AIX バージョン 7.2

拡張アカウンティング・サブシステム

IBM

AIX バージョン 7.2

拡張アカウンティング・サブシステム

IBM

お願い

本書および本書で紹介する製品をご使用になる前に、 49 ページの『特記事項』 に記載されている情報をお読みください。

本書は AIX バージョン 7.2 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

原典： AIX Version 7.2
Advanced Accounting subsystem

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

本書について

本書は、拡張アカウンティング・サブシステムをセットアップ、管理、保守する方法についての概念および手順を示します。対象読者はシステム管理者です。ここには、プロジェクト、ポリシー、トランザクション・アカウンティング、間隔アカウンティング、データ集約などの情報が含まれています。この情報は、オペレーティング・システムに付属のドキュメンテーション CD にも収録されています。

強調表示

本書では、以下の強調表示の規則を使用しています。

太字	コマンド、サブルーチン、キーワード、ファイル、構造体、ディレクトリー、およびシステムによって名前が事前に定義されているその他の項目を示します。また、ユーザーが選択するボタン、ラベル、およびアイコンなどのグラフィカル・オブジェクトも示します。
イタリック	ユーザーが実際の名前や値を指定するパラメーターを示します。
モノスペース	具体的なデータ値の例、画面に表示されるテキストの例、プログラマーとしてユーザーが記述するプログラム・コード部分の例、システムからのメッセージ、またはユーザーが実際に入力する情報を示します。

AIX における大/小文字の区別

AIX® オペレーティング・システムでは、すべて大文字小文字の区別があります。これは、大文字と小文字を区別するということです。例えば、**ls** コマンドを使用するとファイルをリストできます。 `LS` と入力すると、システムはそのコマンドが「`not found` (見つかりませんでした)」と応答します。同様に、**FILEA**、**FiLea**、および **filea** は、同じディレクトリーにある場合でも 3 つの異なるファイル名です。予期しない処理が実行されないように、常に正しい大/小文字を使用するようにしてください。

ISO 9000

当製品の開発および製造には、ISO 9000 登録品質システムが使用されました。

拡張アカウンティング・サブシステム

このトピックでは、拡張アカウンティングをセットアップ、管理、保守する方法についての概念および手順を示します。対象読者はシステム管理者です。

ここには、プロジェクト、ポリシー、トランザクション・アカウンティング、間隔アカウンティング、データ集約などの情報が含まれています。

拡張アカウンティング・サブシステムの概要

拡張アカウンティング・サブシステム (以下、拡張アカウンティングとします) はメインフレーム・テクノロジーに基づくサブシステムで、間隔アカウンティング、データ集約、アカウンティング・データの動的分類といった機能があります。さまざまなコンピューティング環境に合わせて拡張アカウンティングをカスタマイズすることができます。請求アプリケーションが必要とする特定の種類のレコードを生成するよう、拡張アカウンティングを構成できます。

拡張アカウンティングはさまざまなシステム・リソースの使用状況に関する情報を提供するため、包括的なチャージバック・ストラテジーの開発が可能になります。ディスク、ネットワーク・インターフェース、仮想デバイス、ファイルシステム、プロセッサー、メモリーなどのリソースに関するアカウンティング・データを収集できます。間隔アカウンティングによって、システム管理者が定義した時間間隔ごとにこれらのデータを表示して、履歴ビューを開発することができます。これは、キャパシティー・プランニングなどの用途に活用することができます。

また、拡張アカウンティングは、以前のアカウンティング・ツールから新しい統計を生成します。例えば、プロセス・レコードはマイクロ秒レベルの CPU 時間、経過時間に基づくメモリー積算値 (標準的な UNIX アカウンティングでは CPU 時間に基づきます)、ローカルおよび分散論理ファイル入出力、ローカルおよびリモート・ソケット入出力を生成します。

間隔アカウンティングを使って定期的にアカウンティング・データを記録することにより、より正確な請求を生成できるようになります。 アクティブ・プロセスに関する中間的なプロセス・レコードを生成するよう、拡張アカウンティングを構成できます。このようなレコードを完成したプロセス・レコードに追加すれば、システム・リソースの使用量合計を反映した請求を生成することができます。

データ集約は、データ・ファイルに書き込まれるデータの量を制御します。これにより、拡張アカウンティングの実行に必要なリソース負荷の総量が減り、システム・パフォーマンスが改善されます。集約によって、アカウンティング・レコードが追加されるとき、より少ないレコードがアカウンティング・ファイルに書き込まれるので、システムの入出力要件が最小化されます。また、アプリケーションやミドルウェアから見て透過的です。

ポリシーとは、プロセスを自動的に分類するための規則です。分類はユーザー、グループ、アプリケーションに基づいて行われ、請求可能なエンティティーごとにシステム・リソース使用量がカテゴリー化されます。これらのカテゴリーを*プロジェクト* といいます。

提供されている API を使って、アプリケーションやミドルウェアはそれ自体のワークロードのトランザクション特性を記述できるので、サーバー・プロセスのチャージバックが可能になります。これらの API は

トランザクションを定義および記述し、可能な場合にはエンド・ユーザーを識別します。拡張アカウンティングはトランザクションによるリソース使用を測定し (可能な場合)、この情報をすべてアカウンティング・ファイルに記録します。

データ・ファイル

拡張アカウンティングによって記録されるすべての統計は、データ・ファイルに書き込まれます。データ・ファイルがいっぱいになると、請求アプリケーションはそのファイルを処理することができます。ファイルが処理された後、そのファイルは再使用され、このサイクルが繰り返されます。

アカウンティング統計の収集を開始するには、独自のアカウンティング・データ・ファイルを作成しなければなりません。そのためには、企業のニーズを見極め、どの程度のデータを収集するかを判別する必要があります。拡張アカウンティングを開始し、一定期間にわたって実行し続ければ、どの程度の量のデータが生成されるかを判別できます。こうすれば、アカウンティング用にどれほどのディスク・スペースを予約すべきか、およびデータ管理用にどれほどの数のファイルが必要かを把握できます。

拡張アカウンティングを常にアクティブにするために、少なくとも 2 つのデータ・ファイルを指定することをお勧めします (これは必須ではありません)。拡張アカウンティングは一度にただ 1 つのファイルに書き込みますが、書き込み時にはそのファイルへの排他的アクセスを必要とします。ファイルが 2 つあれば、拡張アカウンティングが 1 つのファイルに書き込んでいる間に、請求アプリケーションがもう 1 つのファイルを処理することができます。

