）から実行できます。 体系化されたメニューを使用して、トリガ定義の作成と管理、トリガのプロファイルの定義、ニュークリアスにおけるトリガのアクティビティの監視と一部の制御を行うことができます。

Adabas トリガドライバ

Adabas トリガドライバは、Adabas ニュークリアスの一部として実行され、一般にトリガの実行時処理全体を制御します。 トリガを起動するかどうか、Natural トリガドライバを開始するかどうか、および相互に作用するかどうかを判断して、プロシージャが正確に適切な時間の範囲で処理するようにします。 プロシージャの処理方法の詳細は、「*処理とパフォーマンス*」を参照してください。

初期化

Adabas ニュークリアスは、起動時に ADARUN パラメータ SPT=YES が指定されているかどうかを判断します。指定されている場合は、制御が Adabas トリガドライバに渡され、Adabas トリガドライバを初期化できるようになります。 初期化時に、Adabas トリガドライバは次の処理を行います。

- バッファ用にストレージを確保します。
- 有効な トリガファイルがロードされていることを確認します。
- トリガファイル上の トリガプロファイルを確認し、セッションのトリガとストアドプロシージャの処理時に使用されるセッションパラメータを抽出します。
- トリガファイルに 1 つ以上の トリガ定義が含まれていることを確認します（Adabas トリガドライバの初期化要件）。
- トリガ定義を読み込み、エントリを トリガテーブルに追加します。トリガテーブルは、メモリ上のバッファに格納されます。 ニュークリアスクラスタ環境では、アクティブなニュークリアスからトリガテーブルが取得されます。 プロシージャの処理時は、トリガテーブルでトリガが存在することを確認できるため、トリガファイルを読み込むという効率の悪い方法を使用する必要はありません。

Natural サブシステムの開始

Adabas トリガドライバが開始されると、プロシージャを実際に実行する Natural サブシステムが開始されます。

Natural サブシステムは、ユーザー作成のプロシージャを実行します。 Adabas トリガプロファイルの最大サブシステムパラメータでは、 開始する Natural サブシステムの数（1～10）を指定します。

各 Natural サブシステムは、通常、Adabas アドレススペース内のサブタスクとして実行するように最低限修正されたバッチ Natural ニュークリアスです。 これは、MPM 起動 JCL/JCS で指定されるリージョンサイズに影響します。 この影響は分割された Natural ニュークリアスを使用することで、最小化することができます。

Natural サブシステムが開始されると、Adabas トリガドライバは、サブシステムのステータスやアクティビティの変化を記録します。 ユーザーは、トリガメンテナンスのサブシステムアクティビティ機能を使用して、これらのアクティビティを監視できます。

Natural サブシステムがアクティブになると、次の処理が行われます。

- Natural トリガドライバが制御を取得します。
- Adabas トリガドライバは、ストアドプロシージャ要求またはトリガの起動により発生する可能性があるプロシージャの処理をサブシステムが実行可能な状態であることが通知されます。

トリガの確認

Adabas トリガドライバは、ニュークリアスが受け取るコマンドごとに、トリガを起動する必要があるかどうかを決定します。

プレコマンドトリガの場合、Adabas トリガドライバは Adabas スレッドで処理されるコマンドが選択される前にトリガをチェックします。 Adabas でコマンドが正常に処理されレスポンスコードがゼロになると、Adabas トリガドライバは起動するポストコマンドトリガがあるかどうか調べます。 1 つのコマンドに対して起動できるのは、結果に関係なく 2 つのトリガ（1 つのプレコマンドトリガと 1 つのポストコマンドトリガ）のみです。

コマンドの結果がトリガの起動である場合、またはコマンドがストアドプロシージャ要求であると Adabas トリガドライバが判断した場合、次の場所でエントリが作成されます。

- コマンドが実行されていない場合は、プレトリガキュー。
- コマンドが正常に実行された場合は、ポストトリガキュー。

エントリには、コマンドと、トリガテーブル内の対応するエントリの両方から取得した情報が含まれます。

Natural サブシステムが処理を待機中の場合、 直ちにトリガ要求が渡されます。 それ以外の場合、次のサブシステムが使用可能になるまで、トリガ要求はプレトリガキューまたはポストトリガキュー内に残ります。 サブシステムがトリガ要求を受け入れると、 Natural トリガドライバの制御下で処理が続行されます（「*コンポーネント*」を参照）。

プロシージャの処理結果

プロシージャの実行が完了すると、Natural トリガドライバは、その結果を "トリガ要求エントリ" に格納し、ステータスが適切に更新されます。 Adabas トリガドライバは、この処理を検出すると、トリガ処理を完了しようとします。

プレコマンドトリガとポストコマンドトリガのどちらでも、プロシージャからのリターンコードによって、結果の処理方法が決定されます。 詳細については、「*結果の処理*」を参照してください。

シャットダウン

Adabas トリガドライバは、 失敗した Natural サブシステムのすべてを記録します。 すべてのサブシステムが失敗すると、Adabas トリガドライバは、プロシージャをこれ以上処理することができないと判断し、Adabas トリガプロファイルのエラーアクション値に従って終了します。ニュークリアスクラスタ環境では、 クラスタ内の他の全ニュークリアスにエラーアクション Ignore または Reject が渡されます。

Adabas ニュークリアスが ADAEND または HALT オペレータコマンドを受け取り、Adabas トリガドライバにシャットダウンを指示した場合は、Adabas トリガドライバも終了します。

Natural トリガドライバ

Natural トリガドライバは、Natural サブシステムの起動時に初期化されます。 Natural トリガドライバは、トリガされたプロシージャとストアドプロシージャのすべてを実行します。次のコンポーネントがあります。

STP	Naturalサブシステムの起動時に呼び出されます。STPは、グローバルデータエリアSTPGDAを初期化し、Naturalセッションに必要なすべての設定を確立します。主な機能は、あらゆるエラーからのリカバリを処理すること、および再スタートを確実に完了させることです。
STPPDRIV	Natural トリガドライバのメインルーチンとして機能し、プロシージャの呼び出しを行います。 STPPDRIV は STPNAT を呼び出します。
STPNAT	Adabas トリガドライバとのあらゆる通信と、サブシステムに対するトリガ要求の提供を行います。 Adabas トリガドライバに対して、処理の準備が整ったことを通知します。
SPAENA	（BS2000 のみ）データベースコマンドキューのアドレスを取得します。

パラメータリストの設定

トリガ要求がNaturalサブシステムによって受け入れられると、STPPDRIVは、起動されたトリガによって指定されたパラメータリストに応じて、プロシージャに渡すパラメータリストを設定します。

記録ルーチンの呼び出しSTPUTRAK

Adabas トリガプロファイルでトリガアクティビティの記録設定がアクティブな場合、ルーチン STPUTRAK が呼び出されます。

STPUTRAK は、ユーザー定義のルーチンで、プロシージャを呼び出す各要求を記録します。この記録は、プロシージャを呼び出す前と後の両方で行われます。 このルーチンを使用して、トレースメッセージの書き込みと、各サブシステムのトリガ処理の監査を行うことができます。デフォルトの STPUTRAK ルーチンが用意されています。

プロシージャのパラメータと同類で、トリガに関する情報を含むパラメータが、ワークエリアとして使用可能な 250 バイト領域とともに STPUTRAK ルーチンに渡されます。 STPUTRAK は、このワークエリアを使用して、セッション全体の情報を保持します。ワークエリアはNaturalト

リガドライバによって変更されることはありません。 STPUTRAK には、ワークエリアをプロシージャに渡すことができるオプションが用意されています。

プロシージャの処理

STPUTRAK が処理されて制御が STPPDRIV に戻ると、トリガされたプロシージャが CALLNAT を使用して呼び出されます。

プロシージャが処理を完了すると、STPPDRIV は結果の STPNAT を通知する前に、Adabas トリガプロファイルで "トリガアクティビティの記録" オプション がアクティブに設定されているかどうかを再度確認します。 アクティビティの記録がアクティブな場合、プロシージャの結果を監査できるように STPUTRAK が呼び出されます。

エラーからのリカバリ

プロシージャが NAT0954、NAT3009、NAT1305 などのエラーになると、プロシージャは正常に終了しません。 代わりに Natural トリガドライバはエラーリカバリを実行します。

トリガアクティビティの記録設定に関係なく、エラーの情報は STPUTRAK に渡されます。 この情報は、データベース管理者（DBA）がデバッグや問題分析に使用できます。 そのため、STPUTRAK は Natural トリガドライバによって常に実行可能でなければなりません。 この情報を取得するためにトレースをアクティブ化することをお勧めします。

トリガ要求エントリの更新

プロシージャの結果が STPNAT に渡されると、トリガ要求が更新され、そのステータスは "完了" に変化します。 Adabas トリガドライバは、更新されたエントリを検出するとトリガ処理を完了します。

STPNAT は別のトリガ要求が作成されるのを待機し、サイクル全体が再び開始します。

トリガメンテナンス

トリガメンテナンスは、 トリガ定義の作成とシステム処理の監視を対話的に実行する機能で、フルバージョンの Adabas Online System が必要です。 詳細は、このドキュメントの「トリガメンテナンス」を参照してください。

*trigger definition*は、実行対象のプロシージャの名前と属性、および*triggering event*を起動する選択条件（Adabas コマンドタイプ、ファイルの名前または番号、およびフィールド名）で構成されます。

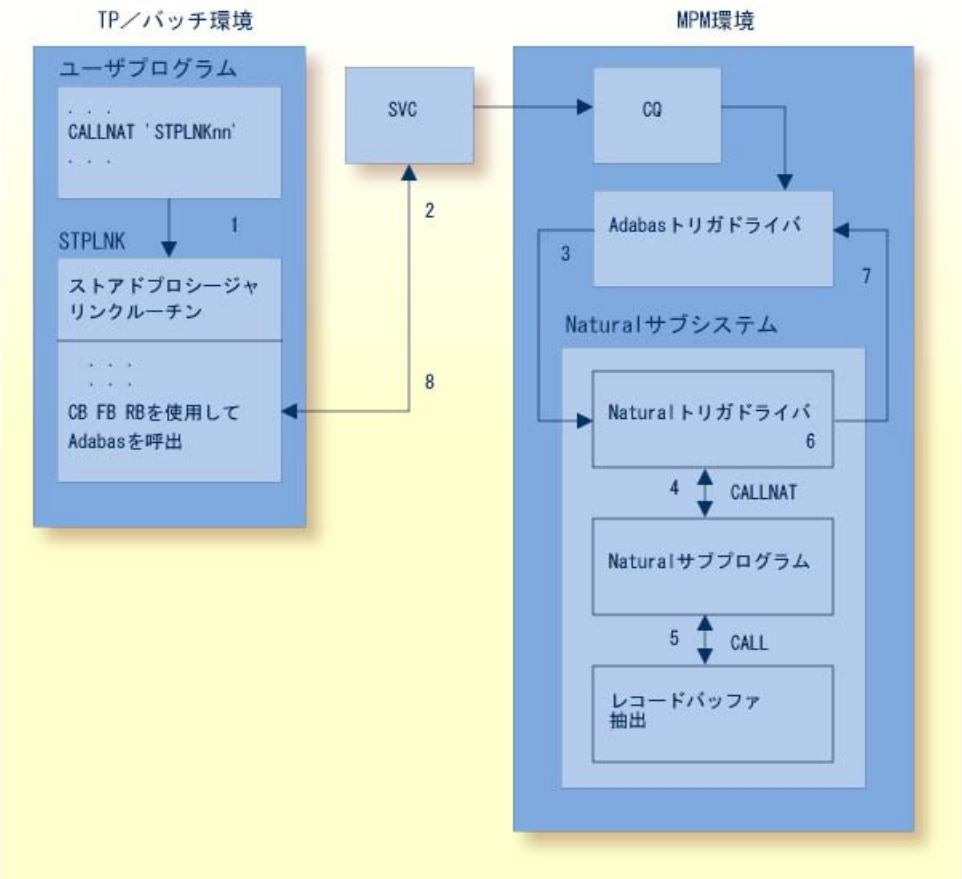
トリガメンテナンスでは、ニュークリアスによるあらゆるトリガ処理の実行を監視および制御する機能も提供します。 Adabas トリガドライバと Natural トリガドライバの両方のアクティビティを調べて、次の処理を実行できます。

- ニュークリアスにおけるトリガ環境の状態を判断します。つまり、アクティブなトリガとそうでないトリガ、バッファサイズ、および Natural サブシステムのステータスを判断します。
- アクティブなセッション中に、アクティビティのタイムアウト、トリガアクティビティの記録、エラーアクションなど、さまざまなパラメータの設定を修正します。
- プレトリガキューおよびポストトリガキューを調べます。これらのトリガキューには、起動されたプレコマンドトリガおよびポストコマンドトリガの実行を待機するプロシージャが格納されています。
- Adabas トリガおよびストアドプロシージャ機能のアクティビティを調べ、何が実行されているのか、問題が存在するかどうか、現在アクティブな Natural サブシステムの数などを判断します。
- 新規、削除、または更新されたトリガ定義で Adabas ニュークリアスのトリガテーブルをリフレッシュします。
- トリガを個々にアクティブ化または非アクティブ化します。 特定のトリガで問題が見つかった場合は、問題を解決する一方で、トリガを一時的または永続的に非アクティブ化することができます。

処理のまとめ

ストアドプロシージャの処理

ストアドプロシージャ処理のステップについて、次の図に示し、「*処理ステップ*」で説明します。



手順 3.1. 処理ステップ

1 ユーザーアプリケーションは、必要なパラメータを設定し、ストアドプロシージャのリンクルーチン STPLNK*nn* に対して CALL を発行します。

2 STPLNK*nn* は、ユーザーの要求を解釈し、標準の Adabas API の形式で DBMS に対する呼び出しを発行します（直接呼び出し）。

3 Adabas トリガドライバはストアドプロシージャ要求を受け取り、Natural トリガドライバに渡します。

4 指定された Natural サブプログラム（ストアドプロシージャ）が呼び出されます。

5 必要に応じて、呼び出し元のパラメータが、レコードバッファ抽出ルーチンによってサブプログラムで利用できるようになります。オプション設定で許可されている場合は、パラメータを修正できます。

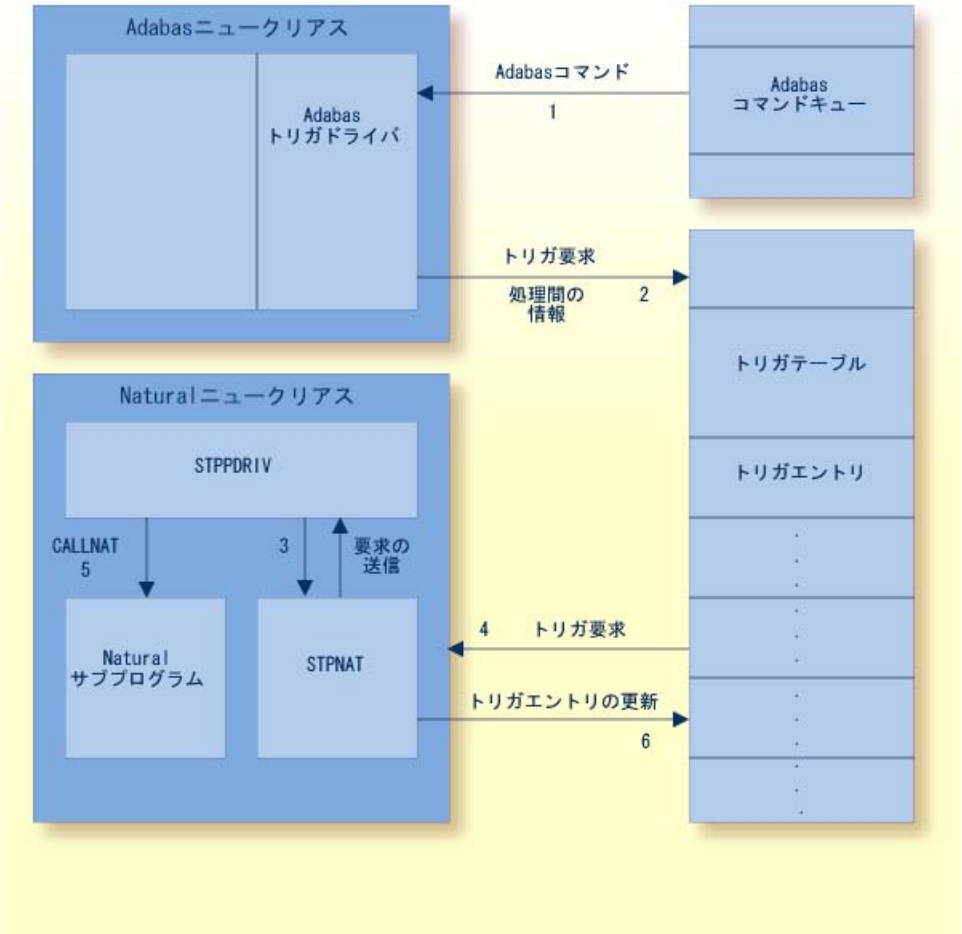
6 完了すると、Natural サブプログラムは制御を Natural トリガドライバに返します。

7 Natural トリガドライバは、ストアドプロシージャ要求の結果を Adabas トリガドライバに返します。

8 ユーザーに、通知と修正されたパラメータが返されます。

トリガ処理

トリガ処理のステップについて、次の図に示し、「*処理ステップ*」で説明します。



手順 3.2. 処理ステップ

1 Adabas トリガドライバは、すべての Adabas コマンドを受け取り、それらのコマンドがトリガイベントに該当するか確認します。

2 コマンドがトリガ条件を満たすと、Adabas トリガドライバはトリガ要求と処理間情報をトリガテーブルに格納します。

3 Natural トリガドライバ制御ルーチン STPPDRIV で別のトリガ要求を処理する準備が整うと、STPPDRIV は STPNAT を呼び出します。

4 STPNAT は、トリガテーブルから次のトリガ要求を取得し、STPPDRIV に渡します。

5 STPPDRIV は、トリガ要求タイプを判断し、CALLNAT をトリガ定義で指定された Natural サブプログラムに対して発行し、関連するパラメータを渡します。

6 STPNAT は、トリガテーブル内のトリガエントリを更新し、トリガが完了したことを示します。

4 インストールと設定

このchapterでは、Adabas トリガとストアドプロシージャ機能のインストール方法について説明します。 NATPARM の設定、プリンタとワークファイルの割り当て、Natural Security 環境の構成、ストアドプロシージャのリンクルーチンの設定について説明します。

このchapterでは、次のトピックについて説明します。

ソフトウェア要件

Adabas に付属の Adabas トリガとストアドプロシージャ機能は、次のオペレーティングシステム用の Natural 3.1 以上で使用できます。

- z/OS
- VSE
- z/VM
- BS2000

この機能では、Adabas Online System（AOS）が必要です。

Natural Optimizer Compiler は必須ではありませんが、トリガとストアドプロシージャの使用時に Natural Optimizer Compiler を使用すると、大幅にパフォーマンスが向上することがあります。

概要

Adabas トリガとストアドプロシージャ機能のインストールは、3 つに分かれています。

- オンラインユーザーインターフェイスである*トリガメンテナンスのインストール*。
- Adabas ニュークリアスコンポーネントである ***Adabas** トリガドライバのインストール*。
- Natural ニュークリアスコンポーネントである ***Natural** トリガドライバのインストール*。

トリガメンテナンスのインストール

1. AOS のインストール

Adabas Online System（AOS）アドオン製品をインストールします。 *Adabas Online System* のドキュメントに記載されている手順を使用します。

2. トリガファイルのロード

ADALOD ユーティリティを使用して、トリガファイルを Adabas データベースにロードします。

トリガファイルのロードを指定するには、ADALOD パラメータでキーワード TRIGGER を指定します。 チェックポイントまたはセキュリティファイルをロードするときは、`CHECKPOINT` または `SECURITY` パラメータを指定して同じ方法で行います。

ADACMP のステップは省略します。FDT 定義はすでにニュークリアスによって認識されるため、この手順は不要です。

この時点でトリガファイルは、Adabas Security を持つことが許可されていません。つまり、Adabas パスワードと暗号化またはセキュリティバイバリューは設定されていません。

3. NATPARM と再リンクの修正

オンラインNaturalニュークリアスでトリガとストアドプロシージャを使用するようにNATPARMを更新し、NATPARM を Natural ニュークリアスに再リンクします。 次のいずれかを使用します。

- 論理ファイル定義。NATPARM で NTFILE マクロ指定とともに使用します。
- `LFILE` パラメータ。セッション初期化時のダイナミックパラメータとして使用します。

トリガファイルの論理ファイル番号は 154 です。

Note: NTFILEマクロまたは`LFILE`パラメータを使用しない場合は、Adabasトリガとストアドプロシージャ機能を開始するたびに、データベース ID とファイル番号を入力する必要があります。

4. 機能の開始とプロファイルの作成

Natural セッションから Adabas トリガとストアドプロシージャ機能を開始し、プロファイルを作成します。

1. Adabas Online System にログオンし、オプション［Trigger Maintenance］を選択します。Enter キーを押します。
2. メインメニューから、［A］（Administrator Functions）を選択し、Enter キーを押します。必要に応じて、データベース ID とファイル番号を入力し、Enter キーを 2 回押します。

   Note: データベース ID とファイル番号は、前述の手順 3 で NTFILE マクロまたは `LFILE` パラメータを使用していない場合、または定義で指定した値が正しくない場合に限り、必要になります。

3. ［Administrator Functions］メニューから、［M］（Modify Profile Information）を選択し、Enter キーを押します。

   ［Modify Profile Information］画面には、上書き可能なデフォルト値が表示されます。 ただし、Adabas トリガとストアドプロシージャ機能を実行する*前*に、プロファイルを正しくインストールしておく必要があります。 詳細については、「*プロファイル情報の表示／修正*」を参照してください。

   Note: トリガメンテナンスは、必要なファイル - フィールドテーブルとトリガ定義が追加されるまで、完全には機能しません。 詳細については、「*トリガメンテナンス*」を参照してください。

Adabas トリガドライバのインストール

モジュール ADATSP と STPEND は、Adabas ロードライブラリ内にあります。 TRGMPMJ は、Adabas ニュークリアスを開始するために使用する JCL/JCS です。

1. ADARUN SPT パラメータの設定

ADARUN `SPT=YES` と設定してニュークリアスで Adabas トリガとストアドプロシージャ機能をアクティブ化するか、`SPT=NO` と設定して非アクティブ化します。

Notes:

1. `SPT=YES` の場合、`PROG=ADANUC` と `MODE=MULTI` を設定する必要があります。 `SPT` は、グローバルパラメータであり、クラスタ内のすべてのニュークリアスで同じ設定にする必要があります。

2. OPENRQ=YES は指定しないでください。 Natural サブシステムで、レスポンスコード 148 を受け取り、初期化できないという問題が発生します。

2. Natural サブシステムの設定

各サブシステムにラベルとジョブ制御割り当てを指定します。

これらは、最大サブシステム数の値、Adabas トリガプロファイルでの CMPRINT 割り当て数、およびプロシージャで使用される出力およびワークファイルに依存します。

3. すべての Natural ワークファイルおよび出力ファイルの設定

指定されたすべての Natural ワークファイルおよび出力ファイルに追加のラベルを指定します。

BS2000 における例外

BS2000 ジョブでは、ラベルを指定する必要はありません。 出力は次のファイルにあります。

```
L.L.db_task_nr.natural_load_name.timestamp
```

上記の意味は次に示すとおりです。

db_task_nr	トリガおよびストアドプロシージャ機能を実行しているデータベースのタスク番号
natural_load_name	バッチ Natural モジュールの名前
timestamp	サブタスクが開始された時間を示す文字

手順 2 と 3 の例

次の例で、前述の手順 2 と 3 を説明します。 この例では、次のように設定されています。

- ダイナミックな CMPRINT 割り当て（Adabas トリガプロファイルで指定するオプション）は TSPRT に設定されています。
- この例で定義されているサブシステムの最大数は 5 です。

z/OS JCL の場合は、次のラベルを指定する必要があります。

```
//TSPRT01  DD  SYSOUT=X
//TSPRT02  DD  SYSOUT=X
//TSPRT03  DD  SYSOUT=X
//TSPRT04  DD  SYSOUT=X
//TSPRT05  DD  SYSOUT=X
```

VSE JCS の場合は、次のラベルを指定する必要があります。

```
// ASSGN SYS041,DISK,VOL=PACK01,SHR
// DLBL TSPRT01,'PRINT.OUTPUT1',0,SD
// EXTENT SYS041,PACK01,1.0,SSSS,LLL
// ASSGN SYS042,DISK,VOL=PACK01,SHR
// DLBL TSPRT02,'PRINT.OUTPUT2',0,SD
// EXTENT SYS042,PACK01,1.0,SSSS,LLL
// ASSGN SYS043,DISK,VOL=PACK01,SHR
// DLBL TSPRT03,'PRINT.OUTPUT3',0,SD
// EXTENT SYS043,PACK01,1.0,SSSS,LLL
// ASSGN SYS044,DISK,VOL=PACK01,SHR
// DLBL TSPRT04,'PRINT.OUTPUT4',0,SD
// EXTENT SYS044,PACK01,1.0,SSSS,LLL
// ASSGN SYS045,DISK,VOL=PACK01,SHR
// DLBL TSPRT05,'PRINT.OUTPUT5',0,SD
// EXTENT SYS045,PACK01,1.0,SSSS,LLL
```

z/VM EXEC の場合は、次のラベルを指定する必要があります。

```
"FILEDEF TSPRT01 DISK TSPRT01 LISTING A"
"FILEDEF TSPRT02 DISK TSPRT02 LISTING A"
"FILEDEF TSPRT03 DISK TSPRT03 LISTING A"
"FILEDEF TSPRT04 DISK TSPRT04 LISTING A"
"FILEDEF TSPRT05 DISK TSPRT05 LISTING A"
```

プロファイルで定義されている5つのサブシステムのそれぞれで、出力ファイルに対してWRITE、PRINT、または DISPLAY を実行できる場合は、MPM JCL で次の定義を指定する必要があります。

z/OS の場合を以下に示します。

```
//CMPRT01  DD  SYSOUT=X
//CMPRT02  DD  SYSOUT=X
//CMPRT03  DD  SYSOUT=X
//CMPRT04  DD  SYSOUT=X
//CMPRT05  DD  SYSOUT=X
```

VSE の場合を以下に示します。

```
// ASSGN SYS051,DISK,VOL=PACK01,SHR
// DLBL CMPRT01,'PRINT.CMPRT01',0,SD
// EXTENT SYS051,PACK01,1.0,SSSS,LLL
// ASSGN SYS052,DISK,VOL=PACK01,SHR
// DLBL CMPRT02,'PRINT.CMPRT02',0,SD
// EXTENT SYS052,PACK01,1.0,SSSS,LLL
// ASSGN SYS053,DISK,VOL=PACK01,SHR
// DLBL CMPRT03,'PRINT.CMPRT03',0,SD
// EXTENT SYS053,PACK01,1.0,SSSS,LLL
// ASSGN SYS054,DISK,VOL=PACK01,SHR
// DLBL CMPRT04,'PRINT.CMPRT04',0,SD
// EXTENT SYS054,PACK01,1.0,SSSS,LLL
// ASSGN SYS055,DISK,VOL=PACK01,SHR
// DLBL CMPRT05,'PRINT.CMPRT05',0,SD
// EXTENT SYS055,PACK01,1.0,SSSS,LLL
```

z/VM の場合を以下に示します。

```
"FILEDEF CMPRT01 DISK CMPRT01 LISTING A"
"FILEDEF CMPRT02 DISK CMPRT02 LISTING A"
"FILEDEF CMPRT03 DISK CMPRT03 LISTING A"
"FILEDEF CMPRT04 DISK CMPRT04 LISTING A"
"FILEDEF CMPRT05 DISK CMPRT05 LISTING A"
```

CMPRTnn ラベルと論理プリンタの割り当てについては、「*プリンタの考慮事項*」を参照してください。

プロファイルで定義されている 5 つのサブシステムのそれぞれで、ワークファイルに対して WRITE WORK または READ WORK を実行できる場合は、MPM JCL で次の定義を指定する必要があります。

z/OS の場合を以下に示します。

```
//CMWKF01 DD DISP=SHR,DSN=WORK.CMWKF01
//CMWKF02 DD DISP=SHR,DSN=WORK.CMWKF02
//CMWKF03 DD DISP=SHR,DSN=WORK.CMWKF03
//CMWKF04 DD DISP=SHR,DSN=WORK.CMWKF04
//CMWKF05 DD DISP=SHR,DSN=WORK.CMWKF05
```

VSE の場合を以下に示します。

```
// ASSGN SYS061,DISK,VOL=PACK01,SHR
// DLBL CMWKF01,'WORK.CMWKF01',0,SD
// EXTENT SYS061,PACK01,1.0,SSSS,LLL
// ASSGN SYS062,DISK,VOL=PACK01,SHR
// DLBL CMWKF02,'WORK.CMWKF02',0,SD
// EXTENT SYS062,PACK01,1.0,SSSS,LLL
// ASSGN SYS063,DISK,VOL=PACK01,SHR
// DLBL CMWKF03,'WORK.CMWKF03',0,SD
// EXTENT SYS063,PACK01,1.0,SSSS,LLL
// ASSGN SYS064,DISK,VOL=PACK01,SHR
// DLBL CMWKF04,'WORK.CMWKF04',0,SD
// EXTENT SYS064,PACK01,1.0,SSSS,LLL
// ASSGN SYS065,DISK,VOL=PACK01,SHR
// DLBL CMWKF05,'WORK.CMWKF05',0,SD
// EXTENT SYS065,PACK01,1.0,SSSS,LLL
```

z/VM の場合を以下に示します。

```
"FILEDEF CMWKF01 DISK CMWKF01 LISTING A"
"FILEDEF CMWKF02 DISK CMWKF02 LISTING A"
"FILEDEF CMWKF03 DISK CMWKF03 LISTING A"
"FILEDEF CMWKF04 DISK CMWKF04 LISTING A"
"FILEDEF CMWKF05 DISK CMWKF05 LISTING A"
```

Notes:

1. プロシージャでワークファイルのサポートまたはプリンタファイルの追加定義が必要ない場合、ファイル名割り当ては必要ありません。
2. MPM JCL/JCS には、特定の Natural サブプログラムまたはプログラムを実行するためにバッチ Natural JCL/JCS で通常使用されるその他の割り当ては含めないでください。

z/VM サポートに必要な変更

z/VM マルチタスキングは、Adabas トリガとストアドプロシージャ機能をサポートするときに使用されます。 そのため、サンプルの RMTNUC EXEC に変更を加える必要があります。

次のサンプルを参考に、**RMTNUC EXEC** を修正してください。

```
/*              Sample RMTNUC Exec                                          */
/*                                                                          */
Address COMMAND
 user.set = word('HT RT',cmsflag(cmstype)+1)
"SET CMSTYPE HT"

trce = 'OFF'
htype = 'HT'
interpret trace trce
/*  The following variable may have the following values:                   */
/*       'DISK' to spool a dump to the virtual printer & read to disk       */
/*       'PRT'  to spool a dump to the virtual printer                      */
/*       'NO'   to suppress a dump                                          */
dump = 'NO'
/*                                                                          */
dbid = '00052'
restart = 'NO'
if 'ARG'() = 1 then parse upper arg dbid restart .
dbid = 'STRIP'(dbid)
if dbid = '' then dbid = '00052'
             else dbid = 'RIGHT'(dbid,5,'0')
signal on error
interpret call adf || dbid 'ADANUC MULTI' trce htype
pull . device
if result = 99 then signal error
/*                                                                          */
errtype ='CMS'
'COPY RUNNUC CONTROL A RDV' || dbid 'CONTROL A ADANUC DDCARD A (REPLACE)'
'COPY RUNMULTI CONTROL A ADANUC DDCARD A (APPEND)'
'COPY RDB' || dbid 'CONTROL A ADANUC DDCARD A (APPEND)'
'COPY STP' || dbid 'CONTROL A ADANUC DDCARD A (APPEND)'
If restart = 'RESTART' then
  'COPY RUNDIB CONTROL A ADANUC DDCARD A (APPEND)'
/*                                                                          */


errtype ='DATADEF'
/*
'DATADEF DDPRINT,DSN=NUC' || dbid'.DDPRINT,MODE=A'
'DATADEF DDDRUCK,DSN=NUC' || dbid'.DDDRUCK,MODE=A'
*/
'DATADEF DDPRINT,DSN=.TEMP,UNIT=SFS,FNAME=NUC' || dbid',FTYPE=DDPRINT'
'DATADEF DDPRINT,DSN=.TEMP,UNIT=SFS,FNAME=NUC' || dbid',FTYPE=DDDRUCK'
'DATADEF DDCARD,DSN=ADANUC.DDCARD,MODE=A'
```

```
select ;
   when dump = 'DISK' then 'DATADEF DUMP,DSN=ADANUC.DUMP,MODE=A'
   when dump = 'PRT'  then 'DATADEF DUMP,UNIT=PRT'
   when dump = 'NO'   then 'DATADEF DUMP,DUMMY'
   otherwise do
      say '..         Invalid dump request'
      signal error
   end
end
signal off error
  /*-----------------------------------------------------------------*/
  /*  Add the filedef's for Adabas triggers and stored procedures    */
  /*-----------------------------------------------------------------*/

  "FILEDEF *      CLEAR"     /*  Let's start clean                   */

/* Filedefs for Adabas triggers & stored procedures facility         */

  "FILEDEF TSPRT01 DISK TSPRT01 LISTING  A"
  "FILEDEF TSPRT02 DISK TSPRT02 LISTING  A"
  "FILEDEF TSPRT03 DISK TSPRT03 LISTING  A"
  "FILEDEF TSPRT04 DISK TSPRT04 LISTING  A"
  "FILEDEF TSPRT05 DISK TSPRT05 LISTING  A"

  "FILEDEF CMPRT01  DISK CMPRT01  LISTING  A"
  "FILEDEF CMPRT02  DISK CMPRT02  LISTING  A"
  "FILEDEF CMPRT03  DISK CMPRT03  LISTING  A"
  "FILEDEF CMPRT04  DISK CMPRT04  LISTING  A"
  "FILEDEF CMPRT05  DISK CMPRT05  LISTING  A"

  "FILEDEF CMWKF01  DUMMY"
  "FILEDEF CMWKF02  DUMMY"
  "FILEDEF CMWKF03  DUMMY"
  "FILEDEF CMWKF04  DUMMY"
  "FILEDEF CMWKF05  DUMMY"

/* End of Filedefs for Adabas triggers & stored procedures facility  */


/*-------------------------------------------------------------------*/
/*  Now set the language file to SSFume                               */
/*-------------------------------------------------------------------*/

"SET LANGUAGE ( ADD SSF USER"

"LOADMOD  SSFRUN"
"PROGMAP  SSFRUN"
  'SSFRUN'                    /* This will run ADARUN               */

   rcode = rc
  "EXECOS"
```

```
  'FINIS * * *'

  Exit rcode
error:
'SET CMSTYPE' origtype
return 99
```

4. ローカルライブラリの指定

MPM JCL STEPLIB / JCS LIBDEF で適切なローカルライブラリを指定します。 次の項目を含めるようにします。

- モジュール STPEND と ADATSP を含む Adabas ロードライブラリ
- ユーザー出口を含むユーザーロードライブラリ
- 特別にリンクされたバッチ Natural ニュークリアスを含む Natural ロードライブラリ

Natural トリガドライバのインストール

Adabas トリガとストアドプロシージャ機能は、Natural サブシステムを使用してユーザー作成のプロシージャを実行します。 これらのサブシステムは、サブタスクとして実行されます。最大で同時に 10 のサブシステムをアクティブにできます。 Natural サブシステムは、基本的にバッチ Natural ニュークリアスです。

サブタスクは、z/OS の場合は Adabas アドレススペースで、VSE の場合は Adabas パーティションで、z/VM の場合は Adabas リージョンで実行されます。 BS2000 の場合、サブタスクは Adabas と同じアドレススペースで実行されませんが、グループの共通メモリと P1 イベント処理を使用して Adabas ニュークリアスと通信する派生ジョブです。

Adabas 実行中のリソースの使用状況を最適化するために、Natural サブシステムで使用されるリソースを最小化するように Natural トリガドライバ（Natural ニュークリアスコンポーネント）を構成することが重要です。

Adabas サンプルジョブ

Adabas のインストールの一部として、テープからデータセット ADA*vrs*.JOBS がロードされています。 その結果、次のサンプルがインストールされています。

名前	説明
ASMNTOS	NATOS アセンブリ（z/OS のみ）
ASMNTVSE	NATVSE アセンブリ（VSE のみ）
ASMNTBS2	NAMBS2 アセンブリ（BS2000 のみ）
ASMPARM	NATPARM アセンブリ
LNKBATCH	Natural ニュークリアスリンク
LNKNATNS	分割ニュークリアス用の非共有 Natural ニュークリアスリンク（BS2000 以外）
LNKNATSH	分割ニュークリアス用の共有 Natural ニュークリアスリンク（BS2000 以外）
TRGMPMJ	起動 MPM JCL（BS2000 以外）
TRGPARM	Adabas トリガとストアドプロシージャ機能に必要な変更が加えられた NATPARM（BS2000 以外）
TSPBLDM	TSP Natural バージョン 4.2.3 モジュールをビルド
TSP$LOAD	TSPBLDM EXEC による呼び出し

Note: サンプルを修正した場合は、別のライブラリに格納するようにしてください。そうしないと、今後のバージョンをインストールしたときに上書きされてしまいます。

バッチ Natural ドライバ NATOS/NATVSE/NAMBS2

Natural のインストールの一部として、データセット NATvrs.SRCE がロードされています。 ソースライブラリのメンバ NATOS/NATVSE/NAMBS2 には、バッチ Natural ドライバが含まれています。このバッチ Natural ドライバは、Adabas トリガとストアドプロシージャ機能でバッチ Natural ニュークリアスを作成するときに使用する必要があります。

バッチ Natural ニュークリアスの準備

Natural を環境非依存のニュークリアスと環境依存のニュークリアスの 2 つに分割することをお勧めします。 詳細は、*Natural のインストールに関するドキュメント*を参照してください。

- *環境非依存*のニュークリアスは、"共有ニュークリアス" とも呼ばれ、オペレーティングシステムの共有エリア（BS2000 では "リエントラント" 部分）に配置されます。
  - z/OS 環境の場合、リンクパック域（LPA）、または拡張リンクパック域（ELPA）。
  - VSE の場合、共用仮想領域（SVA）。
  - BS2000 の場合、共通メモリプール（CMP）。

このようなオペレーティングシステムの特殊なエリアから実行することで、非依存ニュークリアスは、同じオペレーティングシステム内の複数のアドレススペース（またはパーティションやリージョン）から一般にアクセス（共有）することができます。

共有ニュークリアスの利点は、仮想ストレージの負荷軽減です。システム内にニュークリアスのコピーは 1 つしか存在しないためページングアクティビティが減少し、ZAP は 1 回しか適用されないためメンテナンスの手間も少なくなります。

- *環境依存*のニュークリアスは、BS2000 の "フロントエンド" 部分とも呼ばれます。環境非依存部分を取り除いているため、サイズが大幅に縮小されています。バッチアドレススペース（またはパーティションやリージョン）にロードされ、特にトリガとストアドプロシージャで使用するように指定されています。

z/OS 環境

Adabas トリガとストアドプロシージャ機能で使用される Natural ニュークリアスは、NATPARM モジュールと STPDRV および STPRBE のエントリポイントが含まれる必要があります。

STPNAT モジュールとリンクしている必要があります。 リンクでは、オプション RENT または REUSE を使用しないでください。使用すると、予測できない結果が生じます。 また、ストアドプロシージャとトリガ（ADARUN SPT=YES）で使用される ADALNK ルーチンが、NOREUSE および NORENT でリンクされるようにします。そうでないと、予測できない結果が生じます。

リンクモジュールは、MPM JCL STEPLIB に連結されているライブラリに配置する必要があります。

VSE 環境

Adabas トリガとストアドプロシージャ機能で使用される Natural ニュークリアスには、サンプルジョブ ASMPARM を使用して作成された専用の NATPARM モジュールが含まれる必要があります。

また、モジュール STPNAT を Natural リンクに追加してください。 詳細については、サンプルジョブ LNKBATCH を参照してください。

リンクモジュールは、MPM JCS LIBDEF に連結されているライブラリに配置する必要があります。

z/VM 環境

Adabas トリガとストアドプロシージャ機能で使用される Natural ニュークリアスは、サンプルジョブ TSPBLDM を使用して作成された専用の NATPARM モジュールが含まれる必要があります。 TSPBLDM EXEC などを使用して、Adabas トリガとストアドプロシージャ機能で使用する Natural ニュークリアスを作成します。 TSPBLDM EXEC では TSP$LOAD を呼び出すことに注意してください。TSP をサポートするための追加 INCLUDE が含まれています。

BS2000 環境

NAMBS2 マクロパラメータを `ADACOM=ADABAS` に設定します。

ライブラリの命名規則に従って、ASMPARM および ASMNTBS2 モジュールをカスタマイズします。 ASMPARM を使用して、NATPARM モジュールをアセンブルできます。 ASMNTBS2 モジュールは、Natural ドライバ NAMBS2 をアセンブルします。

次のモジュールを Natural にリンクします。

STPNAT SPAENA

バッチ Natural ドライバがアセンブルされたら、LNKBATCH ジョブを使用してリンクします。

1. バッチ Natural ドライバのアセンブル

z/OS の場合は、ADAvrs.JOBS データセットのサンプルジョブ ASMNTOS を使用して、バッチ Natural ドライバをアセンブルします。

VSE の場合は、JCS のサンプル ASMNTVSE を使用します。 Natural ジョブ NATI055 も参照してください。

BS2000 の場合は、ADAvrs.JOBS ライブラリの ASMNTBS2 を使用します。

2. NATPARM モジュールのアセンブル

詳細は「*NATPARM について*」を参照してください。

メンバ STPNAT は、Adabas ロードライブラリで提供されており、3 つのエントリポイント STPDRV、STPRBE、および ADABAS があります。

STPNAT

STPNAT は ADAUSER を置換します。 STPNAT には、同じ Adabas エントリポイントがあり、ADAUSER の通常の機能のすべてを実行します。 また、トリガとストアドプロシージャ用に特化したロジックが含まれています。

z/OS、VSE、および z/VM 環境

次のように、NATPARM モジュールの CSTATIC パラメータ指定に STPDRV と STPRBE を組み込みます。

```
CSTATIC=(STPDRV,STPRBE)
```

VSE および z/VM 環境では、次のパラメータも指定します。

```
ADAPRM=ON
  EXTBUF=10
  MT=0
```

BS2000 環境

次のように、NATPARM モジュールの CSTATIC パラメータ指定に STPNAT と SPAENA を組み込みます。

```
CSTATIC=(STPNAT,SPAENA)
```

3. バッチ Natural ドライバのリンク

手順 1 でアセンブルしたバッチ Natural ドライバをリンクするときは、手順 2 でアセンブルした NATPARM と STPNAT を組み込んで、Adabas トリガとストアドプロシージャ機能のバッチ Natural ニュークリアスを作成します。

Natural ニュークリアスのリンクデッキに INCLUDE STPNAT ステートメントが含まれ、通常の INCLUDE ADAUSER ステートメントが含まれていないことを確認します。 AOSASM モジュールは必要ありません。

BS2000 環境

ADAVrs.JOBS ライブラリで提供される LNKBATCH ジョブを関連のあるライブラリ用にカスタマイズし、ジョブを実行します。

4. REGION/SIZE パラメータ設定の確認

Adabas 起動 JCL/JCS の REGION/SIZE パラメータにサイズ制限が設定されていないことを確認します。

オペレーティングシステムの設定時に、8 メガバイトなどのサイズ制限が指定されている可能性があります。

5. Natural ニュークリアスから Adabas ニュークリアスへのアクセスを可能にする

z/OS 環境

手順 3 で作成した Natural モジュールが、Adabas 初期化時に Adabas ニュークリアスにアクセスできるように、次のいずれかの操作を行います。

- この Natural モジュールを Adabas ロードライブラリの MPM JCL steplib または joblib に配置します。
- Natural ロードライブラリ（モジュールが格納されている）を Adabas MPM JCL steplib または joblib に連結します。

Adabas ロードライブラリ（データセット ADABAS.Vvr.LOAD）には、モジュール STPEND も格納されています。STPEND は、MPM JCL steplib のいずれかのライブラリ内に格納されている必要があります。

例

```
//STEPLIB  DD  DISP=SHR, DSN=ADABAS.Vvr.LOAD
           DD  DISP=SHR, DSN=NATURAL.Vvr.LOAD
```

VSE 環境

手順 3 で作成した Natural モジュールが、Adabas ニュークリアスの起動ジョブの LIBDEF SEARCH ステートメントで使用される Adabas ライブラリにリンクされるようにします。Natural ライブラリ（フェーズが格納されている）を Adabas ニュークリアスの起動ジョブの LIBDEF SEARCH ステートメントに追加します。

BS2000 環境

Adabas ニュークリアスが次のステートメントで実行されるようにします。

```
/START-PROGRAM *M(&ADALIB,ADARUN),-
/  RUN-MODE=ADV(ALT-LIB=YES)
```

次のようにして、バッチニュークリアスが格納されているライブラリにアクセスします。

```
/SET-FILE-LINK DDLNKPAR,DDLNKPAR_file
/SET-FILE-LINK BLSLIBnn,batch_natural_library
```

上記の意味は次に示すとおりです。

nn	00～99。BLSLIB リンクカードで次に使用可能な番号
batch_natural_library	バッチ Natural ライブラリ
DDLNKPAR_file	次のステートメントが記載されたファイル `ADARUN DBID=`*dbid* `ADARUN IDTNAME=`*idtname* *dbid* は Adabas ニュークリアスのデータベース ID、*idtname* は Adabas ニュークリアス実行時の ID テーブル名

Important: *DDLNKPAR_file* は、一時ファイルにしてはなりません。

6. ADALNK リンクオプションの確認

ストアドプロシージャとトリガ（ADARUN SPT=YES）で使用される ADALNK ルーチンが、NOREUSE および NORENT でリンクされるようにします。そうしないと、予測できない結果が生じます。

NATPARM について

Natural パラメータモジュールで Natural ニュークリアスに指定された NATPARM 定義を使用して、Natural セッション用に環境を調整します。

Natural ニュークリアスコンポーネントを正常に実行するには、特定の NATPARM パラメータに適切な値を指定する必要があります。詳細については、Naturalのマニュアルを参照してください。

特殊な要件

ADAPRM
VSE および VM/CMS 環境では、ON に設定する必要があります。

バッファプール
要求やローカル環境の構成に応じて、ローカルバッファプールまたはグローバルバッファプールを使用します。

- ローカルバッファプールを使用する場合、ユーザーが呼び出すプロシージャは、Natural セッション中はローカルバッファプールに残ることがあります。その結果、新しいコピーが無視される可能性があります。
- グローバルバッファプールを使用する場合、できるだけ早くアクティブ化するために、プロシージャがグローバルバッファプールから削除されることがあります。

バッファサイズ

Natural トリガドライバは実行時システムのみである（開発システムではない）ため、プログラムの記述内容に応じて、さまざまなバッファを最小サイズに維持することができます。例えば ESIZE をどの Natural プログラムでも使用できる最大 GDA サイズに設定できます。Adabas トリガとストアドプロシージャ機能で使用される GDA のサイズは、12K です。

CDYNAM

このパラメータは、非 Natural プログラムのダイナミックロードを制御します。 プロシージャは Natural ニュークリアスにリンクされるため、トリガとストアドプロシージャでは無条件で設定します。 デフォルト値は 5 です。

CSTATIC

`CSTATIC` パラメータは、Naturalに静的にリンクされているプログラムを制御します。 デフォルト値はありません。 このパラメータには、Adabas トリガとストアドプロシージャ機能で使用されるルーチンを指定する必要があります。

STPDRV	Natural トリガドライバ
STPRBE	レコードバッファ抽出ルーチン
SPAENA	データベースコマンドキューへのアクセスを可能にします（BS2000 のみ）

DU(mp)

デフォルトの `DU=OFF` では、アベンド時にメモリダンプが生成されません。この設定により、Natural ESTAE がセッション中アクティブになります。 Natural ESTAE がアクティブな場合、すべてのプログラムアベンドがトラップされ、Natural セッションは終了するのではなく、再スタートします。 これは重要なパフォーマンスの考慮事項です。

DYNPARM

デフォルトの DYNPARM=ON は、Natural の起動時に提供されるダイナミックパラメータを処理します。 各バッチ Natural サブシステムは、少なくとも 1 つのパラメータ（`STACK=`）を指定して起動されるため、このデフォルト設定を使用する*必要*があります。 詳細は「*NATPARM のダイナミックな上書き*」を参照してください。

ETA

ETA は、エラートランザクションプログラムです。 指定しないでください。 Adabas トリガとストアドプロシージャ機能では、独自の要件に従って、Natural セッションのエラートランザクションプログラムが設定されます。

Note: Adabas トリガとストアドプロシージャ機能は、単独で 1500 バイトを使用します。そのため、ETA プログラムが GDA で 4K を使用する場合の ESIZE は約 6K になります。

ETID

ETID が使用される場合、つまり `ETID=' '`（空白）を指定しない場合、各タスク（最大で 10 タスク）にユニークな ETID（Adabas ユーザー ID）を割り当てる必要があります。 Natural

Security が使用され、かつ NATPARM 値が ETID に指定されていない場合、ライブラリプロファイルの RESTART オプションおよびユーザープロファイルの ETID オプションを "N" に指定することで、エラーメッセージ NAT3048 および NAT3009 を防ぐことができます。

EXTBUF

VSE および z/VM 環境では、EXTBUF パラメータを 10 以上に設定します。 z/OS 環境で EXTBUF は必須ではありません。

リミットパラメータ

バッチ Natural サブシステムは長時間に渡るトランザクションであるため、LE、LT、MADIO、MAXCL、および MT パラメータを制限付きで設定しないでください。これらのパラメータを制限付きで設定すると、プロシージャの実行時にエラーが発生することがあります。この場合、トリガを起動したコマンドは、レスポンスコード 155 または 156 を受け取ります。

Note: Adabas トリガプロファイルのタイムアウトパラメータによって、長時間に渡るプロシージャの問題は自動的に解決します。 または、DBA がプロシージャの実行でビジーなサブシステムをキャンセルすることもできます。

NTLFILE

NTLFILE パラメータは、Adabas トリガファイルを指すようにコーディングする必要があります。 トリガファイルの論理ファイル番号は 154 です。 または、Natural NTLFILE パラメータを使用してダイナミックに上書きするように NTFILE パラメータを指定します。

PROFILE

PROFILE パラメータは指定しないでください。 このパラメータにより、すべての Natural コントロールブロックが初期化される前に、Adabas コールが発行されます。 それにより、ストアドプロシージャとトリガの初期化時に予期できない結果が生じます。

PROGRAM

PROGRAM パラメータは指定しないでください。 このパラメータはバッチ Natural サブシステムの起動時に使用され、STPEND に設定されます。 このモジュールは Adabas トリガとストアドプロシージャのロードライブラリ内にあり、必要に応じてロードできるように MPMJCL steplib に追加する必要があります。

STACK

STACK パラメータは指定しないでください。 このパラメータはバッチ Natural サブシステムの起動時に使用され、Adabas トリガとストアドプロシージャ機能によって指定されます。

NATPARM のダイナミックな上書き

Natural ニュークリアスで指定される NATPARM 値は、Adabas トリガプロファイルを使用して、または Natural Security がインストールされている場合は Natural Security を使用して、ダイナミックに定義できます。

NATPARM パラメータの値は、Adabas トリガプロファイルから取得され、Natural サブシステムの初期化時に使用されます。 プロファイルで指定される NATPARM 値は、Natural ニュークリアスで指定される値を上書きします。

NATPARM の定義を修正する前に、「*特殊な要件*」を参照してください。 各 Natural サブシステムはさまざまなアプリケーションやファイルに関するトリガ要求を受け入れることができるため、Natural 環境を適切に設定することが重要です。

STACK パラメータを渡す必要があります。Natural サブシステムは、Natural トリガドライバの初期化後に Natural トリガドライバに制御を渡します。 STACK パラメータの値は次のように固定され、ユーザーからは変更できません。

```
STACK=(LOGON:SYSSPT;STP)
```

ここでは次の内容を表しています。

SYSSPT	インストール手順で、ストアドプロシージャアプリケーションに対して INPL が実行されたライブラリ。 これは、実行可能なプロシージャを格納するライブラリでもあります。
STP	Natural トリガドライバの起動ルーチンです。

その他のすべての NATPARM 値は、必要に応じてユーザーが設定する必要があります。

Natural セッションは、Adabas ニュークリアスがアクティブな限り、通常は実行し続けます。したがって、各 Natural サブシステムは長時間に渡るタスクです。Natural セッションのリミットパラメータを適宜設定する必要があります。 Natural サブシステムの実行中は、セッション設定を修正できません。 ただし、特定のセッションパラメータは、通常の Natural プログラミングオプションと同様の方法でプロシージャを使用して修正できます。

プリンタの考慮事項

バッチ Natural サブシステムでは、プリンタは常に CMPRINT ラベルとサブシステムの番号によって指定されます。 単一のアドレススペースで実行している場合は、競合を防ぐために CMPRINT に別の割り当てを設定する必要があります。複数の Natural サブシステムが実行中であり、それらのいずれか、またはすべてで、出力が実行される可能性があるためです。

さまざまなサブシステムで割り当てられたプリンタのオープンとクローズが行われている場合でも、これらが競合しないという保証はありません。 競合が発生すると、オペレーティングシステムでエラーが発生します。

例えば WRITE (nn)、PRINT (nn)、または DISPLAY (nn) オプションを使用する場合、"nn" はすべてのサブシステムに永続的に割り当てられた固有の番号です（CMPRINT の場合と異なる）。

サブシステム 01 で実行中のプロシージャが WRITE (01) を実行し、サブシステム 02 で実行中の別のプロシージャが WRITE (01) を実行すると、オペレーティングシステムでエラーが発生し、最終的には Natural からエラーを受け取ります。このような状況を防ぐ必要があります。

ダイナミックな CMPRINT 割り当て

Naturalサブシステムが起動すると、Adabasトリガドライバは、Adabasトリガプロファイルの "CMPRINT 割り当て" 定義に基づいてダイナミックなプリンタ割り当てを決定します。

トリガプロファイルの CMPRINT 割り当ては 1〜6 バイトのフィールドで、MPM JCL のファイル割り当てと一致する必要があります。 このプリンタ割り当ては、サブシステムの起動時にサブシステムタスク番号の接頭辞として使用されます。

例えば、プリンタ TSPRT を指定する場合、サブシステム 01 と 02 では、MPM 起動 JCL/JCS でラベル TSPRT01 と TSPRT02 が定義されることが想定されます。

Adabasトリガとストアドプロシージャ機能は、バックグラウンドで実行されるサブタスクまたはサブシステムです。 複数の Natural サブシステムがあり、プロシージャを特定のサブシステムから実行されるようにすることは不可能です。情報の出力でWRITE、PRINT、またはDISPLAYステートメントが使用されるときに、CMPRINT がダイナミックに割り当てられるためです。

プロシージャに渡される PDA には、ユニークなサブシステム ID が含まれるため、DECIDE ステートメントを使用して CMPRT01〜CMPRT31 に出力できるようになります。

例

```
DECIDE ON FIRST VALUE OF RQ-TASK    /*check subsystem number
  VALUE  '01' WRITE  (1)  NOTITLE NOHDR  text
  VALUE  '02' WRITE  (2)  NOTITLE NOHDR  text
  VALUE  '03' WRITE  (3)  NOTITLE NOHDR  text
  VALUE  '04' WRITE  (4)  NOTITLE NOHDR  text
  VALUE  '05' WRITE  (5)  NOTITLE NOHDR  text
  NONE        WRITE       NOTITLE NOHDR  text
  END-DECIDE
```

最後の例（NONE）では、CMPRINTラベルのダイナミック割り当てで指定されたプリンタに出力されます。

サブシステム 01 で実行中のプロシージャが出力を実行する場合は、CMPRINT のダイナミック割り当てに出力することも、DECIDE ステートメントで示すように、MPM JCL/JCS で指定されるプリンタ 01（CMPRT01）に出力することもできます。

複数のプリンタが必要な場合は、出力ファイルの範囲を定義できます。 例えば Natural サブシステムが 5 つ定義されている場合、出力ファイルの範囲は CMPRT01〜CMPRT05、または CMPRT06〜CMPRT10 のように定義できます。

BS2000 のみ

宣言しない場合、CMPRINT は SYSLST に書き込みます。 サブタスキングにより、次の構造で自動的にファイルを割り当てられます。

```
L.L.<db_task_number>.<natural_load_name>.<timestamp>
```

ここでは次の内容を表しています。

<db_task_number>	トリガおよびストアドプロシージャ機能を実行しているデータベースのタスク番号
<natural_load_name>	バッチ Natural モジュールの名前
<timestamp>	サブタスクが開始された時間を示す文字

ワークファイルの考慮事項

ワークファイルのラベルの形式は、次のとおりです。

```
CMWKF<i>nn</i>
```

"nn" は、ワークファイル番号です。

ワークファイルの使用方法は、若干の考慮が必要になる可能性があります。しかし、ワークファイルに対する読み書きがない場合は、ワークファイルをJCL/JCSに配置する必要はありません。

ワークファイルが必要な場合、競合が発生しないようにする方法を確立する必要があります。プリンタの考慮と同様のことがワークファイルにも適用されます。

例えば、プロシージャの読み書き操作は、タスク番号、つまりプロシージャが実行されるNaturalサブシステムの番号に応じて、特定のワークファイルがターゲットとなる可能性があります。この場合、タスク 01 は CMWKF01 を読み書きし、タスク 02 は CMWKF02 を読み書きします。

プロシージャが実行されているサブシステムを特定できるため、そのサブシステムに永続的に割り当てられているワークファイルを使用できます。

「*プリンタの考慮事項*」で説明するように、DECIDE ステートメントを使用して、このような割り当てを行えます。 プロシージャの使用方法によっては、別の解決方法を使用することもできます。

Natural Security の考慮事項

このセクションでは、ストアドプロシージャ使用時の Natural サブシステムの実行に関するセキュリティについて説明します。 Natural Security 環境で Adabas トリガとストアドプロシージャを使用するときの情報です。

ログオン

Natural にログオンするときは、AUTO=ON と AUTO=OFF のどちらも使用できます。

- AUTO=OFF を使用するときは、ユーザー ID とパスワードを指定し、Adabas トリガプロファイルの "NATSEC LOGON Required" の値で "Y" を指定する必要があります。 そうでないと、問題が発生します。

  入力するパスワードとユーザー ID は可変でも固定でもかまいません。 可変名の場合は、値 ** を使用する必要があります。 値がユニークになるように、Natural サブシステム番号で置換されます。 例えば USER** と PSWD** を指定した場合、ユーザー ID とパスワードは、次のように生成されます（3 つのタスクがあるとします）。

タスク	ユーザー ID	パスワード
01	USER01	PSWD01
02	USER02	PSWD02
03	USER03	PSWD03

  ユーザー ID は、ETID とバッチユーザー ID を考慮して、Natural Security に対して定義する必要があります。 同じユーザー ID が使用された場合は、レスポンスコード 9 または 48 が返され、Natural セッションは無効になります。 Natural サブシステムはすべて同時に実行されていて、それぞれ相互に独立している必要があることに注意してください。

- AUTO=ON を使用するときは、ジョブ名または割り当てられたバッチユーザー ID が実際のログオン ID に使用されます。 タスクの最大数によっては、複数のユーザーがサインオンします。 Natural サブシステムが発行するレスポンスコード 9 または 48 を受け取らないようにするには、このことを考慮する必要があります。

ライブラリ設定

ライブラリ SYSTRG を定義します。 起動パラメータを、SYSAOS からログオンするときは " "（空白）に設定し、ログオン画面から直接ログオンするときは "menu" に設定します。

ライブラリ SYSSPT を定義します。 Natural サブシステムは、このライブラリにログオンします。

どちらのライブラリにも起動、エラー、または再スタート設定を含めないでください。 これらの設定は、Adabas トリガとストアドプロシージャ機能の初期化時に自動的に確立されます。

ライブラリは保護できます。ただし、Adabas トリガプロファイルを使用して定義されたユーザーには、ログオンして必要な処理すべてを実行するのに十分な権限が必要です。 例えば、あるモジュールでサブシステムのユーザー ID を許可しないと、Natural トリガドライバがそのルーチンを呼び出したときにエラーが発生してしまうため、よい方法とは言えません。

セキュリティの制限

Natural Security では、次のような特定の NATPARM 設定を上書きできます。

- 非アクティビティログオフリミット
- トランザクション経過時間
- CPU 時間（MT=）
- 最大 Adabas 呼び出し数（MADIO=）

Natural サブシステムは、複数の "サブトランザクション" を実行する長いトランザクションの 1 つです。 セキュリティの制限がこれらの 1 つ以上に設定されている場合、制限はそのセッションのすべてのプログラムに適用されます。

パラメータの設定

Adabas トリガとストアドプロシージャサブタスクに影響のある Natural Security パラメータの設定について次に説明します。

Error Program
Error Program は指定しないでください。エラーは、Adabas トリガとストアドプロシージャ機能によって内部的に処理されます。

ETID
トランザクションデータを格納するプロシージャが、異なるサブシステムから続いて呼び出されることがあります。 正しく動作するには、ルーチン GET TRANSACTION DATA がプロシージャの呼び出し元サブシステムを認識している必要があります。 これが問題となる場合、ユーザーは別の形式でデータ回復する方法を構築します。 レスポンスコード 9 または 48 を防ぐため、ETID オプションは慎重に使用してください。

Library Protection
Library Protection は必須ではありませんが、Adabas Online System ユーザーは SYSTRG にログオンできる必要があり、Natural トリガドライバは SYSSPT にログオンできる必要があります。

MADIO
Natural サブシステムの実行時間が長時間にわたる場合があるため、制限を超えないように 0（ゼロ）を設定します。

MAXCL
Natural サブシステムの実行時間が長時間にわたる場合があるため、制限を超えないように 0（ゼロ）を設定します。

Non-Activity Logoff Limit
Natural サブシステムの実行時間が長時間にわたる場合があるため、制限を超えないように 0（ゼロ）を設定します。

Password Change Option
Natural Security ユーザープロファイルには、n 日ごとにユーザーにパスワードの変更を要求するときに使用できるオプションがあります。 このオプションを使用する場合、Adabas トリガプロファイルの NATSEC パスワード も修正する必要があります。

Restart Program
指定しないでください。 再スタートは、Adabas トリガとストアドプロシージャ機能によって内部的に処理されます。

Steplibs
制限はありません。 Adabas トリガとストアドプロシージャ機能は、SYSSPT にログオンします。 そのため、すべてのプロシージャは必要に応じて steplibs 内にある必要があります。

Startup Transaction
指定しないでください。 Startup は、ログオン時に Adabas Online System または Natural トリガドライバによって自動に設定されます。

5 処理とパフォーマンス

Adabas トリガドライバは、Adabas ニュークリアスの一部として実行されます。 一般にトリガの実行時処理全体を制御します。 トリガを起動するかどうか、Natural トリガドライバを開始するかどうか、および相互に作用するかどうかを判断して、プロシージャが正確に適切な時間の範囲で処理するようにします。

このchapterでは、次のトピックについて説明します。

初期化

Adabas ニュークリアスの開始時に、ADARUN パラメータ SPT=YES が指定されているかどうかを判断します。指定されている場合は、Adabas トリガドライバに制御を渡して初期化できるようにします。 初期化時に、Adabas トリガドライバは、以下に説明するようなアクティビティを実行します。

Adabas トリガプロファイルの確認

Adabas トリガドライバは、Adabas トリガプロファイルを確認し、セッションのトリガとストアドプロシージャの処理時に使用されるセッションパラメータを抽出します。 トリガファイルにプロファイルが存在しない場合、ニュークリアスの Adabas トリガとストアドプロシージャ機能の初期化は、適切なエラーメッセージを表示して終了します。

- Adabas トリガドライバは、プロファイルで"トリガステータス"および"ストアドプロシージャステータス" パラメータ設定を確認します。

トリガ	ストアドプロシージャ	処理内容
非アクティブ	非アクティブ	Adabas トリガとストアドプロシージャ機能は、開始が許可されません。 Adabas トリガドライバにとって、"非アクティブ" 設定は、ADARUN パラメータ設定 SPT=NO と同じ意味です。
アクティブ	非アクティブ	トリガ処理は開始しますが、ストアドプロシージャを実行するあらゆる要求は、レスポンスコード 22 で拒否されます。
非アクティブ	アクティブ	ストアドプロシージャ処理は開始しますが、トリガを実行するあらゆる要求は、 レスポンスコード 22 で拒否されます。

- Adabas トリガドライバは、Natural サブシステムに関する情報をプロファイルから取得します。
- サブシステム名。プロセスを実行するリンク先の Natural ニュークリアスの名前、つまり、インストール手順のパート 3 で Natural ニュークリアスに割り当てられた名前に対応する必要があります。
- 起動されるトリガによって生成されるワークロードを処理するために、開始する必要のあるサブシステムの数。

トリガが 1 つ以上存在することの確認

Adabas トリガドライバは、トリガファイルにトリガ定義が 1 つ以上存在することを確認します。 トリガ定義がない場合、トリガステータスが "非アクティブ" に設定されている場合でも、Adabas トリガとストアドプロシージャ機能は初期化されず、適切なエラーメッセージが表示されます。

ストレージの確保

Adabas トリガとストアドプロシージャ機能で必要になるストレージの合計は、次の条件で変わります。

- 必要なワークエリアのサイズ
- トリガテーブル、プレトリガキュー、およびポストトリガキューに必要なバッファサイズ
- Natural サブシステムに必要な容量

Natural サブシステムに必要な容量は、Natural ニュークリアスのサイズと、Natural 環境で必要な各種バッファ（ESIZE、DATSIZE、FSIZE など）のサイズによって決まります。

Adabasニュークリアスのリージョンやアドレススペースの全体サイズが小さすぎる場合、Natural サブシステムを実行できません。 この場合に通常表示されるエラーメッセージは、"ストレージが不足"、またはサブシステムのアベンドを表します（レスポンスコード 40000109 または 40000008 が返される）。 このような場合は、リージョンやアドレススペースのサイズを増やすか、またはプロシージャの実行に使用する Natural サブシステムの数を減らしてください。

Naturalニュークリアスを分割して、ニュークリアスのストレージ要件を最小化することをお勧めします。

トリガテーブルの作成

トリガファイルにはトリガを1つ以上定義する必要があります。そうでない場合、プロセスは続行できません。

Adabas トリガドライバがトリガファイルの正当性を確認したら、トリガ定義が読み込まれ、エントリがトリガテーブルに追加されます。 ニュークリアスクラスタ環境では、トリガテーブルはニュークリアスごとに再読み込みされませんが、すでにアクティブなニュークリアスから取得されます。

システムの整合性を維持するために、REFRESH コマンドがトリガメンテナンス機能から発行されるまで、トリガテーブルは新規、修正、または削除されたトリガで更新されません（「*トリガテーブルの更新*」を参照）。

トリガテーブルにより、パフォーマンスは向上します。 Adabas トリガドライバは、コマンドの処理時にトリガファイル自体でトリガの存在を確認するのではなく、トリガテーブル、つまりメモリ内のバッファを確認するにすぎません。 テーブルの順序によって、トリガドライバは、トリガを起動するかどうかをすばやく判断できるようになります。

トリガファイルを読み込んでトリガテーブルのエントリを判断するときに、Adabas トリガドライバは次の処理を行います。

- トリガ、チェックポイント、またはセキュリティファイルのエントリをすべて無視します。
- 非アクティブ化されたトリガをロードします。ただし、トリガメンテナンス機能のトリガ修正機能からトリガがアクティブ化されるまで無視します。 「*単一トリガの定義*」を参照してください。
- データベースの最大ファイル番号を判断し（使用されている最大のファイル番号+10）、最大番号よりも大きいファイル番号に対するトリガをすべて無視します。

範囲外のファイル番号を無視すると、REFRESH コマンドの使用時に問題が発生する可能性があります。 解決方法として、実際の最大ファイル番号よりも大きいファイル番号のダミーファイルをロードする方法があります。 これで、実際のファイル番号よりも大きく、ダミーファイル番号よりも小さいファイル番号の新しいファイルを追加できるようになります。 ニュークリアスの初期化時に固定サイズバッファを確立することで、このバッファのストレージ要件は最小化されます。2バイトファイル番号を使用する場合、合計の最大サイズが非常に大きくなる可能性があります。

Natural サブシステムの開始

Adabas トリガドライバが開始されると、プロシージャを実際に実行する Natural サブシステムが開始されます。 Adabas トリガプロファイルの "最大サブシステム" パラメータによって、開始するサブシステム数（1～10）が決まります。

各サブシステムは、通常、Adabas アドレススペース内で実行するように最低限修正されたバッチ Natural ニュークリアスです。 これは、MPM 起動 JCL/JCS で指定されるリージョンサイズに影響します。

サブシステムが開始すると、Adabas トリガドライバは、サブシステムのステータスやアクティビティにおけるすべての変更を記録します。そのため、サブシステムが呼び出す Adabas トリガドライバとプロシージャのどちらに対しても、各サブシステムは自身をユニークに識別できます。 ユーザーは、トリガメンテナンスの一部であるサブシステムアクティビティ機能を使用して、これらのアクティビティを監視できます。

Natural サブシステムがアクティブになると、次の処理が行われます。

- Natural トリガドライバが制御を取得します。
- Adabas トリガドライバは、ストアドプロシージャ要求またはトリガの起動により発生する可能性があるプロシージャの処理をサブシステムが実行可能な状態であることが通知されます。

サブシステムキューには、処理を待機している各 Natural サブシステムのエントリが格納されます。 Adabas トリガドライバがサブシステムを必要とすると、サブシステムキューを確認して、利用可能なサブシステムキューを調べます。

プロシージャの確認

Adabas ニュークリアスが初期化されると、ユーザー処理は通常どおり継続します。 Adabas トリガドライバは、ニュークリアスが受け取るコマンドごとに、トリガを起動する必要があるかどうかを決定します。 "非アクティブ" とマークされたトリガテーブル内のエントリは無視されます。

プレコマンドトリガの場合、Adabas トリガドライバは Adabas スレッドで処理されるコマンドが選択される前にトリガをチェックします。 このようなコマンドには、READ、FIND、STORE、DELETE、UPDATE コマンドなどがあります。 これらのコマンドに対して、Adabas トリガドライバは起動するトリガがあるかどうかを判断します。ない場合、コマンド処理は通常どおり続行します。エンドトランザクション（ET）、クローズ（CL）、コマンドIDの解放（RC）などのコマンドは確認されませんが、ニュークリアスに直接渡されて通常の処理が行われます。 ストアドプロシージャ要求に対しては、トリガの確認は行われません。

Adabas でコマンドが正常に処理されレスポンスコードがゼロになると、Adabas トリガドライバは起動するポストコマンドトリガがあるかどうか調べます。 ない場合、ユーザーは通常の方法で通知を受けます。 プレコマンドトリガ、またはポストコマンドトリガの確認の結果、トリガを起動する場合、Adabas トリガドライバはトリガ処理に進みます。

トリガテーブルのスキャン

コマンドがトリガを起動する条件を満たしていると判断されると、トリガテーブルがスキャンされます。 パフォーマンス上の理由から、トリガのスキャン順序はシーケンス、または ユーザーによってトリガに割り当てられたプライオリティによって決まります（「*複数のトリガ定義*」を参照）。

2つのトリガが同じコマンドクラスに存在し、プライオリティが同じ場合、トリガファイルからの読み込み順（つまりファイル上のレコードの ISN 順）でトリガがスキャンされます。 そのため、各トリガのプライオリティを正しく指定することが重要です。

トリガは、次の一般的な順序でスキャンされます。

シーケンス	フィールド	コマンド
1	特定フィールド	特定コマンド
2	任意のフィールド	特定コマンド
3	特定フィールド	任意のコマンド
4	任意のフィールド	任意のコマンド

プレコマンドトリガとポストコマンドトリガのキューエントリの作成

コマンドの結果がトリガの起動である場合、またはコマンドがストアドプロシージャ要求であると Adabas トリガドライバが判断した場合、次の場所でエントリが作成されます。

- コマンドが実行されていない場合は、プレコマンドトリガキュー。
- コマンドが正常に実行された場合は、ポストコマンドトリガキュー。

エントリには、トリガを起動するコマンドと、トリガテーブル内の対応するエントリ（例えば実行するプロシージャの詳細）との両方から取得した情報が含まれます。 また、エントリを使用すると、Adabas トリガドライバは起動されたトリガのステータスを記録できます。

Natural サブシステムが処理を待機中の場合、直ちにトリガ要求が渡されます。 それ以外の場合、次のサブシステムが使用可能になるまで、トリガ要求はプレコマンドトリガキューまたはポストコマンドトリガキュー内に残ります。

プロシージャの処理

トリガ要求がプレトリガキューまたはポストトリガキューに置かれ、サブシステムがその要求を受け入れると、Natural トリガドライバの制御下で処理が続行されます。

1 つのコマンドに対して起動できるのは、結果に関係なく 2 つのトリガ（1 つのプレコマンドトリガと 1 つのポストコマンドトリガ）のみです。

コマンドの結果、トリガが起動すると、トリガが非同期と同期のどちらであるか確認されます。

非同期トリガ

トリガが非同期の場合、コマンドはトリガされたプロシージャの完了を待機しません。 コマンドは解放され、トリガがプレコマンドとポストコマンドのどちらであるかに応じてプロセスは正常に継続されます。

プレコマンド	コマンドは、Adabas スレッド内の処理で使用できます。 トリガされたプロシージャは、コマンドの実行後、またはコマンドの実行と同時に処理できます。
ポストコマンド	トリガされたプロシージャとコマンドは、独立して処理されます。 プロシージャの結果に関係なく、トリガが起動されるとユーザーに通知されます。

同期トリガ

トリガが同期（関与または非関与）の場合、Natural トリガドライバが Adabas トリガドライバに対してプロシージャの実行完了を通知するまで、コマンドは保留されます。

リターンコードがゼロの場合、コマンドは解放されて処理を続行します。

リターンコードがゼロ以外の場合、ユーザーはレスポンスコード 155 または 156 と次の情報を受け取ります。

- アディション3フィールドには、起動したトリガの結果として実行されたプロシージャの名前が格納されます。
- アディション4フィールドの最初の2バイトには、プロシージャからの実際のリターンコードが格納されます。
- アディション4フィールドの次の2バイトには、起動したトリガのタイプを示すサブコードが格納されます。
  - サブコード 15 は、プレコマンドトリガを表します。
  - サブコード 16 は、ポストコマンドトリガを表します。

結果の処理

プロシージャを実行すると、結果がトリガ要求エントリに格納され、ステータスが適切に更新されます。 Adabas トリガドライバは、この処理を検出すると、コマンドのトリガ処理を "ファイナライズ" します。

プレコマンドトリガとポストコマンドトリガのどちらでも、プロシージャからのリターンコードによって、結果の処理方法を指定します。次のセクションで、この結果の処理方法について説明します。

プレコマンドトリガ

リターンコード	処理内容
ゼロ	コマンドは "解放" されて、実行できるようになります。
ゼロ以外	コマンドは実行されず、ユーザーは Adabas コントロールブロックのレスポンスコードフィールドでレスポンスコード 155 を受け取ります。

コマンドがAdabasスレッドによって実行されると、ポストコマンドトリガ処理でそのコマンドを再選択することができます。

同期プレコマンドトリガの特殊な処理

同期トリガでリターンコード "1" は、プロシージャが処理を正常に完了したことを表します。Adabas コントロールブロックのレスポンスコードフィールドは、正常終了を表すゼロに設定されます。 コマンドは、ニュークリアスによる実行から "解放" されていませんが、その代わりにプロシージャの結果が直ちにユーザーに返されます。

この特殊な処理で使用するプロシージャは、レコードバッファに対する読み書きアクセス権を持ち、ストアドプロシージャの場合と同様の方法でコマンドを処理する必要があります。「*フォーマットバッファとレコードバッファの使用*」を参照してください。

コマンドの実行前に、プレコマンドトリガのプロシージャでレコードバッファの内容を修正することができ、これはコマンドを更新したり格納したりするときに便利です。

ポストコマンドトリガ

リターンコード	処理内容
ゼロ	コマンドは成功したとみなされ、ユーザーは Adabas コントロールブロックでレスポンスコード 0（ゼロ）の通知を受けます。
ゼロ以外	Adabas コントロールブロックでレスポンスコード 156 が返されます。 プロシージャはゼロ以外のコードを返しても、実際にはコマンドは成功していることがあります。コマンドの実行結果は、そのコマンドを発行したアプリケーションで解釈する必要があります。

Note: ポストコマンドトリガが起動し、プロシージャからのリターンコードがゼロ以外の場合、コマンドが正常に実行された場合でも、レコードバッファ内のデータは返されません。

同期ポストコマンドトリガの特殊な処理

同期でかつレコードバッファに対して読み書きアクセス権を持つポストコマンドトリガは、実行に成功するかどうかに関係なく、レコードバッファを修正していることがあります。

関与トリガの場合、トリガの結果によって、コマンドの結果が変更されている可能性があります。 例えば正常に実行された UPDATE コマンドによってポストコマンドトリガが起動し、このトリガのプロシージャが正常に完了しなかった場合、BT コマンドが実行されることもあれば、実行されないこともあります。 ユーザーが通知を受ける場合は、レスポンスコード 156 が返されます。 元のコマンドを発行したアプリケーションは、UPDATE コマンドが有効なままであるかどうかを判断して、適切なアクション（ET または BT）を実行する必要があります。

シャットダウン

シャットダウンは、次の状況で発生する可能性があります。

- Adabasトリガドライバは、失敗したサブシステムを記録します。 すべてのサブシステムが失敗すると、Adabasトリガドライバは、プロシージャをこれ以上処理することができないと判断し、終了します。 シャットダウン処理は、"エラーアクション" 値によって異なります（表「エラーアクション」を参照）。
- ニュークリアスが ADAEND または HALT オペレータコマンドを受け取り、Adabasトリガドライバにシャットダウンを指示します。

HALT	Adabas トリガドライバは直ちに終了します。
ADAEND	すべてのサブタスクアクティビティがステータスがETになったときに、Adabasトリガドライバが終了します。 サブタスクがビジーでかつ ET ロジックに関わっている場合、現在のニュークリアスにコマンドを発行している途中でも、完了することができます。そうでない場合は、コンソールにメッセージが表示され（メッセージ ADAN9M を参照）、サブタスクはトランザクションが未完了のまま終了します。

シャットダウン処理のステップ

手順 5.1. シャットダウン処理のステップは次のとおりです。

1 プレトリガキューとポストトリガキューで、待機中のトリガがないかどうか確認されます。同期トリガの実行完了を待機しているすべてのユーザーに対して、レスポンスコード148が発行されます。

2 ユーザーは、通常の方法で、完了済みのポストコマンドトリガについて通知を受けます。

3 完了済みのプレコマンドトリガに対して、レスポンスコード148が発行されます。これらはAdabas スレッドで処理されないためです。 これらのコマンドが ET トランザクションの一部である場合、ユーザーは BT および ET コマンドを適宜発行する必要があります。

4 シャットダウン開始後に検出されたすべてのポストコマンドトリガに対して、レスポンスコード157（"コマンド拒否"）が発行されます。 シャットダウンの開始前にコマンドは実行されましたが、トリガされたプロシージャは実行されません。

5 5秒後にアクティブなままのすべてのサブシステムが強制的に終了し、メッセージがコンソールに表示されます。 実行したままのプロシージャが含まれるサブシステムは"アクティブ" であるとみなされます。Natural プログラムがバッファプール内にあり、プログラムは別のデータベースに対するデータ呼び出しを発行中である可能性も、まったく発行していない可能性もあります。 この場合、現在のデータベースに対して発行された "停止" は、効果がないことがあります。

6 すべてのサブシステムがシャットダウンされます。

7 トリガとストアドプロシージャの合計数がコンソールに書き込まれ、Adabas トリガプロファイルでトリガのステータスフィールドが "非アクティブ" に設定されます。 ニュークリアスは、通常の方法でシャットダウン処理を続行します。

エラーアクション

Adabas トリガドライバ自体によってシャットダウンが要求された場合のシャットダウン処理は、Adabas トリガプロファイルのエラーアクションフィールドに割り当てられた値によって異なります。

処理内容	シャットダウン処理
停止	ニュークリアスも終了する必要があります。 ADAEND 要求は、Adabas トリガドライバ自体から発行されます。 サブシステム内でまだ実行中のすべてのユーザーアプリケーションが終了します。
無視	Adabas トリガとストアドプロシージャ機能によって、Adabas ニュークリアスが終了しますが、ニュークリアスの処理は、ADARUN パラメータ SPT=NO が指定されたのと同様に通常の方法で続行します。 拡張コマンド処理を実行するトリガは起動されず、整合性の問題が発生する可能性があります。
拒否	通常ならトリガが起動されるすべてのコマンドで、レスポンスコード 157 が返されます。 Adabas トリガとストアドプロシージャ機能はアクティブなままですが、すべてのサブシステムはシャットダウンし、プロシージャの処理は中断されます。

ニュークリアスクラスタ環境では、1 つのニュークリアスが "無視" または "拒否" ステータスに設定されると、クラスタ内のすべてのニュークリアスもそのステータスに設定されます。

異常終了

次のコンポーネントがシャットダウンする状況については、「シャットダウン」を参照してください。

- Adabas ニュークリアス、Adabas トリガドライバ、およびすべてのサブシステム
- Adabas トリガドライバとすべてのサブシステム、ただし Adabas ニュークリアスを除く
- すべてのサブシステム、ただし Adabas トリガとストアドプロシージャ機能または Adabas ニュークリアスを除く

Natural ESTAE/STXIT 処理

NATPARM で DU=OFF（デフォルト、アベンド時にメモリダンプが生成されない）が指定された場合、Natural ESTAE/STXIT がセッション時にアクティブになります。

Natural ESTAE/STXIT がアクティブなときに Natural サブシステム内でプログラムのアベンドが発生すると、アベンドがトラップされます。Natural ESTAE/STXIT 出口は制御を取得し、クリーンアップと Adabas トリガドライバへの通知を行い、Natural トリガドライバを再スタートします。

このようにして、Naturalセッションは終了せずに再スタートします。 これは重要なパフォーマンスの考慮事項です。

DU=ON を使用する場合、ESTAE/STXIT はアクティブ化されず、サブシステムは異常終了します。 サブシステムを終了して再スタートするために Adabas トリガドライバを再スタートする必要がある場合は、パフォーマンスが低下します。

Natural サブシステムのアベンド

実行中のプロシージャがタイムアウトリミットを超えたり、または処理の完了前に DBA によってキャンセルされたりすることがあります。（タイムアウトは、CPU タイムの使用に対する経過時間を意味します）。

プロシージャにおけるこれら 2 つの "異常な" 終了は、似ていますが、次の点が異なります。

- キャンセルは、トリガメンテナンス機能から手動で行います。
- タイムアウトは、Adabas トリガプロファイルのアクティビティタイムアウト設定に基づいて、自動的に発生します。

実行中のプロシージャは、待機中、ループしている、または単に想定よりも長く実行中など、あらゆる段階で終了させる必要性が生じる可能性があります。 処理をインターセプトする確実な唯一の方法は、サブシステム自体を終了することです。 この終了を確認するには、トリガメンテナンス機能からサブシステムを監視します。「サブシステムアクティビティ」を参照してください。

トリガが同期の場合、ユーザーは、サブシステムが終了したことを通知される必要があります。トリガタイプ（プレコマンドまたはポストコマンド）に応じて、レスポンスコード 155 または 156 と、タイムアウトを表すサブコード 9 が設定されます。ユーザーはサブシステムがタイムアウトしたのか意図的に終了されたのか常に通知を受けます。コンソールには追加情報のメッセージが書き込まれます。

異常終了が発生した場合は、理由を確認して、問題を修正してください。 それでもこの現象が繰り返し発生する場合は、問題が解決するまでトリガを非アクティブ化してください。プロファイルのトリガアクティビティの記録オプションを使用すると、理由を簡単に特定できます。

Natural サブシステムの再スタート

Natural が回復できず、サブシステムが終了する場合、Adabas トリガドライバはエラーの通知を受け、メッセージがコンソールに書き込まれます。

Natural サブシステムは、終了すると自動的に再スタートします。

- 再スタートに成功すると、サブシステムは通常の方法で再初期化されます。 現在実行する処理がない場合、サブシステムはキューに置かれ、新しいトリガが起動されてプロシージャの処理が必要であることを Adabas トリガドライバが通知するまで待機します。
- 再スタートに失敗すると、さらに 2 回再スタートを試みます。 3 回連続で再スタートに失敗した後で起動に失敗すると、サブシステムは永続的に非アクティブ化され、ユーザーに通知するメッセージがコンソールに表示されます。 ニュークリアスをシャットダウンして再スタートするまで、そのサブシステムは再アクティブ化できません。

Natural トリガドライバを再開する再スタートルーチンは、STP です。

同期トリガによる処理の完了を待機しているユーザーは、トリガが正常に完了しなかったことの通知を受けます。 レスポンスコード 155 または 156 と、アディション 4 フィールドに追加情報が返されます。

コマンドロギング

トリガされたプロシージャはコマンドを拒否することがあるため、Adabas トリガとストアドプロシージャを実行するときは、ユーザーが発行したコマンドが Adabas スレッドで実行されないことも、ニュークリアスのスーパーバイザに認識されないこともあります。

そのため、Adabas トリガとストアドプロシージャ機能で次のログを取得する必要があります。

- PC コマンド（ストアドプロシージャ要求）を受け取ったとき
- コマンドがプレコマンドトリガまたはポストコマンドトリガで選択されたとき
- プレトリガまたはポストトリガされたコマンドがゼロ以外のレスポンスを受け取ったとき

コマンドログレコードを処理するときのログレコードタイプは次のとおりです。

- プレコマンドトリガの場合は X'0005'
- ポストコマンドトリガの場合は X'0006'

ログレコードの Adabas コントロールブロックで、アディション 3 フィールドには呼び出されるプロシージャの名前が格納され、アディション 4 フィールドには、プロシージャに関する次の情報が格納されます。

バイト	内容
1〜2	プロシージャからのレスポンスコード。
3	プレトリガは "P"、プロシージャ呼び出しは "R"、ポストトリガは "S"
4	非同期は "A"、非関与は "N"、関与は "P"
5	読み込みは "R"、検索は "F"、更新は "U"、格納／追加は "S"、削除は "D"、プロシージャ呼び出しは "P"
6	フラグ設定：パラメータなしは "1"、レスポンスコードのみは "2"、制御情報は "4"、特別なストアドプロシージャパラメータは "8"、レコードバッファアクセスは "10"、レコードバッファ更新は "80"
7〜8	トリガに関連付けられたフィールドの名前

Note: ゼロ以外のレスポンスを返す非同期トリガに対して、ログは作成されません。トリガの処理前にコマンドが完了した場合、非同期トリガの CQE アドレス（4 番目のパラメータ）はゼロに設定されます。

6 プログラミングとパフォーマンス

このchapterでは、効率的なプロシージャを作成するためのガイドラインとオプションについて説明します。Naturalを使用する際の特別な要件と、フォーマットバッファとレコードバッファの使用、およびレコードバッファ抽出ルーチン STPRBE についても説明します。

このchapterでは、次のトピックについて説明します。

プロシージャの作成

プロシージャを作成するときは、パフォーマンスについて検討することが重要です。 すべてのユーザーのすべてのコマンドを処理するために使用できるサブシステムの数は、10 までです。著しく長い処理時間を必要とするプロシージャがあると、サブシステムはその他の要求に対応できなくなります。

すべてのサブシステムに重い負荷がかかると、トリガとストアドプロシージャ要求のキューイングが過剰に発生します。 同期のトリガとストアドプロシージャのコマンドは強制的に待機状態になり、ユーザーにとってはレスポンスタイムが悪化する可能性があります。

ストアドプロシージャ

ストアドプロシージャは、ライブラリ SYSSPT に作成することも、SYSSPT の steplib である別のライブラリに作成することもできます。

ストアドプロシージャは、通常の Natural サブプログラムで、次の特徴があります。

- 提供される PDA のパラメータ定義を使用する必要があります。
- Natural コール構造でレベル 1 にならないようにする必要があります。
- エラーが Natural トリガドライバで直接処理される場合は、*ERROR-TA の現在の設定を変更できません。

トリガ

トリガされるプロシージャが実行する機能は、特定のファイルに基づきます。 特定のコマンドやフィールドを使用することもあります。 プロシージャが実行する必要のある操作の数を減らすため、機能は非常に特化した内容にする必要があります。 任意のコマンドで、最大でプレコマンドトリガ 1 つとポストコマンドトリガ 1 つを起動できます。

非同期トリガ

非同期トリガは、実際にイベントが発生したかどうかを判断するときに役立ちます。

- ファイル番号だけが条件である場合、プロシージャを使用してファイル全体に対するアクティビティをレポートしたり監査したりすることができます。

  コマンドに関する情報（つまりAdabasコントロールブロック）は利用できますが、レコードバッファの内容は利用できません。コマンドはプロシージャの実行が完了するまで待機しないで続行するためです。 ただしレコードは、コントロールブロック内のISNを使用して読み込むことができます。
- "コマンド" が条件として追加されると、トリガが起動される回数は少なくなります。

  非同期トリガを使う利点とは、トリガを起動するコマンドがプロシージャの完了を待機する必要がないということです。ただし非同期トリガは、一般にプロシージャのロジックがアプリケーションと独立している場合だけ使用するようにしてください。

同期トリガ

同期トリガは、通常はアプリケーションに関連した依存性があります。つまり、コマンドを処理する前または後で何らかのアクションを実行する必要があります。 コマンドは、プロシージャが完了するまで続行できません。

同期プロシージャが制御を受け取ると、非同期プロシージャの場合と同様の処理を実行できます。 フィールド名が条件として指定されると、プロシージャはその値を受け取って（レコードのISNを読み込むか、STPRBEをコール）、必要な処理を実行することができます。

マルチトリガのサポートの実装

それぞれのコマンドでは、1つのプレコマンドトリガと1つのポストコマンドトリガのみを起動できます。 複数のフィールドでトリガが必要な場合は、"マルチトリガ"（複数の関連するトリガが含まれている単一のトリガ）をサポートするロジックを実装する必要があります。