拡張アカウンティング・データ・ファイルは パーティション・モビリティーをサポートします。移行の前に、ソース区画上で、すべてのアカウンティング・データはアカウンティング・ファイルにフラッシュされます。データ・フラッシュが完了すると、ファイルはクローズされます。移行後、ファイルが使用可能な場合は、新しいアカウンティング・ファイルが宛先区画にオープンします。

データ・ファイルのライフ・サイクル

各データ・ファイルはアカウンティング処理の進行につれて、1 つのライフ・サイクルをたどっていきます。

データ・ファイルのライフ・サイクルは、次のようなプロセスになります。

1. 事前に割り振られた複数のデータ・ファイルからなるプールが作成され、拡張アカウンティングに登録されます。
2. アカウンティング統計がシステムから収集され、アクティブなデータ・ファイルにデータが書き込まれます。
3. データ・ファイルのサイズ (ファイル作成時にあらかじめ決定される) は、ファイルに書き込まれるデータの量を制限します。データ・ファイルがいっぱいになると、そこにはデータを書き込めなくなります。すると、拡張アカウンティングは次の空のデータ・ファイルに切り替えて、そこにデータを書き込み始めます。
4. データ・ファイルがいっぱいになると、請求アプリケーションなどの後処理ツールを使用して、アカウンティング・データを請求処理用の統計にすることができます。
5. データ・ファイルからデータを抽出して処理した後、ファイルは再使用のためにリセットされます。

データ・ファイルの作成

拡張アカウンティングで記録された統計情報を格納するためのデータ・ファイルを作成できます。

このシナリオでは、System Management Interface Tool (SMIT) またはコマンド・ラインのいずれかを使用してデータ・ファイルを作成する方法を説明します。

考慮事項

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。 したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. root ユーザーに切り替えます。
2. コマンド・ラインで `smit create_aacct_file` と入力します。
3. 「**アカウンティング・データ・ファイル**」フィールドに、作成するファイルの名前を入力します。 完全修飾パス名を指定する必要があります。
4. 「**事前割り振りサイズ**」フィールドに、作成するファイルのサイズ (メガバイト) を入力します。
5. Enter キーを押して設定値を保存し、データ・ファイルを作成します。

例えば、コマンド・ラインで `testfile` という名前の 2 MB のデータ・ファイルを作成するには、`acctctl fadd /var/aacct/testfile 2` と入力します。

データ・ファイル管理

データ・ファイルの管理には、コマンド・ラインと SMIT を使用することができます。

次の表には、アカウンティング・データ・ファイルの管理に使用できるコマンドと SMIT 高速パスが示されています。

表 1. データ・ファイル管理コマンド

タスク	コマンド	SMIT 高速パス
指定のファイル名とサイズでアカウンティング・ファイルを割り当てて定義する。デフォルトのサイズ単位はメガバイト。	**acctctl fadd** *file size*	**smit create_aacct_file**
指定したアカウンティング・ファイルをアカウンティング・サブシステムから除去する。この処理によって、ファイルはファイルシステムからは除去されません。	**acctctl frm** *file*	**smit release_aacct_file**
指定したファイルがアカウンティング・サブシステムで再使用可能であることを示す。	**acctctl freset** *file*	**smit reuse_aacct_file**
指定したファイルの状態と現在の使用状況を照会する。ファイルを指定しない場合、すべてのファイルについて照会する。	**acctctl fquery** [*file*]	**smit list_aacct_file**
拡張アカウンティングが新しいアカウンティング・データ・ファイルに切り替えるよう強制する。オプションで、新規ファイルを指定することもできます。	**acctctl fswitch** [*file*]	**smit switch_acct_file**

詳しくは、「*Commands Reference, Volume 1*」を参照してください。

通知メッセージ

拡張アカウンティングは、登録済みのアカウンティング・データ・ファイルに継続的にアカウンティング・データを書き込みます。拡張アカウンティングがデータを記録する場所を常に確保するために、これらのファイルの状態を監視する必要があります。

拡張アカウンティングは、データ・ファイルの状況およびサブシステムの状況をモニターするためのメッセージを生成します。イベントが発生したときにアクションをトリガーするには、データ・ファイルに関連したメッセージを使用してください。ファイルが 90% いっぱいになったとき、および 100% いっぱいになったときにメッセージが生成され、管理アクションが必要であることを示します。アカウンティング・データ・ファイルの管理には **acctctl** コマンドを使用します。 acctctl コマンドの追加情報については、**acctctl** を参照してください。

アカウンティング・データ・ファイルを調査および管理するシェル・スクリプトやコマンドを定期的に実行するには、**cron** 機能を使用できます。デフォルトでは、AIX は **syslog** デーモンを使用して拡張アカウンティング・サブシステムに関する情報を記録します。 **syslog** デーモンの詳細については、「*Commands Reference, Volume 5*」を参照してください。

アカウンティング・データ・ファイルからアカウンティング・レコードを抽出するには、**readaacct** コマンドを使用できます。ただし、これはサンプル・コマンドであり、完全にはテストおよび保守されていないことに注意してください。アカウンティング・データ・ファイルからのアカウンティング・レコードの抽出について、詳しくは 33 ページの『レポートおよび分析』を参照してください。

電子メール通知のメッセージ

拡張アカウンティングでは、データ・ファイル関連の活動状況を通知するメッセージを電子メールで送信します。

電子メール通知を使用するには、手動で構成する必要があります。電子メール通知の構成について、詳しくは 『電子メール通知の構成』を参照してください。

以下のメッセージが電子メール通知を介して送られます。

表 2. 電子メール通知のメッセージ

件名行	本文
AACCT: File nearly full	1400-330: The accounting file is 90% full.
AACCT: File ready	1400-331: The accounting file is ready for processing.
AACCT: Subsystem out of files	1400-332: The Advanced Accounting subsystem has run out of files for use.
AACCT: Subsystem out of kernel buffers	1400-333: The Advanced Accounting subsystem has run out of kernel buffers.
AACCT: File I/O error	1400-334: The accounting file encountered an I/O error while writing.