特定のファイルやコマンドにマルチトリガを定義する場合は、トリガメンテナンス機能ではなく、プロシージャを使用することをお勧めします。 そのようなプロシージャには、次の特徴があります。

- トリガが起動されたときに制御を取得し、その他のプロシージャを呼び出して起動されるトリガをシミュレートします。
- プロシージャの処理順序を正確に定義できます。
- プロシージャがゼロ以外のレスポンスコードを返す場合に実行するアクション（他のトリガの確認、レスポンスコードをユーザーに返すなど）を定義できます。
- 追加の条件（ファイル番号、コマンド、およびフィールド名以外の条件）を導入できます。例えば、フォーマットバッファに2つのフィールドの組み合わせが存在するときだけプロシージャを呼び出すようにすることができます。

マルチトリガの例

トリガ条件	File 11 Command UPDATE Field '** any field **' Procedure PROC001
コマンド A1	Format Buffer='AA,BB,AB,CA,DF,MT,DT.'
プロシージャ	サブプログラムは、AB、GA、DT、AA、LT、CAの順序でマルチトリガを処理します。PROC001は、制御を取得すると、シーケンス内の各フィールドを確認します。フィールドが見つからない場合、そのフィールドのプロシージャは無視され、処理は次のフィールドに移ります。この例では、フィールドGAとLTが見つかりません（フォーマットバッファ内にない）。いずれかのプロシージャからゼロ以外のレスポンスコードが返されるかどうかに関係なく、処理を続行することができます。

単一トリガとマルチトリガ

トリガのリストが非常に長くなる場合、マルチトリガを1つ使用するのではなく、単一トリガを複数使用するほうが効率よいことがあります。

- 必要に応じて、トリガはファイルとコマンドの組み合わせごとに存在します。
- フィールドがグループ化できる場合、各トリガのトリガフィールドをグループ内のフィールドの1つにして、すべてのアクセスや更新で必ず参照されるようにすることができます。

この方法は、ファイルにレコードタイプがある場合、またはアプリケーションがグループ内のファイルに対してデータを読み込んで更新する場合に意味があります。

例

Employeesのようなアプリケーションに、2つの独立した機能があるとします（Adabasのインストール時に提供されるEmployeesサンプルファイルに基づく）。

- 住所や名前などの個人の詳細情報を更新します。電話番号が必ずデータに含まれる場合、EMPLOYEESファイルのTELEPHONEフィールドに対するUPDATEによってトリガを起動できます。
- 部署、休暇状況、役職など仕事の詳細を更新します。EMPLOYEESファイルのJOB-TITLEフィールドまたはPOSITIONフィールドに対するUPDATEによってトリガを起動できます。

Natural 構文の制限

プロシージャをコーディングするときは、通常の Natural 構文を使用できます。 ただし、以下に説明する制限を考慮する必要があります。

エラー処理

STPPDRIV には ON ERROR 節が含まれているため、プロシージャの結果で発生するすべてのエラーは、STPPDRIV でトラップされます。

- STPPDRIV は、STP ルーチンに制御を渡します。
- STP ルーチンは、再スタート処理を実行します。

再スタート処理の一部として、STP は Adabas トリガドライバに対し、処理中のトリガ要求を終了したこと、結果がユーザーに返されることを通知します。

次の規則が適用されます。

- 制御がコール元ルーチン STPPDRIV に返される場合は、ON ERROR 節を使用することができます。
- パフォーマンス上の理由から、STOP や TERMINATE などのステートメントは使用しないでください。 TERMINATE NN 'message' ステートメントがプロシージャによって発行されると、サブシステムがリターンコード NN で終了し、message がコンソールに書き込まれます。その後、Adabas トリガドライバはシステムを再スタートする必要があります。

これらの規則に従わない場合、Natural トリガドライバが制御を失う可能性があり、このときユーザーはAdabas トリガドライバからバッチ Natural サブシステムをキャンセルする必要があります。 Adabas トリガドライバがエラー処理に関わるときは、追加のリソースが必要になります。

プリンタのサポート

レポートやメッセージは、Natural サブシステムから出力できますが、サブシステムはバッチプロセッサのようには機能しません。 プリンタファイルはさまざまなサブシステムに割り当てることができますが、使用するには注意が必要です。 Natural サブシステムは、アプリケーションでどのプロシージャを実行するのか制御しません。

通常、レポートは、完全な形式で利用できます。 ニュークリアスがアクティブでも、スプールシステムがバッファをフラッシュしていないために、最新のメッセージまたはレポートの出力を利用できないことがあります。

ワークファイルのサポート

ワークファイルはサポートされていますが、実行時にどのワークファイルがサブシステムに割り当てられるのかは不明です。 そのため、プロシージャを実行するサブシステムに応じて、特定のワークファイルに対して読み書きするプロシージャを作成する必要があります。

ET ロジック

エンドトランザクション（ET）ロジックは、通常の方法で動作します。ただし、関与トリガによるに影響に注意してください。

- プロシージャから発行された END TRANSACTION（ET）または BACKOUT TRANSACTION（BT）ステートメントは、トリガを起動したコマンドを発行したユーザーによるすべてのレコードの "保留" ステータスに影響します。
- ET または BT ステートメントは、アプリケーションを同期可能な方法で発行してください。例えば、プロシージャでエラーが発生した場合は、BT を発行して、以前更新されたデータ（プロシージャ内で修正されたデータと、トリガコマンドが属するアプリケーションによって修正されたデータ）にバックアウトできるようにする必要があります。

非同期または非関与トリガが実行を完了したときは、BT を発行して、次のトリガを発行できるように、サブシステムを ET ステータスにする必要があります。

Natural のレベル

Natural プログラムレベルは、FETCH RETURN、CALLNAT、または PERFORM ステートメントが外部サブルーチンを処理するときに変更されます。 Natural トリガドライバが Natural セッションを制御し続けられるように、プログラムレベルを維持することが重要です。そのため、FETCH と STOP ステートメント（STACK TOP COMMAND が前提）は使用しないでください。従って、実際にプロシージャやサブプログラムを呼び出す STPPDRIVは、レベル 1 である必要があります。

バッチモードに適したステートメント

プロシージャは、バッチモードで実行されます。 そのため、INPUT などのステートメントは、STACK DATA ステートメントとともに適切に使用する必要があります。 STACK コマンドは、適切ではありません。

CALLNAT パラメータ

Natural トリガドライバは、CALLNAT ステートメントでサブプログラムを呼び出します。 トリガエントリ内の定義に応じて、サブプログラムでは次のいずれかのオプションが使用されることを前提とします。

1. パラメータは渡されません。

   非常に特化した機能を実行する非同期トリガで通常使用されます。 つまり、トリガイベント条件が、プロシージャがタスクを実行するために必要な唯一の情報です。 レスポンスコードは不要のため渡されません。コマンドはすでに完了している可能性があるため、プロシージャからゼロ以外のレスポンスコードを返すことができません。

2. レスポンスコードフィールドだけが渡されます。

   レスポンスコードフィールドは修正可能な4バイトのバイナリ（B4）フィールドです。Natural トリガドライバは、呼び出しが成功したかどうかを判断するために、4バイトすべてを確認します。 ただし、実際のレスポンスコードが格納されているのは、下位 2 バイトだけです。 上位 2 バイトは、理由をレポートするときのサブコードとして使用される場合があります。

   Natural トリガドライバは、同期トリガの場合はレスポンスコードを返しますが、非同期トリガの場合はレスポンスコードを無視します。 非同期トリガは開始コマンドの実行完了後に実行されることがあるため、レスポンスコードは無意味です。

   レスポンスコード値は、ユーザーに返される前に、コマンドのアディション 4 フィールドの上位2バイトに格納されます。Adabas コントロールブロックのレスポンスコードフィールドには格納されません。 この場合のレスポンスコードは、通常 155（プレコマンドトリガを表す）または 156（ポストコマンドトリガを表す）です。

3. レスポンスコードフィールドと制御情報が渡されます。

   レスポンスコードフィールドは、上記と同じ方法で渡されます。

   トリガに関する制御情報は、プロシージャを正常に実行するために不可欠である場合があります。 制御情報は、STPAPARM または STPXPARM パラメータデータエリア（PDA）で提供され、次の情報が含まれます。

   - トリガを開始した Adabas コマンドのコマンドコード
   - 発行されたコマンドが対象とするデータベースの DBID
   - ファイル番号
   - コマンドのレコードバッファの長さ
   - トリガで開始コマンドのレコードバッファを修正できるかどうかを示す設定
   - トリガが非同期、非関与、関与のいずれであるかを示す設定
   - トリガを実行しているバッチ Natural サブシステムのタスク ID

- コマンドの Adabas コントロールブロック（ACB）または拡張 Adabas コントロールブロック（ACBX）のコピー。 非同期トリガでは、最初の 48 バイトのみ、つまりアディション 2 フィールドまでのすべてです。

4. レスポンスコードフィールド、制御情報、およびグローバルユーザーエリアが渡されます。

   上記のレスポンスコードフィールドと制御情報の他に、グローバルユーザーエリアが渡されます。

   グローバルユーザーエリアは、アクティブな Natural サブシステムセッション中、保持されます。 グローバルユーザーエリアが Natural トリガドライバによって修正されることはないため、プロシージャの呼び出し間で情報を渡すために使用できます。 作業用のストレージエリアとして使用することもできます。

フォーマットバッファとレコードバッファの使用

レコードバッファ

- レコードバッファをストアドプロシージャで使用できるのは、パラメータが渡されるときだけです。

  レコードバッファは、呼び出し元からストアドプロシージャ、またはストアドプロシージャから呼び出し元にパラメータを渡すために使用できます。 レコードバッファのレイアウトまたは DSECT は、コール元と実際のストアドプロシージャの間で調整される必要があります。

- レコードバッファをトリガされたプロシージャで使用できるのは、トリガが関与または非関与（同期）の場合に限られ、レコードバッファ抽出ルーチン（STPRBE）を使用する必要があります。 レコードバッファは、トリガコマンドが処理される前でも後でも渡すことができます。

  アクセスや更新コマンドで使用するレコードバッファは、Adabas コントロールブロックのアディション 4 フィールドを使用してコール元が指定します。

パラメータ

レコードバッファを設定するときは、パラメータの定義が重要です。 さまざまな長さの複数のパラメータを次のように渡すことができます。

- 各パラメータの前に別フィールドで長さの値を指定
- パラメータの先頭または最後にフィールドのリストを添付
- 各パラメータの先頭 2 バイト（長さを示すバイトを含む）

パラメータを渡すオプションはさまざまで、柔軟性があります。環境に合わせて、最も効果的または整合性の高い方法を選択してください。

1. パラメータは渡されません。

2. 固定された定義

   パラメータは、ストレージ上の連続した領域に格納され、長さと定義は固定されています。各パラメータは、構造によって変化しないため、やはり長さは固定されています。そのため、呼び出されるプロシージャは、パラメータ用に定義された構造またはDSECTを解釈できる必要があります。 サンプル ***STPLNK01*** を参照してください。

3. パラメータの固定リスト

   複数のパラメータがありますが、STPLNKが呼び出されるときの各パラメータの番号、順序、形式、長さはそれぞれ一定です。 STPLNK では、要求の発行前に、これらのパラメータがストレージ上で連続した領域、つまりレコードバッファに格納されていることだけが必要です。サンプル ***STPLNK02*** を参照してください。

4. パラメータの可変リスト

   パラメータを渡す柔軟性の高い方法です。呼び出されるプロシージャは、レコードバッファに渡されるパラメータの形式を認識する必要があります。 このような情報（つまりプロシージャイベントフィールドDD）を渡すときにフォーマットバッファを使用すると非常に便利です。 レコードバッファ抽出ルーチン STPRBE は、フィールドのフォーマットバッファをスキャンし、同時にレコードバッファを調べて正しい値（開始位置と終了位置）を特定する必要があります。 サンプル ***STPLNK03*** を参照してください。

いずれの場合も、RBOPTパラメータオプションを使用して、レコードバッファに対する読み込みまたは書き込みアクセスが許可される必要があります。

フォーマットバッファ

フォーマットバッファは、レコードバッファで渡されるパラメータの定義を格納するときに、任意で使用できます。

- 構文は通常のコマンドのフォーマットバッファの構文と整合性がある必要があります。または、使用しない場合は "." に設定します。
- プロシージャが各パラメータの値をレコードバッファ抽出ルーチン（STPRBE）から取得できるように、フィールド名は通常はわかりやすい名前にしてください。
- プロシージャで長さが指定されない場合は、長さを使用する必要があります。または、フィールド名が ACB で指定される実際のファイル番号と対応する場合、STPRBE ルーチンでフィールドやパラメータの長さを判断できます。
- 必要に応じて、STPRBE を使用して、実際のフォーマットバッファの内容を抽出できます。
- 複数のプラットフォームでストアドプロシージャを呼び出す場合、各パラメータのフィールドタイプは、次のように指定する必要があります。
  - A は英数字
  - B はバイナリ
  - U はアンパック形式、など

レコードバッファ抽出ルーチン（STPRBE）

STPRBE は、レコードバッファ抽出ルーチンです。 レコードバッファ情報を要求するために Adabas プロシージャで使用できます。 STPRBE をコールできるのは次のとおりです。

- パラメータが渡される場合、ストアドプロシージャから。
- 同期トリガを起動するコマンドから。 STPRBE は、同期トリガのレコードバッファに対するアクセスのみ提供します。

すべての機能で、レコードバッファ抽出ルーチンは次のように呼び出されます。

```
CALL 'STPRBE' FUNCTION REQUEST-PARM REC-BUFFER
```

FUNCTION、REQUEST-PARM、および REC-BUFFER について次に説明します。

REQUEST-PARM（A154）

REQUEST-PARM を構成するフィールドは次のとおりです。

フィールド名	長さ	説明
ERR-MESSAGE	A72	受け取ったエラーのエラーメッセージテキスト
RESPONSE-CODE	I4	このルーチンからのレスポンスコード（サブコードと実際のエラー）
VERSION NUMBER	A4	この構造のバージョン
FIELD NAME	A32	抽出されるフィールドのロングネーム
FIELD FORMAT	A1	抽出されるフィールドの形式
FIELD OPTIONS	A3	特殊なオプション
FIELD LENGTH	I4	抽出されるフィールドの長さ
FIELD SHORT	A2	フィールドの Adabas ショートネーム
未使用	A2	未使用
FIELD OCCUR	I4	PE/MU フィールドのオカレンス番号
GROUP OCCUR	I4	PE フィールド内の MU のオカレンス番号
FIELD OFFSET	I4	抽出するレコードバッファに対するオフセット
未使用	A18	未使用

FUNCTION（A4）

指定しなければならない機能について次の表で説明します。

フィールド	説明
GF	フォーマットバッファの内容を取得します。
GI	インデックス値を取得します（MU または PE のインデックスオカレンスを取得）。
GM	マルチバリューを取得します（PE 内の MU フィールドを取得）。
GR	レコードバッファの範囲を取得します（呼び出しで指定された開始位置と長さに従って、レコードバッファ情報を取得）。
GV	値を取得します（レコードバッファからフラットフィールド値を取得）。
GX	UBX 情報を取得します。 NR 機能とは異なり、ユーザーは ADALNK のユーザー出口 B の定義に従って、ユーザーバッファ拡張を使用して渡されたすべての情報を取得できます。
NR	Natural 情報を取得します（渡されたオフセットと長さに従って、ユーザーバッファから Natural ユーザー情報を取得）。 ADALNK のユーザー出口 B でユーザーバッファ拡張が使用されている場合のみ有効です。
UI	インデックス値を更新します（MU または PE のインデックスオカレンスを変更）。
UM	マルチバリューを更新します（PE 内の MU フィールドを変更）。
UR	レコードバッファの範囲を更新します（呼び出しで指定された開始位置と長さに従って、レコードバッファ情報を変更）。
UV	値を更新します（レコードバッファからフラットフィールド値を変更）。

各機能で必須の値について次の表に示します。

フィールド	GF	GI	GM	GR	GV	GX	NR	UI	UM	UR	UV
ERR-MESSAGE	_/U	_/U	_/U	_/U	_/U	_/U	_/U	_/U	_/U	_/U	_/U
RESPONSE-CODE	_/U	_/U	_/U	_/U	_/U	_/U	_/U	_/U	_/U	_/U	_/U
FIELD NAME		F/A	F/A		F/A			F/A	F/A		F/A
FIELD FORMAT		f/U	f/U		f/U			f/U	f/U		f/U
FIELD OPTIONS		G			C/G			G			G
FIELD LENGTH	F	f/U	f/U		f/U	F		f/U	f/U		f/U
FIELD SHORT		f/U	f/U		f/U			f/U	f/U		f/U
FIELD OCCUR		F/A	F/A					F/A	F/A		
GROUP OCCUR		f/A	F/A					f/A	F/A		
FIELD OFFSET	F		F/A			F/A		F/A			

記号	説明
_	未使用
A	コール後も未変更のまま
F	コール前に設定
f	コール前にオプションで設定
C	フィールドカウント。カウントフィールドの値が必要なとき MU または PE フィールドでのみ有効
G	グループオプション。グループフィールド内のエレメンタリフィールドを抽出／更新
U	明示しない限り、コール後に更新（例えば長さ）

REC-BUFFER

REC-BUFFER は、可変長のフィールドで、次のいずれかが格納されます。

- ユーザーが元のコールで Adabas に渡したレコードバッファ。元のコールとは、トリガされたプロシージャを呼び出したコールです。
- トリガされたプロシージャを実際に処理した後で、ユーザーから返されたレコードバッファ。

レスポンスコード

コード	説明
0	コールは完全に成功しました。
1	内部エラーが発生しました。名前マッピングの取得に失敗しました。
2	名前マッピングでロングネームが見つかりません。ファイル - フィールドテーブル定義に名前が存在しませんでした。
3	予約済みです。
4	フォーマットバッファにフィールドが見つかりません。ユーザーは、フォーマットバッファに存在しないフィールドの値を要求しました。
5	フィールドは、レコードバッファの非常に深いところにあります。レコードバッファの長さは、Adabas コントロールブロック（ACB）のレコードバッファ長（RBL）で判断されました。 フィールドが存在する場合は、定義されている上限を超えています。
6	要求されたオカレンスがフォーマットバッファに存在しません。 MU または PE の特定のオカレンスがフォーマットバッファに見つからなかったことをユーザーに通知します。
7	機能タイプが無効です。FUNCTION に有効な値が設定されていません。
8	レコードバッファにアクセスできません。レコードバッファオプションフィールドが none に設定されているため、アクセスが許可されません。そのため、このプロシージャで利用できるレコードバッファ抽出機能はありません。

コード	説明
9	レコードバッファの修正が拒否されました。レコードバッファへのアクセスは許可されていますが、書き込みアクセスは許可されていません。 そのため、GET 機能のみを要求できます。
10	レコードバッファ長（RBL）が ACB で可能な最大値を超えています。指定された長さは、RBL の合計長を超過しています。 オプション NI にも適用されます。 拡張バッファの 232 バイトのみを取り出すことができます。
11	レコードバッファ長が不適切にゼロに設定されています。レコードバッファ抽出ルーチンでは、ユーザーがこの関数コールで RBL の長さを指定する必要がありましたが、ゼロに設定されています （例えば RAngeReq）。
12	フォーマットバッファの構文が正しくありません。コマンドで有効な構文が、レコードバッファ抽出ルーチンで解決されていない可能性があります。 通常は、n 構文（例えば AB1-n、ABn）が使用されたときに適用されます。ただし、トリガがプレコマンドトリガである場合は、フォーマットバッファ自体が無効である可能性があります。
13	無効なパラメータが指定されました。関数コールで設定されるはずのパラメータのいずれかが設定されていませんでした。 例えば、フィールド名関数コールが設定されていませんでした。
14	PE グループオカレンスが指定されていませんでした。特定の PE または MU オカレンスにアクセス／更新する関数コールが指定されましたが、パラメータでオカレンス番号が渡されませんでした。
15	無効な編集マスクがフォーマットバッファで指定されました。有効な編集マスクだけを指定できます。
16	予約済みです。
17	ユーザー情報は利用できません。リンク B がありません。つまり、ユーザーはユーザーバッファ抽出情報にアクセスしようとしましたが、利用できる情報がありません。
18	現在のファイルのグループ定義レコードが見つかりませんでした。
19	現在のファイルのグループ定義レコードで、フィールドが見つかりませんでした。
60	フォーマットバッファの構文エラーです。ニュークリアスと同様に、レコードバッファ抽出ルーチンでは、フォーマットバッファが一般的な定義規則に準拠する必要がありますが、現在のフォーマットバッファは無効です。エラーメッセージでサブコードを確認してください（『*Adabas メッセージおよびコードマニュアル*』を参照）。
3xxx	ゼロ以外の Adabas レスポンスコードが返されました。xxx は実際の Adabas レスポンスコードです。 レコードバッファ抽出ルーチンがトリガテーブルにアクセスして詳細情報を取得しようとしているときに、このような状態が発生することがあります。
9999	GDA が正しくないか、破損しているか、または存在しません。 これは内部エラーであることを示します。 Natural サブシステムからのメッセージに何らかのエラーを示しているものがないか確認するか、Software AG 技術サポートに連絡してください。 ダンプが必要になります。

オフセットによる値の取得

- レコードバッファをオフセットで移動する範囲要求。

ユーザーは、長さが元のRBL（レコードバッファ長）以下である特定のオフセットに合わせて、レコードバッファにアクセスまたは更新できます。つまり、レコードバッファに格納されている任意の"フィールド"のサイズや定義に関係なく、RBE（レコードバッファ抽出）は、特定の長さの特定の位置にあるレコードバッファの値を返さなければなりません。

- ユーザーバッファ抽出（UBX）情報をオフセットで移動する要求。

  ユーザーは、長さが元のRBL以下である特定のオフセットに合わせて、レコードバッファにアクセスできます。 UBX の場合、これは最大値の 232 です。

- フォーマットバッファをオフセットで移動する要求。

フィールド値の取得

この要求では、Adabas フィールド名が指定される必要があります。

- ショートネームが指定された場合、ロングネームは "**" に設定する必要があります。
- 長さの指定は任意です。長さを指定しない場合は、フィールドに定義された最大サイズ未満の上書き値が代わりに指定されます。

処理時に RBE は、レコードバッファとフォーマットバッファを調べる必要があります。 目的のフィールドを特定する前に、フォーマットバッファで見つかった各フィールドで、RBE はレコードバッファを調べ、実際の移動に備えて正しい位置を探します。

MU フィールドと PE フィールドを調べるには、次のように配列内の要素の合計数を計算する必要があります。

- PE オカレンス数 × MU オカレンス数 × 当該フィールドの実際の長さ

Note: ユーザーがフォーマットバッファ内のオカレンス番号を指定しなかった MU の場合、ユーザーは最初のオカレンスを受け取ります。つまり要求は、"エレメンタリ" フィールドの要求のように扱われます。

RBE は、ユーザーが MU フィールドまたは PE フィールドだけを指定したのか、それとも PE フィールド内の MU を指定したのかを判断します。 フォーマットバッファでサポートされる PE フィールドや使用方法の例を次に示します（m と n は、実際のオカレンス値）。

- MUn または PEn
- PEn(MUn)
- PEn(MUm-n)
- PEm-n(MUn)
- PEm-n(MUm-n)
- PEm-n または MUm-n

レコードバッファから MU 数または PE 数を取得するには、フィールド名かショートネームを指定する必要があります。返される値の長さの指定は任意です。

7 ストアドプロシージャの呼び出し

ストアドプロシージャを使用すると、Adabas ダイレクトコマンド PC を使用してデータベース内に存在するプロシージャを直接呼び出すことができます。

このchapterでは、次のトピックについて説明します。

ストアドプロシージャのリンクルーチン（STPLNKnn）

ストアドプロシージャのリンクルーチン STPLNKnn とともに PC コマンドを使用して、ストアドプロシージャを呼び出します。

STPLNKnn は、SYSSPT ライブラリ内にソース形式で用意されています。

ライブラリ SYSSPT にある STPLNK01、STPLNK02、および STPLNK03 は、呼び出し元のストアドプロシージャで PC コマンドを使用するサンプルです。 各サンプルは、異なる方法でパラメータをルーチンに渡します。

これらのサンプルを使用することも、独自のルーチンを作成することもできます。 サンプルを使用する場合、ルーチンのコードまたは名前を変更して、サイトの標準や要件に合わせることができます。 ルーチン名をインラインコードとしてメイン Natural プログラムに組み込むこともできます。

3 つのサンプルでは、Natural ルーチン CMADA をコールすることで、PC コマンドが呼び出されます。 このエントリ名を直接コーディングしたくない場合は、代わりにライブラリ SYSEXT 内の Natural サブプログラム USR1043N に対して、CALLNAT を発行することもできます。 CALLNAT を使用する方法の利点は、名前 "CMADA" に対する変更からコードを隔離できることです。名前の変更は、時間が経過したりプラットフォームが異なると発生する可能性があります。

PC コマンドの設定

STPLNKnn を使用してストアドプロシージャ要求を呼び出す前に、PC コマンド（ダイレクトコール）の Adabas コントロールブロック（ACB）を設定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。

- PC コマンドの機能と使用方法
- ACB インターフェイスダイレクトコールとバッファの概要
- ACBX インターフェイスダイレクトコールとバッファの概要

- バッファ

PC コマンドの機能と使用方法

PC コマンドは、ストアドプロシージャを呼び出すメカニズムを提供します。

パラメータは、レコードバッファを使用して渡されます。その後、パラメータはストアドプロシージャを使用して更新されて、呼び出し元に返されます。

フォーマットバッファは、プロシージャにパラメータを定義するために使用できます。 そのような情報は、レコードバッファ抽出ルーチンを呼び出すときに使用される可能性があります。

ACB インターフェイスダイレクトコールとバッファの概要

- コントロールブロック
- バッファエリア
- コントロールブロックフィールドの説明

コントロールブロック

フィールド	位置	フォーマット	Adabas コール前	Adabas コール後
	1～2	未使用	未使用	未使用
コマンドコード	3～4	英数字	F	U
コマンド ID	5～8	英数字	F	U
ファイル番号	9～10	英数字	F	U
レスポンスコード	11～12	バイナリ	未使用	A
	13～24	未使用	未使用	未使用
フォーマットバッファ長	25～26	バイナリ	F	U
レコードバッファ長	27～28	バイナリ	F	U
	29～36	未使用	未使用	未使用
アディション 1	37～44	英数字	F	U
アディション 2	45～48	バイナリ／バイナリ		A
アディション 3	49～56	英数字	F	A
アディション 4	57～64	英数字	未使用	A
	65～76	未使用	未使用	未使用
ユーザーエリア	77～80			U

バッファエリア

バッファ	Adabas コール前	Adabas コール後
フォーマット	F	U
レコード	F	A

上記の意味は次に示すとおりです。

F	Adabas コール前にユーザーが入力するフィールド
A	Adabas により入力されるフィールド
U	Adabas コール後も変化なし

コントロールブロックフィールドの説明

コマンドコード（**ACBCMD**）
PC

コマンド **ID**（**ACBCID**）
このフィールドに値 "STPx"（x は任意の値）を設定します。

ファイル番号（**ACBFNR**）
デフォルトでは、トリガファイルのデータベース ID とファイル番号を示します。

1 バイトのデータベース ID の場合は CB-DBID を設定します。2 バイトのデータベース ID の場合は、CB-RSP を設定し、CB-CALL-TYPE を X'30' に設定します。

フォーマットバッファと組み合わせて他のユーザーファイルのファイル番号を指定できます。ファイル番号はフォーマットバッファと一致する必要があるため、レコードバッファ抽出（STPRBE）ルーチンを使用して、ファイル - フィールド定義に応じてフィールド値を解釈したり取得することができます。

読み込むファイルの番号を2進数で指定します。物理ダイレクトコールの場合は、次のようファイル番号を指定します。

- 1 バイトファイル番号の場合は、ファイル番号を第 2 バイト（10）に入力します。第 1 バイト（9）は、バイナリの 0（B'0000 0000'）をセットします。
- 2 バイトファイル番号の場合は、2 バイト（9 と 10）を使います。

> **Note:** 2 バイトファイル番号およびデータベース ID を使用する場合は、コントロールブロックの先頭バイトに X'30' を入力しなければなりません。

レスポンスコード（**ACBRSP**）
Adabas はこのフィールドに、コマンドのレスポンスコードを返します。 レスポンスコード 0 は、このコマンドが正しく実行されたことを示します。 ゼロ以外のレスポンスコードは、

アディション 2 フィールドの下位 2 バイトにサブコードを伴う場合があります。詳細は『*Adabas メッセージおよびコードマニュアル*』を参照してください。

フォーマットバッファ長（**ACBFBL**）
フォーマットバッファ長（バイト単位）。ユーザープログラムに定義した実際のフォーマットバッファエリアは、この指定長と同じか、それ以上でなければなりません。

レコードバッファ長（**ACBRBL**）
レコードバッファ長（バイト単位）。 ユーザープログラムに定義した実際のレコードバッファエリアは、この指定長と同じか、それ以上でなければなりません。

アディション **1 -** ストアドプロシージャ名（**ACBADD1**）
ストアドプロシージャ名。

アディション**2 -** 圧縮および非圧縮レコード長 **-** （**ACBADD2**）
PC コマンドは、このフィールドのバイト 1 および 2 で実行されたプロシージャからのレスポンスを返します。

アディション **3 -** ストアドプロシージャのオプション **-** （**ACBADD3**）

このフィールドは、ストアドプロシージャのリクエストが発行されるときに使用されるオプションを示します。

バイト 1 は Type です。	A = 非同期、P = 関与、N = 非関与
バイト 2 は Parm です。	N = なし、C = コントロール、E = エラー／レスポンス、X = ACBX 付きのコントロール
バイト 3 は RecB です。	N = なし、A = アクセス、U = 更新

アディション **4**（**ACBADD4**）
PC コマンドは、このフィールドのバイト 1 および 2 で実行されたプロシージャからのレスポンスを返します。バイト 3 およびバイト 4 は、ストアドプロシージャを示すため X'0011'（17）に設定されます。

ACBX インターフェイスダイレクトコールとバッファの概要

- コントロールブロック
- バッファエリア

- コントロールブロックフィールドの説明

コントロールブロック

フィールド	位置	フォーマット	Adabas コール前	Adabas コール後
	1〜2	バイナリ	---	---
バージョンインジケータ	3〜4	バイナリ	F	
	5〜6	バイナリ	---	---
コマンドコード	7〜8	英数字	F	U
	9〜10	バイナリ	---	---
レスポンスコード	11〜12	バイナリ	---	A
コマンド ID	13〜16	英数字／バイナリ	F	U
データベース ID	17〜20	数値	F	U
ファイル番号	21〜24	数値	F	U
	25-56	---	---	---
アディション 1	57〜64	英数字	F	U
アディション 2	65〜68	バイナリ	---	A
アディション 3	69〜76	英数字	F	A
アディション 4	77〜84	英数字	---	A
	85〜114	---	---	---
エラーサブコード	115〜116	バイナリ	---	A
	117〜144	---	---	---
コマンドタイム	145〜152	バイナリ	---	A
ユーザーエリア	153〜168	該当なし	---	U
	169〜193	---	---	---

バッファエリア

バッファ	Adabas コール前	Adabas コール後
フォーマット	F	U
レコード	F	A

上記の意味は次に示すとおりです。

F	Adabas コール前にユーザーが入力するフィールド
A	Adabas により入力されるフィールド
U	Adabas コール後も変化なし

コントロールブロックフィールドの説明

バージョンインジケータ（**ACBXVER**）
F2

コマンドコード（**ACBXCMD**）
A1

レスポンスコード（**ACBXRSP**）
Adabas はこのフィールドに、コマンドのレスポンスコードを返します。 レスポンスコード 0 は、このコマンドが正しく実行されたことを示します。 ゼロ以外のレスポンスコードは、エラーサブコード（ACBXERRC）フィールドにサブコードを伴う場合があります。詳細は、*Adabas メッセージおよびコード* のドキュメントを参照してください。

コマンド **ID**（**ACBXCID**）
このフィールドに値 "STP*x*"（*x* は任意の値）を設定します。

データベース **ID**（**ACBXDBID**）
コールに使用するデータベース ID を指定します。

このフィールドがバイナリの 0 に設定されている場合、Adabas API は、DDCARD 入力データの ADARUN カードのデータベース ID を使用するか、または、リンクルーチンにリンクされている LNKGBLS モジュールかリンクルーチンによりロードされた LNKGBLS モジュールのデフォルトのデータベース ID 値を使用します。

ファイル番号（**ACBXFNR**）
デフォルトでは、トリガファイルのファイル番号を示します。

フォーマットバッファと組み合わせて他のユーザーファイルのファイル番号を指定できます。ファイル番号はフォーマットバッファと一致する必要があるため、レコードバッファ抽出（STPRBE）ルーチンを使用して、ファイル - フィールド定義に応じてフィールド値を解釈したり取得することができます。

読み込むファイルの番号を 2 進数で指定します。物理ダイレクトコールの場合は、次のようファイル番号を指定します。

- 1 バイトファイル番号の場合は、ファイル番号を第 2 バイト（10）に入力します。第 1 バイト（9）は、バイナリの 0（B'0000 0000'）をセットします。
- 2 バイトファイル番号の場合は、2 バイト（9 と 10）を使います。

アディション **1 -** ストアドプロシージャ名（**ACBXADD1**）
ストアドプロシージャ名。

アディション**2 -** 圧縮および非圧縮レコード長 **-**（**ACBXADD2**）
コマンドが正常に処理されると、次の情報がこのフィールドに返されます。

- 少なくとも 1 つの有効なフィールド値がレコードバッファにある場合、先頭 2 バイトには、アクセスしたレコードの圧縮レコード長がバイナリ形式で格納されます。
- 後ろ 2 バイトには、フォーマットバッファで選択し、アクセスしたフィールドの非圧縮長がバイナリ形式で格納されます。

Note: プリフェッチ機能の使用時、この長さ情報は返されません。

アディション **3 -** ストアドプロシージャのオプション **-**（**ACBXADD3**）

このフィールドは、ストアドプロシージャのリクエストが発行されるときに使用されるオプションを示します。

バイト 1 は Type です。	A = 非同期、P = 関与、N = 非関与
バイト 2 は Parm です。	N = なし、C = コントロール、E = エラー／レスポンス、X = ACBX 付きのコントロール
バイト 3 は RecB です。	N = なし、A = アクセス、U = 更新

アディション **4**（**ACBXADD4**）
PC コマンドは、このフィールドのバイト 1 および 2 で実行されたプロシージャからのレスポンスを返します。バイト 3 およびバイト 4 は、ストアドプロシージャを示すため X'0011'（17）に設定されます。

エラーサブコード（**ACBXERRC**）
コマンドがゼロ以外のレスポンスコードを返したときは、このフィールドにレスポンスコードの正確な意味を定義したサブコードが含まれます。レスポンスコードとサブコードについては、『*Adabas メッセージおよびコード*』を参照してください。

バッファ

PC コマンドに次のバッファが適用されます。

- フォーマットバッファ

- レコードバッファ

フォーマットバッファ

ユーザーはこのバッファの読み込むフィールドを指定します。 フォーマットバッファの構文形式、および例については、『*Adabas コマンドリファレンスマニュアル*』を参照してください。

フォーマットバッファは、レコードバッファで渡されるパラメータの定義を格納するときに、任意で使用できます。 構文は通常のコマンドのフォーマットバッファの構文と整合性がある必要があります。または、使用しない場合は "." に設定します。

フォーマットバッファで使用されるフィールド名は、通常はわかりやすい名前にして、ストアドプロシージャが各パラメータの値をレコードバッファ抽出（STPRBE）ルーチンから取得できるようにしてください（「*レコードバッファ抽出ルーチン（**STPRBE**）*」を参照）。 ストアドプロシージャルーチンが長さを提供しない場合は、長さを使用する必要があります。 または、フィールド名が ACB で指定される実際のファイル番号と対応する場合、STPRBE ルーチンでフィールドやパラメータの長さを判断できます。

異なるプラットフォーム間でストアドプロシージャを発行するときは、各パラメータのフィールドタイプも指定することが重要です。つまり英数字の場合は "A"、バイナリの場合は "B"、アンパック形式の場合は "U" などです。

詳細については、「*フォーマットバッファ*」を参照してください。

レコードバッファ

Adabasは、バッファに要求されたフィールドの値をこのバッファに返します。 すべての値は、ユーザーがフォーマットバッファに長さやフォーマットを変更して指定していなければ、標準フォーマットと標準長で返されます。

レコードバッファは、呼び出し元からストアドプロシージャ、またはストアドプロシージャから呼び出し元にパラメータを渡すために使用できます。 レコードバッファのレイアウトまたはDSECTは、コール元と実際のストアドプロシージャルーチンの間で調整される必要があります。

レコードバッファは、レコードバッファ抽出（STPRBE）ルーチンを使用することで、関与および非関与（同期）タイプの要求のみで利用できます。 「*レコードバッファ抽出ルーチン（**STPRBE**）*」を参照してください。

レコードバッファをアクセスや更新で使用するかどうかは、アディション4フィールドを使用して呼び出し元が指定します。

詳細については、「*フォーマットバッファとレコードバッファの使用*」を参照してください。

例

このセクションでは、次の表で示すサンプルのプログラムとデータエリアについて説明します。ソースコードはインストール中に提供され、SYSSPT ライブラリに配置されます。

名前	説明
STPLNK01	このストアドプロシージャのリンクルーチンは、パラメータを固定長および固定番号として渡します。
STPLNK02	このストアドプロシージャのリンクルーチンでは、最大で5つのパラメータをプロシージャに渡す可能性があります。各パラメータの長さは、パラメータの最初の2バイトに格納されます。
STPLNK03	STPLNK02 と同様に、最大で 5 つのパラメータをプロシージャに渡す可能性があります。ただし、各パラメータの長さは、先行する別のパラメータに格納されます。

STPLNK01

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Subprogram : STORPROC/STPLNK01
0040 *  Author     : Adabas Development
0050 *
0060 *  Function   : Sample Routine 01 to invoke a stored procedure
0070 *               This example expects fixed parameter definitions
0080 *  Remarks    : This routine will set up the buffers and issue the call
0090 *               to invoke a stored procedure routine directly.
0100 *               Once processing is completed, control is returned to
0110 *               the caller.
0120 *               Parameter RESP must be set to zero if processing is
0130 *               successful.
0140 *
0150 *  Parameters : The following fields in the ACB must be set up to invoke
0160 *               the stored procedure request.
0170 *
0180 *               Command Code: 'PC'
0190 *               Command ID  : 'STPx' - where x is any value
0200 *               Database ID : Database of the respective trigger file
0210 *                             Set CB-DBID for a one byte DBID
0220 *                             Set CB-RSP  for a two byte DBID with
0230 *                               CB-CALL-TYPE set to H'30'
0240 *               File Number : Set to the trigger file number of the
0250 *                             target database (normal one-byte versus
0260 *                             two-byte FNRs is applicable) by default
0270 *                             or any other file used in conjunction with
0280 *                             the format buffer.
0290 *               FB Length   : Length of the format buffer
```

```
0300 *               RB Length   : Length of the record buffer
0310 *               Additions 1 : Name of the stored procedure
0320 *               Additions 3 :
0330 *                  Byte 1   : Type ("A"sync, "P"art, "N"on-Partic)
0340 *                  Byte 2   : Parm ("N"one, "C"ntl, "E"rror/Resp)
0350 *                  Byte 3   : RecB ("N"one, "A"ccess, "U"pdate)
0360 *
0370 *
0380 *  Format Buff: The format buffer is an optional buffer that may be used
0390 *               to convey the definition of the parameters being passed
0400 *               in the record buffer. The syntax must be consistent with
0410 *               that of a format buffer for a normal command, or be set
0420 *               to "." if it not to be used.
0430 *
0440 *               The field names used in the format buffer should
0450 *               normally be meaningful so that the stored procedure can
0460 *               get the values of each parameter from the record buffer
0470 *               extraction (STPRBE) routine. Length must be used if the
0480 *               stored procedure routine does not provide one.
0490 *               Alternatively, if the field names correspond to the
0500 *               actual file number specified in the ACB, then the STPRBE
0510 *               routine will be able to determine the length of the
0520 *               field/parameter.
0530 *
0540 *               When issuing stored procedures across platforms, it is
0550 *               essential to also specify the field type of each
0560 *               parameter; i.e., "A" - alphanumeric, "B" - binary, "U"
0570 *               - unpacked etc.
0580 *
0590 *
0600 *  Record Buff: The record buffer is available for passing any
0610 *               parameters from the caller to the stored procedure
0620 *               and(or) from the stored procedure to the caller.
0630 *               The layout/DSECT of the record buffer must be
0640 *               coordinated between the caller and the actual stored
0650 *               procedure routine itself.
0660 *
0670 *               The record buffer is available for participating
0680 *               and non-participating (sync) type requests via the
0690 *               the record buffer extraction (STPRBE) routine, only.
0700 *
0710 *               Determination of the record buffer being for access or
0720 *               update is specified by the caller via the additions 3
0730 *               field (see above).
0740 *
0750 ************************************************************************
0760 DEFINE DATA PARAMETER
0770  01  REQ-TYPE  (A1)              /* Optional request ID type
0780  01  P-PROC    (A8)              /* Procedure name
0790  01  P-PARM1 (A100)              /* Single parameter
0800  01  P-MSG    (A72)              /* Message corresponding to the RESP
0810  01  RESP      (N4)              /* Response code of proc. request
```