ほとんどの電子メール・プログラムにはフィルター機能があるため、メッセージが受信されたとき、シェル・スクリプトやコマンドは特定のアクションをトリガーすることができます。拡張アカウンティングに関連した着信メッセージを識別するには、文字列 AACCT を使用してください。メッセージの重要度を識別するには、メッセージ本文のメッセージ番号を使用してください。

電子メール通知の構成

データ・ファイルが 90% いっぱいになったことを知らせる通知を送信するように、拡張アカウンティングに対して電子メール通知を構成できます。拡張アカウンティング・サブシステムが新規データ・ファイルに切り替えた場合に、もう 1 つ通知を送信できます。

以下のシナリオでは、「anywhere@anywhere.com」のアドレスに対する電子メール通知を設定します。このシナリオでは、SMIT またはコマンド・ラインのいずれかを使用して電子メール通知を構成する方法を説明します。

考慮事項

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。 したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. root ユーザーに切り替えます。
2. コマンド・ラインで `smit admin_notify` と入力します。
3. 「**管理通知状況**」フィールドにカーソルがある状態で Tab キーを押し、電子メール通知状況をオフからオンに変更します。
4. 「**通知用の電子メール ID**」フィールドに、`anywhere@anywhere.com` と入力します。
5. Enter キーを押して設定値を保存し、電子メール通知をオンにします。

有効な電子メール・アドレスを指定するには、最初に SMIT インターフェースを使用する必要があります。その後、コマンド・ラインで通知をオンまたはオフに設定することができます。コマンド・ラインで電子メール通知を活動化すると、通知は最近使用された電子メール・アドレスに送信されます。コマンド・ラインで電子メール通知を構成するには、次のように入力します。

```
acctctl email anywhere@anywhere.com
acctctl email on
```

プロジェクト

プロジェクトとは、ユーザー、部門、部署、会社、タスクなど、請求可能なエンティティーです。

各プロジェクトは、プロジェクト番号、プロジェクト属性、およびプロジェクト名で構成され、これらをまとめてプロジェクト定義といいます。プロジェクト定義は、プロジェクト定義データベースに入力されます。

プロジェクトはアカウンティング・レコードに書き込まれます。レポート・コマンドおよび分析コマンドは、システム・プロジェクト定義データベースのエントリーを検索することによってプロジェクト番号をプロジェクト名に変換します。理論的には、*プロジェクト* は請求アプリケーションによって保守される重要なビジネス・データ (顧客名、請求先住所、アカウント番号、サービス・レベル契約など) への索引です。

プロジェクト番号は、プロジェクト割り当て規則の集合であるプロジェクト割り当てポリシーによって割り当てられます。それぞれの規則には分類基準が含まれ、これが完全に満たされると分類結果を生成します。*分類結果* とは、分類対象のオブジェクト (通常はプロセス) に論理的に割り当てられるプロジェクト・リストです。分類基準は、ポリシーのタイプに依存します。

プロジェクト・リストは、プロジェクトの手動割り当てを可能にします。複数プロジェクトからなるリストが指定された場合、ユーザーは現在のプロジェクト割り当てを、リスト内の他のプロジェクトに変更できます。これによって、さまざまなプロジェクトの下でジョブを起動することが可能になり、複数のクライアントに関する処理を行うときに役立ちます。システム管理者は複数のプロジェクトを任意のユーザーに割り当てた後、手動でプロジェクト割り当てを変更することができます。

プロジェクト分類のセマンティクス

プロジェクト分類のセマンティクスは、プロジェクトを分類および割り当てるときに使用されます。

それぞれのサブルーチン exec()、initp()、setuid()、および setgid() ごとに、プロジェクト割り当て規則を使ってプロセスが再分類され、現在のプロジェクト割り当てを変更すべきかどうか判別されます。プロジェクト割り当て規則がロードされていない場合、または規則が正常に適用されない場合には、現在のプロジェクト ID が使用されます。

デフォルトのプロジェクト・システム ID はゼロ (0) です。これは、アカウンティングが使用可能になる前に基本システム・プロセスに適用され、一般的なシステム・オーバーヘッドを表すために使用されることもあります。これが割り当てられた後、**fork()** カーネル・サービスと **creatp()** カーネル・サービスを使用して親プロセスから子プロセスへプロジェクトが継承されます。

アプリケーション・フィルターの使用法は、**initp()** カーネル・サービスと exec() サブルーチンの間で異なります。前者の場合、開始されるカーネル・プロセスのコマンド名を使って分類が実行されます。コマンド名を表示するには、**ps** コマンドを使用します。後者の場合、ポリシーによって指定された完全修飾パス名の FID とともに、実行可能ファイルの FID (i ノード + デバイス番号) を使って分類が実行されます。ポリシーのロード時に算出可能な FID だけが受け入れられます。

アプリケーションを明示的に指定する規則を介してプロセスが分類される場合、作業のブロックにラベルを付けるのが目的であるため、そのプロジェクト ID はプロセスと子に適用される必要があります。つまり、プロセス内の後続の exec()、setgid()、setuid() サブルーチンおよびそれらの子は、プロジェクトの再分類を実行しません。内部的に、これらのプロセスは、アプリケーションを明示的に指定した規則を介して分類されたことを示すスティッキー・ビットによって識別されます。スティッキー・ビットを表示するには、**ps -P** コマンドを使用します。

以下は、**ps -P** コマンドを実行したときの出力例です。 *dev* プロジェクトの前のアスタリスク (*) は、スティッキー・ビットが付加されていることを示します。

```
UID    GID    PID    TTY    TIME    PROJECT    SUBPROJ    CMD
0      0      16922  pts/1  0:00    *dev       0          ps
0      0      19206  pts/1  0:00    *dev       0          acctras
0      0      22286  pts/1  0:00    *dev       0          ksh
```

新しいポリシー・ファイルをロードすると、システム内のスティッキー・ビットを持たないすべてのプロセスが再分類されます。スティッキー・ビットを持つプロセスは、変更不能です。

異なるプロジェクト ID がプロセスに割り当てられた場合、プロセス・アカウンティング・レコードは **acct** ファイルに書き込まれ、プロジェクトによるリソース使用量が正確に報告されます。これが行われるときには常に、プロセスに関するアカウンティング統計がゼロにリセットされます。

プロジェクトの手動分類

ユーザーは現在のプロジェクト割り当てを手動で変更することができます。

特権を持たないユーザーの場合、権限はポリシー・ファイルを介して提供されます。規則内に最初にリストされたプロジェクトはデフォルト・プロジェクトと見なされ、ユーザーが現在のプロジェクト割り当てをリスト内の他のプロジェクトに変更しない限り、それが自動的に選択されます。 現在のプロジェクト割り当てを変更すると現行セッションにだけ適用され、他のジョブに暗黙的に適用されることはありません (ただし、シェルのような親プロセスは例外となる可能性があります)。

以下の規則は、プロジェクト・リストの例です。

表 3. プロジェクト・リストの例

ユーザー	グループ	アプリケーション	プロジェクト (複数の場合もあり)
User1	-	-	Chemistry, Biology

リスト内の最初のプロジェクトは Chemistry (化学) であるため、これがデフォルト・プロジェクトです。User1 は、現在の自分のプロジェクト割り当てを Biology (生物学) に変更できます。現在のプロジェクト割り当てはプロジェクト分類のパラメーターとして使用され、これにはプロジェクト割り当てとして Biology が含まれる可能性があります。

また、アプリケーションを指定する規則でプロジェクト・リストを指定することもできます。こうすれば、管理者はアプリケーションの特定のインスタンスにプロジェクトを関連付けることができます。サーバー・プロセスの場合には、これが重要です。コスト構造が異なる特定のリソースを管理するために、サーバー・プロセスは外部的に構成されることがあるためです。

システム管理者はこのメカニズムを利用して、使用量ベースのアカウンティングを直接サポートしないリソースに関して間接的に課金することができます。例えば、システム管理者が特定のデータベースへのアクセスに対して別の価格で課金したい場合、これらのデータベースを管理するために別の Oracle インスタンスを開始することができます。

以下の規則は、アプリケーションを指定する方法を例示しています。

表 4. アプリケーション指定の例

ユーザー	グループ	アプリケーション	プロジェクト (複数の場合もあり)
-	-	/usr/bin/Oracle	Dataset1, Dataset2
User1	-	-	Chemistry, Biology, Dataset2

最初の規則では、アプリケーション Oracle に 2 つのプロジェクトが関連付けられています。 Oracle が開始され、現在のプロジェクトが Dataset2 でない場合には、Dataset1 が選択されます。そうでない場合、Dataset2 が選択されます。 User1 の現在のプロジェクト割り当てが Chemistry または Biology である場合、このユーザーが自分のデータベース・アプリケーションを開始すると、プロジェクト ID Dataset1 が割り当てられます。また、User1 は自分の現在のプロジェクト割り当てを Dataset2 に変更して、データベース・アプリケーションを再起動することもできます。こうすると、プロジェクト割り当てが Dataset2 になります。

特権ユーザー (root ユーザー、または拡張アカウンティング機能を持つユーザー) には、これらの規則を無視できる強制オプション (**-f** フラグ) があります。

環境変数によるプロジェクト分類

プロセスの分類に環境変数を使用することができます。

プロセスを分類するには、環境変数 PROJECTNAME=*project name* および PROJECTID=*project id* を使用することができます。root 権限または拡張アカウンティング管理者機能を持つ場合、プロジェクト割り当て規則への照会なしで割り当てが行われます。これに該当しない場合、プロジェクトはロードされたポリシーに照らして検証されます。 PROJECTNAME と PROJECTID の両方が指定されている場合は、PROJECTNAME が PROJECTID より優先されます。

プロジェクトの相対的分類

拡張アカウンティング機能を持つユーザー ID やグループ ID に関連させて、プロジェクトを割り当てることができます。

相対的にプロジェクトを分類するには、プロジェクト・リスト内で $UID および $GID キーワードを使用することにより、プロジェクト・コードの算出をシステムに指示します。この場合、「キーワード」または「キーワード + 定数」という単純式を使用します (定数 は 10 進数または 16 進数 0xffff)。

以下の規則は、プロジェクトの相対的分類の使用法を示しています。

表 5. プロジェクトの相対的分類

ユーザー	グループ	アプリケーション	プロジェクト
*	-	-	$UID + 1000000

アカウンティングの使用不可

分類プロセスを使用して、選択したプロセスのアカウンティングを使用不可にすることができます。

アカウンティングを使用不可にするには、「NoAccounting」プロジェクト・リストを指定します。この属性は、親プロセスから子プロセスに継承されます。

以下の例は、これを行う方法を示します。

表 6. プロジェクト・リストでの NoAccounting 指定の例

ユーザー	グループ	アプリケーション	プロジェクト
Oracle	Oracle	/usr/*/oracle	NoAccounting
Root	Root	kbiod	NoAccounting

最初の規則では、Oracle がサブルーチン **exec()** によって分類されます。 2 番目の規則の NFS kproc は、サブルーチン **initp()** によって分類されます。

プロジェクトの作成

拡張アカウンティングにプロジェクトを作成することができます。

プロジェクトは、請求可能なエンティティーを表します。業務上の請求対象である各ユーザー、部門、部署、または会社は、拡張アカウンティングのプロジェクトによって表されなければなりません。プロジェクトを作成するには、**projctl add** コマンドを使用します。作成されたプロジェクトは、プロジェクト定義ファイルに保管されます (デフォルトの場所は **/etc/project** ディレクトリー内の **projdef** ファイル)。

注: プロジェクト定義を追加するとき、必ず集約を使用不可にしてください。プロジェクト定義の集約を使用可能にするには、まず集約をオフにした状態でプロジェクト定義を作成した後、そのプロジェクトの集約を手動で使用可能にします。データ集約についての詳細は、31 ページの『データ集約』を参照してください。

プロジェクト定義の作成

請求可能なエンティティーを含むプロジェクト定義を作成できます。

この手順では、プロジェクト定義を作成する方法を説明します。プロジェクトは、請求可能なエンティティーを表します。各プロジェクトは、プロジェクト番号、プロジェクト属性、およびプロジェクト名で構成され、これらをまとめてプロジェクト定義といいます。以下のシナリオでは、SMIT とコマンド・ラインの両方を使用してプロジェクト定義を作成する方法を説明します。ここでは、mktg というプロジェクト定義を作成する方法を説明します。