```
0820            LOCAL USING STPLCB
0830            LOCAL
0840  01  FB    (A16) INIT<'AA,100,A.'>
0850  01  ET-CNT         (P3)
0860 END-DEFINE
0870 FORMAT PS=0
0880 *
0890 RESET CB                          /* Clear the ACB
0900 MOVE 'STP'   TO CB-CID            /* Command ID
0910 MOVE 'PC'    TO CB-CMD            /* Command code
0920 MOVE 222     TO CB-DBID           /* Database ID
0930 MOVE 12      TO CB-FNR            /* Default to TRG file number
0940 MOVE 9       TO CB-FBL            /* FB length
0950 MOVE 100     TO CB-RBL            /* RB length
0960 IF P-PROC = ' '                   /* Did we get a procedure name?
0970   DO
0980     MOVE 1 TO RESP
0990     MOVE 'Invalid Procedure Name specified' TO P-MSG
1000     ESCAPE ROUTINE
1010   DOEND
1020 MOVE P-PROC     TO CB-ADD1        /* Stored procedure name
1030 MOVE 'NCA     ' TO CB-ADD3        /* Options: N - Sync (non-partic)
1040 *                                 /*          C - Control parms
1050 *                                 /*          A - RecBuff for access
1060 *
1070 CALL 'CMADA' USING CB FB P-PARM1 /* Invoke the stored procedure
1080 *
1090 MOVE CB-RSP TO RESP
1100 MOVE 'Check Response code returned for this request' TO P-MSG
1110 * PRINT (CD=YE) 'Resp ..' (YEI) CB-RSP(EM=HH) 'Add2' CB-ADD2(EM=H(8))
1120 *              'Add4' CB-ADD4(EM=H(8))
1130 *
1140 END
```

STPLNK02

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Subprogram : STORPROC/STPLNK02
0040 *  Author     : Adabas Development
0050 *
0060 *  Function   : Sample routine 02 to invoke a stored procedure
0070 *               This example expects up to 5 different variable-length
0080 *               parameters. The length of each parameter is specified
0090 *               as the first two bytes of each parameter. Length is
0100 *               inclusive of the two-byte length itself.
0110 *  Remarks    : This routine will set up the buffers and issue the call
0120 *               to invoke a stored procedure routine directly.
0130 *               Once processing is completed, control is returned to
0140 *               the caller.
```

```
0150 *               Parameter RESP must be set to zero if processing is
0160 *               successful.
0170 *
0180 *  Parameters : The following fields in the ACB must be set up to invoke
0190 *               the stored procedure request.
0200 *
0210 *               Command Code: 'PC'
0220 *               Command ID  : 'STPx' - where x is any value
0230 *               Database ID : Database of the respective trigger file
0240 *                             Set CB-DBID for a one-byte DBID
0250 *                             Set CB-RSP  for a two-byte DBID with
0260 *                               CB-CALL-TYPE set to H'30'
0270 *               File Number : Set to the trigger file number of the
0280 *                             target database (normal one-byte versus
0290 *                             two-byte FNRs is applicable) by default
0300 *                             or any other file used in conjunction with
0310 *                             the format buffer.
0320 *               FB Length   : Length of the format buffer
0330 *               RB Length   : Length of the record buffer
0340 *               Additions 1 : Name of the stored procedure
0350 *               Additions 3 :
0360 *                  Byte 1   : Type ("A"sync, "P"art, "N"on-Partic)
0370 *                  Byte 2   : Parm ("N"one, "C"ntl, "E"rror/Resp)
0380 *                  Byte 3   : RecB ("N"one, "A"ccess, "U"pdate)
0390 *
0400 *
0410 *  Format Buff: The format buffer is an optional buffer that may be used
0420 *               to convey the definition of the parameters be ingpassed
0430 *               in the record buffer. The syntax must be consistent with
0440 *               that of a format buffer for a normal command, or be set
0450 *               to "." if it not to be used.
0460 *
0470 *               The field names used in the format buffer should
0480 *               normally be meaningful so that the stored procedure can
0490 *               get the values of each parameter via the record buffer
0500 *               extraction (STPRBE) routine. Length must be used if the
0510 *               stored procedure routine does not provide one.
0520 *               Alternatively, if the field names correspond to the
0530 *               actual file number specified in the ACB, then the STPRBE
0540 *               routine will be able to determine the length of the
0550 *               field/parameter.
0560 *
0570 *               When issuing stored procedures across platforms, it is
0580 *               essential to also specify the field type of each
0590 *               parameter; i.e., "A" - alphanumeric, "B" - binary, "U"
0600 *               - unpacked etc.
0610 *
0620 *
0630 *  Record Buff: The record buffer is available for passing any
0640 *               parameters from the caller to the stored procedure
0650 *               and(or) from the stored procedure to the caller.
0660 *               The layout/DSECT of the record buffer must be
```

```
0670 *               coordinated between the caller and the actual stored
0680 *               procedure routine itself.
0690 *
0700 *               The record buffer is available for participating
0710 *               and non-participating (sync) type requests via the
0720 *               the record buffer extraction (STPRBE) routine, only.
0730 *
0740 *               Determination of the record buffer being for access or
0750 *               update is specified by the caller via the additions 3
0760 *               field (see above).
0770 *
0780 ************************************************************************
0790 DEFINE DATA PARAMETER
0800  01  REQ-TYPE   (A1)
0810  01  P-PROC     (A8)              /* Procedure name
0820  01  P-OPTIONS  (A8)
0830  01  REDEFINE P-OPTIONS
0840    02  P-TYPE   (A1)             /* Async versus sync procedure
0850    02  P-PARMS  (A1)             /* Parm type for procedure
0860    02  P-RECB   (A1)             /* Rec buffer access
0870  01  P-PARM1(A1/1:V)             /* Variable-length parameter
0880 *                                    first 2 bytes set to incl. length
0890  01  P-PARM2(A1/1:V)             /* Variable-length parameter 2
0900  01  P-PARM3(A1/1:V)             /* Variable-length parameter 3
0910  01  P-PARM4(A1/1:V)             /* Variable-length parameter 4
0920  01  P-PARM5(A1/1:V)             /* Variable-length parameter 5
0930  01  P-MSG     (A72)             /* Message corresponging to the RESP
0940  01  RESP       (N4)             /* Response code of proc request
0950            LOCAL USING STPLCB
0960            LOCAL
0970  01  SUB        (I2)
0980  01  SUB1       (I2)
0990  01  SUB2       (I2)
1000  01  SUB3       (I2)
1010  01  SUB4       (I2)
1020  01  FB        (A48)
1030  01  REDEFINE FB
1040    02  FB-FIELD  (8)
1050      03  FB-FLD (A3)
1060      03  FB-LEN (N3)
1070  01  RB    (A1/1000)            /* Max length for all parms
1080  01  W-ADD3     (A8)
1090  01  REDEFINE W-ADD3
1100    02  W-TYPE   (A1)
1110    02  W-PARMS  (A1)
1120    02  W-RECB   (A1)
1130  01  #LENGTH    (B2)
1140  01  REDEFINE #LENGTH
1150    02  #LENG  (A1/2)
1160  01  W-LENG   (P5/5)
1170 END-DEFINE
1180 FORMAT PS=0
```

```
1190 *
1200 * In this example, we will say that each parameter has an individual
1210 * maximum length of 200; however, the limit may be established as a
1220 * total of all parameters. Since our max. record buffer is 1000 then the
1230 * maximum of all parameters cannot exceed 1000. This may be changed as
1240 * required by the user.
1250 *
1260 FOR SUB1 1 5                        /* Get all the parameter lengths
1270   DECIDE ON FIRST VALUE OF SUB1
1280     VALUE 1  MOVE P-PARM1(1:2) TO #LENG(1:2) /* Get Parm1 length
1290              IF #LENGTH < 3 /* Min length with inclusive length
1300                DO
1310                  MOVE 16 TO RESP
1320                  MOVE 'Invalid Length for Parameter 1. Must be 3-200'
1330                       TO P-MSG
1340                  ESCAPE ROUTINE
1350                DOEND
1360     VALUE 2  MOVE P-PARM2(1:2) TO #LENG(1:2) /* Get Parm2 length
1370     VALUE 3  MOVE P-PARM3(1:2) TO #LENG(1:2) /* Get Parm3 length
1380    VALUE 4  MOVE P-PARM5(1:2) TO #LENG(1:2) /* Get Parm4 length
1390    VALUE 5  MOVE P-PARM1(1:2) TO #LENG(1:2) /* Get Parm5 length
1400    ANY      IF #LENGTH = H'4040'  /* Is length Blanks?
1410               RESET #LENGTH       /*   yes, then treat as dummy parm
1420             MOVE #LENGTH TO W-LENG(SUB1)
1430             IF W-LENG(SUB1) > 202 /* For our example, we limit the length
1440               DO
1450                 MOVE 4 TO RESP
1460                 MOVE SUB1 TO FB-LEN(SUB1)
1470                 COMPRESS 'Invalid Length for Parameter' FB-LEN(SUB1)
1480                          '. Max is 200.' INTO P-MSG
1490                 ESCAPE ROUTINE    /* Terminate processing with error
1500               DOEND
1510             SUBTRACT 2 FROM W-LENG(SUB1) /* ACTUAL parm length
1520    NONE     IGNORE
1530  END-DECIDE
1540 CLOSE LOOP (1260)
1550 *
1560 IF P-PROC = ' '                    /* Did we get a procedure name?
1570   DO
1580     MOVE 1 TO RESP
1590     MOVE 'Invalid Procedure Name specified' TO P-MSG
1600     ESCAPE ROUTINE
1610   DOEND
1620 IF NOT (P-TYPE = 'A' OR= 'N' OR= 'P' OR= ' ')
1630   DO                               /* Async, participating, non-partic.
1640     MOVE 2 TO RESP
1650     MOVE 'Proc Type must be A, N, P or " "' TO P-MSG
1660     ESCAPE ROUTINE
1670   DOEND
1680 IF NOT (P-PARMS = 'C' OR= 'E' OR= 'N' OR= ' ')
1690   DO                               /* Cntrl, Error/Resp, None
1700     MOVE 3 TO RESP
```

```
1710     MOVE 'Parameter Type must be C, E, N or " "' TO P-MSG
1720     ESCAPE ROUTINE
1730   DOEND
1740 IF NOT (P-RECB = 'A' OR= 'N' OR= 'U' OR= ' ')
1750   DO                                /* Access, None, Update
1760     MOVE 3 TO RESP
1770     MOVE 'Parameter access must be Access, None or Update' TO P-MSG
1780     ESCAPE ROUTINE
1790   DOEND
1800 *
1810 * Next we merge all the passed parameters into a single contiguous
1820 * buffer which will be used as the record buffer for the call. The
1830 * format buffer will also be set up to indicate the 'structure' of the
1840 * record buffer for use by the invoked procedure.
1850 *
1860 MOVE 1 TO SUB
1870 *
1880 FOR SUB3 1 5                       /* Step through all parameters
1890   IF W-LENG(SUB3) < 3              /* Check min. length of a parameter
1900     DO
1910       MOVE '.' TO FB-FLD(SUB3)
1920       ESCAPE BOTTOM                /* None, so assume we have all parms
1930     DOEND
1940   MOVE W-LENG(SUB3) TO SUB1
1950   ADD SUB1 TO SUB2
1960   DECIDE ON FIRST VALUE OF SUB1    /* Move parms into the RB
1970     VALUE 1 MOVE 'P1,' TO FB-FLD(1)
1980             MOVE P-PARM1 (3:SUB1) TO RB(SUB:SUB2)
1990     VALUE 2 MOVE 'P2,' TO FB-FLD(2)
2000             MOVE P-PARM2 (3:SUB1) TO RB(SUB:SUB2)
2010     VALUE 3 MOVE 'P3,' TO FB-FLD(3)
2020             MOVE P-PARM3 (3:SUB1) TO RB(SUB:SUB2)
2030     VALUE 4 MOVE 'P4,' TO FB-FLD(4)
2040             MOVE P-PARM4 (3:SUB1) TO RB(SUB:SUB2)
2050     VALUE 5 MOVE 'P5,' TO FB-FLD(5)
2060             MOVE P-PARM5 (3:SUB1) TO RB(SUB:SUB2)
2070     ANY     ADD SUB1 TO SUB
2080             MOVE SUB1 TO FB-LEN(SUB3)
2090     NONE    IGNORE
2100   END-DECIDE
2110 *
2120 CLOSE LOOP (1880)
2130 *
2140 * Now we start setting up the CB and do some additional validation.
2150 * When moving in the procedure options, we allow for defaults.
2160 *
2170 RESET CB                          /* Clear the ACB
2180 MOVE 'STP'   TO CB-CID            /* Command ID
2190 MOVE 'PC'    TO CB-CMD            /* Command code
2200 MOVE 77      TO CB-DBID           /* Database ID
2210 MOVE 22      TO CB-FNR            /* File number
2220 MOVE 48      TO CB-FBL            /* FB length
```

```
2230 MOVE 1000    TO CB-RBL            /* RB length
2240 MOVE P-PROC  TO CB-ADD1           /* Stored procedure name
2250 *
2260 MOVE 'A' TO W-TYPE                 /* Set the default options
2270 MOVE 'C' TO W-PARMS
2280 MOVE 'N' TO W-RECB
2290 IF NOT (P-TYPE = ' ')             /* Should we default to Async?
2300   MOVE P-TYPE TO W-TYPE
2310 IF NOT (P-PARMS = ' ')            /* Should we default to Contrl?
2320   MOVE P-PARMS TO W-PARMS
2330 IF NOT (P-RECB = ' ')             /* Should we default to None?
2340   MOVE P-RECB TO W-RECB
2350 MOVE W-ADD3  TO CB-ADD3           /* Options for request
2360 *
2370 CALL 'CMADA' USING CB FB RB(1)   /* Invoke the stored procedure
2380 *
2390 IF CB-RSP NE 0
2400   DO
2410     PRINT (CD=YE) 'Resp ..' (YEI) CB-RSP(EM=HH) 'Add2' CB-ADD2(EM=H(4))
2420                   'Add3' CB-ADD3(EM=H(8)) 'Add4' CB-ADD4(EM=H(8))
2430     ESCAPE ROUTINE
2440   DOEND
2450 *
2460 * Now we need to restore the parameters back into the user's area,
2470 * in case the data was modified. This can happen only if the record
2480 * buffer was modifiable; i.e., P-RECB was set to 'U'.
2490 *
2500 IF  CB-RSP = 0                    /* Was everything okay
2510 AND P-RECB = 'U'                  /* Update: Parms may have been updated
2520   DO
2530     MOVE 1 TO SUB
2540     RESET SUB2
2550     FOR SUB1 1 5
2560       ADD  W-LENG(SUB1) TO SUB2
2570       MOVE W-LENG(SUB1) TO SUB3
2580       DECIDE ON FIRST VALUE OF SUB1 /* Restore parm from RB
2590         VALUE 1   ASSIGN P-PARM1 (3:SUB3) = RB(SUB:SUB2)
2600         VALUE 2   ASSIGN P-PARM2 (3:SUB3) = RB(SUB:SUB2)
2610         VALUE 3   ASSIGN P-PARM3 (3:SUB3) = RB(SUB:SUB2)
2620         VALUE 4   ASSIGN P-PARM4 (3:SUB3) = RB(SUB:SUB2)
2630         VALUE 5   ASSIGN P-PARM5 (3:SUB3) = RB(SUB:SUB2)
2640         ANY       ADD W-LENG(SUB1) TO SUB /* Get next position
2650         NONE      IGNORE
2660       END-DECIDE
2670     CLOSE LOOP(2550)
2680   DOEND
```

```
2690 *
2700 END
```

STPLNK03

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Subprogram : STORPROC/STPLNK03
0040 *  Author     : Adabas Development
0050 *
0060 *  Function   : Sample routine 03 to invoke a stored procedure
0070 *               This example expects up to five different variable-length
0080 *               parameters. Parameter lengths are passed as extra
0090 *               parameters.
0100 *  Remarks    : This routine will set up the buffers and issue the call
0110 *               to invoke a stored procedure routine directly.
0120 *               Once processing is completed, control is returned to
0130 *               the caller.
0140 *               Parameter RESP must be set to zero if processing is
0150 *               successful.
0160 *
0170 *  Parameters : The following fields in the ACB must be set up to invoke
0180 *               the stored procedure request.
0190 *
0200 *               Command Code: 'PC'
0210 *               Command ID  : 'STPx' - where x is any value
0220 *               Database ID : Database of the respective trigger file
0230 *                             Set CB-DBID for a one-byte DBID
0240 *                             Set CB-RSP  for a two-byte DBID with
0250 *                               CB-CALL-TYPE set to H'30'
0260 *               File Number : Set to the trigger file number of the
0270 *                             target database (normal one-byte versus
0280 *                             two-byte FNRs is applicable) by default
0290 *                             or any other file used in conjunction with
0300 *                             the format buffer.
0310 *               FB Length   : Length of the format buffer
0320 *               RB Length   : Length of the record buffer
0330 *               Additions 1 : Name of the stored procedure
0340 *               Additions 3 :
0350 *                  Byte 1   : Type ("A"sync, "P"art, "N"on-Partic)
0360 *                  Byte 2   : Parm ("N"one, "C"ntl, "E"rror/Resp)
0370 *                  Byte 3   : RecB ("N"one, "A"ccess, "U"pdate)
0380 *
0390 *
0400 *  Format Buff: The format buffer is an optional buffer that may be used
0410 *               to convey the definition of the parameters being passed
0420 *               in the record buffer. The syntax must be consistent with
0430 *               that of a format buffer for a normal command, or be set
0440 *               to "." if it not to be used.
0450 *
```

```
0460 *               The field names used in the format buffer should
0470 *               normally be meaningful so that the stored procedure can
0480 *               otain the values of each parameter via the record buffer
0490 *               extraction (STPRBE) routine. Length must be used if the
0500 *               stored procedure routine does not provide one.
0510 *               Alternatively, if the field names correspond to the
0520 *               actual file number specified in the ACB, then the STPRBE
0530 *               routine will be able to determine the length of the
0540 *               field/parameter.
0550 *
0560 *               When issuing stored procedures across platforms, it is
0570 *               essential to also specify the field type of each
0580 *               parameter; i.e., "A" - alphanumeric, "B" - binary, "U"
0590 *               - unpacked etc.
0600 *
0610 *
0620 *  Record Buff: The record buffer is available for passing any
0630 *               parameters from the caller to the stored procedure
0640 *               and(or) from the stored procedure to the caller.
0650 *               The layout/DSECT of the record buffer must be
0660 *               coordinated between the caller and the actual stored
0670 *               procedure routine itself.
0680 *
0690 *               The record buffer will be available for participating
0700 *               and non-participating (sync) type requests via the
0710 *               the record buffer extraction (STPRBE) routine, only.
0720 *
0730 *               Determination of the record buffer being for access or
0740 *               update is specified by the caller via the additions 3
0750 *               field (see above).
0760 *
0770 ************************************************************************
0780 DEFINE DATA PARAMETER
0790  01  REQ-TYPE   (A1)
0800  01  P-PROC     (A8)             /* Procedure name
0810  01  P-OPTIONS  (A8)
0820  01  REDEFINE P-OPTIONS
0830    02  P-TYPE   (A1)             /* Async versus sync procedure
0840    02  P-PARMS  (A1)             /* Parm type for procedure
0850    02  P-RECB   (A1)             /* Rec buffer access
0860  01  P-LEN1     (P3)             /* Length of Parm1
0870  01  P-PARM1(A1/1:V)             /* Variable-length parameter 1
0880  01  P-LEN2     (P3)             /* Length of Parm2
0890  01  P-PARM2(A1/1:V)             /* Variable-length parameter 2
0900  01  P-LEN3     (P3)             /* Length of Parm3
0910  01  P-PARM3(A1/1:V)             /* Variable-length parameter 3
0920  01  P-LEN4     (P3)             /* Length of Parm4
0930  01  P-PARM4(A1/1:V)             /* Variable-length parameter 4
0940  01  P-LEN5     (P3)             /* Length of Parm5
0950  01  P-PARM5(A1/1:V)             /* Variable-length parameter 5
0960  01  P-MSG     (A72)             /* Message corresponging to the RESP
0970  01  RESP       (N4)             /* Response code of proc request
```

```
0980            LOCAL USING STPLCB
0990            LOCAL
1000  01  SUB        (I2)
1010  01  SUB1       (I2)
1020  01  SUB2       (I2)
1030  01  SUB3       (I2)
1040  01  FB        (A64)
1050  01  REDEFINE FB
1060    02  FB-FIELD  (8)
1070      03  FB-FLD (A3)
1080      03  FB-LEN (N3)
1090      03  FB-COMM(A1)
1100  01  RB    (A1/1000)             /* Max length for all parms
1110  01  W-ADD3     (A8)
1120  01  REDEFINE W-ADD3
1130    02  W-TYPE   (A1)
1140    02  W-PARMS  (A1)
1150    02  W-RECB   (A1)
1160  01  #LENGTH    (B2)
1170  01  REDEFINE #LENGTH
1180    02  #LENG  (A1/2)
1190  01  W-LENG   (P3/5)
1200 END-DEFINE
1210 FORMAT PS=0
1220 *
1230 MOVE P-LEN1 TO W-LENG(1)
1240 MOVE P-LEN2 TO W-LENG(2)
1250 MOVE P-LEN3 TO W-LENG(3)
1260 MOVE P-LEN4 TO W-LENG(4)
1270 MOVE P-LEN5 TO W-LENG(5)
1280 *
1290 * In this example, we will say that each parameter has an individual
1300 * maximum length of 200; however, the limit may be established as a
1310 * total of all parameters. Since our max. record buffer is 1000, the
1320 * maximum of all parameters cannot exceed 1000. This may be changed as
1330 * required by the user.
1340 *
1350 FOR SUB1 1 5                      /* Validate all parameter lengths
1360  IF W-LENG(SUB1) > 16448          /* Does length contain X'4040'
1370    RESET W-LENG(SUB1)             /*   yes, then must be dummy parm
1380  IF W-LENG(SUB1) > 200            /* For our example we limit the length
1390    DO
1400      MOVE 15   TO RESP
1410      MOVE SUB1 TO FB-LEN(SUB1)
1420      COMPRESS 'Invalid Length for Parameter' FB-LEN(SUB1)
1430               '. Max is 200.' INTO P-MSG
1440      ESCAPE ROUTINE               /* Terminate processing with error
1450    DOEND
1460 CLOSE LOOP
1470 *
1480 * Now we validate the parameters, as required. Of course, these may
1490 * be changed as per the user's requirement and may vary from one stored
```

```
1500 * procedure link routine to another.
1510 *
1520 IF P-PROC = ' '                    /* Did we get a procedure name?
1530  DO
1540    MOVE 1 TO RESP
1550    MOVE 'Invalid Procedure Name specified' TO P-MSG
1560    ESCAPE ROUTINE
1570  DOEND
1580 IF NOT (P-TYPE = 'A' OR= 'N' OR= 'P' OR= ' ')
1590  DO                               /* Async, Participating, Non-Partic
1600    MOVE 2 TO RESP
1610    MOVE 'Proc Type must be A, N, P or " "' TO P-MSG
1620    ESCAPE ROUTINE
1630  DOEND
1640 IF NOT (P-PARMS = 'C' OR= 'E' OR= 'N' OR= ' ')
1650  DO                               /* Cntrl, Error/Resp, None
1660    MOVE 3 TO RESP
1670    MOVE 'Parameter Type must be C, E, N or " "' TO P-MSG
1680    ESCAPE ROUTINE
1690  DOEND
1700 IF NOT (P-RECB = 'A' OR= 'N' OR= 'U' OR= ' ')
1710  DO                               /* Access, None, Update
1720    MOVE 3 TO RESP
1730    MOVE 'Parameter access must be Access, None or Update' TO P-MSG
1740    ESCAPE ROUTINE
1750  DOEND
1760 IF P-LEN1 < 3                     /* Min. length with inclusive length
1770  DO                               /* Anything less indicates no parm
1780    MOVE 4 TO RESP
1790    MOVE 'First Parameter MUST be valid. Length must be
3-200' TO P-MSG
1800    ESCAPE ROUTINE
1810  DOEND
1820 *
1830 * Next we merge all the passed parameters into a single contiguous
1840 * buffer which will be used as the record buffer for the call. The
1850 * format buffer will also be set up to indicate the 'structure' of the
1860 * record buffer for use by the invoked procedure.
1870 *
1880 MOVE 1 TO SUB
1890 RESET SUB2
1900 *
1910 FOR SUB1 1 5                       /* Step through all parameters
1920  IF W-LENG(SUB1) < 3               /* Check min. length of a parameter
1930    DO
1940      MOVE '.' TO FB-FLD(SUB1)
1950      ESCAPE BOTTOM                 /* None, so assume we have all parms
1960    DOEND
1970  ADD W-LENG(SUB1)  TO SUB2         /* Get end position
1980  MOVE W-LENG(SUB1) TO SUB3         /* Set index for MOVE statement
1990  DECIDE ON FIRST VALUE OF SUB1     /* Move next parm into the RB
2000     VALUE 1   MOVE P-PARM1 (1:SUB3) TO RB(SUB:SUB2)
```

```
2010               MOVE 'P1,' TO FB-FLD(1)
2020     VALUE 2   MOVE P-PARM2 (1:SUB3) TO RB(SUB:SUB2)
2030               MOVE ','   TO FB-COMM(SUB1 - 1)
2040               MOVE 'P2,' TO FB-FLD(2)
2050     VALUE 3   MOVE P-PARM3 (1:SUB3) TO RB(SUB:SUB2)
2060               MOVE ','   TO FB-COMM(SUB1 - 1)
2070               MOVE 'P3,' TO FB-FLD(3)
2080     VALUE 4   MOVE P-PARM4 (1:SUB3) TO RB(SUB:SUB2)
2090               MOVE ','   TO FB-COMM(SUB1 - 1)
2100               MOVE 'P4,' TO FB-FLD(4)
2110     VALUE 5   MOVE P-PARM5 (1:SUB3) TO RB(SUB:SUB2)
2120               MOVE ','   TO FB-COMM(SUB1 - 1)
2130               MOVE 'P5,' TO FB-FLD(5)
2140               MOVE '.'   TO FB-COMM(5)
2150     ANY       ADD W-LENG(SUB1) TO SUB  /* Get new position
2160               MOVE W-LENG(SUB1) TO FB-LEN(SUB1)
2170     NONE      IGNORE
2180  END-DECIDE
2190 *
2200 CLOSE LOOP
2210 *
2220 * Now we set up the CB for the actual stored procedure call.
2230 *
2240 RESET CB                          /* Clear the ACB
2250 MOVE 'STP'   TO CB-CID            /* Command ID
2260 MOVE 'PC'    TO CB-CMD            /* Command Code
2270 MOVE 77      TO CB-DBID           /* Database ID
2280 MOVE 22      TO CB-FNR            /* File Number
2290 MOVE 64      TO CB-FBL            /* FB Length
2300 MOVE 1000    TO CB-RBL            /* RB length
2310 MOVE P-PROC  TO CB-ADD1           /* Stored procedure name
2320 *
2330 * If any options were not passed, we use a pre-specified default.
2340 *
2350 MOVE 'A' TO W-TYPE                 /* Set the default options
2360 MOVE 'C' TO W-PARMS
2370 MOVE 'N' TO W-RECB
2380 IF NOT (P-TYPE = ' ')             /* Should we default to Async?
2390  MOVE P-TYPE TO W-TYPE
2400 IF NOT (P-PARMS = ' ')            /* Should we default to Contrl?
2410  MOVE P-PARMS TO W-PARMS
2420 IF NOT (P-RECB = ' ')             /* Should we default to None?
2430  MOVE P-RECB TO W-RECB
2440 MOVE W-ADD3  TO CB-ADD3           /* Options for request 2450 *
2460 CALL 'CMADA' USING CB FB RB(1)   /* Invoke the stored procedure
2470 *
2480 IF CB-RSP NE 0
2490  DO
2500    PRINT (CD=YE) 'Resp ..' (YEI) CB-RSP(EM=HH) 'Add2' CB-ADD2(EM=H(4))
2510                  'Add3' CB-ADD3(EM=H(8)) 'Add4' CB-ADD4(EM=H(8))
2520    ESCAPE ROUTINE
2530  DOEND
```

```
2540 *
2550 * Now we need to restore the parameters back into the user's area,
2560 * in case the data was modified. This can happen only if the record
2570 * buffer was modifiable; i.e., P-RECB was set to 'U'.
2580 *
2590 IF  CB-RSP = 0                    /* Was everything okay
2600 AND P-RECB = 'U'                 /* Update: Parms may have been updated
2610  DO
2620    MOVE 1 TO SUB
2630    RESET SUB2
2640    FOR SUB1 1 5
2650      ADD  W-LENG(SUB1) TO SUB2
2660      MOVE W-LENG(SUB1) TO SUB3
2670      DECIDE ON FIRST VALUE OF SUB1 /* Restore parm from RB
2680        VALUE 1   ASSIGN P-PARM1 (1:SUB3) = RB(SUB:SUB2)
2690        VALUE 2   ASSIGN P-PARM2 (1:SUB3) = RB(SUB:SUB2)
2700        VALUE 3   ASSIGN P-PARM3 (1:SUB3) = RB(SUB:SUB2)
2710        VALUE 4   ASSIGN P-PARM4 (1:SUB3) = RB(SUB:SUB2)
2720        VALUE 5   ASSIGN P-PARM5 (1:SUB3) = RB(SUB:SUB2)
2730        ANY       ADD W-LENG(SUB1) TO SUB /* Get next position
2740        NONE      IGNORE
2750      END-DECIDE
2760    CLOSE LOOP(2640)
2770  DOEND
2780 *
2790 END
```

8 トリガメンテナンス

Note: トリガメンテナンスサブシステムは、Adabas Online System（AOS）アドオン製品がインストールされている場合のみ利用できます。 AOS デモバージョンでは利用できません。

トリガメンテナンス機能は、次の操作が可能な Natural アプリケーションです。

- トリガ定義の追加、削除、修正、およびリスト表示
- ファイル - フィールドテーブルの生成、修正、表示、および削除
- 実行時トリガシステム定義のプロファイル情報の表示および修正
- トリガ操作の監視と、トリガの永続的または一時的なアクティブ化または非アクティブ化

手順 8.1. トリガメンテナンスを開始する手順

- Adabas Online System（AOS）メインメニューから［Trigger Maintenance］を選択するか、ライブラリ SYSTRG の Natural NEXT プロンプトで「MENU」と入力します。

  トリガメンテナンスのメインメニューが表示されます。

```
HH:MM:SS              ***** TRIGGER MAINTENANCE *****       YYYY-MM-DD
User USR01                       - Main Menu -              DBnr 105

                     Code  Function
                     ----  ---------------------------------
                       A   Administrator Functions
                       F   File-Field Table Definitions
                       R   Procedure Reports
                       T   Create/Modify Trigger Definitions
                       ?   Help
                       .   Exit
                     ----  ---------------------------------

                Code ...


Command ==>
Enter--PF1--PF2--PF3--PF4---PF5---PF6---PF7--PF8--PF9--PF10--PF11--PF12
```

```
        Help      Exit Field Trigr Admin Procs          FTRG  FDIC  Canc
```

次の表で、メインメニューの機能について説明します。

コード	機能	説明
A	管理者機能	プロファイル情報（トリガとストアドプロシージャの実行時パラメータ設定）を表示または修正します。 Natural サブシステム情報を取得します。 サブシステムアクティビティとトリガアクティビティを監視します。
F	ファイル - フィールドテーブル定義	ファイル - フィールドテーブルを表示、生成、修正、削除、または選択します。 ファイル - フィールドテーブルのソースを表示します。
R	プロシージャレポート	定義済みトリガのプロシージャのリストをファイル順または名前順に表示します。
T	トリガ定義の作成／修正	トリガ定義を追加、表示、修正、削除、または選択します。
?	ヘルプ	トリガメンテナンス機能のヘルプ情報を表示します。
.	出口	Natural NEXT 画面に戻ります。 メインメニューに戻るには、「TRIGGERS」と入力します。

Note: トリガメンテナンス機能では、データベース ID とファイル番号が必要です。 これらの値が NATPARM LFILE パラメータまたは NTFILE マクロを使用してデフォルトで指定されていない場合は、メインメニューを使用する前に、入力する必要があります。 メインメニューで PF10 を押すと、これらの値を入力することができます。そうでない場合は、入力を要求するメッセージが自動的に表示されます。

このchapterでは、次のトピックについて説明します。

概要

トリガメンテナンス機能は、データ画面やポップアップウィンドウを表示する一連のメニューとサブメニューで構成されます。 それぞれのメニューには、機能と機能コードのリスト、入力フィールドのグループ、コマンド行、PF キーのセット、およびメッセージエリアがあります。 Adabas Online System（AOS）は、同じ環境にインストールする必要があります。

ワイルドカード表記

ワイルドカード表記は、可能な限り許可されます。 例えば、ほとんどのメニューで、ファイル名を指定する必要があります。 ファイル名がわからない場合は、代わりにワイルドカード文字を入力できます。 値の範囲がポップアップウィンドウに表示され、その範囲から項目またはエントリを選択できます。

"ワイルドカード" という語句は、次のように特殊文字を使用することを意味します。

文字	例	次の条件に当てはまる名前を選択
*	PERS*	PERS で始まる名前
>	PERS>	PERS よりも大きい値が含まれる
<	PERS<	PERS よりも小さい値が含まれる

入力フィールド

トリガメンテナンス機能の各メニューには、次の入力フィールドが含まれます。

フィールド	入力	
コード	機能を表すコードを入力します。 例えば、メインメニューのコードは次のとおりです。	
	A	管理者機能
	F	ファイル - フィールドテーブル定義
	R	プロシージャレポート
	T	トリガ定義の作成／修正
	機能に応じて、サブメニュー、ポップアップ、またはデータ画面が表示されます。	
ファイル名	ファイル名を入力するか、ワイルドカードを入力してファイル名と番号の選択リストを表示します（例えば、「a*」と入力すると、名前がaで始まるすべてのファイルのリストを返します）。	
ファイル番号	ファイル番号を入力します。 FDTからファイル-フィールドテーブルを生成するときは、実際のファイル番号の代わりに「99999」と入力すると、有効なファイルのリストを取得できます。	

その他の入力フィールドは変化します。 例としては、次のようなものがあります。

- ［File-Field Table Definitions］メニューには、［Generation Type］入力フィールドがあります。 ［Generation Type］は、ファイル - フィールドテーブルを生成するソースを指定します（Natural DDM、Adabas FDT、または Predict Adabas ファイル定義）。
- ［Trigger Definitions］メニューには、［Field Name］と［Command Type］入力フィールドがあります。 ［File Name］、［Command Type］、および［Field Name］は、トリガ定義の作成時に指定する選択条件を構成します。

メッセージ

画面の上部または下部にあるメッセージエリアには、1行のエラーメッセージまたはアクションメッセージが表示されます。 各メッセージには、メッセージ番号と短い説明が含まれます。 詳細については、『*Adabas メッセージおよびコードマニュアル*』を参照してください。

コマンド

次の表で、コマンド行に入力できるコマンドについて説明します。 コマンドによっては、すべての画面に適用されない場合があります。

コマンドは大文字と小文字を区別しないため、大文字、小文字、または大文字と小文字を混在させて入力することができます。 処理される前に、コマンドは大文字に変換されます。

コマンドで下線の付いた部分は、使用できる最短の省略形を示します。

コマンド	表示する内容
ACTIVITY または DISPLAY ACTIVITY	［Current Trigger Activity］画面。現在実行中のトリガに関する情報が表示されます。
DISPLAY PROFILE	［Display Profile Information］画面。トリガメンテナンス機能のプロファイルが表示されます。
DISPLAY TASKS または DISPLAY SYSTEMS	［Subsystem Activity］画面。現在実行中の Natural サブシステムに関する情報が表示されます。
MENU	メインメニュー。
EXIT、QUIT、または .（ピリオド）	前の画面。
FIELD	［File-Field Table Definitions］メニュー。
TRIGGER	［Trigger Definitions］メニュー。
ADMIN	［Administrator］メニュー。
PROFILE	［Profile Information］メニュー。
SET FILE	現在のトリガファイルのデータベースとファイルの割り当て。別のデータベースとファイルを入力できますが、有効なトリガファイルである必要があります。
SET FDIC	現在の Predict ファイルのデータベースとファイルの割り当て。 別の Predict システムファイルを入力できます。
FORWARD または +	次のページ。
TOP、または --	上部（先頭）に戻ります。例えば、ファイル - フィールドテーブルリストの最初の部分。
BACK、または -	前のページ。

PF キー

標準の PF キーについて次の表で説明します。

キー	ラベル	説明	メニューまたは画面
PF1	ヘルプ	現在の機能に関する情報を表示します。	全画面
PF2	Menu	メインメニューに戻ります。	全画面
PF3	出口	前の画面に戻ります。ほとんどの場合、更新オプションが選択されていなければ、すべての更新が無視されます。	全画面
PF4	Field	［File-Field Table Definitions］メニューを表示します。	メイン、［Profile Information］、［Procedure Reporting］、［Trigger Definitions］の各メニュー
PF5	Trigr	［Trigger Definitions］メニューを表示します。	メイン、［File-Field Table Definitions］、［Profile Information］、［Procedure Reporting］の各メニュー、［Define Trigger Info］、［Add Function］の各ポップアップ
PF6	Admin	［Administrator］メニューを表示します。	メイン、［File-Field Table Definitions］、［Trigger Definitions］の各メニュー
PF7	Procs	［Procedure Reporting］メニューを表示します。	メインメニュー
PF10	FTRG	［File Assignments for Triggers］ポップアップを表示します。現在使用しているデータベースとファイルとは別のデータベースとファイルを入力できます。	メインメニューと［File-Field Table Definitions］メニュー
PF11	FDIC	Predict ファイルの設定の値を変更します（トリガの場合のみ）。	メインメニューと［File-Field Table Definitions］メニュー
PF12	Canc	現在のプロセスをキャンセルし、前の画面に戻ります。更新オプションが選択されていなければ、すべての更新が無視されます。	全画面

ファイル - フィールドテーブル

トリガでは、ファイル - フィールドテーブルへのアクセスが必要です。ファイル - フィールドテーブルは、ロングファイル名からファイル番号へ、およびフィールド名から 2 文字の Adabas フィールド ID へのマッピングを行います。

ファイル - フィールドテーブルには、次のエントリが含まれる必要があります。

- トリガ定義で使用されるすべてのフィールド

- トリガされたプロシージャのアクセスまたは更新コマンドで参照されるすべてのフィールド
- レコードバッファ抽出ルーチン（STPRBE）でクエリされる必要のあるすべてのフィールド

ファイル - フィールドテーブルは、Adabas FDT、Natural DDM、または Predict Adabas ファイルから生成できますが、次の制限があります。

- スーパーディスクリプタ、サブディスクリプタ、ハイパーディスクリプタ、およびフォネティックディスクリプタは、ヒストグラム（L9）のみでサポートされます。

ファイル-フィールドテーブルの定義後は、外部フィールドの名前と内部フィールドの名前および番号を関連付けることができ、トリガをファイル-フィールドテーブルの任意のフィールドに対して定義できるようになります。 例えば、EMPLOYEES ファイルと SALARY フィールドを参照する場合、Adabas ではこれらをファイル 1 とフィールド AS として識別できるようになります。

レコードバッファ抽出

レコードバッファ抽出ルーチン（STPRBE）でレコードバッファ内のフィールドの値を抽出する場合、適切な位置と長さで値を抽出する必要があります。正しい位置を特定するために、STPRBE では不要な各フィールドを通過する必要があり、そのため各フィールドの長さを認識している必要があります。

- フォーマットバッファでフィールドの長さを明示的に指定しない場合は、ファイル-フィールドテーブル定義から取得する必要があります。
- Naturalのように、フィールドの長さが常にフォーマットバッファで指定されている場合は、ファイル - フィールドテーブルにフィールドを含める必要はありません。

フィールドの長さがフォーマットバッファで指定されていない場合、定義がトリガファイル内に存在しなければ、エラーが発生します。 プロシージャは処理を続行できずに終了します。

グループフィールドテーブル

フォーマットバッファでグループ名が指定されている場合、STPRBE ではグループフィールドテーブルを使用して、そのグループに属するエレメンタリフィールドを特定します。 グループフィールドテーブル内の各エントリには、エレメンタリフィールドとグループにおけるオフセットに関する情報が含まれます。 オフセットは、最大で 7 グループレベルまで維持されます。 特定のファイルのグループフィールドテーブルで、最大で 50 のエレメンタリフィールドにエントリを設定できます。

グループフィールドテーブルのエントリは、ファイル-フィールドテーブルの生成機能、またはファイル - フィールドテーブルの修正機能から生成できます。

［File-Field Table Definitions］メニュー

ファイル - フィールドテーブルは、トリガメンテナンス機能のファイル - フィールドテーブル定義機能を使用して生成および保守できます。任意のフィールドの組み合わせをファイル - フィールドテーブルに追加したり順次削除したりすることができます。

手順 8.2. ファイル - フィールドテーブル定義機能にアクセスする手順

■ メインメニューで「F」と入力します。

表示される画面は、次の例のようになります。

```
HH:MM:SS            ***** TRIGGER MAINTENANCE *****        YYYY-MM-DD
 User USR01       - File-Field Table Definitions Menu -     DBnr 105

                 Code  Function
                 ----  -----------------------------------
                   D   Display File-Field Table
                   G   Generate File-Field Table
                   M   Modify File-Field Table
                   P   Delete File-Field Table
                   R   Rename File-Field Table
                   S   Select File-Field Table
                   V   View the Origin for File-Field Table
                   ?   Help
                   .   Exit
                 ----  -----------------------------------

     Code ......... _
     File Name .... ____________________
     File Number .. _____
     Gen. Type .... __

 Command ==>
 Enter-PF1--PF2--PF3--PF4--PF5---PF6---PF7---PF8--PF9--PF10--PF11--PF12
       Help Menu Exit      Trigr Admin Procs           FTRG  FDIC  Canc
```

次の表で［File-Field Table Definitions］メニューのそれぞれの機能を説明します。

機能	説明
Display File-Field Table	指定されたファイルのテーブルを表示します。
Generate File-Field Table	フィールドを追加または削除して、新しいテーブルを生成します。生成タイプコードが必要です。
Modify File-Field Table	1 つ以上のフィールドを削除して、テーブルを修正します。トリガ定義で使用しているフィールドは削除できません。
Delete File-Field Table	テーブル全体を削除します。トリガ定義で使用しているフィールドが 1 つ以上含まれているテーブルは削除できません。

機能	説明
Rename File-Field Table	テーブルの名前を変更します。指定されたテーブルのすべてのフィールドとトリガの名前が変更されます。
Select a File-Field Table	選択リストからテーブルを選択します。
View the Origin for File-Field Table	テーブルの生成元となったFDT、DDM、またはPredictファイルを表示します。 選択したフィールドをテーブルに追加できます。

ファイル - フィールドテーブルの表示

手順 8.3. ファイル - フィールドテーブルを表示する手順

■ ［File-Field Table Definitions］メニューで「D」と入力します。

```
HH:MM:SS                ***** TRIGGER MAINTENANCE *****       YYYY-MM-DD
  User USR01                - Display File-Field Table -         DBnr 105

  File Name ... AUTOMOBILES                                      Fnr ... 4

 Field Name  Long Field Name                    Status  Message
 ----------  --------------------------------  ------  ------------------
     AC      BODY-TYPE                         Triggr
     CA      COLOR                             Triggr
     AA      MAKE                              Active
     FB      MILEAGE                           Triggr
     AB      MODEL                             Active
     BD      WEIGHT                            Active
     DA      YEAR                              Active



                 Enter 'F'(Fwd), 'T'(Top), 'B'(Bck), '.'(Exit)
 Command==>
 Enter-PF1---PF2---PF3---PF4--PF5---PF6--PF7--PF8--PF9---PF10---PF11--PF12
       Help  Menu  Exit       Next  --   -    +    Grp   GFld         Canc
```

- この画面の情報を確認するには、Enter キーを押してください。
- グループテーブルを表示するには、PF9 キーを押すか、スクロールし続けます。ファイルのグループテーブルが生成された場合は、ファイル-フィールドテーブルの最後に表示されます。
- グループフィールドテーブルを表示するには、PF10 キーを押します。

グループテーブルの表示

手順 8.4. グループテーブルを表示する手順

■ PF9（Grp）キーを押すか、［Display File-Field Table］画面の最後までスクロールします。

```
HH:MM:SS                 ***** TRIGGER MAINTENANCE *****            YYYY-MM-DD
 User USR01                    - Display Group Table -                DBnr 105

 File Name ... AUTOMOBILES                                            Fnr ... 4

 Name Level Type Length      Name Level Type Length      Name Level Type Length
 ---- ----- ---- ------      ---- ----- ---- ------      ---- ----- ---- ------
  AB    1    G       60
  A1    1    G       53
  A2    1    G       21
  AQ    1    P       13
  A3    1    G        4
  AW    1    P       12



                Enter 'F'(Fwd), 'T'(Top), 'B'(Bck), '.'(Exit)
 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Menu  Exit              --    -     +           GFld  FFT   Canc
```

■ この画面の情報を確認するには、Enter キーを押してください。

■ グループフィールドテーブルを表示するには、PF10 キーを押すか、スクロールし続けます。 グループフィールドテーブルがグループテーブルの最後に表示されます。

■ ［Display File-Field Table］画面に戻るには、PF11 キーを押してください。

ファイルの各グループフィールドがリストされます。 ［Type］列の［G］は、単一グループフィールドを表します。［P］は、ピリオディック（PE）グループを表します。 グループフィールドテーブルエントリは、フィールドがグループまたは PE グループの一部である場合にのみ作成できます。

［Length］列には次の情報が表示されます。

■ 単一グループの場合は、グループに含まれるすべてのフィールドの合計長

■ PE グループの場合は、各オカレンスの合計長

グループフィールドテーブルの表示

手順 8.5. グループフィールドテーブルを表示する手順

■ ［Display File-Field Table］画面で PF10（GFld）キーを押します。

Or:

［Display Group Table］画面の最後までスクロールします。

```
HH:MM:SS                ***** TRIGGER MAINTENANCE *****          YYY-MM-DD
 User USR01               - Display Group-Field Table -           DBnr 105

 File Name ... AUTOMOBILES                                        Fnr ... 4

 Field              Group Offsets                          Message
----------------   ------------------------------------ --------------------
 2,AS,5,P           AQ(3)
 2,AT,5,P           AQ(8)
 2,AU,2,U           A3(0)
 2,AV,2,U           A3(2)
 2,AX,6,U           AW(0)
 2,AY,6,U           AW(6)



           Enter 'F'(Fwd), 'T' (Top), 'B' (Bck), or '.'(Exit)

 Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
```

- この画面の情報を確認するには、Enter キーを押してください。
- グループテーブルを表示するには、PF9 キーを押してください。
- ［Display File-Field Table］画面に戻るには、PF11 キーを押してください。

フィールドは ADACMP FNDEF フォーマットで表示されます。表示される内容は、レベル、名前、長さ、フォーマット［オプション］です。 詳細は、『*Adabas* ユーティリティマニュアル』を参照してください。

［Group Offsets］列は、フィールドが参加するグループおよびそのグループ内のフィールドのオフセットを示します。 フィールドが複数のグループのメンバである場合は、追加グループ（およびグループ内のフィールドのオフセット）もリストされます。

例えば、AS フィールドは AQ グループに属しており、オフセット 3 にあります。 また、AT フィールドも AQ グループに属していますが、オフセット 8 にあります。

ファイル - フィールドテーブルの修正

手順 8.6. ファイル - フィールドテーブルを修正する手順

■ ［File-Field Table Definitions］メニューで「M」と入力します。

```
HH:MM:SS                ***** TRIGGER MAINTENANCE *****       YYYY-MM-DD
User USR01                - Modify File-Field Table -         DBnr 105

 File Name ... AUTOMOBILES                                     Fnr ...4

 Sel  Field Name  Long Field Name                    Status  Message
----  ----------  --------------------------------  ------  --------------
 __       AC      BODY-TYPE                         Triggr
 __       CA      COLOR                             Triggr
 __       AA      MAKE                              Active
 __       FB      MILEAGE                           Triggr
 __       AB      MODEL                             Active
 __       BD      WEIGHT                            Active
 __       DA      YEAR                              Active