考慮事項

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。 したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. root ユーザーに切り替えます。
2. コマンド・プロンプトで `smit add_proj` と入力します。
3. 「プロジェクト名」フィールドに、`mktg` と入力します。
4. プロジェクトにプロジェクト番号を割り当てます。 例えば、「プロジェクト番号」フィールドに、`10` と入力します。
5. 「集約」フィールドが「**オフ**」に設定されていることを確認します。 SMIT では「集約」フィールドが「**オン**」に設定されているとプロジェクトを作成できません。
6. 「**プロジェクト・コメント**」フィールドに、適切なコメントを追加します。 このシナリオでは、「プロジェクト・コメント」フィールドに、「`This is the default project for the Marketing department (これはマーケティング部門のデフォルトのプロジェクトです)`」と入力します。
7. Enter キーを押して設定値を保存し、プロジェクト定義を作成します。

コマンド・ラインから `mktg` というプロジェクト定義を作成するには、`projctl add mktg 10 off "This is the default project for the Marketing department"` と入力します。

プロジェクト・コマンドと高速パス

プロジェクトはコマンド・ラインまたは SMIT 高速パスを使って管理することができます。

以下の表は、プロジェクトを管理するためのコマンドを示しています。

表 7. プロジェクトのコマンド

タスク	コマンド	SMIT 高速パス
プロジェクト定義の追加。	**projctl add** *projname projnumber* [*comment*] [{**-d** *projpath* | **-p** [*DN*]}]	**smit add_proj**
プロジェクト定義のロードまたは再ロード。	**projctl ldprojs** [{**-g** [*DN*] | **-p** [*DN*]}] [**-d** *projpath*] [**-r**] [**-a**]	**smit load_proj**
指定したプロジェクト定義の除去。	**projctl rm** *projname* [{**-d** *projpath* | **-p** [*DN*]}]	**smit rename_proj**
アクティブなプロジェクト定義の表示または変更。	**projctl chg** *projname* [**-p** *pid* [, *pid*]] [**-f**]	**smit show_chg_proj**
2 つのプロジェクト定義のマージ。	**projctl merge** *sourceprojpath* [**-d** *targetprojfile*]	なし
プロジェクト割り当てのあるプログラムの開始。	**projctl exec** *projname* [*cmd line*] [**-f**]	**smit start_proj_prg**
プロジェクトの集約を使用可能または使用不可にする。	**projctl chattr agg** *projname* {**-s**|**-u**} [{**-d** *projpath* | **-p** [*DN*]}]	なし
システムにロードされているポリシーの表示。	**ojctl qpolicy** [**-g** [*DN*]]	
アクティブなプロジェクト定義をすべてリストする。	**projctl qprojs** [**-n**]	なし
指定したプロジェクト定義をリストする。	**projctl qproj** *projectname*	**smit show_chg_proj**
プロセスのプロジェクト割り当ての変更。	**projctl chg** *projname* [**-p** *pid* [, *pid*]] [**-f**]	**smit chg_proj_proc**
プログラムのプロジェクト割り当ての表示。	**projctl qapp** *appname*	**smit show_proj_pgm**
プロジェクトの管理。	なし	**smit work_project**
プロジェクト定義の除去。	なし	**smit remove_admin_proj**

詳しくは、「*Commands Reference, Volume 4*」を参照してください。

ポリシー

ポリシーはプロジェクト割り当てを自動化します。ポリシーは分類基準と分類結果からなります。プロジェクト割り当てはサブルーチンおよびカーネル・サービス (**exec**、**initp**、**setuid**、**setgid** など) を使用して行われます。

ポリシーを使用すると、ユーザー、グループ、アプリケーションごと、またはこれらの属性の組み合わせに基づいてデータを分類できます。作成されるポリシーのタイプに応じて、管理者はユーザー名、グループ名、アプリケーション名、およびプロジェクト・リストをポリシー・ファイル内に指定できます。ただし、ポリシーが機能するには、この 4 つの構成要素がすべて揃っていなくても差し支えありません。

プロセスを割り当てるには、以下の 2 つの方法があります。

- プロセス分類属性が変更されたとき、割り当て規則を使用する。これは、プロセスを分類する最も一般的な方法です。
- 必要な権限を持つユーザーが手動でクラスに割り当てる。

管理ポリシー

管理ポリシーはプロセス属性としてユーザー名、グループ名、およびアプリケーション名を使ってプロセスを分類します。管理ポリシーはアプリケーションに基づき、アプリケーション・レベルでのアカウンティング統計の収集を可能にします。

管理ポリシーは、ほとんどのフィルター、および Korn シェル構文ワイルドカードの使用をサポートします。管理ポリシーを構成および保守するには、スタンドアロンの管理ツール (例えば SMIT やエディター) が必要です。

デフォルトでは、管理ポリシーは **/etc/project** ディレクトリーに入っています。代わりの管理ポリシーを独自にいくつか作成して、さまざまな状況で使用することができます。例えば、月曜日から金曜日の間に実行される管理ポリシーと、土曜日および日曜日に実行される別の管理ポリシーを作成することができます。代替的な管理ポリシーは **/etc/project/alter** ルート・ディレクトリーのサブディレクトリーに保管されます。新しい管理ポリシーを作成するとき、サブディレクトリーの名前を指定する必要があります。