Mark fields with 'A' Add, 'D' Delete, 'G' Generate, 'I' Info, or '.' Exit

Command==>
Enter-PF1---PF2---PF3---PF4--PF5---PF6--PF7--PF8--PF9---PF10---PF11--PF12
      Help  Menu  Exit       Next  --    -    +    Grp   GFld         Canc
```

■ この画面の情報を確認するには、Enter キーを押してください。

■ グループテーブルを表示するには、PF9 キーを押すか、最後までスクロールします。 グループテーブルを修正することはできません。

■ グループフィールドテーブルを修正するには、PF10 キーを押します。

［Modify File-Field Table］画面で、［Status］が

■ ［Triggr］であるフィールドは、トリガ定義で使用されています。

■ ［Active］であるフィールドは、ファイル - フィールドテーブルに含まれていますが、トリガ定義では使用されていません。

フィールドの修正

［Modify File-Field Table］画面では、グループフィールドテーブルのエントリを生成したり、フィールド属性を表示したり、ファイル-フィールドテーブルからフィールドを削除したりすることができます。

手順 8.7. グループフィールドテーブルのエントリを生成する手順

- フィールド名の隣の［Sel］列に「G」を入力して、エントリを生成する各フィールドにマークを付けます。 その後、Enter キーを押します。

  グループフィールドテーブルエントリは、フィールドがグループまたは PE グループの一部である場合にのみ作成できます。

手順 8.8. 各フィールドの属性を表示する手順

- フィールド名の隣の［Sel］列に「I」を入力して、フィールドにマークを付けます。 その後、Enter キーを押します。

  フィールドのグループフィールドエントリが生成されている場合は、結果のウィンドウに、フィールド名、フォーマット、長さ、タイプ（スーパーディスクリプタの場合は SP、サブディスクリプタの場合は SB）、およびインジケータが表示されます。

手順 8.9. 1 つ以上のフィールドを削除する手順

- フィールド名の隣の［Sel］列に「D」を入力して、削除する各フィールドにマークを付けます。 その後、Enter キーを押します。

  次の規則が適用されます。

  - Enter キーを押す*前*に、フィールド名の隣の［Sel］列に「A」と入力すると、フィールドの削除をキャンセルできます。
  - ［Active］のフィールドだけを削除できます。
  - ［Triggr］のフィールドは、トリガ定義を先に削除しなければ削除できません。

グループフィールドテーブルの修正

手順 8.10. グループフィールドテーブルを修正する手順

■ ［Modify File-Field Table］画面で PF10（GFld）キーを押します。

Or:

［Display Group Table］画面の最後までスクロールします。

```
HH:MM:SS                 ***** TRIGGER MAINTENANCE *****           YYY-MM-DD
User USR01                - Modify Group-Field Table -             DBnr 105

 File Name ... AUTOMOBILES                                         Fnr ... 4

 Sel Field              Group Offsets                          Message
 --- ----------------   ------------------------------------ ---------------
  _   2,AS,5,P           AQ(3)
  _   2,AT,5,P           AQ(8)
  _   2,AU,2,U           A3(0)
  _   2,AV,2,U           A3(2)
  _   2,AX,6,U           AW(0)
  _   2,AY,6,U           AW(6)




           Mark Fields with 'D'(Delete) or '.'(Exit)

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Menu  Exit              --    -     +     Grp         FFT   Canc
```

■ この画面の情報を確認するには、Enter キーを押してください。

■ グループテーブルを表示するには、PF9 キーを押してください。 グループテーブルを修正することはできません。

■ ［Modify File-Field Table］画面に戻るには、PF11 キーを押してください。

フィールドは ADACMP FNDEF フォーマットで表示されます。表示される内容は、レベル、名前、長さ、フォーマット［オプション］です。 詳細は、『*Adabas* ユーティリティマニュアル』を参照してください。

［Group Offsets］列は、フィールドが参加するグループおよびそのグループ内のフィールドのオフセットを示します。 フィールドが複数のグループのメンバである場合は、追加グループ（およびグループ内のフィールドのオフセット）もリストされます。

例えば、AS フィールドは AQ グループに属しており、オフセット 3 にあります。 また、AT フィールドも AQ グループに属していますが、オフセット 8 にあります。

手順 8.11. 1 つ以上のエントリを削除する手順

- フィールド名の隣の［Sel］列に「D」を入力して、削除する各エントリにマークを付けます。 その後、Enter キーを押します。

ファイル - フィールドテーブルの削除

ファイル - フィールドテーブルの*全体*を削除する場合に限り、［File-Field Table Definitions］メニューの削除機能を使用します。 ファイル - フィールドテーブルを削除すると、そのファイルに関連付けられたグループフィールドテーブルもすべて削除されます。

手順 8.12. ファイル - フィールドテーブル全体を削除する手順

- ［File-Field Table Definitions］メニューで［Delete File-Field Table］を表す「P」を入力します。

  TRG0109 エラーが発生した場合、このエラーは、すべてのトリガが削除されるまでテーブルを削除できないことを表します。 ファイル - フィールドテーブルを表示して、テーブルのいずれかのフィールドに［Triggr］（トリガ定義で使用されている）とマークが付いているかどうか確認してください。 マークが付いていない場合、トリガ定義を確認する必要があります。 "すべてのフィールド" オプション（フィールド名 **）のトリガがこのファイルに定義されている可能性があります。

ファイル - フィールドテーブルの生成

手順 8.13. ファイル - フィールドテーブルを生成する手順

- ［File-Field Table Definitions］メニューで「G」と入力し、ファイル名と次に示す生成タイプコードを入力します。

コード	ソース
F	Adabas FDT
D	Natural DDM
P	Predict ファイル

  - FDT 生成は、常にトリガファイルの現在の設定に対するデータベースの DBID から行われます。
  - DDM と Predict の生成は、常に Predict ファイルのデータベースとファイル番号から行われます。 これはFDICの現在の設定から取得できます。または PF10 キーを使用したり、トリガメンテナンス機能のコマンド行からコマンド SET FDIC を発行したりして上書きすることもできます。

Note: FDT ファイル名には、ワイルドカード表記を使用できません。 ただし、生成機能に限って、ファイル番号に「99999」と入力すると、データベースにロードされているすべてのファイルのリストを表示できます。

表示される画面には、ファイル-フィールドテーブルを生成するためのソース（FDT、DDM、または Predict ファイル）が含まれます。 ファイル - フィールドテーブルに追加したり削除したりするフィールドを選択できます。 「*FDT のサンプル*」を参照してください。

ファイル-フィールドテーブルが生成されると、フィールド名（ロングネーム）が変更されたかどうか、Adabas ショートネームを基に確認されます。 変更された場合、ステータスフィールドは値［Alias］になり、新しいフィールドの名前が、先頭にコロン（：）を付けてメッセージフィールドに表示されます。 フィールド名の隣の［Sel］列に「A」を入力して、エントリを新しい名前で更新します。

FDT 生成の場合、フィールド名は常に Adabas ショートネームに "-FIELD" を付けた名前として生成されます。 トリガメンテナンス機能が生成されなかった名前を検出すると、メッセージでは "xx-FIELD" という形式のロングフィールド名として使用されます。 もう一度［Sel］列で「A」を入力し、変更を確認します。 「*FDT のサンプル*」には、各フィールドのステータスがあります。［Active］とマークが付いたフィールドは、ファイル - フィールドテーブルにすでに含まれます。

生成タイプが［F］（Adabas FDT）の場合、フィールド名は "xx-FIELD" のように生成されます（"xx" は Adabas フィールド名）。 その後、DDM または Predict Adabas ファイルのファイル番号とファイル名が FDT 定義でのファイル番号とファイル名と同じ場合、これらのフィールドは、DDM または Adabas ファイルのユーザー定義名で更新されることがあります。

Note: ［Field Definition］列のフィールドは、ADACMP FNDEF フォーマットで表示されます。"U" はアンパック形式の数値を表し、フィールド長は文字数ではなくバイト数です。

```
HH:MM:SS              ***** TRIGGER MAINTENANCE *****          YYYY-MM-DD
 User USR01         - Generate/Modify File-Field Table -        DBnr 105

  FDT File Name .. AUTOMOBILES                                FDT Fnr ... 4

  Sel Field Definition    Long Field Name   Status   Message
  --- -----------------   ---------------   ------   --------------
   _  01,AA,020,A,DE,NU   MAKE              Active
   _  01,AB,020,A,DE,NU   MODEL             Active
   _  01,AC,015,A,DE,NU   BODY-TYPE         Active
   _  01,BA,002,U,DE,NU   BA-FIELD
   _  01,BB,003,U,DE,NU   BB-FIELD
   _  01,BC,005,U,NU      BC-FIELD
   _  01,BD,005,U,NU      WEIGHT            Active
   _  01,CA,010,A,DE,NU   COLOR             Active
   _  01,DA,002,U,DE,NU   YEAR              Active
   _  01,DB,016,A,NU      DB-FIELD
```

```
  _  01,FA,006,U,DE,NU   FA-FIELD
  _  01,FB,006,U,DE,NU   MILEAGE           Active

  Mark Fields with 'A' Add, 'D' Delete, 'G'Generate, 'I' Info, or '.' Exit

Command==>
Enter-PF1---PF2---PF3---PF4--PF5---PF6--PF7--PF8--PF9--PF10--PF11--PF12
      Help  Menu  Exit       Next  --    -    +                    Canc
```

手順 8.14. 1 つ以上のフィールドを追加する手順

- フィールド名の隣の［Sel］列に「A」を入力して、追加する各フィールドにマークを付けます。 その後、Enter キーを押します。

  すべてのフィールドをファイル - フィールドテーブルに追加する場合は、コマンド行に「ALL」と入力します。

手順 8.15. 1 つ以上のフィールドを削除する手順

- フィールド名の隣の［Sel］列に「D」を入力して、削除する各フィールドにマークを付けます。 その後、Enter キーを押します。

手順 8.16. グループフィールドテーブルのエントリを生成する手順

- フィールド名の隣の［Sel］列に「G」を入力して、エントリを生成する各フィールドにマークを付けます。 その後、Enter キーを押します。

  グループフィールドテーブルエントリは、フィールドがグループまたは PE グループの一部である場合にのみ作成できます。

手順 8.17. 各フィールドの属性を表示する手順

- フィールド名の隣の［Sel］列に「I」を入力して、フィールドにマークを付けます。 その後、Enter キーを押します。

  フィールドのグループフィールドエントリが生成されている場合は、結果のウィンドウに、フィールド名、フォーマット、長さ、タイプ（スーパーディスクリプタの場合は SP、サブディスクリプタの場合は SB）、およびインジケータが表示されます。

トリガ定義

Note: TRGMAIN と、ユーザープログラムからトリガを保守する API については、「*TRGMAIN: トリガを保守するための API*」を参照してください。

概念上、トリガは、2 つの部分から成ります。トリガするイベントと、トリガされるプロシージャです。 トリガするイベントは、一連の選択条件で定義されます。 条件が満たされると、それに応じてトリガされるプロシージャが実行されます。

選択条件（ファイル名、コマンドタイプ、フィールド名）は、トリガ定義の一部として対象のデータベースに格納されます。ファイル-フィールドテーブルは、これらを対応する物理 Adabas ファイル番号と 2 文字のフィールド ID にマッピングします。 ファイル名とフィールド名は、最大 32 文字のわかりやすい名前にします。

Note: トリガの結果として実行されるプロシージャからのデータベースコマンドは、開始 Adabas コマンドが実行されたデータベースに限定されません。

ファイル名

ファイル名では、開始 Adabas コマンドが操作するファイルを指定します。 トリガが対象とする Adabas ファイルは、1 つだけです。 トリガを複数のファイルに適用する場合は、ファイル名以外は同一のトリガを複数（ファイルごとに 1 つ）定義します。

コマンドタイプ

コマンドタイプでは、開始 Adabas コマンドのコマンドクラスを指定します。 トリガは、指定されたコマンドタイプに基づいて実行するように定義されます。

次の Adabas コマンドクラスの 1 つまたはすべてを単一のトリガ定義に指定できます。

コマンドクラス	コマンドコード
FIND	S1、S2、S4
READ	L1、L2、L3、L4、L5、L6、L9
UPDATE	A1、A4
INSERT	N1、N2
DELETE	E1、E4

トリガをすべてのコマンドクラスに適用する場合は、コマンドタイプフィールドを空白のままにします。 デフォルトでは "All"（すべてのコマンド）です。

トリガをすべてのコマンドクラスではなく、複数のコマンドクラスに適用する場合は、コマンドクラス以外は同一のトリガを複数（クラスごとに 1 つ）定義します。

フィールド名

トリガは、ファイル内の単一のフィールドまたはすべてのフィールドで操作するコマンドと関連付けることができます。 例えば、EMPLOYEES ファイルの SALARY フィールドで UPDATE コマンドが実行されるたびに起動するトリガを定義することが可能です。

フィールドを指定することが常に適切であるとは限りません。 例としては、次のようなものがあります。

- 特定のフィールドを DELETE コマンドと関連付けることは無意味です。DELETE コマンドはフォーマットバッファを必要としないからです。
- プレコマンドトリガの読み込みコマンドの場合、フィールドにはデータが含まれません。 そのため、読み込むフィールド、または要求を発行したユーザーのユーザー ID のいずれかを検証しない場合、フィールドを指定する必要はありません。

トリガをすべてのフィールドではなく、複数のフィールドに適用する場合は、フィールド以外は同一のトリガを複数（フィールドごとに 1 つ）定義します。

トリガは、イベントの選択条件に従って、1つのフィールドだけ（つまりフォーマットバッファで指定されているフィールド）に対して起動します。 ただし、トリガを複数のフィールドに対して起動する必要がある場合は、特定のコマンドとフィールドに対してトリガファイルを定義し、プロシージャ自体でその他のフィールドの存在を確認することができます。 「*マルチトリガのサポートの実装*」を参照してください。

プロシージャは、他のプロシージャを呼び出すかどうか、また呼び出すのであればどのフィールドに対して呼び出すのかを確認できます。 このメカニズムを利用すれば、"メインの" プロシージャで、エラーを処理したり、前に実行したプロシージャでエラーが発生した場合でも他のプロシージャを呼び出す必要があるかどうかを判断したりすることができます。 そのためユーザーは、プロシージャを起動するかどうかを判断するために精度の高いルールセットが必要な状況を柔軟に制御することができます。

［Trigger Definitions］メニュー

［Trigger Definitions］メニュー（下図参照）には、トリガ定義の作成と保守に使用できる機能があります。

手順 8.18. ［Trigger Definitions］メニューを表示する手順

■ メインメニューで［Create/Modify Trigger Definitions］を表す「T」を入力します。

```
HH:MM:SS                ***** TRIGGER MAINTENANCE *****          YYYY-MM-DD
 User USR01               - Trigger Definitions Menu -            DBnr 105

                            Code  Function
                            ----  -------------------------
                              A   Add Trigger Definition
                              D   Display Trigger Definition
                              M   Modify Trigger Definition
                              P   Delete Trigger Definition
                              S   Select Trigger Definition
                              ?   Help
                              .   Exit
                            ----  -------------------------
               Code ........ _       Active/Deactive Opt... _
               File Name ... ________________
               Cmd Type .... _
               Field Name .. ________________

 Command ==>
 Enter PF1---PF2---PF3---PF4---PF5---PF6---PF7--PF8--PF9--PF10--PF11--PF12--
       Help  Menu  Exit  Field       Admin Procs          FTRG  FDIC  Canc
```

［Trigger Definitions］メニューでは、トリガ定義を追加、表示、修正、または削除できます。 画面に同じファイルに対する複数のトリガ定義を表示したり、ポップアップウィンドウにファイル内の特定フィールドの単一トリガ定義を表示したりすることができます。

手順 8.19. 特定ファイルのすべてのトリガ定義にアクセスする手順

- 機能コード（A は追加、D は表示、M は修正、P はパージ／削除）とファイル名を入力します。 コマンドタイプとフィールド名にはワイルドカード値を入力します。

  そのファイルのすべてのトリガ定義が画面に表示されます。 この画面にアクセスするために入力した機能コード（A、D、M、または P）に応じて、1 つ以上のトリガ定義を追加、表示、修正、または削除できます。 「*複数のトリガ定義*」を参照してください。

手順 8.20. 特定のトリガ定義にアクセスする手順

- 機能コード（A、D、M、または P）、ファイル名、およびフィールド名を入力します。

  トリガ定義がポップアップウィンドウに表示されます。 機能（A、D、M、または P）に応じて、トリガ定義を追加、表示、修正、または削除できます。 「*単一トリガの定義*」を参照してください。

［Trigger Definitions］メニューを使用して、1 つ以上のトリガを選択することもできます。

手順 8.21. トリガを 1 つ以上選択する手順

1 機能コード「S」と、ファイル名またはファイル名にワイルドカード値（例えば「>G」）を入力します。

表示される画面は、次の例のようになります。

Note: 特定のファイル名を選択した場合、画面にはファイル名ではなくロングファイル名が表示されます。

```
HH:MM:SS          ***** TRIGGER MAINTENANCE *****             YYYY-MM-DD
User USR01             - List Trigger Definitions -           DBnr 105

Sel File Name / Long Field Name    Commnd  Type      ProcName Para   RecB
--- --------------------------    ------- ----      -------- ------ ----
_ MISCELLANEOUS                   Delete  Pre Non-P SYMP0003 Cont   None
   ** Any Field **
_  VEHICLES-FILE                  Delete  Pre Non-P SYMP0003 Cont   None
   ** Any Field **
_  VEHICLES-FILE                  Read    Pre Async ANYONE   Contrl None
   ** Any Field **
_  VEHICLES-FILE                  Read    Pre Async MAKE0001 Cont   None
   MAKE
_  VEHICLES-FILE                  Read    Pre Async MODEL001 Cont   None
   MODEL
_  VEHICLES-FILE                  Read    Pre Async COLOR001 Contrl None
   COLOR
_  VEHICLES-FILE                  Read    Pre Async CLASS001 Contrl None
   CLASS
_  GDMUSIC                        (All)   Pre Async PROC008  Contrl None
   ** Any Field **

Command ==>
Enter PF1---PF2---PF3---PF4---PF5---PF6---PF7--PF8---PF9---PF10--PF11--PF12
           Menu  Exit              --    -    +                       Canc
```

2 ファイル名の隣の［Sel］フィールドに、表示の場合は「D」、修正の場合は「M」、選択の場合は「S」、パージの場合は「P」を入力します。

「D」、「P」、または「M」を入力すると、トリガ定義がポップアップウィンドウに表示されます。「*単一トリガの定義*」を参照してください。

「S」を入力すると、そのトリガが選択され、［Trigger Definitions］メニューが表示されます。

複数のトリガ定義

手順 8.22. 1 つ以上のトリガ定義を追加、表示、修正、または削除する手順

1 ［Trigger Definitions］メニューで、コード（A、D、M、またはP）とファイル名を入力します。コマンドタイプの指定は任意です。空白のままにしておくと、デフォルトでAll（すべてのコマンド）になります。

入力したコードに応じて、次のいずれかの画面が表示されます。

- ［Add Trigger Definitions］
- ［Display Trigger Definitions］
- ［Modify Trigger Definitions］
- ［Delete Trigger Definitions］

これらの画面には、指定したファイルのトリガ定義が表示されます。例えば［Modify Trigger Definitions］画面は、次のようになります。

```
HH:MM:SS                ***** TRIGGER MAINTENANCE *****          YYYY-MM-DD
User USR01              - Modify Trigger Definitions -           DBnr 105

                                                          Pre-Post .. Pre
File ... VEHICLES-FILE (3)                                Command ... Read

Prty Long Field Name                  Fld Type   ProcName Params RecBuff Msg
---- -------------------------------- --- -----  -------- ------ -------
010  COLOR____________________________ AF  Async  COLOR001 Contrl None___
020  CLASS____________________________ AH  Async  CLASS001 Contrl None___
030  MAKE_____________________________ AD  Async  MAKE0001 Contrl None___
040  MODEL____________________________ AE  Async  MODEL001 Contrl None___
050  ** Any Field **__________________ **  Async  SAMP0002 Contrl None___
___  _________________________________ __  _____  ________ ______ _______
___  _________________________________ __  _____  ________ ______ _______
___  _________________________________ __  _____  ________ ______ _______
___  _________________________________ __  _____  ________ ______ _______
___  _________________________________ __  _____  ________ ______ _______
___  _________________________________ __  _____  ________ ______ _______
Modify Entries, or enter '.'(Exit), or '?'(Help) to see options ... _

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12
      Help  Menu  Exit        Updat       -     +           Reseq Post  Canc
```

2 適切な情報を入力したら、PF5 キーを押してトリガテーブルを更新します。

更新が確認されたことがメッセージにより通知されます。 入力した情報が無効または不完全な場合は、エラーメッセージが表示されます。

エントリフィールド

トリガ定義の画面のエントリフィールドについて次に説明します。

Note: ロングフィールド名と［Fld］（ショートフィールド名）の両方を入力する必要はありません。一方を入力すると、もう一方はファイル-フィールドテーブルエントリから設定されます。

Prty（プライオリティ）
Adabasトリガドライバは、各トリガに定義された選択条件と一致するかどうか、フォーマットバッファをスキャンします。同じファイルとコマンドに複数のフィールド名が指定されている場合、各フィールドに割り当てられるプライオリティによって、処理される順序が決まります。一致が見つかると、トリガが起動されます。［Prty］フィールドでは、シーケンスを設定または修正できます。

最高のプライオリティは 1 です。 ただし、プライオリティは 10 刻みで 10～900 の範囲で表されます。 それ以外の値が入力されると、フィールドの順序、つまり "プライオリティ" が変化します。 10 刻みの途中の値を入力すると、次の 10 刻みの値へと順番に並べ替えられます（RESEQ コマンドを使用、「コマンド」を参照）。例えばフィールドにプライオリティとして 11 と 12 を入力すると、それぞれの値は 20 と 30 として並べ替えられます。

フィールドのプライオリティを変更するには、1～9 の値を指定します。

- プライオリティを下げるには、そのフィールドの表す値より大きい値を指定します。
- プライオリティを上げるには、そのフィールドの表す値より小さい値を指定します。

例

［List Trigger Definitions］画面で MAKE のプライオリティの値を 30 から 10 に変更し、それに応じて他のフィールドのプライオリティを変更する場合は、MAKE に 1～9 の値を割り当てます。

RESEQ コマンドを使用すると、すべてのフィールドのプライオリティが 10～900 に変更され、MAKE が 10、COLOR が 20、CLASS が 30 というようになります。

Long Field Name
フィールドの Adabas ロングフィールド名です。 フィールド名の選択リストを表示するには、ワイルドカードを入力します。 ロングフィールド名が不明な場合は、［Fld］（ショートフィールド名）から設定できます。

Fld（ショートフィールド名）
Adabas ショートフィールド名、つまり特定ファイルの特定フィールドを識別するために DBMS で使用されるユニークな名前です。フィールドのロングフィールド名と対応する必要があります。 ショートフィールド名が不明な場合は、ロングフィールド名から設定できます。

Type
［Type］は、非同期、関与、または非関与です。 デフォルト値は非同期です。

ProcName
トリガの選択条件が満たされたときに呼び出される Natural サブプログラムの名前です。 値は、1～8 文字の有効な Natural サブプログラム名です。 デフォルト値はありません。

Important: トリガを呼び出すユーザージョブの名前は、トリガの ProcName とは異なる必要があります。

Params

トリガのプロシージャが呼び出されるときに渡されるパラメータは次のとおりです。

Contrl	ACB インターフェイスを使用します。トリガ要求、トリガコマンド、および修正可能なレスポンスコードフィールドに関する情報を渡すために、制御パラメータが使用されます。Contrl がデフォルト値です。
Contrx	ACB または ACBX インターフェイスを使用します。トリガ要求、トリガコマンド、および修正可能なレスポンスコードフィールドに関する情報を渡すために、制御パラメータが使用されます。
Resp	プレコマンドトリガの場合には、コマンドの実行を防ぐために、修正可能なレスポンスコードフィールドが使用されます。レスポンスコードは同期トリガでのみ使用されます。レスポンスコードには、すでに完了している可能性のある非同期トリガに対する値や意味はありません。
None	パラメータは渡されません。

RecBuff
値は access（読み取り専用）、update（読み書き）、または update（なし）です。

Msg
エラーが発生したときに、このフィールドにエラーメッセージが表示されることがあります。 エラーの説明は画面の下部に表示されます。

コマンド
次の表で、トリガ定義の画面のコマンド行に入力できるコマンドについて説明します。

Note: コマンドは、大文字、小文字、または大文字と小文字を混在させて入力することができます。

コマンド	説明
PRTY	［Modify Trigger Definitions］画面を表示します。トリガに割り当てられたプライオリティを修正できます。
UPDATE	入力した値でトリガファイルを更新します。
BACK、-	前のページを表示します。
FWD、+	次のページを表示します。
RESEQ	トリガ定義のリストをプライオリティ順に並べ替えます。

コマンド	説明
POST	このファイルとコマンドのポストコマンドトリガを表示します。
PRE	このファイルとコマンドのプレコマンドトリガを表示します。
ACTIVATE	プレコマンドトリガまたはポストコマンドトリガをアクティブ化します。
DEACTIVATE	プレコマンドトリガまたはポストコマンドトリガを非アクティブ化します。
DELETE	選択したトリガ定義をトリガファイルから削除します。
MODIFY	トリガ定義を修正します。

単一トリガの定義

手順 8.23. 単一のトリガ定義を追加、表示、修正、または削除する手順

■ ［Trigger Definitions］メニューで、コード（A、D、M、またはP）、ファイル名、およびフィールド名を入力します。

コマンドタイプの指定は任意です。空白のままにしておくと、デフォルトで All（すべてのコマンド）になります。

入力したコードに応じて、追加、表示、修正、または削除機能のポップアップウィンドウが表示されます。 これらのウィンドウには、指定したトリガ定義に関するトリガ情報とプロシージャ情報が表示されます（「修正機能画面」を参照）。

■ トリガ情報には、ファイルの名前と番号、コマンドタイプ（読み込み、更新など）、ロングファイル名、および Adabas フィールド名があります。 また、次の表に示すようなトリガの現在の状態も表示されます。

ステータス	説明
アクティブ	トリガファイルと現在アクティブなニュークリアスに対するトリガが現在アクティブです。
非アクティブ	トリガの永続的な非アクティブ化が要求されました。アクティブ化されるまで、トリガはこの状態のままになります。
Temp Inactive	トリガは実行中のニュークリアスでのみ非アクティブです。 一時的な "非アクティブ化" が要求されました。"アクティブ化" が発行されるまで、トリガはこの状態のままになります。
File Not Used	トリガ定義で指定されたファイル番号が、データベースの最大ファイル番号（使用されている最大のファイル番号+10）よりも大きい値です。「*トリガテーブルの作成*」を参照してください。
Not Loaded	最後にニュークリアスがアクティブ化された後で、トリガがファイルに追加されました。 次回 REFRESH コマンドが発行されるか、ニュークリアスが再スタートすると、トリガはアクティブになります。
Not Checked	トリガメンテナンス機能はニュークリアスでアクティブではありません。そのため、トリガのステータス確認を実行できません。

ステータス	説明
Inaccessible	Adabas ニュークリアスは、トリガステータスに関するすべての要求を受け入れていません。

- プロシージャ情報には、プロシージャ名、プレコマンドステータスまたはポストコマンドステータス、トリガタイプ（非同期、関与、非関与）、CALLNAT パラメータタイプ（cntrl、resp、または none）、およびレコードバッファオプション（access、update、または none）があります。

手順 8.24. トリガ定義を追加する手順

1 プロシージャの名前を入力します。

2 必要に応じてオプションを修正します。

3 PF5 キーを押して追加を確認します。

手順 8.25. トリガ定義を修正する手順

1 ［Modify Function］ポップアップは、次のようになります。

```
Modify Details and press 'PF5' to Confirm Update.

  HH:MM:SS   *** Define Trigger Info ***   YYYY-MM-DD
                  - Modify Function -

  Trigger Information   ** Active **
    File Number ....... 4
    File Name ......... AUTOMOBILES
    Command Type ...... Read
    Long Field Name ... BODY-TYPE
    Adabas Field ...... AC
    Field Prty/Seq .... 010
  Procedure Information
    Name (Subpgm)...... _________
    Pre Cmd Select .... N  (Post)
    Trigger Type ...... N  (Non-Participating)
    CALLNAT Params .... C  (Cntl Info + Resp)
    RecBuffer Access .. A  (May be Accessed)

  Command ==>
  Enter-PF1--PF2--PF3--PF4--PF5--PF6--PF7--PF8--
        Help Menu Exit Prty Updat     Act  Deact
```

2 ［Procedure Information］の［Field Prty/Seq］（プライオリティまたはシーケンス）などのフィールドを修正します。

3 トリガをアクティブ化または非アクティブ化する場合は、コマンド行に「ACTIVATE」または「DEACTIVATE」を入力します。 次に、トリガのステータスを一時的にするか、永続的にするかを選択します。

コマンド		意味
アクティブ化	一時的	ニュークリアスでトリガをアクティブ化しますが、トリガファイルでは非アクティブステータスのままにします。
	永続的	ニュークリアスでトリガをアクティブ化し、トリガファイルから非アクティブステータスを削除します。
非アクティブ化	一時的	データベースに対して発行されたすべてのコマンドのイベント条件を確認するトリガを無視します。つまり、この定義に対してトリガを起動できません。
	永続的	ステータス "非アクティブ" でトリガファイルにトリガを格納します。 ニュークリアスが開始してもアクティブ化されません。 ただし、トリガを後で開始することはできます。 アクティブな Adabas セッションは直ちに非アクティブ化されます。

4 PF5 キーを押して更新を確認します。

手順 8.26. トリガ定義を削除する手順

- PF5 キーを押して削除を確認します。

手順 8.27. ファイルのすべてのフィールドをプライオリティ順に表示する手順

- PF4 キーを押すか、コマンド行に「PRTY」を入力します。

  Note: トリガでコマンドタイプ "Delete" を指定した場合は、ユーザーがプレトリガ定義やポストトリガ定義を同時に定義することができる画面が表示されます。

プロシージャレポート

次に示す［Procedure Reporting］メニューでは、トリガされたプロシージャのリストを取得できます。このリストは、ファイルまたはプロシージャ名をアルファベット順にソートしたリストです。 レポートは、特定のファイルに制限することも、すべてのファイルを含めることもできます。 プロシージャレポートを使用すると、例えば重複したプロシージャを特定したり、特定のプロシージャが使用されている各インスタンスを識別したりすることができます。

手順 8.28. ［Procedure Reporting］メニューを表示する手順

- メインメニューで、［Procedure Reports］を表す「R」を入力します。

```
HH:MM:SS               ***** TRIGGER MAINTENANCE *****       YYYY-MM-DD
User USR01              - Procedure Reporting Menu -         DBnr 105

                          Code  Function
                          ----  -------------------------
                            F   Display Procedures by File
                            N   Display Procedures by Name
                            ?   Help
                            .   Exit
                          ----  ------------------------

             Code ......... N
             File Number .. _____
             Procedure .... PROC0001




 Command ==>
 Enter PF1---PF2--PF3--PF4---PF5---PF6---PF7---PF8--PF9--PF10--PF11--PF12
       Help  Menu Exit Field Trigr Admin                            Canc
```

手順 8.29. 特定のファイルのみのプロシージャをリストする手順

■ ［Display Procedures by File］を表すコード「F」とファイル番号を入力します。

手順 8.30. 特定の名前から始まるプロシージャをリストする手順

■ ［Display Procedures by Name］を表すコード「N」とプロシージャ名を入力します。

どちらの場合も表示される画面には、次の例で示すように、情報のタイプが含まれます。

```
HH:MM:SS               ***** TRIGGER MAINTENANCE *****       YYYY-MM-DD
User USR01                 - List Procedure Names -          DBnr 105

 File Name ... AUTOMOBILES                                      Fnr ... 4


 Sel  ProcName   Command  Field Name                  When  Type   ParmTy
 --- --------   -------  --------------------------  ----  -----  ------
  _   PROC0001   Read     BODY-TYPE                    Post  NonP   Cntrl
  _   PROC0001   Update   ** Any Field **              Pre   Async  Cntrl
      PRO001     (All)    MILEAGE                      Pre   Async  Cntrl
  _   SUBPGM     Read     COLOR                        Pre   Async  Cntrl
```

```
 Select 'D' to Display, or enter 'F'(Fwd), 'T'(Top), 'B'(Bck), '.'(Exit)
 Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6--PF7--PF8---PF9--PF10--PF11--PF12
      Help  Menu  Exit              --    -    +                    Canc
```

プロシージャレポートには、データベース番号、ファイル番号、およびファイル名と、プロシージャごとに次の情報が示されます。

フィールド	説明
ProcName	トリガされたプロシージャの名前。
Command	トリガを開始した Adabas コマンド。
Field Name	トリガフィールド名。
When	プロシージャを実行するタイミング。 ■ "pre-command": 開始 Adabas コマンドの前にプロシージャを実行します。 ■ "post-command": 開始 Adabas コマンドの後にプロシージャを実行します。
Type	トリガのタイプ。 ■ "non-participating": トリガされるプロシージャは、開始 Adabas コマンドのトランザクションロジックに関与しません。 ■ "participating": トリガされるプロシージャは、開始 Adabas コマンドのトランザクションロジックに関与することがあります。 ■ "asynchronous": Adabas コマンドとトリガされるプロシージャを個々に実行し、お互いに影響し合いません。トリガされるプロシージャのトランザクションロジックは、Adabas コマンドトランザクションロジックに関与しません。
ParmTy	プロシージャが呼び出されるときに渡されるパラメータのタイプ。 ■ "control": トリガ要求、トリガコマンド、および修正可能なレスポンスコードフィールドに関する情報を渡すために、ACB形式の制御パラメータが使用されます。これがデフォルト値です。 ■ "controlx": トリガ要求、トリガコマンド、および修正可能なレスポンスコードフィールドに関する情報を渡すために、ACB または ACBX 形式の制御パラメータが使用されます。 ■ "response": プレコマンドトリガの場合にコマンドの実行を防ぐために、修正可能なレスポンスコードフィールドが使用されます。 レスポンスコードは同期トリガでのみ使用されます。レスポンスコードには、すでに完了している可能性のある非同期トリガに対する値や意味はありません。 ■ "none": パラメータは渡されません。

手順 8.31. 特定のプロシージャの詳細情報を取得する手順

■ プロシージャ名の隣の［Sel］列に「D」を入力します。

ポップアップウィンドウに、トリガ定義とプロシージャに関する情報が次のように表示されます。

```
Trigger Information currently displayed

               HH:MM:SS   *** Define Trigger Info ***   YYYY-MM-DD
                             - Display  Function -
               Trigger Information
                 File Number ....... 4
                 File Name ......... AUTOMOBILES
                 Command Type ...... Read
                 Long Field Name ... BODY-TYPE
                 Adabas Field ...... AC
                 Field Prty/Seq .... ____
               Procedure Information
                 Name (Subpgm)...... PROC0001
                 Pre Cmd Select .... N  (Post)
                 Trigger Type ...... N  (Non-Participating)
                 CALLNAT Params .... C  (Cntl Info + Resp)
                 RecBuffer Access .. A  (May be Accessed)

               Command ==>
               Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8-
                     Help  Menu  Exit  Prty
```

- トリガ情報には、トリガイベント条件、つまりファイルの名前と番号、コマンドタイプ、ロングファイル名、および Adabas フィールド名があります。
- プロシージャ情報には、Naturalサブプログラム名、プロシージャがプレコマンドとポストコマンドのどちらであるか、トリガタイプ（非同期、関与、非関与）、CALLNAT パラメータタイプ（cntrl、resp、または none）、および RecBuffer アクセスステータス（A= アクセス、N= アクセスなし、U= アクセス／更新）があります。

管理者機能

管理者機能では、トリガアクティビティを監視したり、プロファイルを表示および修正したり、ジョブステータス設定やバッファサイズを保守したりすることができます。

手順 8.32. ［Administrator Functions］メニューを表示する手順

■ メインメニューで、［Administrator］を表す「A」を入力します。

```
HH:MM:SS              ***** TRIGGER MAINTENANCE *****          YYYY-MM-DD
User USR01            - Administrator Functions Menu -         DBnr 105


                      Code  Function
                      ----  --------------------------
                        A   Active Session Settings
                        D   Display Profile Information
                        M   Modify Profile Information
                        S   Subsystem Activity
                        T   Trigger Activity
                        ?   Help
                        .   Exit
                      ----  --------------------------

               Code ... D

Command ==>
Enter PF1---PF2---PF3---PF4---PF5---PF6--PF7--PF8---PF9---PF10--PF11--PF12--
      Help  Menu  Exit  Field Trigr      Procs            FTRG        Canc
```

アクティブなセッションの設定

アクティブなセッションの設定には、ジョブステータス設定やバッファサイズなどが含まれます。

手順 8.33. アクティブなセッションの設定を修正する手順

1 ［Administrator Functions］メニューで、［Active Session Settings］を表す「A」を入力します。

2 フィールドの値を修正し、PF5 キーを押して設定を更新します。

ニュークリアスクラスタ環境では、アクティブなセッションの設定に加えた変更がクラスタ内のすべてのアクティブなニュークリアスに反映されます。

［Active Session Settings］画面は、次の例のようになります。

```
HH:MM:SS               ***** TRIGGER MAINTENANCE *****           YYYY-MM-DD
User USR01               - Active Session Settings -             DBnr 105

 Job Name ..........SAGDT077          Trigger File Number ......12

 SVC Number ........217                Max File to be accepted...60
 Nucleus .......... Active__
 Triggers ......... Active__          Session Buffer Sizes in Bytes
 Stored Proc. ..... Active__            Trigger Table Buffer....8192
 Error Action ..... Halt                Pre Trigger Queue.......15244
 Trigger Logging .. Active__            Post Trigger Queue......2960
 Activity Timeout.. 600                 Waiting Subsys Queue....80
 Subsystems                             Acquired Storage........31232
   Maximum ........5                    Used Storage............31232
   Active .........5
   Inactive ...... 0
   Waiting ........5
   In Progress ....0

Change Parameters as required or press 'PF3' to Exit

Command ==>
Enter PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12-
      Help  Menu  Exit  Sact  Updat                                      Canc
```

Note: ［Subsystem Activity］画面を表示するには、PF4 キーを押します。

［Active Session Settings］のフィールドについて次の表で説明します。

フィールド	説明
Job Name	Adabas ニュークリアスのジョブ名。
SVC Number	データベースで使用している SVC の番号。
Nucleus	Adabas トリガとストアドプロシージャが現在のデータベースでアクティブであるかどうかを示します。
Triggers	"active"（デフォルト）の場合、このデータベースでトリガを実行できます。"inactive" の場合、実行できません。 このフィールドは変更できます。「refresh」と入力すると、トリガテーブルを更新できます（「*トリガテーブルの更新*」を参照）。
Stored Procedures	"active"（デフォルト）の場合、このデータベースでストアドプロシージャを実行できます。"inactive" の場合、実行できません。 このフィールドは変更できます。
Error Action	処理のエラーが発生したときに、トリガメンテナンスで実行されるアクション（"reject"、"halt"、または "ignore"）。 このフィールドは変更できます。
Trigger Logging	ロギング機能が "active" と "inactive" のどちらであるかを示します。

フィールド	説明	
Activity Timeout	タスクがキャンセルされる前の秒数。 デフォルト値は 60 で、最大値は 9999 です。 このフィールドは変更できます。 ゼロに設定すると、Adabas TT パラメータ値のデフォルト値になります。	
Subsystems	Maximum	割り当てられる Natural サブシステムの数。 値は 01～10 です。
	Active	開始した Natural サブシステムの数。 システムを回復できないエラーが発生し、サブシステムが非アクティブな場合、この値は可能な最大値より小さくなることがあります。
	Inactive	システムを回復できないエラーが発生したために現在終了またはシャットダウンしている Natural サブシステムの数。
	Waiting	現在処理を待機中のサブシステムの数。
	In Progress	現在トリガを処理中のサブシステムの数。
Trigger File Number	データベースのトリガファイルのファイル番号。	
Max File to be accepted	Adabas トリガとストアドプロシージャが受け入れ可能な最大ファイル番号。 データベースの起動時や初期化時に、ロード済みの最大のファイル番号に 10 を足した値として設定されます。 その後、さらに大きいファイル番号がデータベースに追加された場合、ファイルで検出されたすべてのトリガが無視されます。 Max File 値を超えるトリガをアクティブ化するには、データベースをシャットダウンしてから再スタートする必要があります。	
Session Buffer Sizes in Bytes	Trigger Table Buffer	トリガテーブルバッファのサイズ。 このフィールドは、プロファイル修正機能を使用して修正できます。
	Pre-Trigger Queue	任意の時点で処理中であるすべてのプレコマンドトリガのキューのサイズ。
	Post-Trigger Queue	任意の時点で処理中であるすべてのポストコマンドトリガのキューのサイズ。
	Waiting Subsys Queue	トリガの処理を待機しているタスクのキューのサイズ。 現在のところ、80 バイトで固定です。
	Acquired Storage	Adabas トリガとストアドプロシージャで使用するために取得されるストレージの合計サイズ。
	Used Storage	処理時に Adabas トリガとストアドプロシージャで使用されているストレージの実際のサイズ。

トリガテーブルの更新

REFRESH コマンドを使用すると、データベースをシャットダウンせずに新しいトリガをトリガテーブルに追加できます。 ニュークリアスクラスタ環境では、REFRESH コマンドですべてのクラスタニュークリアスのトリガテーブルが更新されます。

ロードされる追加トリガの数が、最初にロード済みの数よりも非常に大きい場合は、増加したトリガを処理するのに十分な追加スペースを手動で割り当てます。 トリガテーブルバッファが十分大きくない場合、Adabas トリガプロファイルのエラーアクションフィールドの値に基づいて Adabas トリガとストアドプロシージャが終了するような不整合が発生することがあります。

必要なトリガを使用するアプリケーションを実装する前に、それらのトリガをあらかじめロードしておくと、トリガテーブルの更新を回避できます。

手順 8.34. トリガテーブルを更新する手順

■ コマンド行で「REFRESH」と入力するか、トリガフィールドに「refresh」と入力します。

バッファサイズの計算

トリガテーブル、プレトリガキュー、およびポストトリガキューには、バッファが必要です。

トリガテーブル

トリガテーブルのバッファサイズは、次のように計算します。

```
((TOT TRG + TOT FILE)) * TRG ELEMENT) + 4096 = TOTAL  SIZE
```

ここでは次の内容を表しています。

TOT TRG	トリガテーブルで定義されたトリガの合計数
TOT FILE	トリガが存在するファイルの合計数
TRG ELEMENT	トリガ要素のサイズ（28 バイト）

Adabas トリガドライバで TOTAL SIZE を計算すると、結果は 256 の倍数に切り捨てられます。

プレトリガキューとポストトリガキュー

データベースに渡される 1 秒当たりのコマンド数、実際にプロシージャが実行される時間、およびトリガが同期と非同期のどちらであるかによって、キューイングが起こる場合と起こらない場合とがあります。 各バッファは、プレトリガとポストトリガを個別にキューイングするように設定されています。 キューがいっぱいになると、トリガを起動するそれ以降のコマンドは、レスポンスコード 154 を受け取ります。 キューイングが消去された場合、DBA はキューサイズを大きくすることを検討してください。

バッファサイズを設定する場合は、プレトリガとポストトリガの比率についても考慮してください。 例えば、まったくプレトリガを使用しない場合、プレトリガキューは必要ないため、すべてのバッファスペースをポストトリガキューに割り当てる必要があります。

- プレトリガキューが必要になるのは、プレコマンドトリガがトリガファイルで定義されている場合に限られます。 プレトリガキューのバッファサイズは、次のように計算します。

  ```
  (NC / 2) * 148 = TOTAL SIZE
  ```

  "NC" は、ADARUN パラメータ NC の値です。

  この計算は、すべてのトリガが同期の場合に有効です。 非同期トリガを使用する場合、コマンドは、前のコマンドによるプロシージャが完了する前に、連続して発行される可能性があります。 このため、キューイングが発生し、さらに大きなバッファサイズが必要になります。

  バッファの量を増やすかどうかは、発行されるコマンドの数と発行される速さ（つまり、コマンドのレスポンスと次のコマンドの発行までの時間間隔）、および各トリガのプロシージャが完了するまでにかかる時間によります。 例えば、バッチジョブが非同期トリガを起動する数千ものコマンドを発行する場合、非常に大きなバッファが必要になります。
- ポストトリガキューが必要になるのは、ポストコマンドトリガがトリガファイルで定義されている場合に限られます。 バッファサイズは、プレトリガキューのバッファサイズとまったく同じ方法で計算します。サイズに関して同様の検討が必要です。

プロファイル情報の表示／修正

プロファイルには、Adabas トリガとストアドプロシージャで必要なシステム情報があり、指定したデータベース ID とファイル番号から生成されます。 プロファイルが生成されたら、そのプロファイルを表示したり、プロファイル内の値を修正したりすることができます。

手順 8.35. プロファイル表示する手順

- ［Administrator Functions］メニューで、［Display Profile Information］を表す「D」を入力します。

  ［Display Profile Information］画面が次の例のように表示されます。

```
HH:MM:SS              ***** TRIGGER MAINTENANCE *****         YYYY-MM-DD
User USR01            - Display Profile Information -         DBnr 105

 Triggers Status .......... Active__                 Total Triggers ..34
 Stored Proc. Status ...... Active__

 Natural Subsystem Parameters
   Batch Natural Name ..... NATAPT
   Maximum Subsystems ..... 6_
   Activity Timeout ....... 600/80
   NATPARM Parameters ..... DU=OFF,INTENS=1,ETID='  ' __________
```

```
                          ____________________________________

  Fixed NATPARM Parm ..... STACK=(LOGON:SYSSPT;STP),PROGRAM=STPEND
  CMPRINT Assignment ..... TSPRT

 Required .. N     UserID ... USER**__  Password .. PSWD**__

 Adabas Session Parameters
  Error Action ..........  Halt__   Trigger Table Size ....... __10K
  Log Trigger Activity ... Active__ Pre Trigger Queue Size ... __35K
                                    Post Trigger Queue Size .. __50K


Command ==>
Enter PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12-
      Help  Menu  Exit        Mod                                      Canc
```

Adabas トリガとストアドプロシージャによって、プロファイル内のフィールドにデフォルト値が割り当てられ、これらの値が初期化時に使用されます。この情報は、初期化時（つまり Adabas ニュークリアスの開始時）のみ有効です。プロファイルを修正した場合、新しい値は、ニュークリアスが次回バウンスされるときに有効になります。

手順 8.36. プロファイルを修正する手順

1 ［Display Profile Information］画面で、PF5 キーを押します。［Administrator Functions］メニューで、［Modify Profile Information］を表す「M」を入力します。

［Modify Profile Information］画面が表示されます。

2 新しい値を入力し、PF5 キーを押すか、コマンド行に「UPDATE」と入力します。

プロファイルが正常に更新されたことがメッセージにより通知されます。

プロファイルのフィールド

プロファイルのフィールドについて次の表で説明します。

フィールド	説明	
DBnr	このプロファイルを適用するトリガファイルのデータベース ID。値は、NTFILE、LFILE、または PF10（FTRG）を使用して入力します。	
Trigger Status	このフィールドは変更できます。	
	アクティブ	（デフォルト）このデータベースに対してトリガを実行できます。
	非アクティブ	このデータベースに対してトリガを実行できません。
Stored Proc. Status	このフィールドは変更できます。	

フィールド	説明	
	アクティブ	（デフォルト）このデータベースに対してストアドプロシージャを実行できます。
	非アクティブ	このデータベースに対してストアドプロシージャを実行できません。
Total Triggers	このデータベース ID に対して定義されたトリガの合計数。 デフォルトで、トリガテーブルバッファのサイズを計算するために使用できます。 値は、トリガファイルに追加されたトリガ定義から取得されます。 この値が正しいかどうかは、`NUMBER` コマンドを発行して確認できます。	
Batch Natural Name	プロシージャを実際に実行する 1～10 の Natural サブシステムを起動するために、Adabas トリガドライバによって開始される Natural ニュークリアス。 この名前は、インストール処理中に、Natural ニュークリアスコンポーネントに割り当てられます。	
Maximum Subsystems	このフィールドは変更できます。	
	指定された Adabas セッションでアクティブ化される Natural サブシステムの数。 値は 01～10 です。	
Activity Timeout	このフィールドは変更できます。 タスクがキャンセルされる前の秒数。 デフォルトは 60 で、最大値は 9999 です。 ゼロに設定すると、デフォルトで Adabas TT パラメータ値になります。	
NATPARM Parameters	Natural ニュークリアスにリンクされた NATPARM モジュールのダイナミックなパラメータ上書き（NATPARM モジュールは、Natural セッションで有効になるオプションを指定する）。 「*NATPARM について*」を参照してください。	
Fixed NATPARM	この値は、Adabas トリガとストアドプロシージャによって生成されます。 `STACK=(LOGON:SYSSPT;STP)` SYSSPT はプロシージャが実行されるライブラリです。STP は Natural ドライバを呼び出すために使用されます。	
CMPRINT Assignment	このフィールドは変更できます。 CMPRINT ラベルのダイナミックな割り当て。 デフォルト値は TSPRT です。 Natural 構文で特定のプリンタ番号表記（PRINT(01)、DISPLAY(02)、WRITE(03) など）が使用されない限り、特定のサブシステム内にある任意のプロシージャからの出力は、このラベルに送信される必要があります。	
NATSEC LOGON Required	このフィールドは Natural サブシステムが Natural Security 配下で実行されている場合にのみ適用されます。 Natural Security へのログオンが必要か（Y）必要でないか（N）を示します。 それぞれ `AUTO=OFF`、`AUTO=ON` に対応します。	
ユーザー ID	このフィールドは変更できます。 このフィールドは Natural サブシステムが Natural Security 配下で実行されている場合にのみ適用されます。 これは、Natural Security ユーザー ID です。 デフォルト値は USER** です。 ** は、サブシステムのダイナミックタスク番号（値 01～10）で置き換えられます。 デフォルトでは ** を接尾辞として使用しますが、空白以外の文字が 1 つ以上指定されていれば、ユーザー ID の任意の位置に指定できます。	
パスワード	このフィールドは変更できます。 このフィールドは Natural サブシステムが Natural Security 配下で実行されている場合にのみ適用されます。 これは Natural Security パスワードです。 デフォルト値は PSWD** です。 ** は、サブシステムのダイナミックタスク番号で置き換えられます。 デフォルトでは ** を接尾辞として使用しますが、空白以外の文字が 1 つ以上指定されていれば、ユーザー ID の任意の位置に指定できます。	

フィールド	説明	
Error Action	このフィールドは変更できます。 このフィールドは、システムが復元できなかった処理エラーの後で、Adabasトリガとストアドプロシージャによって実行されるアクションです。	
	停止	（デフォルト）クラスタとクラスタ以外の両方のニュークリアスに対して ADAEND が発行された場合と同様に、Adabas ニュークリアスを停止します。
	拒否	Adabasトリガとストアドプロシージャはアクティブなままにします。ただし、トリガを起動するあらゆるコマンドがレスポンスコード 157 を受け取ります。ニュークリアスの 1 つが "拒否" に切り替わると、その他のニュークリアスもすべて切り替わります。 ただしリフレッシュ時に問題が発生し、エラーアクションが "拒否" の場合、ニュークリアスはシャットダウンします。トリガテーブルが破損または不完全な場合は、コマンドを拒否できません。
	無視	Adabasトリガとストアドプロシージャをシャットダウンします。ただし、Adabas トリガとストアドプロシージャの状態に関わらず、ニュークリアスはアクティブなままにします。 クラスタ環境でニュークリアスの1つが"無視"に切り替わると、その他のニュークリアスもすべて切り替わります。 リフレッシュ時に問題が発生し、エラーアクションが "無視" の場合、すべてのニュークリアスが "無視" に切り替わります。 **注意:** "無視" オプションを指定すると、アプリケーションの整合性に問題が発生する可能性があります。
Log Trigger Activity	このフィールドは変更できます。	
	アクティブ	トリガアクティビティをログに記録します。
	非アクティブ	（デフォルト）トリガアクティビティをログに記録しません。
	トリガやストアドプロシージャが呼び出されると、ユーザーの定義に従って、プロシージャに関する情報が常に出力スプールに出力されます。 この情報は、監査やデバッグに役立ちます。	
Trigger Table Size	このフィールドは変更できます。 これは、トリガテーブルバッファのサイズ（バイト単位）です。 追加のトリガがトリガテーブルにロードされる場合は、バッファサイズを増やす必要がある可能性があります（「*トリガテーブルの更新*」を参照）。 デフォルト値は、トリガファイルに定義されたトリガの数に基づいて計算されます。 「*バッファサイズの計算*」を参照してください。	
Pre-Trigger Queue Size	このフィールドは変更できます。 これは、プレトリガキューバッファのサイズ（バイト単位）です。このバッファは、処理の選択前にプレコマンドトリガを格納します。 デフォルト値が計算されます（「*バッファサイズの計算*」を参照）。	
Post-Trigger Queue Size	このフィールドは変更できます。 これは、ポストトリガキューバッファのサイズ（バイト単位）です。このバッファは、処理の選択前にポストコマンドトリガを格納します。 デフォルト値が計算されます（「*バッファサイズの計算*」を参照）。	

サブシステムアクティビティ

サブシステムアクティビティでは、現在実行中のNaturalサブシステムに関する情報が表示されます。

手順 8.37. サブシステムアクティビティ情報を表示する手順

■ ［Administrator Functions］メニューで、［Subsystem Activity］を表す「S」を入力します。

［Subsystem Activity］画面は、次の例のようになります。

```
HH:MM:SS               - Subsystem Activity -            YYYY-MM-DD

 Nmbr  Started Status Type  Start  / TimeOut of Trigger  Trig Count
 ----  ------- ------ ----- --------------------------- ----------
 01   09:25:57 Busy   Sync  09:41:37 09:46:37                  350
 02   09:25:57 Busy   Async 09:41:39 09:46:39                  280
 03   09:25:57 Wait                                           212
 04   09:25:57 Wait                                           156
 05   09:25:57 Wait                                            51




Subsystems . .  5 of  5
Command ==>
```

［Subsystem Activity］画面の情報について次の表で説明します。

フィールド	説明
Nmbr	Natural サブシステムの番号。
Started	サブシステムが初期化された時刻。 リストされている他のサブシステムと［Started］の時刻が異なる場合、タイムアウト、キャンセル、または終了が発生したことを表します。 この場合、照合を行う必要があります。
Status	サブシステムの現在のステータス。busy（ビジー）、active（アクティブ）、wait（処理を待機中）、shutdown（シャットダウン）、cancelled（キャンセル）、またはabended（アベンド）（エラー発生時）。
Type	トリガのタイプ。sync（同期）または async（非同期）。
Start	プロシージャが実行を開始した時刻。
Timeout	プロシージャが最大タイムリミットを超えて実行される場合にプロシージャがキャンセルされる時間。タイムアウト前にエラーが発生した場合は、エラーが表示されます。
Trig Count	初期化後にサブシステムによって実行されたトリガの数。

フィールド	説明
Command	DBA がアクティブまたは待機中のサブシステムをキャンセルするには、CANCEL コマンドを入力し、次にキャンセルするサブシステムの［Status］にカーソル位置を合わせて「C」と入力します。

手順 8.38. アクティブまたは待機中のサブシステムをキャンセルする手順

1 コマンド行に「CANCEL」と入力します。

2 キャンセルするサブシステムの［Status］フィールドに、「C」（キャンセル）と入力します。

3 終了するには、PF3 キーを押します。

トリガアクティビティ

トリガアクティビティでは、現在実行中のトリガに関する情報が表示されます。

手順 8.39. トリガアクティビティ情報を表示する手順

■ ［Administrator Functions］メニューで、［Trigger Activity］を表す「T」を入力します。

［Current Trigger Activity］画面は、次の例のようになります。

```
HH:MM:SS               ***** TRIGGER MAINTENANCE *****          YYYY-MM-DD
User USR01               - Current Trigger Activity -           DBnr 105

Nmbr Status   Cmd  Fnr  Field  ProcName Type        RecBu   UserID (hex)
---- ------   ---  ---  -----  -------- ----------  -----  -------------
 01  Active                   PROC0002  Pre Non-P  None   ABD6EA375DE96A01
     Waiting  L3   11   **     SYMP0002  Pre Async  None
     Waiting  L3   11   **     SYMP0002  Pre Async  None
     Waiting  L3   11   **     SYMP0002  Pre Async  None
     Waiting  L3   11   **     SYMP0002  Pre Async  None
     Waiting  L3   11   **     SYMP0002  Pre Async  None
     Waiting  L3   11   **     SYMP0002  Pre Async  None
     Waiting  L3   11   **     SYMP0002  Pre Async  None
     Waiting  S4   11   **     SYMP0002  Pre Async  None
     Waiting  N1   11   LE     SYMP0001  Pre Non-P  Upd


Command ==>
Enter PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12-
           Menu  Exit  Sact  Refr  --     -     +          >          Canc
```

［Current Trigger Activity］画面の情報について次の表で説明します。

フィールド	説明
Nmbr	トリガを実行する Natural サブシステムの数。 有効な値の範囲は 01～10 です。
Status	サブシステムの現在のステータス（アクティブまたは待機中）。
Cmd	トリガを開始した Adabas コマンド。
Fnr	コマンドが呼び出されたファイル番号。 このフィールドの値がトリガファイルと同じである場合、これはストアドプロシージャです。
Field	トリガの起動元となったフィールドの名前。
ProcName	トリガされたプロシージャの名前。
Type	トリガのタイプ。pre（プレコマンド）、post（ポストコマンド）、non-P（非関与）、part（関与）、または async（非同期）。
RecBu	トリガの RecBuff 設定。access（読み取り専用）、update（読み書き）、または none（なし）です。
UserID	実際の Adabas コマンドを発行したユーザーの ID（16 進数値）。

手順 8.40. 追加情報を表示する手順

■ PF10 キーを押します。

画面が右にスクロールし、次の図のようになります。

```
HH:MM:SS              ***** TRIGGER MAINTENANCE *****         YYYY-MM-DD
User USR01              - Current Trigger Activity -          DBnr 105

Nmbr Status   Cmd  Fnr  Field  ProcName CID  CID (hex) ISN in ACB Timeout
---- ------   ---  ---  -----  -------- ---  --------- ---------- ---------
 01  Active             **     PROC0002      00000000             11:30:41
     Waiting  L3   11   **     SYMP0002  ??? 00200101
     Waiting  L3   11   **     SYMP0002  ??? 00200101  1
     Waiting  L3   11   **     SYMP0002  ??? 00200101  16
     Waiting  L3   11   **     SYMP0002  ??? 00200101  15
     Waiting  L3   11   **     SYMP0002  ??? 00200101  14
     Waiting  L3   11   **     SYMP0002  ??? 00200101  20
     Waiting  L3   11   **     SYMP0002  ??? 00200101  23
     Waiting  L3   11   **     SYMP0002  ??? 00200101  2
     Waiting  S4   11   **     SYMP0002 ???? 02100101
     Waiting  N1   11   LE     SYMP0001      00000000


Command ==>
Enter PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12-
           Menu  Exit  Sact  Refr  --     -     +           <          Canc
```

フィールド	説明
Nmbr	トリガを実行する Natural サブシステムの数。 有効な値の範囲は 01〜10 です。
Status	サブシステムの現在のステータス（アクティブまたは待機中）。
Cmd	トリガを開始した Adabas コマンド。
Fnr	コマンドが呼び出されたファイル番号。このフィールドの値がトリガファイルと同じである場合、これはストアドプロシージャです。
Field	トリガの起動元となったフィールドの名前。
ProcName	トリガされたプロシージャの名前。
CID	Adabas コントロールブロック（ACB）のコマンド ID。
CID (hex)	16 進数形式で再表示したコマンド ID。
ISN in ACB	Adabas コントロールブロック（ACB）からの ISN。
Timeout	プロシージャが最大タイムリミットを超えて実行される場合にプロシージャがキャンセルされる時間。

9 TRGMAIN: トリガを保守するための API

Adabas トリガとストアドプロシージャ機能には、ユーザー作成のプログラムからトリガ定義を保守するために呼び出し可能なルーチン TRGMAIN が用意されています。 TRGMAIN の機能は次のとおりです。

- 新しいトリガの追加
- 既存のトリガの修正
- 既存のトリガの削除
- 特定のトリガ定義の表示
- トリガのアクティブ化
- トリガの非アクティブ化

手順 9.1. トリガメンテナンスルーチンを呼び出す手順

- 次のように入力します。

```
CALL 'TRGMAIN' function calling-parameters
```

ここで、function（機能）と calling-parameter（呼び出しパラメータ）について、次のセクションで説明します。

このchapterでは、次のトピックについて説明します。

機能（フォーマット A5）

FUNC パラメータは、トリガ定義で実行されるアクションを指定します。 正しい値は次のとおりです。

ADD	追加
MOD	修正
DEL	削除
DISP	表示
ACT	アクティブ化
DEACT	非アクティブ化

コールパラメータ（フォーマット A209）

CALLING-PARMSを構成するパラメータは、アクションの実行に必要な情報を指定します。これらのパラメータは、オンライントリガメンテナンス機能で要求される情報と整合性があり、次のフィールドが含まれます（デフォルト値に下線付き）。

フィールド名	長さ	説明
TRG-DBNR	N5	トリガファイル DBnr
FILE-NAME	A32	FFT で定義されたファイル名
CMD-TYPE	A1	トリガを起動するコマンドタイプ D 削除 F 検索 I 挿入 R 読み込み U 更新 * すべてのコマンド
FIELD-NAME	A32	FFT で定義されたフィールド名
TRIGGER-PROG	A8	Natural トリガプログラムの名前
PRIORITY	N3	トリガのプライオリティ
PRE-CMD-SELECT	A1	コマンドの実行に関するトリガのタイミング N ポストコマンドトリガ Y プレコマンドトリガ
PART-NON-FLAG	A1	関与フラグ。 正しい値は次のとおりです。 A 非同期（デフォルト） N 非関与 P 関与
CALLNAT-FLAG	A1	CALLNAT パラメータ 正しい値は次のとおりです。 C ACBありの制御パラメータとレスポンスコード（デフォルト） E レスポンスコードのみ N 渡されるパラメータなし X ACBXありの制御パラメータとレスポンスコード
RECBUFF-OPTION	A1	レコードバッファアクセス。 正しい値は次のとおりです。 A アクセスオンリー

フィールド名	長さ	説明	
		N	レコードバッファへのアクセスなし（デフォルト）
		U	更新を許可
ACT-TYPE	A4	処理されるアクティブ化／非アクティブ化のタイプ	
		PERM	永続的
		TEMP	一時的
TRG-RSV	A39	予約済み（将来的に使用される予定）	
RESP	N4	API から返されるレスポンスコード	
RETURN-MSG	A68	レスポンスコードのテキストによる説明	
VERSION	N3	GCB のバージョン	
PRODVERN	N3	データベースのバージョン	
NUCID	P5	ニュークリアス ID	

サンプルユーザープログラム

```
     ************************************************************
      Program     UMAINT       Library SYSTRG
0010 ************************************************************
0020 *  Application .. Trigger Maintenance
0030 *  Program ...... UMAINT
0040 *  Function ..... Sample call program to call API TRGMAIN that
0050 *                 maintains trigger definitions.
0060 *                 (Add,Delete,Modify,Display,Activate,Deactivate).
0070 *
0080 *  Parameters ... The following parameters are passed when calling
0090 *                 the API:
0100 *      FUNC           (A5)  Action to perform on Trigger Definition
0110 *                           Valid values are ADD - Add
0120 *                                            MOD - Modify
0130 *                                            DEL - Delete
0140 *                                            DISP - Display
0150 *                                            ACT - Activate
0160 *                                            DEACT - Deactivate
0170 *      TRG-DBNR       (N5)  Trigger file DBnr
0180 *      FILE-NAME     (A32)  File name defined in the FFT
0190 *      CMD-TYPE       (A1)  Command type that causes trigger to fire
0200 *                           Valid values are R - Read
0210 *                                            F - Find
0220 *                                            I - Insert
0230 *                                            U - Update
0240 *                                            D - Delete
0250 *                                            * - All commands
```

```
0260 *      FIELD-NAME    (A32)  Field name defined in the FFT
0270 *      TRIGGER-PROG   (A8)  Name of the Natural trigger program
0280 *      PRIORITY       (N3)  Trigger priority
0290 *      PRE-CMD-SELECT (A1)  Pre-trigger or post-trigger
0300 *                           Valid values are Y - Pre trigger
0310 *                                            N - Post trigger
0320 *      PART-NON-FLAG  (A1)  Participation flag
0330 *                           Valid values are A - Asyncronous
0340 *                                            N - Non-participating
0350 *                                            P - Participating
0360 *      CALLNAT-FLAG   (A1)  CALLNAT parameters
0370 *                           Valid values are N - No parmeters passed
0380 *                                            E - Response code only
0390 *                                            C - Control parms and
0400 *                                                response code
0410 *      RECBUFF-OPTION (A1)  Record buffer access
0420 *                           Valid values are A - Access only
0430 *                                            U - Updates allowed
0440 *                                            N - No access to RB
0450 *      ACT-TYPE       (A4)  Type of activation/deactivation
0460 *                           Valid values are: TEMP - Temporary
0470 *                                             PERM - Permanent
0480 *      RESP           (N4)  Response code returned from the API
0490 *      RETURN-MSG    (A68)  Text description of the response code
0500 ************************************************************************
0510 DEFINE DATA LOCAL
0520 01 FUNC (A5)
0530 01 CALLING-PARMS (A161)
0540 01 REDEFINE CALLING-PARMS
0550   02 TRG-DBNR           (N5)
0560   02 FILE-NAME         (A32)
0570   02 CMD-TYPE           (A1)
0580   02 FIELD-NAME        (A32)
0590   02 TRIGGER-PROGRAM    (A8)
0600   02 PRIORITY           (N3)
0610   02 PRE-CMD-SELECT     (A1)
0620   02 PART-NON-FLAG      (A1)
0630   02 CALLNAT-FLAG       (A1)
0640   02 RECBUFF-OPTION     (A1)
0650   02 ACT-TYPE           (A4)
0660   02 RESP               (N4)
0670   02 RETURN-MSG        (A68)
0680 01 MAP-MSG (A68)
0690 01 HOLD-FUNC (A5)
0700 01 HOLD-PRE-CMD-SELECT     (A1)
0710 01 PAGE-TITLE (A50)
0720 01 #ATTR (C)
0730 END-DEFINE
0740 RESET CALLING-PARMS MAP-MSG
0750 MOVE 233 TO TRG-DBNR
0760 SET KEY PF3
0770 **
```

```
0780 ** Request function and required fields
0790 **
0800 REPEAT
0810   INPUT (AD=TMIL'_' CD=NE)
0820     MAP-MSG (IP=OFF AD=O)
0830     / 9T 'API Maintenance of Trigger Definitions' (YEI)
0840     // 'Function.........' FUNC   30T '(Add, Del, Mod, Disp, or ".")'
0850     // 'File Name........' FILE-NAME
0860     / 'Field Name........' FIELD-NAME
0870     / 'Command Type......' CMD-TYPE 30T '(R, F, I, U, D, or *)'
0880     / 'Pre-Command.......' PRE-CMD-SELECT 30T '(Y, N, or blank)'
0890   RESET MAP-MSG
0900 **
0910 ** Escape out of here
0920 **
0930   IF FUNC = MASK('.') OR *PF-KEY = 'PF3'
0940   ESCAPE BOTTOM
0950 ** Set up Page Titles
0960   DECIDE ON FIRST VALUE OF FUNC
0970     VALUE 'DISP' MOVE 'Display' TO PAGE-TITLE
0980     VALUE 'ADD'  MOVE 'Add' TO PAGE-TITLE
0990     VALUE 'MOD'  MOVE 'Modify' TO PAGE-TITLE
1000     VALUE 'DEL'  MOVE 'Delete' TO PAGE-TITLE
1010     VALUE 'X'    ESCAPE BOTTOM
1020     NONE REINPUT 'Invalid Function Code'
1030   END-DECIDE
1040   COMPRESS PAGE-TITLE 'Trigger Definition' INTO PAGE-TITLE
1050   IF FILE-NAME = ' '
1060   REINPUT 'File Name cannot be BLANK' MARK *FILE-NAME
1070 **
1080 ** Handle Request to Display a Trigger
1090 **
1100   IF FUNC = 'DISP'
1110   DO
1120     PERFORM GET-TRIGGER
1130     INPUT (AD=O CD=NE)
1140       MAP-MSG (IP=OFF AD=O)
1150       / 9T 'API Maintenance of Trigger Definitions' (YEI)
1160       // 10T PAGE-TITLE (IP=OFF)
1170       // 'File name.........' FILE-NAME
1180       / 'Field Name........' FIELD-NAME
1190       / 'Command Type......' CMD-TYPE
1200       // 'Trigger Program...' TRIGGER-PROGRAM
1210       / 'Priority..........' PRIORITY
1220       / 'Pre-Command.......' PRE-CMD-SELECT
1230       / 'Trigger Type......' PART-NON-FLAG
1240       / 'CALLNAT Params....' CALLNAT-FLAG
1250       / 'RecBuffer Access..' RECBUFF-OPTION
1260     RESET FUNC
1270     MOVE RETURN-MSG TO MAP-MSG
1280   DOEND
1290 **
```

```
** Handle Request to Alter Trigger Definitions
**
  IF FUNC = 'ADD' OR= 'MOD' OR= 'DEL'
  DO
    IF (FUNC = 'MOD' OR= 'DEL')
    PERFORM GET-TRIGGER
    IF FUNC = 'DEL'
    DO
      MOVE 'Press ENTER to Delete or PF-3 to Cancel' TO MAP-MSG
      MOVE (AD=P) TO #ATTR
    DOEND
    ELSE DO
      MOVE 'Press ENTER to confirm data or PF-3 to Cancel' TO MAP-MSG
      MOVE (AD=D CD=NE) TO #ATTR
    DOEND
    REPEAT
      INPUT (AD=TMIL'_' CD=NE)
        MAP-MSG (IP=OFF AD=O)
        / 9T 'API Maintenance of Trigger Definitions' (YEI)
        // 10T PAGE-TITLE (AD=O IP=OFF)
        // 'File Name.........' FILE-NAME (AD=O)
        / 'Field Name........' FIELD-NAME (CV=#ATTR)
        / 'Command Type......' CMD-TYPE (CV=#ATTR)
        // 'Trigger Program...' TRIGGER-PROGRAM (CV=#ATTR)
        / 'Priority.........' PRIORITY (CV=#ATTR)
        / 'Pre-Command.......' PRE-CMD-SELECT (CV=#ATTR)
        / 'Trigger Type......' PART-NON-FLAG (CV=#ATTR)
        / 'CALLNAT Params....' CALLNAT-FLAG (CV=#ATTR)
        / 'RecBuffer Access..' RECBUFF-OPTION (CV=#ATTR)
      RESET MAP-MSG
      IF *PF-KEY = 'PF3'
      DO
        MOVE 'Function cancelled per user request' TO MAP-MSG
        ESCAPE BOTTOM
      DOEND
**
** Perform the update of data (add, del, or mod)
** and handle the response
**
      CALLNAT 'TRGMAIN' FUNC CALLING-PARMS
      DECIDE ON FIRST VALUE OF RESP
        VALUE 0  MOVE RETURN-MSG TO MAP-MSG
          EXAMINE MAP-MSG FOR 'confirmed' REPLACE 'successful'
          ESCAPE BOTTOM
        VALUE 20 REINPUT WITH TEXT RETURN-MSG MARK *FIELD-NAME
        VALUE 23 REINPUT WITH TEXT RETURN-MSG MARK *FIELD-NAME
        VALUE 25 REINPUT WITH TEXT RETURN-MSG MARK *CMD-TYPE
*       VALUE 37 REINPUT WITH TEXT RETURN-MSG MARK *PRIORITY
*       VALUE 38 REINPUT WITH TEXT RETURN-MSG MARK *PRIORITY
        VALUE 39 REINPUT WITH TEXT RETURN-MSG MARK *TRIGGER-PROGRAM
        VALUE 40 REINPUT WITH TEXT RETURN-MSG MARK *PRE-CMD-SELECT
        VALUE 41 REINPUT WITH TEXT RETURN-MSG MARK *PART-NON-FLAG
```

```
1820         VALUE 42 REINPUT WITH TEXT RETURN-MSG MARK *CALLNAT-FLAG
1830         VALUE 43 REINPUT WITH TEXT RETURN-MSG MARK *RECBUFF-OPTION
1840         NONE     MOVE RETURN-MSG TO MAP-MSG
1850       END-DECIDE
1860     LOOP
1870     RESET FUNC
1880   DOEND
1890 ** Loop back up to display the starting screen
1900 LOOP
1910 ******************************
1920 ** Subroutine to retrieve the *
1930 ** trigger information        *
1940 ******************************
1950 DEFINE SUBROUTINE GET-TRIGGER
1960 MOVE FUNC TO HOLD-FUNC              /* Go get the existing
1970 MOVE 'DISP' TO FUNC                 /* trigger information
1980 MOVE PRE-CMD-SELECT TO HOLD-PRE-CMD-SELECT
1990 CALLNAT 'TRGMAIN' FUNC CALLING-PARMS
2000 MOVE HOLD-FUNC TO FUNC
2010 IF RESP NE 0
2020 REINPUT WITH TEXT RETURN-MSG
2030 IF RESP = 0
2040 DO
2050   IF PRE-CMD-SELECT NE HOLD-PRE-CMD-SELECT
2060     AND HOLD-PRE-CMD-SELECT NE ' '
2070   DO
2080     MOVE HOLD-PRE-CMD-SELECT TO PRE-CMD-SELECT
2090     MOVE 'Trigger does not exist' TO MAP-MSG
2100     IF HOLD-PRE-CMD-SELECT = 'Y'
2110     COMPRESS 'Pre-' MAP-MSG INTO MAP-MSG
2120     ELSE COMPRESS 'Post-' MAP-MSG INTO MAP-MSG
2130     MOVE HOLD-FUNC TO FUNC
2140     ESCAPE TOP
2150   DOEND
2160   IF CMD-TYPE = ' '
2170   MOVE '*' TO CMD-TYPE
2180 DOEND
2190 RETURN
2200 *** End of Subroutine ***
2210 END
2220 **
```

レスポンスコード

コード	説明
000	機能は正常に終了しました。
013	無効なファイル-フィールドエントリが指定されました。 トリガ機能では、ファイル-フィールドテーブルへのアクセスが必要です。ファイル-フィールドテーブルは、ロングファイル名からファイル番号へ、およびフィールド名から 2 文字の Adabas フィールド ID へのマッピングを行います。 詳細については、「*ファイル- フィールドテーブル*」を参照してください。
016	この条件のトリガ定義は見つかりませんでした。
020	削除コマンドクラスでフィールド名は空白にする必要があります。
023	このファイルのフィールド名は見つかりませんでした。 指定されたフィールド名はこのファイルのファイル-フィールドテーブルに存在しません。 詳細については、「*ファイル- フィールドテーブル*」を参照してください。
025	コマンドの種類が無効です。 有効な値については上記のリストを参照してください。
027	このファイルおよびコマンドの種類のフィールドが見つかりませんでした。
029	このファイル、コマンド、およびフィールドの条件のトリガが見つかりませんでした。
037	プライオリティは 1～900 の間でなければなりません。
038	Adabas フィールド名が ** の場合は、プライオリティを設定できません。
039	Natural サブプログラム名が無効です。
040	プレトリガの場合は "Y"、ポストトリガの場合は "N" を指定してください。
041	トリガタイプは A、N、または P のみです。
042	CALLNAT タイプは C、E、または N のみです。
043	レコードバッファの使用は A、N、または U のみです。
044	無効なパラメータの組み合わせです。 非同期トリガで使用できるレコードバッファがありません。
045	無効なパラメータの組み合わせです。 読み込みまたは検索コマンドのプレトリガで利用可能なレコードバッファがありません。
046	無効なパラメータの組み合わせです。 削除コマンドで利用可能なレコードバッファがありません。
047	このパラメータの組み合わせにトリガはすでに存在します。
048	無効な要求です。 トリガはすでに指定された状態です。
052	トリガ状態を変更することは現在できません。
103	ファイル名を空白にできません。
111	無効な機能コードです。
112	TYPE は TEMP または PERM にする必要があります。 トリガのアクティブ化／非アクティブ化のタイプは、一時（TEMP、このニュークリアスセッション間のみ持続）または永続（PERM）のどちらかにする必要があります。
1xxx	Adabas レスポンス 22 が返されました。*xxx* は、サブコードを表します。

コード	説明
3xxx	ゼロ以外の Adabas レスポンスコードが返されました。*xxx* は実際の Adabas レスポンスコードです。
9999	機能は成功しませんでした。パラメータおよび既存のトリガ定義を確認してください。Software AG 技術サポートに連絡してください。

10 TRGUNLD と TRGLOAD ユーティリティ

Adabas トリガとストアドプロシージャ機能のアンロードとロードユーティリティは、オンライントリガメンテナンス機能の一部であり、Natural 環境で実行されます。

ユーティリティ	使用目的
TRGUNLD	トリガ定義と、関連するファイル - フィールドテーブルエントリを Adabas トリガファイルからアンロードし、それらをワークファイルに書き込みます。
TRGLOAD	トリガ定義と、関連するファイル - フィールドテーブルエントリを TRGUNLD ワークファイルからトリガファイルにロードします。

TRGUNLD のワークファイルは、Natural ワークファイル（ワークファイル 1）であり、次のいずれかで定義されます。

- バッチファイル内
- アンロードユーティリティを実行している Natural 環境

TRGUNLD ワークファイルは、TRGLOAD ユーティリティに対する入力として使用されます。

処理されたトリガについてまとめたレポートがアンロードまたはロードの最後に準備されます。

このchapterでは、次のトピックについて説明します。

ユーティリティの起動

手順 10.1. ユーティリティを呼び出す手順

- ユーティリティの名前（TRGUNLD または TRGLOAD）をコマンドとして発行します。任意で、パラメータリストを続けることもできます。

  パラメータは、処理されるトリガを制限するときに使用します。

  個々のパラメータの値は、入力デリミタ（ID）で区切る必要があります。デフォルトの ID はコンマ "," です。 入力モードパラメータ `IM` は、デリミタモード `IM=D` に設定します。

  ユーティリティをバッチジョブとして実行するときは、選択したユーティリティとそのパラメータリストを実行する前に、バッチジョブが SYSTRG ライブラリにログオンする必要があります。

TRGUNLD

TRGULND ユーティリティがオンラインシステムからパラメータリストなしでコマンド行から呼び出されると、次のウィンドウが表示されます。

```
17:51:42          ***** A D A B A S TRIGGER MAINTENANCE *****       YYYY-MM-DD
User
     |-----------------------------------------------------------------------|
     | 17:51:46  **** A D A B A S TRIGGER MAINTENANCE ****   YYYY-MM-DD   |
     | -----------------  Trigger  Unload  Utility  -------------------   |
     |                                                                       |
     |       Specify the following information to identify                  |
     |       the Triggers to be unloaded to Work File 1:                    |
     |                                                                       |
     |        File Name ..... _____________________________________         |
     |        Field ......... _____________________________________         |
     |        Pre-triggers .. ____                                          |
     |        Command Type .. __                                            |
     |        Active State .. _                                             |
     |        Trigger Type .. _                                             |
     |                                                                       |
     |       Use 'PF3' to cancel or hit  Enter' when ready                  |
     |-----------------------------------------------------------------------|


Command ==> trgunld
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Field Trigr Admin Procs             FTRG  FDIC  Canc
```

必要に応じてパラメータの値を入力し、アンロードするトリガ定義を制限します。 パラメータが空白のままである場合は、デフォルト値を使用すると想定されます。

TRGUNLD でパラメータが指定されない場合、トリガファイルのすべてのトリガと、関連するファイル - フィールドテーブルエントリがアンロードされます。

TRGLOAD

TRGLOAD ユーティリティがオンラインシステムからパラメータリストなしでコマンド行から呼び出されると、次のウィンドウが表示されます。

```
18:13:43          ***** A D A B A S TRIGGER MAINTENANCE *****       YYYY-MM-DD
User DBAU |-----------------------------------------------------------|nr 105
          |                                                           |
          |  ---------------  Trigger  Load  Utility  -------------  |
          |                                                           |
          |     Specify the following information to identify        |
          |     the Triggers to be loaded from Work File 1:          |
          |                                                           |
          |        File ...... _____________________________________  |
```

```
          |        Field ..... _____________________________________     |
          |        Replace ... ___                                      |
          |        With FFT .. ___                                      |
          |                                                             |
          |     Use 'PF3' to cancel or hit  Enter' when ready           |
          |                                                             |
          |-------------------------------------------------------------|


Command ==> trgload
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Field Trigr Admin Procs             FTRG  FDIC  Canc
```

TRGLOAD でパラメータが指定されない場合、TRGUNLD ワークファイルで見つかったすべてのトリガおよび関連するファイル-フィールドテーブルエントリがトリガファイルにロードされます。

ユーティリティのパラメータ

ワイルドカード表記

各ユーティリティでは、以降のセクションで説明する FILE と FIELD パラメータで、指定する値に次のワイルドカード（特殊文字）表記を使用できます。

文字	例	トリガをロードするファイルまたはフィールドの名前
*	PERS*	PERS で始まる名前
>	PERS>	PERS よりも大きい値が含まれる名前
<	PERS<	PERS よりも小さい値が含まれる名前

TRGUNLD パラメータ

次のパラメータを使用して、TRGUNLD の処理中にトリガファイルからワークファイルにアンロードされるトリガを制限できます（各パラメータのデフォルト値は "*" で、すべての可能性のあるトリガが含まれます）。

パラメータ	説明
FILE	トリガ定義で見つかったファイルの名前。 正しい値は次のとおりです。 ファイル名（ワイルドカード表記が可能） * 全ファイル（デフォルト）
FIELD	トリガ定義にあるフィールド名。 正しい値は次のとおりです。 フィールド名（ワイルドカード表記は可能）

パラメータ	説明	
	*	全フィールド（デフォルト）
PRE	プレトリガまたはポストトリガを指定します。 正しい値は次のとおりです。	
	Y	プレトリガ
	N	ポストトリガ
	*	両方
CMD	トリガを起動するコマンドタイプ。 正しい値は次のとおりです。	
	R	読み込み
	F	検索
	I	挿入
	U	更新
	D	削除
	*	すべてのコマンドタイプ（デフォルト）
ACT	処理されるトリガのステータス。 正しい値は次のとおりです。	
	A	アクティブ
	D	アクティブでない（非アクティブ）
	*	両方
TYPE	ユーザーの ET ロジックでのトリガの関与タイプ。 正しい値は次のとおりです。	
	A	非同期
	N	非関与
	P	関与
	*	すべての関与タイプ

TRGUNLD パラメータの例

```
TRGUNLD FILE=EMPLOYEES
```

FILE パラメータを指定すると、アンロードされるトリガが指定されたファイル（この場合は EMPLOYEES ファイル）で定義されたトリガに制限されます。

```
TRGUNLD FILE=VEHICLES,PRE=Y,ACT=A,TYPE=N
```

パラメータを組み合わせると、アンロードされるトリガを細かく制限できます。この場合では、VEHICLESファイルに定義されたアクティブで非関与のプレトリガだけがアンロードされます。パラメータは、Natural環境のIDパラメータで定義した入力デリミタで区切る必要があります。

TRGLOAD パラメータ

次のパラメータを使用して、TRGLOAD の処理中に TRGUNLD ワークファイルからトリガファイルにロードされるトリガを制限できます。

パラメータ	説明
FILE	トリガ定義で見つかったファイルの名前。 正しい値は次のとおりです。 ファイル名（ワイルドカード表記が可能） * 全ファイル（デフォルト）
FIELD	トリガ定義にあるフィールド名。 正しい値は次のとおりです。 フィールド名（ワイルドカード表記は可能） * 全フィールド（デフォルト）
FFT	TRGUNLD 入力データセットで見つかったファイル - フィールドテーブルエントリをトリガでロードするように要求します。 正しい値は次のとおりです。 Y はい（デフォルト） N ×
REPLACE	データベースにすでに存在するロードされているトリガ定義で古いトリガ定義を置換するように要求します。 正しい値は次のとおりです。 Y はい。既存の定義を新しい定義で置き換えます。 N いいえ（デフォルト）。既存の定義を置き換えず、エラーメッセージを返します

TRGLOAD パラメータの例

```
TRGLOAD FILE=EMP*
```

EMP で始まるファイルについて、TRGUNLD 入力ワークファイルで見つかったすべてのファイル - フィールドテーブルエントリとトリガをトリガファイルにロードします。

```
TRGLOAD FFT=N,REPLACE=Y
```

ワークファイルから読み込まれたすべてのファイル - フィールドテーブルを無視します。 ワークファイルから読み込まれたトリガ定義に既存のトリガ定義と同じ指定がされている場合は、既存の定義をワークファイルからの定義で置き換えます。

処理終了レポート

アンロードレポート（TRGUNLD）

アンロード処理の終了時にレポートが作成され、アンロードされたトリガ定義の数とファイル - フィールドテーブルの数が通知されます。

例

レポートの1ページ目には、トリガをアンロードするソースファイルとデータベースが示され、選択条件の概要が表示されます。 その後、ワークファイルにアンロードされている各レコードのリストが続きます。

```
18:12:00            *****  TRIGGERS UNLOAD UTILITY *****            YYYY-MM-DD
DBAUSER
Unloading from Trigger File     12 on Database    105
Rec  File  Details
---  ----- ------------------------------------------------------------
Unloading Pre and Post Triggers for file(s) *
and field name(s) * for all command types

FFT     45 ZB-FIELD (ZB,2,A)
FFT     45 ZF-FIELD (ZF,2,A)
FFT     45 ZZ-FIELD (ZZ,4,P)
TRG     45 CMD=R FLD=ZB PRTY=01 PGM=RBEGIMU PRE=S TYP=P PRM=C RB=U
GRP      4 Group Record
FFT      4 ADDRESS-LINE (AI,20,A)
FFT      4 AREA-CODE (AN,6,A)
FFT      4 BIRTH (AH,6,U)
.
.
.
```

レポートの最後には、ワークファイルに書き込まれたレコード数の概要がリストされます。 書き込まれた物理ワークファイルには、これらのレコードと、内部使用された追加レコード2つが含まれます。

```
18:13:04           *****  TRIGGERS UNLOAD UTILITY *****           YYYY-MM-DD
 DBAUSER
 Unloading from Trigger File     12 on Database    105
 Rec  File  Details
 ---  ----- ------------------------------------------------------------
 FFT      7 NA-FIELD (NA,40,A)
 FFT      7 NT-SUPER (NT,110,A)
 FFT      7 TI-FIELD (TI,70,A)
 FFT      7 TY-FIELD (TY,10,A)
 TRG      7 CMD=R FLD=** PRTY=90 PGM=NACNN200 PRE=S TYP=A PRM=C RB=N

 Total records written to Work File 1

 File-Field Table Entries: ...    141
 Group Control Records: ......      2
 Trigger Definitions: ........     13

 Total Records: ..............    156

 ***  TRGUNLD completed successfully.  ***
```

レポートの最後のページでは、アンロードされたトリガをカテゴリ別に要約します。このレポートは、TRGLOAD ユーティリティで照合を行うときに便利です。

```
**** Trigger Unload Statistics ****

Number of Triggers Unloaded by Categories

Pre or Post :           Pre:      7
                       Post:      6

Trigger Type:  Asynchronous:      6
              Participating:      3
          Non-Participating:      4

Command Type:          READ:      6
                       FIND:      0
                     INSERT:      0
                     DELETE:      2
                     UPDATE:      1
                      *ANY*:      4

Field Criteria:       *ANY*:      4
                    Specific:     9
```

ロードレポート（TRGLOAD）

レポートがロード処理の終了時に作成され、ロードされたトリガ定義の数とファイル-フィールドテーブルの数が通知されます。

例

レポートの 1 ページ目には、トリガをロードするデータベース、ワークファイルが作成されたソースデータベース、および選択条件の概要が表示されます。 その後、ワークファイルにロードされている各レコードのリストが続きます。

```
20:05:05            *****  TRIGGERS LOAD UTILITY  *****           YYYY-MM-DD
DBAUSER
Rec  File  Details
---  ----- ------------------------------------------------------------

Loading Trigger Definitions into database    106 file     14
from data set unloaded from database    105 file     12
created on 1999-07-26 20:04 from version  711

FFT     50 AA-FIELD (AA,8,A)
FFT     50 AC-FIELD (AC,20,A)
FFT     50 AD-FIELD (AD,20,A)
FFT     50 AE-FIELD (AE,20,A)
FFT     50 AF-FIELD (AF,1,A)
FFT     50 AG-FIELD (AG,1,A)
FFT     50 AH-FIELD (AH,6,U)
FFT     50 AI-FIELD (AI,20,A)
FFT     50 AJ-FIELD (AJ,20,A)
FFT     50 AK-FIELD (AK,10,A)
.
.
.
```

レポートの最後は、ワークファイルから読み込まれたレコード数の集計と、データベースにロードされた方法です。

```
20:08:43            *****  TRIGGERS LOAD UTILITY  *****           YYYY-MM-DD
 DBAUSER
 Rec  File  Details
 ---  ----- ------------------------------------------------------------
 TRG      5 CMD=R FLD=AA PGM=CAROL PRE=P TYP=A PRM=C RB=N

 Total records loaded from Work File 1

 File-Field Table Entries: ...    109
 Group Control Records: ......      1
 Trigger Definitions: ........     65
```

```
 Total Records Loaded: .......    175
 Records found with errors ...      0
 Total Records Read: .........    178

 ***  TRGLOAD completed successfully.  ***
```

レポートの最後のページでは、ロードされたトリガをカテゴリ別に要約します。 このレポートは、TRGUNLD ユーティリティで照合を行うときに便利です。

```
******* Trigger Load Statistics *******

Number of Triggers Loaded by Categories

Pre or Post :           Pre:     34
                       Post:     31

Trigger Type:  Asynchronous:     23
              Participating:     32
          Non-Participating:     10

Command Type:          READ:     25
                       FIND:      6
                     INSERT:     14
                     DELETE:      2
                     UPDATE:      3
                      *ANY*:     15