管理ポリシーの割り当て規則

管理ポリシーは、管理ポリシー・ファイルに保管される 1 つ以上の割り当て規則からなります。

管理ポリシーの割り当て規則は、次の構文に従っている必要があります。

user name:*group name*:*application name*:*Project List*::コメント *(オプション)*

以下の表は、割り当て規則の各フィールドに使用できる値について詳しく説明しています。

表 8. ユーザー、グループ、およびアプリケーションの規則

規則のタイプ	説明
user (ユーザー)	ハイフン (-)、または **/etc/passwd** ファイルで定義された少なくとも 1 つの有効なユーザー名を含めることができます。特定のグループをクラスから除外するために、名前の前に感嘆符 (!) を使用することができます。ユーザー名のリストは、コンマ (,) で区切られた 1 つ以上の名前で構成されます。完全な Korn シェル・パターン・マッチング構文を使用して、複数のユーザー名からなるセットに一致するパターンを指定できます。無効または誤ったユーザー名を使用した場合、拡張アカウンティングはその規則の全体を無視します。ハイフン (-) を使用した場合、拡張アカウンティングはその規則の次のフィールドにスキップします。
group (グループ)	ハイフン (-)、または **/etc/groups** ファイルで定義された少なくとも 1 つの有効なグループ名を含めることができます。特定のユーザーをクラスから除外するために、名前の前に感嘆符 (!) を使用することができます。グループ名のリストは、コンマ (,) で区切られた 1 つ以上の名前で構成されます。完全な Korn シェル・パターン・マッチング構文を使用して、複数のグループ名からなるセットに一致するパターンを指定できます。誤ったグループ名を使用した場合、拡張アカウンティングはその規則の全体を無視します。ハイフン (-) を使用した場合、拡張アカウンティングはその規則の次のフィールドにスキップします。
application (アプリケーション)	ハイフン (-)、複数のアプリケーション・パス名からなるリスト、またはカーネル・プロセスのコマンド名を含めることができます。これは、クラスに含まれるプロセスによって実行されるアプリケーション (プログラム) のパス名です。アプリケーション名には、絶対パス名、またはパス名に一致する Korn シェル・パターンを指定します。アプリケーション名のリストは、コンマ (,) で区切られた 1 つ以上のパス名で構成されます。特定のアプリケーションを除外するために、名前の前に感嘆符 (!) を使用することができます。ロード時に、リスト内の少なくとも 1 つのアプリケーションが見つかる必要があります。そうでない場合、規則は無視されます。この理由で最初に無視された規則は、リストの内の 1 つ以上のアプリケーションを含むファイルシステムがマウントされれば、後で有効になる可能性があります。

プロセス割り当てポリシーの場合、分類基準は **/etc/passwd** ファイルにリストされたユーザー名、**/etc/groups** ファイルにリストされたグループ名、および完全修飾アプリケーション名です。分類結果はプロジェクト・リストです。ユーザーが現在のプロジェクト割り当てをリスト内の他のプロジェクトに変更しない限り、デフォルトでは、リスト内の最初のプロジェクトが使用されます。

属性値が変更されると、常に分類が行われます。それは、これらのプロセス属性の値と、クラス割り当て規則ファイルに含まれる可能な値のリスト (これを*規則* という) を比較することで行われます。比較によって、プロセス属性の現在の値にどの規則が一致するかが判別されます。

プロセスを分類するために、拡張アカウンティングはアクティブな構成に対する最上位の管理ポリシーを調べて、プロセスがどのクラスに属するかを判別します。拡張アカウンティングは、ファイル内のそれぞれの規則ごとに、プロセス属性の現在の値と、規則で指定された値 (および値のリスト) を調べます。拡張アカウンティングは管理ファイル内にリストされた順番ですべての規則を調べ、プロセスに最初に一致する規則に応じてプロジェクト内のプロセスを分類します。したがって、規則ファイル内の規則の順序は重要です。

以下のリストは、プロセス属性値が管理ポリシー・ファイルの同じ属性フィールド値に一致するかどうかを判別する基準です。

- 規則ファイル内のフィールドの値がハイフン (-) であれば、対応するプロセス属性の任意の値が一致します。
- いずれかのフィールド値の前に感嘆符 (!) が付いていれば、その値は常に除外されます。
- いずれかのフィールド値の後にアスタリスク (*) が付いている場合、その値に一致するすべてのケースが認識されます。

管理ポリシー規則の例

以下の例は、管理ポリシー規則の使用法を示します。

管理ポリシーでは、複数のユーザー、グループ、またはアプリケーションをそれぞれのフィールドで指定できます。例えば、Frank、Bob、および John に同じ属性を割り当てたい場合、以下の構文のように指定します。

```
User1,User2,User3:-:-:Project List::Comments
```

上記の構文では、User1、User2、および User3 がこの規則で同様に扱われるようになります。グループ名とアプリケーション・フィールドのダッシュ (-) は、ワイルドカードです。また、アスタリスク (*) を使用することもできます。既に述べたように、ワイルドカードを使用して、特定の属性のすべての値を含めることもできます。例えば、B で始まるすべてのユーザー名を含めたい場合には、ユーザー名フィールドに B* と入力します。規則のすべてのフィールドで、完全な Korn シェル・パターン・マッチング構文を使用できます。

また、以下のようにして、特定の何ユーザーかを含め、それ以外のユーザーを除外するように管理ポリシーを設定できます。

```
User1,!User2,User3:-:-:Project List::Comments
```

上記の構文では、User1 と User3 が同じ属性を持つようになりますが、User2 はポリシーから除外されます。

注:

1. カーネルは数値だけを読み取ります。上記の例では、ユーザー名 User1、User2、および User3 がカーネルにロードされた後、数値に変換されます。
2. ポリシーを何らかの形で変更した場合、**projctl** コマンドを使ってポリシーをカーネルに再ロードする必要があります。

管理ポリシーの別名

別名割り当てによってユーザーまたはグループのリストが簡単に参照されるようになり、管理ポリシーが簡略化されて、見やすく保守しやすいものになります。

ユーザーまたはグループのリストの別名を定義すれば、ユーザー、グループ、またはアプリケーション名の完全なリストを規則に入力する必要がなくなります。それぞれの管理ポリシーごとに別個の別名ファイルが使用され、それらは関連する管理ポリシーと同じディレクトリーに保管されます。

User1、User2、および User3 が 1 つの別名 dev1 にグループ化されるような別名を作成するには、以下のように定義します。

```
dev1:User1,User2,User3::Development team 1
```

別名を使用するには、管理ポリシーの中で別名の前にドル記号 ($) を付ける必要があります。例えば、管理ポリシーで User1、User2、および User3 の代わりに別名 dev1 を使用する場合、以下のようにします。

```
$dev1:-:-:Project List::This is a rule within a user alias
```

また、複数の別名をコンマで区切って使用することもできます。管理ポリシー規則内で別名の前に感嘆符 (!) を付けると、その別名は除外されます。

代替的な管理ポリシー

代わりの管理ポリシーを独自にいくつか作成して、ユーザーが定義するさまざまな状況に適用することができます。

一日の時間帯、または週の曜日ごとに異なる請求ストラテジーを実施するために、複数の管理ポリシーを作成することが可能です。代替的な管理ポリシーをロードするには、**projctl** コマンドを使用します。新しい