Field Criteria:       *ANY*:     19
                    Specific:     46
```

ユーティリティのレスポンスコード

コード	説明
000	機能は正常に終了しました。
013	無効なファイル-フィールドテーブルエントリが指定されました。 トリガでは、ファイル-フィールドテーブルへのアクセスが必要です。 ファイル - フィールドテーブルは、ロングファイル名からファイル番号へ、およびフィールド名から 2 文字の Adabas フィールド ID へのマッピングを行います。 詳細については、「*ファイル - フィールドテーブル*」を参照してください。
016	この条件のトリガ定義は見つかりませんでした。
023	このファイルのフィールド名は見つかりませんでした。 指定されたフィールド名はこのファイルのファイル - フィールドテーブルに存在しません。 詳細については、「*ファイル - フィールドテーブル*」を参照してください。
025	コマンドの種類が無効です。 有効な値については上記のリストを参照してください。
027	このファイルおよびコマンドの種類のフィールドが見つかりませんでした。

コード	説明
029	このファイル、コマンド、およびフィールドの条件のトリガが見つかりませんでした。
039	Natural サブプログラム名が無効です。
041	トリガタイプは A、N、または P のみです。
042	CALLNAT タイプは C、E、または N のみです。
043	レコードバッファの使用は A、N、または U のみです。
047	トリガはこの条件ですでに存在します。
103	ファイル名を空白にできません。
110	ファイルでファイル - フィールドテーブルエントリはすでに存在します。
1xxx	Adabas レスポンスコード 22 が返されました。xxx は、サブコードを表します。
3xxx	ゼロ以外の Adabas レスポンスコードが返されました。xxx は実際の Adabas レスポンスコードです。
9999	機能は成功しませんでした。 パラメータおよび既存のトリガ定義を確認してください。 詳細については、Software AG 技術サポートに連絡してください。

A 例

このdocumentでは、次の表で示すサンプルのプログラムとデータエリアについて説明します。ソースコードはインストール中に提供され、SYSSPT ライブラリに配置されます。

ストアドプロシージャのリンクルーチン STPLNK01、STPLNK02、および STPLNK03 については、「例」を参照してください。

名前	説明
SAMPINT1	このプログラムは、トリガを起動するコマンドを発行し、特定のファイルに対するレコードの更新または格納が書き込まれると、追加処理を実行します。
SAMPPRC1	SAMPINT1のストアドプロシージャバージョンです。 データはパラメータとして渡され、このパラメータから抽出された情報が返されます。
SAMPP001	SAMPPRC1 から呼び出されるストアドプロシージャです。 入力パラメータからの情報を使用して、コール元に返す追加情報を抽出して作成します。
STPLCB	STPLNK02 や STPLNK03 などのサンプルルーチンで、ローカルデータエリア（LDA）として使用される Adabas ACB コントロールブロックレイアウトです。
STPLCBE	ローカルデータエリア（LDA）として使用される Adabas ACBX コントロールブロックレイアウトです。
STPLRBE	レコードバッファ抽出ルーチン（STPRBE）が呼び出されるときに使用する必要のあるパラメータデータエリア（PDA）のサンプルレイアウトです。
STPUTPRM	サブプログラム STPUTRAK に渡されるパラメータデータエリアの例です。
STPUTRAK	プロシージャを実行する要求を受け取ったときに、Natural トリガドライバが呼び出すルーチンです。 このルーチンは、トリガメンテナンスプロファイルでトリガアクティビティの記録オプションが active に設定されている場合のみ呼び出されます。 このルーチンの名前は変更しないでください。
STPAPARM	プロシージャが呼び出されるときに、Natural トリガドライバから渡されるパラメータデータエリア（PDA）のサンプルレイアウトです。
STPAPRM1	プロシージャが呼び出されていて、STPUTRAKが拡張情報も渡すことを要求したときに、Natural トリガドライバから渡されるパラメータデータエリア（PDA）のサンプルレイアウトです。 拡張エリアの長さは、250 バイトです。 トリガドライバのグローバルデータエリア（GDA）の一部として保守され、トリガドライバによるプロシージャの呼び出し間で変更されません。
SAMP0001	SAMPINT1から発行されるコマンドで起動するトリガの結果として呼び出されるプロシージャです。
SAMP0002	CONTACTS ファイルに対して発行されるすべてのコマンドで呼び出されるプロシージャです。CONTACTS ファイルは、SAMPINT1 プログラムのサンプルファイルとして使用されます。 このプロシージャは、すべてのコマンドを監査するために、メッセージを CMPRINT に書き込み、監査ファイルにログレコードを作成します。
SAMP0003	SAMP0002 と同様に、このプロシージャは、CONTACTS ファイルに対して発行されるコマンドを監査します。ただし監査対象は、関係のあるコマンドのみです。
SAMP0004	VEHICLES ファイルに対して行われるあらゆる削除で、SAMPREF1 ルーチンが実行されるときに呼び出されるプロシージャです。 参照ロジックタイプがRestrictであるサンプルアプリケーションです。

名前	説明
SAMP0005	EMPLOYEES ファイルに対する更新で、SAMPREF2 ルーチンが実行されるときに呼び出されるプロシージャです。 これは、参照保全タイプが Cascade の場合の例です。
SAMPREF1	このルーチンは、VEHICLES ファイルの参照保全性の検証を実行するトリガを呼び出すコマンドを発行します。 関連するプロシージャは SAMP0004 です。このプロシージャは、削除されるレコードで制限チェックを実行します。
SAMPREF2	このルーチンは、EMPLOYEES ファイルでプライマリキーに対する更新の参照保全検証を実行するトリガを呼び出すコマンドを発行します。関連するプロシージャはSAMP0005です。このプロシージャは、VEHICLES と MISCELLANEOUS ファイルの外部キーのプライマリキーに対するあらゆる更新をカスケードします。

SAMPINT1

```
0010 ************************************************************************
0020 *  Application: Adabas Triggers
0030 *      Program: SAMPINT1
0040 *     Function: Routine to add names to file (CONTACTS). A trigger
0050 *               will be established to perform additional processing
0060 *               whenever the name is updated or added.
0070 * Trigger Defn: The definition on the trigger file (one for an update
0080 *               and one for the STORE/INSERT) is as follows:
0090 *                 File Number ....... 11
0100 *                 File Name ......... CONTACTS
0110 *                 Command Type ...... Update + Insert
0120 *                 Long Field Name ... CONTACT-NAME
0130 *                 Adabas Field ...... LE
0140 *                 Field Prty/Seq .... 10_
0150 *               Procedure Information
0160 *                 Name (Subpgm)...... SAMP0001
0170 *                 Pre Cmd Select .... Y  (Pre)
0180 *                 Trigger Type ...... P  (Participating)
0190 *                 CALLNAT Params .... C  (Cntl Info + Resp)
0200 *                 RecBuffer Access .. U  (May be Updated)
0210 *
0220 *               NOTE: An additional trigger also exists for this file
0230 *                     whereby any commands to the file will result in a
0240 *                     trigger being fired. This definition is:
0250 *
0260 *                 File Number ....... 11
0270 *                 File Name ......... CONTACTS
0280 *                 Command Type ...... ** All Command **
0290 *                 Long Field Name ... ** Any Field **
0300 *                 Adabas Field ...... **
0310 *                 Field Prty/Seq .... ___
0320 *               Procedure Information
0330 *                 Name (Subpgm)...... SAMP0002
```

```
0340 *                 Pre Cmd Select .... Y  (Pre)
0350 *                 Trigger Type ...... A  (Asynchronous)
0360 *                 CALLNAT Params .... C  (Cntl Info + Resp)
0370 *                 RecBuffer Access .. N  (No RecBuff Access)
0380 *
0390 *       Author: Adabas Development
0400 *         Date: December 1995
0410 ************************************************************************
0420 DEFINE DATA LOCAL
0430  01  #NAME     (A60)
0440  01  RESP       (N4)
0450  01  CONTACTS VIEW OF CONTACTS    /* file 11 for this example
0460    02  CONTACT-NAME               /* field LE,A,60,NU,DE
0470    02  CONTACT-UPPER              /* field LO,A,60,NU,DE
0480    02  KEYWORDS (20)              /* field LM,A,20,NU,MU
0490 END-DEFINE
0500 *
0510 REPEAT
0520 *
0530 INPUT (AD=WMIL'_' CD=NE)
0540      'Trigger Example for Data Consistancy' (YEI)
0550   // 'Name ...' (TU) #NAME
0560 *
0570 IF #NAME = MASK('.')              /* exit?
0580   STOP                            /* yes
0590 IF #NAME = ' '                    /* name must be specified
0600   REINPUT 'Invalid Name specified'
0610 *
0620 FIND CONTACTS WITH CONTACT-NAME = #NAME /* Find the name
0630   IF NO RECORDS FOUND             /* does it exist?
0640     DO                            /* no, so we should add it
0650 *
0660 *     Although only the CONTACT-NAME is being added, the other fields to
0670 *     be used in the Store are included. In this way, the format buffer
0680 *     and record buffer have reference to these files, since this example
0690 *     results in a pre-trigger that sets the values in these fields.
0700 *     Of course a post-trigger would also have worked; however, it would
0710 *     have been necessary for the procedure to do an additional read and
0720 *     update of the record. In this example, better performance is
0730 *     achieved with a pre-trigger.
0740 *
0750      RESET CONTACT-UPPER KEYWORDS(*)
0760      MOVE #NAME TO CONTACT-NAME    /* move value into view
0770 *
0780      STORE CONTACTS                /* insert the new record on the file
0790      END TRANSACTION               /* and commit the transaction
0800      IF *ISN(0780) = 0             /* we can check that a new ISN exists
0810        DO
0820          WRITE 'Store was unsuccessful' *ISN(0780)
0830          ESCAPE BOTTOM
0840        DOEND
0850      GET CONTACTS *ISN(0780)       /* now we refresh the record buffer
```

```
0860      INPUT (AD=O CD=TU)            /* and show the results of the Store
0870            '*** Results ***' (YEI)
0880          / 'Name ......' (GRI) CONTACT-NAME
0890          / 'Upper .....' (GR)  CONTACT-UPPER
0900          / 'Keywords ..' (GR) KEYWORDS (1:3)
0910          / '           ' KEYWORDS (4:6)
0920          / '           ' KEYWORDS (7:9)
0930          / '           ' KEYWORDS (10:12)
0940          / '           ' KEYWORDS (13:15)
0950          / '           ' KEYWORDS (16:18)
0960          / '           ' KEYWORDS (19:20)
0970      ESCAPE BOTTOM                 /* and exit the Add Record logic
0980    DOEND
0990  SET KEY PF3 PF5                  /* activate a couple of PF-keys
1000  INPUT (AD=O CD=TU)               /* allow the name to be changed
1010            '*** Make required Changes and PRESS PF5 to Update:' (YEI)
1020          / 'Name ......' (TU) CONTACT-NAME (AD=WMIL'_' CD=NE)
1030          / 'Upper .....' (TU) CONTACT-UPPER
1040          / 'Keywords ..' (TU) KEYWORDS (1:3)
1050          / '           ' KEYWORDS (4:6)
1060          / '           ' KEYWORDS (7:9)
1070          / '           ' KEYWORDS (10:12)
1080          / '           ' KEYWORDS (13:15)
1090          / '           ' KEYWORDS (16:18)
1100          / '           ' KEYWORDS (19:20)
1110    IF *PF-KEY = 'PF5'
1120      DO
1130        UPDATE(0620)                /* do the modification
1140        GET CONTACTS *ISN(0620)     /* now we refresh the record buffer
1150        INPUT (AD=O CD=TU)          /* and show the results of the update
1160              '*** Results of the Update ***' (YEI)
1170            / 'Name ......' (GRI) CONTACT-NAME
1180            / 'Upper .....' (GR)  CONTACT-UPPER
1190            / 'Keywords ..' (GR) KEYWORDS (1:3)
1200            / '           ' KEYWORDS (4:6)
1210            / '           ' KEYWORDS (7:9)
1220            / '           ' KEYWORDS (10:12)
1230            / '           ' KEYWORDS (13:15)
1240            / '           ' KEYWORDS (16:18)
1250            / '           ' KEYWORDS (19:20)
1260        MOVE CONTACT-NAME TO #NAME /* and reset the name
1270        ESCAPE BOTTOM               /* and exit the Update Record logic
1280      DOEND
1290  ESCAPE BOTTOM                    /* no update done
1300 CLOSE LOOP(0620)
1310 END TRANSACTION                   /* my job is to confirm and release
1320 CLOSE LOOP(0510)
```

```
1330 *
1340 END
```

SAMPPRC1

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *      Program: SAMPPRC1
0040 *     Function: Routine to input a name and then invoke a stored
0050 *               procedure to populate additional fields based on the
0060 *               name passed.
0070 *
0080 *       Author: Adabas Development
0090 *         Date: December 1995
0100 ************************************************************************
0110 DEFINE DATA LOCAL
0120  01  #NAME     (A60)
0130  01  RESP       (N4)
0140  01  CONTACTS-INFORMATION         /* could be a file
0150    02  CONTACT-NAME  (A60)
0160    02  REDEFINE CONTACT-NAME
0170      03  PARM1     (A1/60)
0180    02  CONTACT-UPPER (A60)
0190    02  REDEFINE CONTACT-UPPER
0200      03  PARM2     (A1/60)
0210    02  KEYWORDS (A20/1:20)
0220    02  REDEFINE KEYWORDS
0230      03  PARM3    (A1/400)
0240  01  LINK-ROUTINE-PARMS           /* parameters for the link routine
0250    02  P-FUNC         (A1)
0260    02  P-PROC         (A8)        /* SAMP0001
0270    02  P-OPTIONS      (A8)        /* 'PCU'
0280    02  P-LEN        (P3/5)        /* lengths for the parameters
0290    02  P-MSG         (A72)        /* response message
0300    02  P-RESP         (N4)        /* response code
0310 END-DEFINE
0320 *
0330 MOVE '2'        TO P-FUNC         /* function....not relevant for this
0340 MOVE 'SAMPP001' TO P-PROC         /* procedure name
0350 MOVE 'NCU'      TO P-OPTIONS      /* non-partic + ctrl parms + upd RB
0360 MOVE 60         TO P-LEN(1) P-LEN(2)
0370 MOVE 200        TO P-LEN(3)
0380 MOVE 100        TO P-LEN(4) P-LEN(5)
0390 RESET P-MSG P-RESP
0400 *
0410 REPEAT
0420 *
0430 * In this example, the routine prompts the end user for an organization
```

```
0440 * name and in response, extracts some keywords from the value.
0450 * This is similar to SAMPINT1 (except that no file is being used) and
0460 * is a possible alternative to it.
0470 *
0480 INPUT (AD=WMIL'_' CD=NE)
0490 'Stored Procedure Example for Data Consistency' (YEI)
0500 // 'Name ...' (TU) #NAME
0510 *
0520 IF #NAME = MASK('.') /* exit?
0530 STOP /* yes
0540 IF #NAME = ' ' /* name must be specified
0550 REINPUT 'Invalid Name specified'
0560 *
0570 RESET CONTACT-UPPER KEYWORDS(*)
0580 MOVE #NAME TO CONTACT-NAME /* move value into view
0590 *
0600 CALLNAT 'STPLNK03' P-FUNC P-PROC P-OPTIONS P-LEN(1) PARM1(1:60)
0610 P-LEN(2) PARM2(1:60) P-LEN(3) PARM3(1:200)
0620 P-LEN(4) PARM3(201:300) P-LEN(5) PARM3(301:400)
0630 P-MSG P-RESP
0640 *
0650 INPUT (AD=O CD=TU) /* and show the results of the Store
0660 '*** Results ***' (YEI)
0670 / 'Name ......' (GRI) CONTACT-NAME
0680 / 'Upper .....' (GR) CONTACT-UPPER
0690 / 'Keywords ..' (GR) KEYWORDS (1:3)
0700 / ' ' KEYWORDS (4:6)
0710 / ' ' KEYWORDS (7:9)
0720 / ' ' KEYWORDS (10:12)
0730 / ' ' KEYWORDS (13:15)
0740 / ' ' KEYWORDS (16:18)
0750 / ' ' KEYWORDS (19:20)
0760 *
0770 CLOSE LOOP(0410)
0780 *
0790 END
```

SAMPP001

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Subprogram : SAMPP001
0040 *  Author     : Adabas Development
0050 *  Date       : August 1995
0060 *  Function   : Sample routine of processing by a procedure
0070 *  Remarks    : This routine converts a name into uppercase andextracts
0080 *               all the keywords associated with it. Once processing is
```

```
0090 *               completed, control is returned to the caller.
0100 *
0110 *               Parameter RESP must be set to zero if processing is
0120 *               successful.
0130 *
0140 *  Parameters : Name1        (A60)
0150 *               Name2        (A60)
0160 *               Keyword(A20/01:10)
0170 *               Keyword(A20/11:15)
0180 *               Keyword(A20/16:20)
0190 *
0200 *  Rec Buffer : The record buffer will be available for update via a
0210 *               CALL to the external routine STPRBE.
0220 *
0230 ************************************************************************
0240 DEFINE DATA PARAMETER USING STPAPARM
0250            LOCAL     USING STPLRBE      /* parms for the Call routine
0260            LOCAL
0270  01  REC-BUFFER(A20/1:26)               /* max rec buffer passed to STPRBE
0280  01  REDEFINE REC-BUFFER                /* redefine this to get the def.
0290    02  INPUT-NAME   (A60)
0300    02  OUTPUT-NAME  (A60)
0310    02  KEYWORDS(A20/1:20)
0320  01  FUNC            (A4)
0330  01  SUB             (I2)
0340  01  SUB1            (I2)
0350  01  SUB2            (I2)
0360  01  W-UPPER        (A61)
0370  01  REDEFINE W-UPPER
0380    02  #UPPER       (A60)
0390    02  REDEFINE #UPPER
0400      03  CHAR   (A1/1:60)
0410  01  #KEYS     (A40/1:20)
0420 END-DEFINE
0430 *
0440 *  Option below is to audit any procedure activity.
0450 *
0460 * CALLNAT 'SAMP0002' REQ-AREA RESP
0470 *
0480 * Since the record buffer information is available to us, we can
0490 * now call the record buffer extraction routine (STPRBE) to obtain
0500 * the contents of the buffer.
0510 *
0520 * Function 'GR' -- GET RB Value using RB offset + length
0530 *     This enables the caller to obtain information based on a
0540 *     certain location; hence, RBE-OFFSET specifies the start
0550 *     position, and RBE-LENGTH specifies the length.
0560 *
0570 MOVE 1   TO RBE-OFFSET               /* start at the beginning
0580 MOVE 520 TO RBE-LENGTH               /* for a max length of 520 bytes
0590 MOVE 'GR' TO FUNC
0600 CALL 'STPRBE' 'GR' RBE-AREA REC-BUFFER(1)
```

```
0610 IF RBE-RESP NE 0
0620   PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
0630        RBE-ERROR 'subcode' RBE-SUBCODE 'for func GR'
0640   MOVE RBE-RESP TO RESP
0650   ESCAPE ROUTINE
0660 END-IF
0670 * PERFORM PRINT-REC-BUFFER            /* option to print the parms
0680 *
0690 * Change all lowercase to UPPERcase
0700 *
0710 MOVE INPUT-NAME TO #UPPER
0720 *
0730 EXAMINE #UPPER AND TRANSLATE INTO UPPER CASE
0740 *
0750 MOVE #UPPER TO OUTPUT-NAME            /* save the uppercase name
0760 *
0770 FOR SUB 1 60                          /* loop to remove all special chars.
0780   IF CHAR(SUB) = MASK(S)
0790     MOVE ' ' TO CHAR(SUB)
0800     ESCAPE TOP
0810   END-IF
0820 END-FOR
0830 *
0840 * We are now ready to extract keywords from our name. This sample is
0850 * very basic and may be made as complex as required.
0860 * This routine assumes a max. length of 20 and a max. num. of 20 keywords
0870 *
0880 EXAMINE FULL W-UPPER FOR FULL ' A '   REPLACE ' '
0890 EXAMINE FULL W-UPPER FOR FULL ' AND ' REPLACE ' '
0900 EXAMINE FULL W-UPPER FOR FULL ' AS '  REPLACE ' '
0910 EXAMINE FULL W-UPPER FOR FULL ' AT '  REPLACE ' '
0920 EXAMINE FULL W-UPPER FOR FULL ' ARE ' REPLACE ' '
0930 EXAMINE FULL W-UPPER FOR FULL ' BE '  REPLACE ' '
0940 EXAMINE FULL W-UPPER FOR FULL ' DO '  REPLACE ' '
0950 EXAMINE FULL W-UPPER FOR FULL ' FOR ' REPLACE ' '
0960 EXAMINE FULL W-UPPER FOR FULL ' HERE ' REPLACE ' '
0970 EXAMINE FULL W-UPPER FOR FULL ' IF '  REPLACE ' '
0980 EXAMINE FULL W-UPPER FOR FULL ' IN '  REPLACE ' '
0990 EXAMINE FULL W-UPPER FOR FULL ' IS '  REPLACE ' '
1000 EXAMINE FULL W-UPPER FOR FULL ' IT '  REPLACE ' '
1010 EXAMINE FULL W-UPPER FOR FULL ' OF '  REPLACE ' '
1020 EXAMINE FULL W-UPPER FOR FULL ' ON '  REPLACE ' '
1030 EXAMINE FULL W-UPPER FOR FULL ' OR '  REPLACE ' '
1040 EXAMINE FULL W-UPPER FOR FULL ' TO '  REPLACE ' '
1050 EXAMINE FULL W-UPPER FOR FULL ' THE ' REPLACE ' '
1060 EXAMINE FULL W-UPPER FOR FULL ' TOO ' REPLACE ' '
1070 EXAMINE FULL W-UPPER FOR FULL ' WAS ' REPLACE ' '
1080 EXAMINE FULL W-UPPER FOR FULL ' WITH ' REPLACE ' '
1090 EXAMINE #UPPER FOR FULL ' ' REPLACE ',' /* put delimiters in the string
1100 *
1110 RESET KEYWORDS(*)
1120 STACK TOP DATA #UPPER                   /* now we will separate each word
```

```
1130 INPUT (AD=I IP=ON) #KEYS(01:03) / #KEYS(04:06) / #KEYS(07:09)
1140                  / #KEYS(10:12) /* #KEYS(13:15) / #KEYS(16:18)
1150                  / #KEYS(19:20)
1160 *
1170 MOVE 1 TO SUB2
1180 MOVE #KEYS(1) TO KEYWORDS(1)
1190 FOR SUB 2 20                          /* now we remove all duplicates
1200   FOR SUB1 1 SUB
1210     IF #KEYS(SUB) = KEYWORDS(SUB1)
1220       RESET #KEYS(SUB)
1230     END-IF
1240   END-FOR
1250   IF #KEYS(SUB) NE ' '
1260     ADD 1 TO SUB2
1270     MOVE #KEYS(SUB) TO KEYWORDS(SUB2)   /* and finally save the value
1280   END-IF
1290 END-FOR
1300 *
1310 * Function 'UR' -- Update RB value using RB offset + length
1320 *     This enables the caller to change information based on a
1330 *     certain location; hence, RBE-OFFSET specifies the start
1340 *     position and RBE-LENGTH specified the length.
1350 *
1360 * PERFORM PRINT-REC-BUFFER              /* print the final results
1370 MOVE 1   TO RBE-OFFSET                 /* start at the beginning
1380 MOVE 520 TO RBE-LENGTH              /* for a max. length of 520 bytes
1390 MOVE 'UR' TO FUNC                   /* req to update all changes
1400 CALL 'STPRBE' 'UR' RBE-AREA REC-BUFFER(1)
1410 IF RBE-RESP NE 0
1420   PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
1430        RBE-ERROR 'subcode' RBE-SUBCODE 'for func UR'
1440   MOVE RBE-RESP TO RESP
1450   ESCAPE ROUTINE
1460 END-IF
1470 *
1480 * Return to the caller: everything went okay
1490 *
1500 ESCAPE ROUTINE
1510 *
1520 DEFINE SUBROUTINE PRINT-REC-BUFFER
1530 *-----------------------------------------------------------------------*
1540 *
1550 *  For testing purposes, display the information returned from STPRBE
1560 *  This routine assumes a maximum of three subsystems running.
1570 *
1580 *-----------------------------------------------------------------------*
1590   DECIDE ON FIRST VALUE OF RQ-TASK
1600     VALUE '01'
1610       WRITE (1) NOTITLE NOHDR (AD=L CD=TU)
1620           '**** RECORD BUFFER EXTRACTION: Function' FUNC '****'
1630           'Stored Procedure RBE' *PROGRAM '****'
1640         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
```

```
1650         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1660         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1670         / '  Message .......' (TU) RBE-MSG(AL=60)
1680         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1690         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1700     VALUE '02'
1710       WRITE (2) NOTITLE NOHDR (AD=L CD=TU)
1720           '**** RECORD BUFFER EXTRACTION: Function' FUNC '****'
1730           'Stored Procedure RBE' *PROGRAM '****'
1740         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1750         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1760         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1770         / '  Message .......' (TU) RBE-MSG(AL=60)
1780         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1790         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1800     VALUE '03'
1810       WRITE (3) NOTITLE NOHDR (AD=L CD=TU)
1820           '**** RECORD BUFFER EXTRACTION: Function' FUNC '****'
1830           'Stored Procedure RBE' *PROGRAM '****'
1840         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1850         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1860         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1870         / '  Message .......' (TU) RBE-MSG(AL=60)
1880         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1890         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1900     NONE
1910       WRITE NOTITLE NOHDR (AD=L CD=TU)
1920           '**** RECORD BUFFER EXTRACTION: Function' FUNC '****'
1930           'Stored Procedure RBE' *PROGRAM '****'
1940         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1950         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1960         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1970         / '  Message .......' (TU) RBE-MSG(AL=60)
1980         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1990         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * '
2000   END-DECIDE
2010 *
2020 END-SUBROUTINE
```

```
2030 *
2040 END
```

STPLCB

```
0010 ************************************************************************
0020 **
0030 **Local data area  'STPLCB'
0040 **describes Adabas control block
0050 **
0060 ************************************************************************
0070 DEFINE DATA LOCAL
0080 1 CB         (B80)
0090 1 REDEFINE CB
0100 2 CB-DSECT             /* ACB definition
0110 3 CB-CALL-TYPE(B1)
0120 3 CB-HOST-ID  (B1)
0130 2 CB-CMD      (A2)     /* command code
0140 2 CB-CID      (A4)     /* command ID
0150 2 CB-FILE     (B2)         /* file number
0160 2 REDEFINE CB-FILE
0170 3 CB-DBID     (B1)         /* one-byte DBNR
0180 3 CB-FNR      (B1)         /* one-byte FNR
0190 2 CB-RSP      (B2)         /* response code
0200 2 CB-ISN(B4)               /* ISN value
0210 2 CB-ISLL(B4)              /* ISN lower limit
0220 2 CB-ISQ(B4)               /* ISN quantity
0230 2 CB-FBL(B2)               /* format buffer length
0240 2 CB-RBL(B2)               /* record buffer length
0250 2 CB-SBL(B2)               /* search buffer length
0260 2 CB-VBL(B2)               /* value buffer length
0270 2 CB-IBL(B2)               /* ISN buffer length
0280 2 CB-CO1(A1)               /* command option 1
0290 2 CB-CO2(A1)               /* command option 2
0300 2 CB-ADD1(A8)              /* additions 1
0310 2 CB-ADD2(A4)              /* additions 2
0320 2 CB-ADD3(A8)              /* additions 3
0330 2 CB-ADD4(A8)              /* additions 4
0340 2 CB-ADD5(A8)              /* additions 5, reserved
0350 2 CB-CT(B4)                /* command time
0360 2 CB-UA(B4)                /* user area
0370 ************************************************************************
```

```
0380 ******* END OF LOCAL DATA AREA ************************************
0390 ************************************************************************
```

STPLCBE

```
0010   * * ******************************** * **** ********************************
0020   * *
0030   * * LOCAL DATA AREA  'STPLCBE'
0040   * * DESCRIBES ADABAS CONTROL BLOCK
0050   * * EXTENDED
0060   * *
0070   * * ******************************** * **** ********************************
0080     1 CB                               B  192
0090   R 1 CB
0100     2 CB-DSECT                                 /* ACBE DEFINITION
0110     3 CB-TYPE                          B    1
0120     3 CB-HOST                          A    1
0130     2 CB-VERSION                       A    2 /* ACBE VERSION
0140     2 CB-LENGTH                        B    2 /* ACBE LENGTH
0150     2 CB-CMD                           A    2 /* COMMAND
0160     2 CB-NUCID                         B    2 /* NUCID
0170     2 CB-RSP                           B    2 /* RESPONSE CODE
0180     2 CB-CID                           B    4 /* COMMAND ID
0190     2 CB-DBID                          B    4 /* DBID
0200     2 CB-FNR                           B    4 /* FILE NUMBER
0210     2 CB-RESERVED-1                    B    4 /* UNUSED
0220     2 CB-ISN                           B    4 /* ISN VALUE
0230     2 CB-RESERVED-2                     B    4 /* UNUSED
0240     2 CB-ISLL                           B    4 /* ISN LOWER LIMIT
0250     2 CB-RESERVED-3                     B    4 /* UNUSED
```

```
0260     2 CB-ISQ                              B    4 /* ISN QUANTITY
0270     2 CB-CO1                              A    1 /* COMMAND OPTION 1
0280     2 CB-CO2                              A    1 /* COMMAND OPTION 2
0290     2 CB-CO3                              A    1 /* COMMAND OPTION 3
0300     2 CB-CO4                              A    1 /* COMMAND OPTION 4
0310     2 CB-CO5                              A    1 /* COMMAND OPTION 5
0320     2 CB-CO6                              A    1 /* COMMAND OPTION 6
0330     2 CB-CO7                              A    1 /* COMMAND OPTION 7
0340     2 CB-CO8                              A    1 /* COMMAND OPTION 8
0350     2 CB-ADD1                             A    8 /* ADDITIONS 1
0360     2 CB-ADD2                             B    4 /* ADDITIONS 2
0370     2 CB-ADD3                             A    8 /* ADDITIONS 3
0380     2 CB-ADD4                             A    8 /* ADDITIONS 4
0390     2 CB-ADD5                             B    8 /* ADDITIONS 5
0400     2 CB-ADD6                             A    8 /* ADDITIONS 6
0410     2 CB-RESERVED-4                       B    4 /* RESERVED
0420     2 CB-ERROR                            B   16 /* SUPPLEMENTAL ERROR INFO
0430   R 2 CB-ERROR
0440     3 CB-ERROR-G                          B    4 /* OFFSET IN BUFFER 64-BIT
0450     3 CB-ERROR-A                         B    4 /* OFFSET IN BUFFER 32-BIT

0460     3 CB-ERROR-B                         A    2 /* ERROR CHARACTER FIELD

0470     3 CB-ERROR-C                          B    2 /* SUBCODE

0480     3 CB-ERROR-D                         A    1 /* ERROR BUFFER ID

0490     3 CB-ERROR-E                         B    3 /* BUFFER SEQ. NUMBER

0500     2 CB-SUB                             B    8 /* SUBCOMPONENT ERROR INFO

0510   R 2 CB-SUB

0520     3 CB-SUB-R                           B    2 /* SUBCOMPONENT RSP CODE

0530     3 CB-SUB-S                           B    2 /* SUBCOMPONENT REASON CD

0540     3 CB-SUB-T                           A    4 /* SUBCOMPONENT ERROR TEXT

0550     2 CB-LCMP                            B    8 /* COMPRESSED RECORD LNGTH

0560     2 CB-LDEC                            B    8 /* DECOMPRESSED LENGTH

0570     2 CB-TIME                             B    8 /* COMAND TIME

0580     2 CB-USER                             B   16 /* USER FIELD

0590     2 CB-ROUTER                           B    1 /* ROUTER FLAGS

0600     2 CB-RESERVED-5                       B   23 /* RESERVED

0610   * * ******************************** * **** ********************************
```

```
0620   * * ***** END OF LOCAL DATA AREA *** * **** ********************************
0630   * * ******************************** * **** ********************************
```

STPLRBE

```
***** DEFINE DATA LOCAL
0010 1  RBE-AREA(A154)      /* record buffer extraction area
0020 1  REDEFINE RBE-AREA
0030 2  RBE-MSG(A72)            /* error text for errors
0040 2  RBE-RESP(B4)            /* error number
0050 2  REDEFINE RBE-RESP
0060 3  RBE-SUBCODE(B2)         /* error subcode
0070 3  RBE-ERROR(B2)           /* actual error code
0080 2  RBE-VERNO(A4)           /* structure version
0090 2  RBE-FIELD-NAME(A32)     /* long name of field
0100 2  RBE-FORMAT(A1)      /* field format
0110 2  RBE-OPTS(A3)            /* special options
0120 2  RBE-LENGTH(B4)      /* field/RB length
0130 2  RBE-ADA-FIELD(A2)       /* Adabas short name
0140 2  RBE-RESRV2(A2)      /* reserved
0150 2  RBE-FIELD-OCC(B4)       /* field occurrence for MU or PE
0160 2  RBE-GROUP-OCC(B4)       /* PE occurrence for MU within PE
0170 2  RBE-OFFSET(B4)      /* offset into RB
0180 2  RBE-UNUSED(A18)         /* not used
***** END-DEFINE
```

STPUTPRM

```
***** DEFINE DATA PARAMETER
0010 1  CALL-TYPE(A1)       /* type of call
0020 **                     /* 'B' before invoking
0030 **                     /* 'A' after  invoking
0040 **                     /* 'E' error  incurred
0050 1  REQ-AREA(A200)      /* request area
0060 1  REDEFINE REQ-AREA
0070 2  RQ-VERNO(A4)        /* structure version
0080 2  RQ-TASK(A2)         /* subsystem number
0090 2  RQ-PROC(A8)         /* procedure name
0100 2  RQ-USER(A32)        /* user identification
0110 2  RQ-CMD(A2)          /* trigger command
0120 2  RQ-DBID(B2)         /* Trigger DBID
0130 2  RQ-FNR(B2)          /* trigger target file number
0140 2  RQ-FIELD(A2)        /* trigger field (short name)
0150 2  RQ-SYNC(A1)         /* sync/async request
```

```
0160 2  RQ-PARTIC(A1)       /* participating/non-participating request
0170 2  RQ-LENGTH(B2)       /* record buffer length
0180 2  RQ-UPD(A1)          /* RB update indicator
0190 2  RQ-TTYP(A1)         /* trigger type "P"re or Po"S"t
0200 2  RQ-RESP(B4)
0210 2  REDEFINE RQ-RESP
0220 3  RESP-CODE(B2)
0230 3  SUB-CODE(B2)
0240 2  RQ-PDA-TYPE(A1)     /* calling type
0250 2  RQ-RESRVED2(A55)
0260 2  RQ-CB(A80)          /* trigger control block
0270 1  ERR-INFO(A72)       /* error information for CALL-TYPE 'E'
0280 1  REDEFINE ERR-INFO
0290 2  ERR-NR(N4)          /* error number
0300 2  ERR-LINE(N4)        /* line number of error
0310 2  ERR-STAT(A1)        /* error status indicator
0320 2  ERR-PROG(A8)        /* error program
0330 2  ERR-LEVEL(N2)       /* not used
0340 2  ERR-TYPE(A24)       /* error identification
0350 1  REQ-GLOBAL-WS(A250) /* global WS area
0360 1  RESP(B4)            /* procedure response
***** END-DEFINE
```

STPUTRAK

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Program    : STPUTRAK
0040 *  Function   : Routine that is invoked if triggers is running with
0050 *               'Trigger Logging' set to Active.
0060 *               Invoked (CALL-TYPE):
0070 *                  'I' - when the subsystem is initialized
0080 *                  'T' - when the subsystem is being terminated
0090 *                  'B' - before a procedure is invoked
0100 *                  'A' - after a procedure has completed
0110 *                  'E' - whenever an error occurs
0120 *  NOTE       : If logging is ctive, then the module (cataloged object)
0130 *               must exist; otherwise, a NAT0082 occurs.
0140 *  Author     : Adabas Development
0150 *  Date       : June 1994
0160 ************************************************************************
0170 DEFINE DATA PARAMETER USING STPUTPRM
0180            LOCAL
0190 01  EVENT      (A14)              /* event criteria
0200 01  REDEFINE EVENT
0210   02  E-FNR     (N5)
0220   02  E-F1      (A2)
0230   02  E-CMD     (A2)
```

```
0240   02  E-F2      (A2)
0250   02  E-FIELD   (A2)
0260 01  PARM-TYPE   (A7)
0270 01  TRIG-TYPE  (A10)
0280 01  REDEFINE TRIG-TYPE
0290   02  PRE-POST  (A4)
0300   02  FILL      (A1)
0310   02  PROC-TYPE (A5)
0320 01  UQE-ID     (A28)
0330 01  RB-TYPE     (A6)
0340 01  RBLEN       (N5)
0350 01  RESP-BIN    (B4)
0360 01  REDEFINE RESP-BIN
0370   02  RESP-SUBC (B2)
0380   02  RESP-CDE  (B2)
0390 END-DEFINE
0400 *
0410 FORMAT PS=0 LS=133                /* set report attributes
0420 *
0430 IF CALL-TYPE = 'I'                /* subsystem initialization
0440   WRITE NOTITLE (CD=TU)
0450         '***** Triggers and Stored Procedures *****' (GRI)
0460     / '      - Natural Subsystem Initialization -' (YEI)
0470    // 'Program + Library Location ...' (TU) *PROGRAM *LIBRARY-ID
0480     / 'Task Initialization Time .....' (TU) *DATX *TIMX
0490     / 'Task Identification Number ...' (TU) RQ-TASK
0500     / 'Task User Identification .....' (TU) *INIT-USER *INIT-ID
0510     / '********************** INIT ********************' (GRI)
0520  NEWPAGE                           /* setup for proper headings
0530  ESCAPE ROUTINE                    /* return control
0540 END-IF
0550 *
0560 IF CALL-TYPE = 'T'                 /* subsystem termination
0570  WRITE NOTITLE (CD=TU)
0580       '***** Triggers and Stored Procedures *****' (GRI)
0590     / '        - Natural Subsystem Termination -' (YEI)
0600    // 'Task Termination Time ........' (TU) *DATX *TIMX
0610     / 'Task Identification Number ...' (TU) RQ-TASK
0620     / 'Task User Identification .....' (TU) *INIT-USER *INIT-ID
0630     / '********************** EXIT ********************' (GRI)
0640  ESCAPE ROUTINE                    /* return control
0650 END-IF
0660 *
0670 IF CALL-TYPE = 'A'                 /*  after invoking procedure
0680   MOVE RESP TO RESP-BIN
0690   WRITE RQ-TASK 2X *DATX *TIMX 'complete' 16X RQ-PROC 6X 'Resp:'
0700         RESP-CDE RESP-SUBC
0710 END-IF
0720 *
0730 IF CALL-TYPE = 'B'                 /*  before invoking procedure
0740  MOVE RQ-RESRVED2 TO UQE-ID
0750* MOVE RB-RBL   TO RBLEN
```

```
0760  MOVE RQ-LENGTH TO RBLEN
0770  IF RQ-TTYP = 'P'
0780    MOVE 'Pre ' TO PRE-POST
0790  ELSE
0800    MOVE 'Post' TO PRE-POST
0810  END-IF
0820  IF RQ-SYNC = 'A'
0830    MOVE 'ASync' TO PROC-TYPE
0840  END-IF
0850  IF RQ-SYNC = 'S'
0860    MOVE 'Sync' TO PROC-TYPE
0870    IF RQ-PARTIC = 'P'
0880      MOVE 'Part' TO PROC-TYPE
0890    END-IF
0900    IF RQ-PARTIC = 'N'
0910      MOVE 'Non-P' TO PROC-TYPE
0920    END-IF
0930  END-IF
0940  DECIDE ON FIRST VALUE OF RQ-PDA-TYPE
0950    VALUE 'C' MOVE 'Control' TO PARM-TYPE
0960    VALUE 'N' MOVE 'No Parm' TO PARM-TYPE
0970    VALUE 'R' MOVE 'Resp'    TO PARM-TYPE
0980    NONE      MOVE 'Unknown' TO PARM-TYPE
0990  END-DECIDE
1000  DECIDE ON FIRST VALUE OF RQ-UPD
1010    VALUE 'A' MOVE 'Access' TO RB-TYPE
1020    VALUE 'N' MOVE 'No Rec' TO RB-TYPE
1030    VALUE 'U' MOVE 'Update' TO RB-TYPE
1040    NONE      MOVE '??????' TO RB-TYPE
1050  END-DECIDE
1060  IF RQ-CMD = 'PC'
1070    MOVE '*Stored Proc.*' TO EVENT
1080  ELSE
1090    MOVE RQ-FNR TO E-FNR
1100    MOVE RQ-CMD TO E-CMD
1110    MOVE RQ-FIELD TO E-FIELD
1120  END-IF
1130  DISPLAY NOTITLE (CD=TU)
1140      'Tsk' RQ-TASK 'Date' *DATX 'Time' *TIMX 'User' RQ-USER(AL=8)
1150      'Fnr    Cmd Fld' (TU) EVENT 'Proc' (TU) RQ-PROC
1160      'Type' (TU) TRIG-TYPE 'Parms' (TU) PARM-TYPE
1170      'RecBuf' RB-TYPE
1180 * PRINT 5X 'UserID ...' UQE-ID(EM=H(28)) /* UQE-ID
1190 *
1200 * A special overwrite option allows the user to have the procedure
1210 * called with the additional parameter of the RQ-GLOBAL-WS.
1220 * This is valid only if RQ-PDA-TYPE is set to 'C'.
1230 * The procedure should expect parameters to be passed as specified in
1240 * STPAPRM1 i.e. CALLNAT 'procname' REQ-AREA REQ-GLOBAL-WS RESP
1250 *
1260 * MOVE 'W' TO CALL-TYPE
1270 *
```

```
1280 * RQ-GLOBAL-WS is an area that is not changed between each call to
1290 * the procedures. It may be used for keeping statistics or whatever.
1300 *
1310 END-IF
1320 *
1330 IF CALL-TYPE = 'E'                  /* subsystem error notification
1340   WRITE NOTITLE (CD=TU)
1350        '***** Triggers and Stored Procedures *****' (GRI)
1360      / '        - Natural Subsystem Error Info -' (YEI)
1370     // 'Task Termination Time .......' (TU) *DATX *TIMX
1380     / 'Task Identification Number ..' (TU) RQ-TASK
1390       '<<< Interrupted with an ERROR <<<<<'
1400    // '* Subsystem Error Number ....' ERR-NR
1410   41T '* Stored Proc ....' RQ-PROC
1420     / '*           Active Module ...' ERR-PROG
1430   41T '* UserID .........' RQ-USER
1440     / '*           Line Number .....' ERR-LINE (EM=9999)
1450   41T '* Trigger Cmd ....' RQ-CMD
1460     / '*           Error Level .....' ERR-LEVEL
1470   41T '* Trigger Fnr ....' RQ-FNR (AD=L)
1480     / '*           Error Status ....' ERR-STAT
1490   41T '* Trigger Type ...' RQ-TTYP RQ-PARTIC 'opt' RQ-PDA-TYPE RQ-UPD
1500     / '*           Error Type ......' ERR-TYPE (AL=14)
1510*  41T '* Field Name .....' RQ-TRG-FIELD '+' RB-RBL
1520*    / '*           UQE Ident. ......' CA-USER
1530     /  '*** Processing for this request ABNORMALLY terminated ***'
1540    // '********************* ERROR ********************' (GRI)
1550  ESCAPE ROUTINE                     /* return control
1560 END-IF
1570 *
1580 END
```

STPAPARM

```
***** DEFINE DATA PARAMETER
0010 1  REQ-AREA(A200)      /* request area
0020 1  REDEFINE REQ-AREA
0030 2  RQ-VERNO(A4)        /* structure version
0040 2  RQ-TASK(A2)         /* subsystem number
0050 2  RQ-PROC(A8)         /* procedure name
0060 2  RQ-USER(A32)        /* user identification
0070 2  RQ-CMD(A2)          /* trigger command
0080 2  RQ-DBID(B2)         /* Trigger DBID
0090 2  RQ-FNR(B2)          /* trigger target file number
0100 2  RQ-FIELD(A2)        /* trigger field (short name)
0110 2  RQ-SYNC(A1)         /* sync/async request
0120 2  RQ-PARTIC(A1)       /* participating/non-participating request
0130 2  RQ-LENGTH(B2)       /* record buffer length
```

```
0140 2  RQ-UPD(A1)          /* RB update indicator
0150 2  RQ-TTYPE(A1)        /* pre- or post-trigger
0160 2  RQ-RESP(B4)         /* subcode(B2) + resp code(B2)
0170 2  RQ-PDA-TYPE(A1)     /* calling parameters type
0180 2  RQ-USERID(A28)      /* user ID from Adabas CQE
0190 2  RQ-RESRVED2(A27)
0200 2  RQ-CB(A80)          /* trigger control block
0210 1  RESP(B4)            /* procedure response
***** END-DEFINE
```

STPAPRM1

```
***** DEFINE DATA PARAMETER STPAPRM1 LIBRARY SYSSPT
0010 1  REQ-AREA(A200)           /* Request area
0020 1  REDEFINE REQ-AREA
0030 2  RQ-VERNO(A4)             /* structure version
0040 2  RQ-TASK(A2)              /* subsystem number
0050 2  RQ-PROC(A8)              /* procedure name
0060 2  RQ-USER(A32)             /* user identification
0070 2  RQ-CMD(A2)               /* trigger command
0080 2  RQ-DBID(B2)              /* Trigger DBID
0090 2  RQ-FNR(B2)               /* trigger target file number
0100 2  RQ-FIELD(A2)             /* trigger field (short name)
0110 2  RQ-SYNC(A1)              /* sync/async request
0120 2  RQ-PARTIC(A1)            /* participating/non-participating request
0130 2  RQ-LENGTH(B2)            /* record buffer length
0140 2  RQ-UPD(A1)               /* RB update indicator
0150 2  RQ-RESRVED1(A1)          /* not used
0160 2  RQ-RESP(B4)
0170 2  RQ-PDA-TYPE(A1)
0180 2  RQ-RESRVED2(A55)
0190 2  RQ-CB(A80)               /* trigger control block
0200 1  REQ-GLOBAL-WS(A250)      /* global WS area
0210 1  RESP(B4)                 /* procedure response
***** END-DEFINE
```

STPXPARM

```
0010     1 REQ-AREA                          A  200 /* Request Area
0020   R 1 REQ-AREA                                 /* Redef. begin : REQ-AREA
0030     2 RQ-VERNO                          A    4 /* Structure Version
0040     2 RQ-TASK                           A    2 /* Subsystem Number
0050     2 RQ-PROC                           A    8 /* Procedure Name
0060     2 RQ-USER                           A   32 /* User Identification
0070     2 RQ-CMD                            A    2 /* Trigger Cmd
0080     2 RQ-DBID                           B    2 /* Trigger DBID
0090     2 RQ-FNR                            B    2 /* Trigger Target File Nr
0100     2 RQ-FIELD                          A    2 /* Trigger Field (Short Name)
0110     2 RQ-SYNC                           A    1 /* Sync/Async Request
0120     2 RQ-PARTIC                         A    1 /* Part/Non-participating Req
0130     2 RQ-LENGTH                         B    2 /* Record Buffer Length
0140     2 RQ-UPD                            A    1 /* RB Update Indicator
0150     2 RQ-TTYPE                          A    1 /* Pre or Post Trigger
0160     2 RQ-RESP                           B    4 /* Subcode(B2) + Resp Code(B2)
0170     2 RQ-PDA-TYPE                       A    1 /* Calling Parameters Type
0180     2 RQ-USERID                         A   28 /* UserID from ADABAS UQE
0190     2 RQ-RESRVED2                       A   27 /*
0200     2 RQ-CB                             A   80 /* Trigger Control Block
0210     1 RQ-CBX                            A  192 /* X Verion of CB
```

```
0220     1 RESP                              B    4 /* Procedure Response
```

SAMP0001

```
0010 ************************************************************************
0020 *  Application: ADASTP
0030 *  Subprogram : SAMP0001
0040 *  Author     : Adabas Development
0050 *  Date       : August 1995
0060 *  Function   : Sample routine of processing by a procedure
0070 *  Remarks    : This routine converts the CONTACT-NAME into uppercase
0080 *               and extracts all the keywords associated with it.
0090 *               Once processing is completed, control is returned to
0100 *               the caller.
0110 *               Parameter RESP must be set to zero if processing is
0120 *               successful.
0130 *
0140 *  Parameters : REQ-AREA (A200)
0150 *               RESP       (B4)
0160 *
0170 *  Trigger Typ: The type of trigger will be PARTICIPATING; i.e.,
0180 *               synchronous.
0190 *  Rec Buffer : The record buffer will be available for update via a
0200 *               call to the external routine STPRBE.
0210 * Trigger Defn: Definition on the trigger file (note that there is a
0220 *               trigger for the insert and update) is as follows:
0230 *                 File Number ....... 11
0240 *                 File Name ......... CONTACTS
0250 *                 Command Type ...... Update + Insert
0260 *                 Long Field Name ... CONTACT-NAME
0270 *                 Adabas Field ...... LE
0280 *                 Field Prty/Seq .... 10_
0290 *               Procedure Information
0300 *                 Name (Subpgm)...... SAMP0001
0310 *                 Pre Cmd Select .... Y  (Pre)
0320 *                 Trigger Type ...... P  (Participating)
0330 *                 CALLNAT Params .... C  (Cntl Info + Resp)
0340 *                 RecBuffer Access .. U  (May be updated)
0350 *
0360 ************************************************************************
0370 DEFINE DATA PARAMETER USING STPAPARM
0380            LOCAL     USING STPLRBE /* parms for the call routine
0390            LOCAL
0400   01  REC-BUFFER(A20/1:26)         /* max, record buffer passed to STPRBE
0410   01  REDEFINE REC-BUFFER          /* redefine this to get the definition
0420     02  INPUT-NAME   (A60)
0430     02  OUTPUT-NAME  (A60)
```

```
0440     02  KEYWORDS(A20/1:20)
0450   01  FUNC            (A4)
0460   01  SUB             (I2)
0470   01  SUB1            (I2)
0480   01  SUB2            (I2)
0490   01  W-UPPER        (A61)
0500   01  REDEFINE W-UPPER
0510     02  #UPPER       (A60)
0520     02  REDEFINE #UPPER
0530       03  CHAR   (A1/1:60)
0540   01  #KEYS     (A40/1:20)
0550 END-DEFINE
0560 *
0570 *  First, all procedures for this file must go through the audit procedure
0580 *  because our example requires a trace of all commands to this file.
0590 *
0600 CALLNAT 'SAMP0003' REQ-AREA RESP
0610 *
0620 * Since the record buffer information is available to us, we can now call
0630 * the record buffer extraction routine (STPRBE) to obtain the contents of
0640 * the buffer.
0650 *
0660 * Function 'GR' -- GET RB value using RB offset + length
0670 *     This enables the caller to obtain information based on a
0680 *     certain location; hence, RBE-OFFSET specifies the start
0690 *     position and RBE-LENGTH specifies the length.
0700 *
0710 MOVE 1   TO RBE-OFFSET            /* start at the beginning
0720 MOVE 520 TO RBE-LENGTH            /* for a max. length of 520 bytes
0730 MOVE 'GR' TO FUNC
0740 CALL 'STPRBE' 'GR' RBE-AREA REC-BUFFER(1)
0750 IF RBE-RESP NE 0
0760   PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
0770        RBE-ERROR 'subcode' RBE-SUBCODE 'for func GR'
0780   MOVE RBE-RESP TO RESP
0790   ESCAPE ROUTINE
0800 END-IF
0810 * PERFORM PRINT-REC-BUFFER         /* option to print the parameters
0820 *
0830 * Change all lowercase to UPPERcase
0840 *
0850 MOVE INPUT-NAME TO #UPPER
0860 *
0870 EXAMINE #UPPER AND TRANSLATE INTO UPPER CASE
0880 *
0890 MOVE #UPPER TO OUTPUT-NAME        /* save the uppercase name
0900 *
0910 FOR SUB 1 60                      /* loop to remove all special chars.
0920   IF CHAR(SUB) = MASK(S)
0930     MOVE ' ' TO CHAR(SUB)
0940     ESCAPE TOP
0950   END-IF
```

```
0960 END-FOR
0970 *
0980 * We are now ready to extract keywords from our name. This sample is
0990 * very basic and may be made as complex as required.
1000 * This routine assumes a max. length of 20 and a max. num. of 20 keywords
1010 *
1020 EXAMINE FULL W-UPPER FOR FULL ' A '   REPLACE ' '
1030 EXAMINE FULL W-UPPER FOR FULL ' AND ' REPLACE ' '
1040 EXAMINE FULL W-UPPER FOR FULL ' AS '  REPLACE ' '
1050 EXAMINE FULL W-UPPER FOR FULL ' AT '  REPLACE ' '
1060 EXAMINE FULL W-UPPER FOR FULL ' ARE ' REPLACE ' '
1070 EXAMINE FULL W-UPPER FOR FULL ' BE '  REPLACE ' '
1080 EXAMINE FULL W-UPPER FOR FULL ' DO '  REPLACE ' '
1090 EXAMINE FULL W-UPPER FOR FULL ' FOR ' REPLACE ' '
1100 EXAMINE FULL W-UPPER FOR FULL ' HERE ' REPLACE ' '
1110 EXAMINE FULL W-UPPER FOR FULL ' IF '  REPLACE ' '
1120 EXAMINE FULL W-UPPER FOR FULL ' IN '  REPLACE ' '
1130 EXAMINE FULL W-UPPER FOR FULL ' IS '  REPLACE ' '
1140 EXAMINE FULL W-UPPER FOR FULL ' IT '  REPLACE ' '
1150 EXAMINE FULL W-UPPER FOR FULL ' OF '  REPLACE ' '
1160 EXAMINE FULL W-UPPER FOR FULL ' ON '  REPLACE ' '
1170 EXAMINE FULL W-UPPER FOR FULL ' OR '  REPLACE ' '
1180 EXAMINE FULL W-UPPER FOR FULL ' TO '  REPLACE ' '
1190 EXAMINE FULL W-UPPER FOR FULL ' THE ' REPLACE ' '
1200 EXAMINE FULL W-UPPER FOR FULL ' TOO ' REPLACE ' '
1210 EXAMINE FULL W-UPPER FOR FULL ' WAS ' REPLACE ' '
1220 EXAMINE FULL W-UPPER FOR FULL ' WITH ' REPLACE ' '
1230 EXAMINE #UPPER FOR FULL ' ' REPLACE ',' /* put delimiters in the string
1240 *
1250 RESET KEYWORDS(*)
1260 STACK TOP DATA #UPPER                   /* now we will separate each word
1270 INPUT (AD=I IP=ON) #KEYS(01:03) / #KEYS(04:06) / #KEYS(07:09)
1280                  / #KEYS(10:12) /* #KEYS(13:15) / #KEYS(16:18)
1290                  / #KEYS(19:20)
1300 *
1310 MOVE 1 TO SUB2
1320 MOVE #KEYS(1) TO KEYWORDS(1)
1330 FOR SUB 2 20                            /* now we remove all duplicates
1340   FOR SUB1 1 SUB
1350     IF #KEYS(SUB) = KEYWORDS(SUB1)
1360       RESET #KEYS(SUB)
1370     END-IF
1380   END-FOR
1390   IF #KEYS(SUB) NE ' '
1400     ADD 1 TO SUB2
1410     MOVE #KEYS(SUB) TO KEYWORDS(SUB2)  /* and finally save the value
1420   END-IF
1430 END-FOR
1440 *
1450 * Function 'UR' -- Update RB value using RB offset + length
1460 *     This enables the caller to change information based on a
1470 *     certain location; hence, RBE-OFFSET specifies the start
```

```
1480 *     position and RBE-LENGTH specified the length.
1490 *
1500 * PERFORM PRINT-REC-BUFFER             /* print the final results
1510 MOVE 1   TO RBE-OFFSET                 /* start at the beginning
1520 MOVE 520 TO RBE-LENGTH                 /* for a max. length of 520 bytes
1530 MOVE 'UR' TO FUNC                      /* request to update all changes
1540 CALL 'STPRBE' 'UR' RBE-AREA REC-BUFFER(1)
1550 IF RBE-RESP NE 0
1560   PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
1570        RBE-ERROR 'subcode' RBE-SUBCODE 'for func UR'
1580   MOVE RBE-RESP TO RESP
1590   ESCAPE ROUTINE
1600 END-IF
1610 *
1620 * Return to the caller: everything went okay
1630 *
1640 ESCAPE ROUTINE
1650 *
1660 DEFINE SUBROUTINE PRINT-REC-BUFFER
1670 *----------------------------------------------------------------------*
1680 *
1690 *  For testing purposes, display the information returned from STPRBE
1700 *  This routine assumes a maximum of three subsystems running.
1710 *
1720 *----------------------------------------------------------------------*
1730   DECIDE ON FIRST VALUE OF RQ-TASK
1740     VALUE '01'
1750       WRITE (1) NOTITLE NOHDR (AD=L CD=TU)
1760           '>>>> RECORD BUFFER EXTRACTION: Function' FUNC '<<<<'
1770         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1780         / '             ....' (TU) RBE-ADA-FIELD RBE-FIELD-OCC
1790         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1800         / '  Message .......' (TU) RBE-MSG(AL=60)
1810         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1820         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1830     VALUE '02'
1840       WRITE (2) NOTITLE NOHDR (AD=L CD=TU)
1850           '>>>> RECORD BUFFER EXTRACTION: Function' FUNC '<<<<'
1860         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1870         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1880         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1890         / '  Message .......' (TU) RBE-MSG(AL=60)
1900         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
1910         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
1920     VALUE '03'
1930       WRITE (3) NOTITLE NOHDR (AD=L CD=TU)
1940           '>>>> RECORD BUFFER EXTRACTION: Function' FUNC '<<<<'
1950         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
1960         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
1970         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR '<<<<<'
1980         / '  Message .......' (TU) RBE-MSG(AL=60)
1990         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
```

```
2000         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
2010     NONE
2020       WRITE NOTITLE NOHDR (AD=L CD=TU)
2030           '>>>> RECORD BUFFER EXTRACTION: function' FUNC '<<<<'
2040         / '  Field Info ....' (TU) RBE-FIELD-NAME RBE-FORMAT RBE-LENGTH
2050         / '             ....' (TU) RBE-ADA-FIELD  RBE-FIELD-OCC
2060         / '  Resp + Error ..' (TU) RBE-RESP RBE-ERROR'<<<<<'
2070         / '  Message .......' (TU) RBE-MSG(AL=60)
2080         / '  Rec Buffer ....' (TU) / REC-BUFFER(1)(AL=79)
2090         / '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * '
2100   END-DECIDE
2110 *
2120 END-SUBROUTINE
2130 *
2140 END
```

SAMP0002

```
0010 ************************************************************************
0020 *  Application: Adabas Triggers
0030 *   Subprogram: SAMP0002
0040 *     Function: Sample routine of processing by a stored procedure
0050 *               The requirement is to audit all commands for a file
0060 *               by writing out an audit record to a file/printer.
0070 *               For this example, the audit is the Natural system file.
0080 * Trigger Defn: Trigger Information
0090 *                 File Number ....... 11
0100 *                 File Name ......... CONTACTS
0110 *                 Command Type ...... ** All Command **
0120 *                 Long Field Name ... ** Any Field **
0130 *                 Adabas Field ...... **
0140 *                 Field Prty/Seq .... ___
0150 *               Procedure Information
0160 *                 Name (Subpgm)...... SAMP0002
0170 *                 Pre Cmd Select .... Y  (Pre)
0180 *                 Trigger Type ...... A  (Asynchronous)
0190 *                 CALLNAT Params .... C  (Cntl Info + Resp)
0200 *                 RecBuffer Access .. N  (No RecBuff Access)
0210 *
0220 *       AUTHOR: Adabas Development
0230 *         DATE: December 1995
0240 ************************************************************************
0250 DEFINE DATA PARAMETER USING STPAPARM
0260            LOCAL     USING STPLCB /* DSECT of the Adabas control block
0270            LOCAL
0280  01  #SRCID     (A18)            /* key of the audit record
0290  01  REDEFINE #SRCID
0300    02  SRC-LIB   (A8)            /* to be placed on a Natural system file
```

```
0310    02  SRC-PGM   (A8)
0320    02  SRC-SEQ   (B2)
0330  01  #DATE       (A8)
0340  01  #TIME       (A8)
0350  01  LOG-AREA VIEW OF SYSTEM2     /* write information to the FNAT file
0360    02  SRCID
0370    02  SRCTX      (1)
0380  01  W-USERID   (A28)             /* user ID from originating command
0390  01  REDEFINE W-USERID
0400    02  W-F1     (A20)
0410    02  W-USER    (A8)             /* TP USID of the user ID
0420  01  #TEXT (A72)                  /* text message to be written
0430  01  REDEFINE #TEXT
0440    02  TX-LNO    (B2)
0450    02  TX-F1     (A1)
0460    02  TX-DATE   (A8)
0470    02  TX-F2     (A1)
0480    02  TX-TIME   (A5)
0490    02  TX-F3     (A1)
0500    02  TX-USER   (A8)
0510    02  TX-F4     (A1)
0520    02  TX-CMD    (A2)
0530    02  TX-F5     (A1)
0540    02  TX-PRE    (A3)
0550    02  TX-F6     (A1)
0560    02  TX-FNR    (N3)
0570    02  TX-F7     (A1)
0580    02  TX-RBL    (N4)
0590    02  TX-F8     (A1)
0600    02  TX-SYNC   (A5)
0610    02  TX-F9     (A1)
0620    02  TX-TASK   (A2)
0630    02  TX-F10    (A1)
0640    02  TX-FIELD  (A2)
0650    02  TX-F11    (A1)
0660    02  TX-PROC   (A8)
0670    02  TX-F12    (A1)
0680    02  TX-USR2   (A8)
0690 END-DEFINE
0700 *
0710 ASSIGN #SRCID =  'AUDIT   LOGINFO' /* set the target lib and pgm name
0720 MOVE H'0000' TO SRC-SEQ
0730 *
0740 MOVE RQ-CB   TO CB                 /* move ACB into our CB layout
0750 MOVE H'0010' TO TX-LNO             /* line number of Natural source
0760 MOVE *DATE   TO TX-DATE
0770 MOVE *TIMX   TO TX-TIME
0780 MOVE RQ-USERID TO W-USERID         /* user ID of the command
0790 MOVE W-USER  TO TX-USER            /* may be a batch user
0800 IF NOT TX-USER = MASK(PPPPPPPP)    /* printable user ID?
0810  MOVE RQ-USER TO TX-USER           /* no, so use the jobname or TPname
0820 END-IF
```

```
0830 MOVE RQ-CMD  TO TX-CMD             /* information from the CB layout
0840 MOVE RQ-FNR  TO TX-FNR
0850 MOVE RQ-TASK TO TX-TASK            /* subsystem number
0860 IF RQ-LENGTH > 9999                /* exceed max. size in audit message?
0870  MOVE 9999 TO TX-RBL
0880 ELSE
0890  MOVE RQ-LENGTH TO TX-RBL          /* the real record buffer length
0900 END-IF
0910 MOVE *PROGRAM  TO TX-PROC          /* originating procedure. This subpgm
0920 IF RQ-TTYPE = 'P'                  /* trigger type
0930  MOVE 'Pre' TO TX-PRE
0940 ELSE
0950  MOVE 'Pos' TO TX-PRE
0960 END-IF
0970 IF RQ-SYNC = 'A'                   /* processing type
0980  MOVE 'ASync' TO TX-SYNC
0990 ELSE
1000  IF RQ-PARTIC = 'P'                /* trigger logic type
1010    MOVE 'Part'  TO TX-SYNC
1020  ELSE
1030    MOVE 'Non-P' TO TX-SYNC
1040  END-IF
1050 END-IF
1060 *
1070 * Now we do some logic to write the information out to a report
1080 * Here we support up to five subysytems
1090 * Contents are a one-line display to minimize output
1100 *
1110 DECIDE ON FIRST VALUE OF RQ-TASK
1120    VALUE '01'
1130      DISPLAY (1) NOTITLE (AD=L CD=TU)
1140           'Procedure' *PROGRAM
1150           'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1160           'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1170           'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1180           'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1190    VALUE '02'
1200      DISPLAY (2) NOTITLE (AD=L CD=TU)
1210           'Procedure' *PROGRAM
1220           'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1230           'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1240           'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1250           'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1260    VALUE '03'
1270      DISPLAY (3) NOTITLE (AD=L CD=TU)
1280           'Procedure' *PROGRAM
1290           'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1300           'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1310           'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1320           'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1330    VALUE '04'
1340      DISPLAY (4) NOTITLE (AD=L CD=TU)
```

```
1350            'Procedure' *PROGRAM
1360            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1370            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1380            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1390            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1400    NONE
1410      DISPLAY (5) NOTITLE (AD=L CD=TU)
1420            'Procedure' *PROGRAM
1430            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1440            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1450            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1460            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1470  END-DECIDE
1480 *
1490 * Finally, we write this information to a 'audit' File. In this case, we
1500 * use the Natural FNAT file for simplicity. Realistically, a separate
1510 * 'audit' file should be used.
1520 *
1530  MOVE #TEXT TO LOG-AREA.SRCTX (1.1)
1540  MOVE H'0001' TO SRC-SEQ
1550  ASSIGN LOG-AREA.SRCID = #SRCID
1560  STORE LOG-AREA
1570  END TRANSACTION                   /* required for non-participating
1580 *                                  /* and asynchronous triggers
1590 *
1600 END
```

SAMP0003

```
0010 ************************************************************************
0020 *  Application: Adabas Triggers
0030 *   Subprogram: SAMP0003
0040 *     Function: Sample routine of processing by a stored procedure
0050 *               The requirement is to audit  all commands for a file
0060 *               by writing out an audit record to a file/printer.
0070 *               For this example, the audit is the Natural system file.
0080 *
0090 *               This routine is called by participating triggers and
0100 *               will contain no ET logic; hence, it must have been
0110 *               invoked as a result of a update/delete/store command
0120 * Trigger Defn: None because it will be invoked directly from another
0130 *               procedure. In this case SAMP0001.
0140 *
0150 *       Author: Adabas Development
0160 *         Date: December 1995
0170 ************************************************************************
0180 DEFINE DATA PARAMETER USING STPAPARM
0190            LOCAL     USING STPLCB /* DSECT of the Adabas control block
```

```
0200            LOCAL
0210  01  #SRCID     (A18)             /* key of the audit record
0220  01  REDEFINE #SRCID
0230    02  SRC-LIB   (A8)             /* to be placed on a Natural system file
0240    02  SRC-PGM   (A8)
0250    02  SRC-SEQ   (B2)
0260  01  #DATE       (A8)
0270  01  #TIME       (A8)
0280  01  LOG-AREA VIEW OF SYSTEM2     /* write information to the FNAT file
0290    02  SRCID
0300    02  SRCTX      (1)
0310  01  W-USERID   (A28)             /* user ID from originating command
0320  01  REDEFINE W-USERID
0330    02  W-F1     (A20)
0340    02  W-USER    (A8)             /* TP USID of the user ID
0350  01  #TEXT (A72)                  /* text message to be written
0360  01  REDEFINE #TEXT
0370    02  TX-LNO    (B2)
0380    02  TX-F1     (A1)
0390    02  TX-DATE   (A8)
0400    02  TX-F2     (A1)
0410    02  TX-TIME   (A5)
0420    02  TX-F3     (A1)
0430    02  TX-USER   (A8)
0440    02  TX-F4     (A1)
0450    02  TX-CMD    (A2)
0460    02  TX-F5     (A1)
0470    02  TX-PRE    (A3)
0480    02  TX-F6     (A1)
0490    02  TX-FNR    (N3)
0500    02  TX-F7     (A1)
0510    02  TX-RBL    (N4)
0520    02  TX-F8     (A1)
0530    02  TX-SYNC   (A5)
0540    02  TX-F9     (A1)
0550    02  TX-TASK   (A2)
0560    02  TX-F10    (A1)
0570    02  TX-FIELD  (A2)
0580    02  TX-F11    (A1)
0590    02  TX-PROC   (A8)
0600    02  TX-F12    (A1)
0610    02  TX-USR2   (A8)
0620 END-DEFINE
0630 *
0640 ASSIGN #SRCID =  'AUDIT   LOGINFO' /* set the target lib and program name
0650 MOVE H'0000' TO SRC-SEQ
0660 *
0670 MOVE RQ-CB   TO CB                 /* move ACB into the CB layout
0680 MOVE H'0010' TO TX-LNO             /* line number of Natural source
0690 MOVE *DATE   TO TX-DATE
0700 MOVE *TIMX   TO TX-TIME
0710 MOVE RQ-USERID TO W-USERID         /* user ID of the command
```

```
0720 MOVE W-USER  TO TX-USER            /* may be a batch user
0730 MOVE RQ-USER TO TX-USR2            /* jobname or TPname
0740 MOVE RQ-CMD  TO TX-CMD             /* information from the CB layout
0750 MOVE RQ-FNR  TO TX-FNR
0760 MOVE RQ-TASK TO TX-TASK            /* subsystem number
0770 IF RQ-LENGTH > 9999                /* exceed max. size in audit message?
0780   MOVE 9999 TO TX-RBL
0790 ELSE
0800   MOVE RQ-LENGTH TO TX-RBL         /* the real record buffer length
0810 END-IF
0820 MOVE *PROGRAM  TO TX-PROC          /* originating procedure. This subpgm
0830 IF RQ-TTYPE = 'P'                  /* trigger type
0840   MOVE 'Pre' TO TX-PRE
0850 ELSE
0860   MOVE 'Pos' TO TX-PRE
0870 END-IF
0880 IF RQ-SYNC = 'A'                   /* processing type
0890   MOVE 'ASync' TO TX-SYNC
0900 ELSE
0910   IF RQ-PARTIC = 'P'               /* trigger logic type
0920     MOVE 'Part'  TO TX-SYNC
0930   ELSE
0940     MOVE 'Non-P' TO TX-SYNC
0950   END-IF
0960 END-IF
0970 *
0980 * Now we do some logic to write the information out to a report
0990 * Here we support up to five subsystems
1000 * Contents are a one-line display to minimize output
1010 *
1020 DECIDE ON FIRST VALUE OF RQ-TASK
1030    VALUE '01'
1040      DISPLAY (1) NOTITLE (AD=L CD=TU)
1050            'Procedure' *PROGRAM
1060            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1070            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1080            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1090            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1100            'Usr2' (TU) TX-USR2
1110    VALUE '02'
1120      DISPLAY (2) NOTITLE (AD=L CD=TU)
1130            'Procedure' *PROGRAM
1140            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1150            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1160            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1170            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1180            'Usr2' (TU) TX-USR2
1190    VALUE '03'
1200      DISPLAY (3) NOTITLE (AD=L CD=TU)
1210            'Procedure' *PROGRAM
1220            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1230            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
```

```
1240            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1250            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1260            'Usr2' (TU) TX-USR2
1270    VALUE '04'
1280      DISPLAY (4) NOTITLE (AD=L CD=TU)
1290            'Procedure' *PROGRAM
1300            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1310            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1320            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1330            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1340            'Usr2' (TU) TX-USR2
1350    NONE
1360      DISPLAY (5) NOTITLE (AD=L CD=TU)
1370            'Procedure' *PROGRAM
1380            'Date' (TU) TX-DATE 'Time'  (TU) TX-TIME
1390            'Task' (TU) RQ-TASK 'UserID'(TU) TX-USER
1400            'Cmd'  (TU) TX-CMD  'Fld'   (TU) RQ-FIELD
1410            'PreP' (TU) TX-PRE  'Fnr'   (TU) TX-FNR
1420            'Usr2' (TU) TX-USR2
1430  END-DECIDE
1440 *
1450 * Finally we write this info to a 'audit' file. In this case, we use the
1460 * Natural FNAT file for simplicity. Realistically, a separate 'audit' file
1470 * should be used. End Transaction (ET) must  not be issued because this
1480 * will be controlled by the application and not the trigger procedure.
1490 *
1500  MOVE #TEXT TO LOG-AREA.SRCTX (1.1)
1510  MOVE H'0001' TO SRC-SEQ
1520  ASSIGN LOG-AREA.SRCID = #SRCID
1530  STORE LOG-AREA
1540 *
1550 END
```

SAMP0004

```
0010 ************************************************************************
0020 *  Application: Adabas Triggers
0030 *   Subprogram: SAMP0004
0040 *     Function: Sample routine of processing by a stored procedure
0050 *               referential integrity - RESTRICT
0060 *               (assume that the primary key is on the EMPLOYEES file and
0070 *                the foreign key on the VEHICLES + MISCELLANEOUS files).
0080 * Trigger Defn: Definition on the trigger file is as follows:
0090 *                 File Number ....... 3
0100 *                 File Name ......... VEHICLES-FILE
0110 *                 Command Type ...... Delete
0120 *                 Long Field Name ... ** Any Field **
0130 *                 Adabas Field ...... **
```

```
0140 *                Field Prty/Seq .... ___
0150 *              Procedure Information
0160 *                Name (Subpgm)...... SAMP0004
0170 *                Pre Cmd Select .... Y  (Pre)
0180 *                Trigger Type ...... N  (Non-Participating)
0190 *                CALLNAT Params .... C  (Cntl Info + Resp)
0200 *                RecBuffer Access .. N  (No RecBuff Access)
0210 *
0220 *      Invoked: Invoked with deletes from VEHICLES/MISCELLANEOUS files
0230 *               Sample Routine: SAMPREF1
0240 *       Author: Adabas Development
0250 *         Date: December 1995
0260 ************************************************************************
0270 DEFINE DATA PARAMETER USING STPAPARM
0280           LOCAL
0290   01  VEHICLES  VIEW OF VEHICLES
0300     02  PERSONNEL-ID               /* foreign key: field AC
0310   01  MISCELLANEOUS VIEW OF MISCELLANEOUS
0320     02  PERSONNEL-ID               /* foreign key: field CA
0330   01  EMPLOYEES VIEW OF EMPLOYEES
0340     02  PERSONNEL-ID               /* primary key: field AA
0350   01  #FILE        (P5)
0360   01  #ISN        (P10)
0370   01  #PERS-NUM    (A8)
0380   01  CONTRL-BLK  (A80)
0390   01  REDEFINE CONTRL-BLK
0400     02  CB-FIL1   (A12)
0410     02  CB-ISN     (B4)
0420 END-DEFINE
0430 *
0440 *  First we extract the foreign key information
0450 *  i.e., get the ISN of the record in the ACB and read this record
0460 *    to extract the required information; i.e., the foreign key info.
0470 *    NOTE: With a delete, no data is passed in the record buffer.
0480 *
0490 MOVE RQ-CB  TO CONTRL-BLK         /* get the ACB of the originating cmd
0500 MOVE RQ-FNR TO #FILE              /* find out which file has the delete
0510 MOVE CB-ISN TO #ISN               /* ISN of the record to be deleted
0520 *
0530 IF #FILE = 3                      /* identify the file: Vehicles
0540   DO
0550     GET VEHICLES #ISN             /* get the value of the foreign key
0560     MOVE PERSONNEL-ID(0550) TO #PERS-NUM    /* get the key
0570   DOEND
0580 ELSE
0590   IF #FILE = 2                    /* or the Miscellaneous file
0600     DO
0610       GET MISCELLANEOUS #ISN      /* get the value of the foreign key
0620       MOVE PERSONNEL-ID(0610) TO #PERS-NUM  /* get the key
0630     DOEND
0640   ELSE                            /* a check for the unexpected...
0650     DO                            /* a trigger may have been defined wrong
```

```
0660       MOVE 913 TO RQ-RESP         /* either ignore or return an error
0670       ESCAPE ROUTINE              /* and exit
0680     DOEND
0690 *
0700 RESET RQ-RESP
0710 *
0720 * Now we check the primary file to see if the value exists. If yes
0730 * then we cannot allow this deletion; hence, we prevent any deletions
0740 * of the foreign key files if a record with the same key exists on the
0750 * primary file.
0760 *
0770 * NOTE: With the setting of RESP, consideration should be given to
0780 *       ambiguities. While the command will receive a response 155
0790 *       (pre-trigger) or 156 (post-trigger), the additions field will
0800 *       contain the error returned from this procedure. The value
0810 *       could be in the form of an Adabas response (1-255) or a
0820 *       Natural error (e.g., 954 or 935 or 3009); therefore, a
0830 *       user-specified error from the procedure should be something
0840 *       outside these ranges........for simplicity.
0850 *
0860 FIND EMPLOYEES WITH PERSONNEL-ID = #PERS-NUM
0870   MOVE 901 TO RESP                 /* it does: delete may not be done
0880   ESCAPE ROUTINE
0890 CLOSE LOOP(0860)
0900 *
0910 END
```

SAMP0005

```
0010 ************************************************************************
0020 *  Application: ADASTP
0030 *   Subprogram: SAMP0005
0040 *     Function: Sample routine of processing by a stored procedure
0050 *               referential integrity - CASCADE
0060 *               (assume that the primary key is on the EMPLOYEES file
0070 *               and foreign keys on the VEHICLES + MISCELLANEOUS files).
0080 * Trigger Defn: Definition on the trigger file is as follows:
0090 *                 File Number ....... 4
0100 *                 File Name ......... EMPLOYEES
0110 *                 Command Type ...... Update
0120 *                 Long Field Name ... PERSONNEL-ID
0130 *                 Adabas Field ...... AA
0140 *                 Field Prty/Seq .... 010
0150 *               Procedure Information
0160 *                 Name (Subpgm)...... SAMP0005
0170 *                 Pre Cmd Select .... Y  (Pre)
0180 *                 Trigger Type ...... P  (Participating)
0190 *                 CALLNAT Params .... C  (Cntl Info + Resp)
```

```
0200 *                 RecBuffer Access .. A  (May be Accessed)
0210 *
0220 *      Invoked: Invoked with updates to the EMPLOYEES PERSONNEL-ID
0230 *               Sample routine: SAMPREF2
0240 *       Author: Adabas Development
0250 *         Date: December 1995
0260 ************************************************************************
0270 DEFINE DATA PARAMETER USING STPAPARM
0280            LOCAL     USING STPLRBE      /* parameters for call to STPRBE
0290            LOCAL
0300  01  EMPLOYEES VIEW OF EMPLOYEES
0310    02  PERSONNEL-ID                     /* primary key: field AC
0320  01  MISCELLANEOUS VIEW OF MISCELLANEOUS
0330    02  PERSONNEL-ID                     /* foreign key: field CA
0340  01  VEHICLES  VIEW OF VEHICLES
0350    02  PERSONNEL-ID                     /* foreign key: field AA
0360  01  FUNC         (A4)
0370  01  #ISN        (P10)
0380  01  #PERS-NUM    (A8)
0390  01  CONTRL-BLK  (A80)
0400  01  REDEFINE CONTRL-BLK
0410    02  CB-FIL1   (A12)
0420    02  CB-ISN     (B4)
0430 END-DEFINE
0440 *
0450 *  First we extract the foreign key information
0460 *  There are two ways to pick this up:
0470 *
0480 *      1) Since the value is in the record buffer, we can use STPRBE to
0490 *         extract the required information; i.e., the primary key
0500 *         information. There are three ways to do this...in this case:
0510 *
0520 *             A) identify the field by its long name; i.e., PERSONNEL-ID
0530 *             B) identify the field by its short name; i.e., AA
0540 *             C) identify the location and length in the record buffer
0550 *
0560 *      2) Get the ISN of the record in the ACB and read this record to
0570 *         extract the required information; i.e., the primary key
0580 *         information. However, this is the old value and cannot be used
0590 *         in this example.
0600 *
0610 *
0620 * OPTION 1A
0630 *
0640 * Function 'GV' -- GET field value using the long field name
0650 *     This enables the caller to obtain information about a specific
0660 *     field which is determined according to the long field name
0670 *     passed in the parameters to STPRBE.
0680 *
0690 RESET #PERS-NUM
0700 MOVE 'GV'           TO FUNC
0710 MOVE 'PERSONNEL-ID' TO RBE-FIELD-NAME /* and identify the corresponding
```

```
0720 *                                        field for this file
0730 MOVE 8              TO RBE-LENGTH     /* default or give override length
0740 *                                     /* length in FB could have been used
0750 CALL 'STPRBE' FUNC RBE-AREA #PERS-NUM
0760 PRINT *PROGRAM 'Option 1A returned ..' #PERS-NUM 'resp' RBE-RESP
0770 IF RBE-RESP NE 0                      /* successful ?
0780  DO
0790    PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
0800          RBE-ERROR 'subcode' RBE-SUBCODE 'for func GV'
0810    MOVE RBE-RESP TO RESP              /* indicate this
0820    ESCAPE ROUTINE                     /* and exit
0830  DOEND
0840 *
0850 * OPTION 1B
0860 *
0870 * Function 'GV' -- GET field value using short field name
0880 *     This enables the caller to obtain information about a specific
0890 *     field which is determined according to the short field name
0900 *     passed in the parameters to STPRBE.
0910 *     NOTE: '**' in field name means user-supplied details in short name
0920 *
0930 RESET #PERS-NUM
0940 MOVE 'GV'     TO FUNC
0950 MOVE '**'     TO RBE-FIELD-NAME    /* special notation for this request
0960 MOVE RQ-FIELD TO RBE-ADA-FIELD     /* get field name that fired the
0970 *                                     trigger from the parm area...OR.....
0980 IF NOT (RQ-FIELD = 'AA')           /* if we know the field....
0990   MOVE 'AA'   TO RBE-ADA-FIELD     /* identify the specific field name
1000 MOVE 8        TO RBE-LENGTH        /* for a maximum length of 8 bytes
1010 CALL 'STPRBE' FUNC RBE-AREA #PERS-NUM
1020 PRINT *PROGRAM 'Option 1B returned ..' #PERS-NUM 'resp' RBE-RESP
1030 IF RBE-RESP NE 0                   /* successful
1040  DO
1050    PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
1060          RBE-ERROR 'subcode' RBE-SUBCODE 'for func GV'
1070    MOVE RBE-RESP TO RESP           /* indicate this
1080    ESCAPE ROUTINE                  /* and exit
1090  DOEND
1100 *
1110 * OPTION 1C
1120 *
1130 * Function 'GR' -- GET RB value using RB offset + length
1140 *     This enables the caller to obtain information based on a
1150 *     certain location; hence, RBE-OFFSET specifies the start
1160 *     position and RBE-LENGTH specifies the length.
1170 *
1180 RESET #PERS-NUM
1190 MOVE 'GR' TO FUNC
1200 MOVE 1    TO RBE-OFFSET            /* start at the beginning
1210 MOVE 8    TO RBE-LENGTH            /* for a max. length of 50 bytes
1220 CALL 'STPRBE' FUNC RBE-AREA #PERS-NUM
1230 PRINT *PROGRAM 'Option 1C returned ..' #PERS-NUM 'resp' RBE-RESP
```

```
1240 IF RBE-RESP NE 0
1250  DO
1260    PRINT *PROGRAM 'received an error from the STPRBE routine. Error:'
1270          RBE-ERROR 'subcode' RBE-SUBCODE 'for func GR'
1280    MOVE RBE-RESP TO RESP
1290    ESCAPE ROUTINE
1300  DOEND
1310 *
1320 *  NOTE: Only one of the options need be used to extract the value
1330 *
1340 RESET RQ-RESP
1350 *
1360 * Now, we read the original record, which is not yet changed; hence the
1370 * reason for setting this up as a pre-trigger, to see if the value
1380 * (PERSONNEL-ID in this case) has changed.
1390 *
1400 MOVE RQ-CB  TO CONTRL-BLK          /* get the original ACB of the A1/4
1410 MOVE CB-ISN TO #ISN               /* extract the ISN of the record
1420 GET EMPLOYEES #ISN                /* read the, so far, unchanged data
1430 IF PERSONNEL-ID(1420) = #PERS-NUM  /* have the numbers changed?
1440   ESCAPE ROUTINE                  /* no, then exit
1450 *
1460 * Now that we have observed that the primary key has changed, we must
1470 * read all the files with a foreign key and CASCADE the update.
1480 *
1490 FIND VEHICLES WITH PERSONNEL-ID = PERSONNEL-ID(1420) /* Vehicles file
1500  ASSIGN PERSONNEL-ID(1490) = #PERS-NUM
1510  UPDATE (1490)
1520 CLOSE LOOP(1490)
1530 *
1540 FIND MISCELLANEOUS WITH PERSONNEL-ID = PERSONNEL-ID(1420) /* Misc file
1550  ASSIGN PERSONNEL-ID(1540) = #PERS-NUM
1560  UPDATE (1540)
1570 CLOSE LOOP(1540)
1580 *
1590 * Issuing an ET now, is not valid with a participating trigger because
1600 * the originating command (A1/Update) has not yet been executed and
1610 * the originating user expects to do the ET once the update is complete.
1620 * If this ET were done here, the A1/4 (pre-trigger) would receive a
1630 * response 144 because the ISN would be released. If the originating
1640 * user had to do other updates, then a misplaced ET (End Transaction)
1650 * could cause a loss of data integrity across the files.
```

```
1660 *
1670 END
```

SAMPREF1

```
0010 ************************************************************************
0020 * Application: Adabas Triggers
0030 *     Program: SAMPREF1 - Example of referential integrity (restrict)
0040 *    Function: SPT routine to delete records from the Vehicles file
0050 *              Invoked with a delete trigger as shown below:
0060 *
0070 *              Trigger Information
0080 *                File Number ....... 3
0090 *                File Name ......... VEHICLES-FILE
0100 *                Command Type ...... Delete
0110 *                Long Field Name ... ** Any Field **
0120 *                Adabas Field ...... **
0130 *                Field Prty/Seq .... ___
0140 *              Procedure Information
0150 *                Name (Subpgm)...... SAMP0004
0160 *                Pre Cmd Select .... Y  (Pre)
0170 *                Trigger Type ...... N  (Non-Participating)
0180 *                CALLNAT Params .... C  (Cntl Info + Resp)
0190 *                RecBuffer Access .. N  (No RecBuff Access)
0200 *
0210 ************************************************************************
0220 DEFINE DATA LOCAL
0230   01  #NUMBER    (A8)
0240   01  VEHICLES VIEW OF VEHICLES
0250     02  PERSONNEL-ID
0260 END-DEFINE
0270 *
0280 INPUT (AD=TMIL'_' CD=NE)
0290      'Trigger Example for Referential Integrity - RESTRICT' (YEI)
0300   // 'Enter Personnel Number ..' (TU) #NUMBER
0310 *
0320 IF #NUMBER = MASK('.')                    /* exit?
0330   STOP                                    /* yes
0340 IF #NUMBER = ' '                          /* a number must be specified
0350   REINPUT 'Invalid Number specified'
0360 *
0370 FIND VEHICLES WITH PERSONNEL-ID = #NUMBER /* find the record to be deleted
0380   DELETE(0370)                            /* issue the Delete request
0390   END TRANSACTION                         /* finalize the delete
0400   REINPUT 'Record has now been deleted'   /* confirm and restart
0410 CLOSE LOOP
0420 IF *NUMBER(0370) = 0                      /* validate existence of number
0430   REINPUT 'Invalid Personnel Number specified'
```

```
0440 *
0450 * Below, any error handling may be done. With a trigger, a procedure
0460 * could return a non-zero response. This would result in the trigger
0470 * command (the Delete in this case) receiving a response 155. Pre-triggers
0480 * receive a response 155 and post-triggers receive a response 156.
0490 *
0500 ON ERROR                          /* handle any errors from the trigger
0510   DO
0520     BACKOUT TRANSACTION           /* release the held record/ISN
0530     INPUT (AD=O CD=YE) 8X '*** Warning ***' (REI)
0540       // 'Personnel Number' (YE) #NUMBER 'NOT deleted' (YEI)
0550        / 'Response' (YE) *ERROR-NR 'received for this request' (YE)
0560       // 4X 'Press Enter to continue' (REI)
0570     STACK TOP COMMAND *PROGRAM    /* return to start of this routine
0580     STOP
0590   DOEND
0600 END
```

SAMPREF2

```
0010 ************************************************************************
0020 * Application: Adabas Triggers
0030 *     Program: SAMPREF2 - Example of referential integrity (Cascade)
0040 *    Function: SPT routine to update records on the Employees file
0050 *              Invoked with an update trigger as shown below:
0060 *
0070 *              Trigger Information
0080 *                File Number ....... 4
0090 *                File Name ......... EMPLOYEES
0100 *                Command Type ...... Update
0110 *                Long Field Name ... PERSONNEL-ID
0120 *                Adabas Field ...... AA
0130 *                Field Prty/Seq .... 10_
0140 *              Procedure Information
0150 *                Name (Subpgm)...... SAMP0005
0160 *                Pre Cmd Select .... Y  (Pre)
0170 *                Trigger Type ...... P  (Participating)
0180 *                CALLNAT Params .... C  (Cntl Info + Resp)
0190 *                RecBuffer Access .. A  (May be Accessed)
0200 *
0210 ************************************************************************
0220 DEFINE DATA LOCAL
0230   01  #NUMBER    (A8)
0240   01  EMPLOYEES VIEW OF EMPLOYEES
0250     02  PERSONNEL-ID
0260     02  FIRST-NAME
0270     02  NAME
0280     02  MIDDLE-NAME
```

```
0290 END-DEFINE
0300 *
0310 REPEAT
0320 *
0330 INPUT (AD=TMIL'_' CD=NE)
0340       'Trigger Example for Referential Integrity - CASCADE' (YEI)
0350    // 'Enter Personnel Number ..' (TU) #NUMBER
0360 *
0370 IF #NUMBER = MASK('.')                    /* exit ?
0380   STOP                                    /* yes
0390 *
0400 IF #NUMBER = ' '                          /* a number must be specified
0410   REINPUT 'Invalid Number specified'
0420 *
0430 FIND EMPLOYEES WITH PERSONNEL-ID = #NUMBER /* read the record
0440   INPUT (AD=MIL CD=NE)                    /* show data for doing updates
0450         'Enter Employee Details Below for Update:-' (YEI)
0460     // 'Personnel Number ...' (TU) PERSONNEL-ID
0470      / 'Last Name ..........' (TU) NAME
0480      / 'First Name .........' (TU) FIRST-NAME
0490      / 'Middle Name ........' (TU) MIDDLE-NAME
0500 *
0510 * Validation of the changes may now be done as required
0520 *
0530   UPDATE(0430)                    /* make the database changes
0540   END TRANSACTION                 /* and finalize them
0550   ESCAPE BOTTOM
0560 CLOSE LOOP
0570 IF *NUMBER(0430) = 0
0580   REINPUT 'Invalid Personnel Number specified'
0590 ELSE
0600   INPUT NO ERASE ////// 4X 'Record has now been updated' (YEI)
0610 *
0620 CLOSE LOOP(0310)                  /* repeat loop
0630 *
0640 * Below, any error handling may be done. With a trigger, a procedure
0650 * could return a non-zero response. This would result in the trigger
0660 * command (the update in this case) receiving a response 155. Pre-triggers
0670 * receive a response 155 and post-triggers receive a response 156.
0680 *
0690 ON ERROR                           /* handle any errors from the trigger
0700   DO
0710     BACKOUT TRANSACTION            /* release the held record/ISN
0720     INPUT (AD=O CD=YE) 8X '*** Warning ***' (REI)
0730       // 'Personnel Number' (YE) #NUMBER 'NOT Updated' (YEI)
0740        / 'Response' (YE) *ERROR-NR 'received for this request' (YE)
0750       // 4X 'Press Enter to continue' (REI)
0760     STACK TOP COMMAND *PROGRAM     /* return to start of this routine
0770     STOP
0780   DOEND
0790 END
```

索引

P

R

S

T

い

か

き

こ

さ

す

て

と

は

ひ

ふ

ほ

ま

ゆ

り

れ

わ