ポリシーをロードするために、それまでロードされていた管理ポリシーをアンロードする必要はありません。特定の時点でポリシーをロードするには、**cron** 機能を使用します。

代替的な管理ポリシーは **/etc/project/alter** ディレクトリーに格納されます。例えば、「weekend」という名前の代替的な管理ポリシーは、管理ファイル **/etc/project/alter/weekend/admin** に格納されます。このポリシーが別名を使用する場合、別名は **/etc/project/alter/weekend/alias** 別名ファイルに格納されます。管理ポリシーを作成、変更、表示、ロード、アンロード、および除去するには、SMIT を使用します。SMIT で代替ポリシーにアクセスするには、現行フォーカスをポリシー名に変更します。

管理ポリシーの作成

管理ポリシーは、プロジェクト割り当てを自動化するために使用されます。

このポリシーによって、アプリケーション・レベルでのアカウンティング統計の収集が可能になります。デフォルトでは、管理ポリシーの名前は **admin** で、**/etc/project** ディレクトリーに保管されます。これ以外の管理ポリシーを作成した場合、それらは **/etc/project/alter** ディレクトリーのサブディレクトリー内に保管されます。サブディレクトリーを作成するには、**mkdir** コマンドまたは SMIT を使用します。

ユーザーおよびグループのポリシー

ユーザー・ポリシーおよびグループ・ポリシーは、ユーザーおよびグループに関するプロセス属性を使用します。ユーザー・ポリシーまたはグループ・ポリシー内のプロジェクト割り当て規則は、ユーザー名またはグループ名とプロジェクト・リストで構成されます。ユーザー・ポリシー・データはセキュリティー・データベースに保管されるため、ユーザー・ポリシーまたはグループ・ポリシーに関連したファイルはありません。

ユーザー・ポリシーとグループ・ポリシーは、(ポリシーに応じて) ユーザー名またはグループ名だけに基づいてプロジェクトを割り当てます。これらはユーザー管理に統合されています。ユーザーやグループが作成されるとき、ユーザーおよびグループに対してプロジェクト・リストを指定することができます。

以下の表は、ユーザー・ポリシーの構造を示しています。

表 9. ユーザー・ポリシーの構造

ユーザー名	プロジェクト・リスト
User1	project1,project 2
User2	biology,chemistry

グループ・ポリシーを作成するには、次のステップを実行します。

1. 以下を入力して、グループを作成します。

   ```
   mkgroup staff
   ```

2. 以下を入力して、プロジェクトを作成します。

   ```
   projctl add biology_dept 1200 Project Comment
   ```

3. プロジェクト `biology_dept` を新しく作成したグループ `staff` と関連付けます。

   ```
   chgroup projects=biology_dept staff
   ```

注: ユーザー・ポリシーを作成するには、ステップ 1 からステップ 3 までを完了します。その際、**mkuser** と **chuser** をそれぞれ **mkgroup** と **chgroup** の代わりに指定します。

ユーザーまたはグループが作成されると、プロジェクトがそれに関連付けられます。 1 つのユーザーまたはグループに複数のプロジェクトを関連付けることが可能ですが、一度にアクティブにできるのは 1 つだけです。

ユーザー・ポリシーまたはグループ・ポリシーの作成

ユーザー・ポリシーとグループ・ポリシーを作成するには、プロジェクトをユーザーまたはグループに関連付けます。

ユーザー・ポリシーとグループ・ポリシーは、ユーザーおよびグループに対するプロジェクト割り当てを自動化するために使用されます。プロジェクトをユーザーやグループに関連付けるには、**chuser** コマンドおよび **chgroup** コマンドを使用します。

ポリシー・コマンドと高速パス

ポリシーはコマンド・ラインまたは SMIT 高速パスを使って管理することができます。

以下の表は、ポリシーを管理するためのコマンドを示しています。

表 10. ポリシーのコマンド

タスク	コマンド	SMIT 高速パス
システムにロードされているポリシーの表示。	**projctl qpolicy**	**smit query_policy**
ポリシーのロード。	**projctl ldadm** [{**-g** [{[*name:*]*DN* | *name*}] | **-p** [{[*name:*]*DN* | *name*}] }] [**-d** *admproj*] [**-r**] [**-a**] **projctl ldusr** [**-a**] [**-r**] **projctl ldgrp** [**-a**] [**-r**] **projctl ld** [**-a**] [**-r**] **projctl ldall** [**-d** *localadm*] [**-r**] [**-a**]	**smit load_admin** **smit load_users** **smit load_groups**
ポリシーのアンロード。	**projctl** {**unldusr**|**unldgrp**|**unldall** | {{**unldrojs** | **unldadm**} [{**-p** [*name*] | **-g**}] [**-f**]}} [**-a**]	**smit unload_admin** **smit unload_users** **smit unload_groups**
管理ポリシーの作成。	なし	**smit create_admin**
現在のフォーカス・ポリシーの表示または変更。	なし	**smit change_show_focus**
管理ポリシーの除去。	なし	**smit remove_admin**
規則の追加。	なし	**smit add_admin_rule**
規則の除去。	なし	**smit remove_admin_rule**
ユーザー別名の追加。	なし	**smit add_usr_alias**
グループ別名の追加。	なし	**smit add_grp_alias**
指定した別名の表示または変更。	なし	**smit chg_alias**
指定した別名の除去。	なし	**smit remove_alias**
ユーザー用のプロジェクト・リストの作成。	**chuser** *user*	**smit create_user**
グループ用のプロジェクト・リストの作成。	**chgroup** *group*	**smit create_group**

表 10. ポリシーのコマンド (続き)

タスク	コマンド	SMIT 高速パス
指定したユーザー用の指定したプロジェクト・リストの表示または変更。	**chuser projects=***projectlist user*	**smit change_show_user_list**
グループ用のプロジェクト・リストの表示または変更。	**chgroup** *group*	**smit change_show_group_list**
指定したユーザー用のプロジェクト・リストの除去。	**chuser projects=***user*	**smit remove_user**
指定したグループ用のプロジェクト・リストの除去。	**chgroup projects=***group*	**smit remove_group**
すべてのユーザーのプロジェクト・リストの表示。	**lsuser -a ALL**	なし
すべてのグループのプロジェクト・リストの表示。	**lsgroup -a projects ALL**	なし

詳しくは、「*Commands Reference, Volume 4*」を参照してください。

アプリケーション・リソース管理インターフェース

アプリケーションはワークロードのトランザクション機能を記述するために、アプリケーション・リソース管理 (ARM) インターフェースを使用します。

拡張アカウンティングは ARM インターフェースを介して提供される情報をアカウンティング・データ・ファイルに記録することにより、ARM インターフェースをサポートします。チャージバックのためにこの情報を利用するには、ARM インターフェースに関連したプログラミング・モデルを理解し、重要なビジネス情報を拡張アカウンティング・サブシステムに渡すメカニズムを理解して、請求アプリケーション用にこの情報が保持されるようにする必要があります。

ARM インターフェースの構造

ARM プログラミング・モデルの基本的な構造は、階層構造です。

トランザクション・インスタンスは、トランザクション登録時に定義されるトランザクション定義から派生します。アプリケーション・インスタンスは、アプリケーション登録時に定義されるアプリケーション定義から派生します。トランザクションの開始時に、アプリケーション・インスタンス内で使用すべきトランザクション定義が指定され、各トランザクション・インスタンスに関する完全な情報のセットを定義できるようになります。例えば、すべてのトランザクションにはアプリケーション名およびグループ名があります。

拡張アカウンティングは、以下のようなレコード階層を介して ARM インターフェースをサポートします。

- アプリケーション環境レコード
- トランザクション環境レコード
- トランザクション・インスタンス・レコード

アプリケーション環境レコードは、アプリケーション名、アプリケーション・グループ名、属性 (ID とコンテキストの両方) など、アプリケーション情報の固有な組み合わせを記述します。それぞれのアプリケーション環境レコードには固有 ID が割り当てられるため、それをシンボリックに参照できます。この ID をアプリケーション*環境 ID* といい、トランザクション・インスタンス・レコード内に含まれます。

トランザクション環境レコードは、トランザクション名、属性など、トランザクション情報の固有な組み合わせを記述します。それぞれのトランザクション環境レコードには固有 ID が割り当てられ、それをシンボリックに参照できます。この ID をトランザクション*環境 ID* といい、トランザクション・インスタンス・レコードに含まれます。

アプリケーション環境 ID とトランザクション環境 ID は長く持続しますが、永続的なものではありません。システムがブートするたびに、これらの ID が再生成されます。この問題を回避するために、拡張アカウンティングはアプリケーション環境とトランザクション環境を各ファイルに記録します。こうして、レポート・コマンドおよび分析コマンドは現在のファイルのエントリーを使用して、トランザクションに適用されるべき値の固有な組み合わせを判別します。

トランザクション・インスタンス・レコードは、それぞれのトランザクションを記述するために使用されます。上記で定義されたアプリケーション環境 ID とトランザクション環境 ID がこれに含まれます。これらの ID は対応するアプリケーション環境レコードとトランザクション環境レコードを検索するために使用され、アプリケーション名、アプリケーション・グループ名、およびトランザクション名をトランザクション・インスタンスに関連付けることが可能になります。この目的は各トランザクション・インスタンスに関して記録する必要があるデータの量を最小限に抑えることです (ほとんどのデータの性質は静的であるため)。

また、拡張アカウンティングは ARM アカウンティング・データの集約もサポートします。データ集約についての詳細は、31 ページの『データ集約』を参照してください。

ARM インターフェースのプログラミング・モデル

ARM インターフェースのプログラミング・モデルには、基本機能を実行するルーチンが含まれます。

以下の表は、基本的なプログラミング・モデルについて説明しています。

表 11. ARM インターフェースのプログラミング・モデル

インターフェース	説明
arm_register_application	このルーチンはアプリケーション ID を ARM インプリメンテーションに登録します。アプリケーション ID は、基本的な計測対象アプリケーション・スコープ、および後続の ARM 4.0 呼び出しの基盤を提供します。このルーチンはアプリケーションの初期設定時に呼び出されます。
arm_start_application	このルーチンは ARM 4.0 トランザクション呼び出しの準備段階として、始動済みアプリケーション・インスタンスを ARM インプリメンテーションに確立します。このインターフェースは、ARM へのアプリケーション登録後、アプリケーションの初期設定時に呼び出されます。
arm_register_transaction	このルーチンはトランザクション ID を ARM インプリメンテーションに登録します。トランザクション ID は、トランザクションの監視および計測用の後続の ARM 4.0 呼び出しにおいて登録済みアプリケーションの下で実行されるトランザクションのカテゴリー (通常、トランザクション・タイプと呼ばれる) を提供します。このインターフェースは、ARM アプリケーション・インスタンスの開始後、アプリケーションの初期設定時に呼び出されます。
arm_start_transaction	このルーチンは始動済みトランザクション・インスタンスを ARM インプリメンテーションに確立します。始動済みトランザクション・インスタンスは、トランザクションの監視および計測の基盤を提供します。このルーチンは、トランザクション処理中に呼び出されます。

表 11. ARM インターフェースのプログラミング・モデル (続き)

インターフェース	説明
arm_block_transaction	このルーチンは、始動済みトランザクション・インスタンスが特定のイベントによってブロックされた状態であることを示します。このルーチンは、トランザクション処理中に呼び出されます。
arm_unblock_transaction	このルーチンは、始動済みトランザクション・インスタンスをブロックしていたイベントが解放されたことを示します。このルーチンは、トランザクション処理中に呼び出されます。
arm_bind_transaction	このルーチンは、現在のスレッドが始動済みトランザクション・インスタンスのために実行されていることを示します。バインディングによって、スレッドとトランザクションとの間の排他的処理関係が確定されるため、システムはトランザクションによるプロセッサー使用状況を測定できるようになります。このルーチンは、トランザクション処理中に呼び出されます。
arm_unbind_transaction	このルーチンは、現在のスレッドがもはや始動済みトランザクション・インスタンスのために実行されていないことを示します。このルーチンは、トランザクション処理中に呼び出されます。
arm_stop_transaction	このルーチンは、ARM インプリメンテーションによって認識されている始動済みトランザクション・インスタンスを終了します。計測対象アプリケーションが開始済みトランザクション・インスタンスを終了する際には、通常、このインターフェースを呼び出します。
arm_stop_application	このルーチンは、ARM インプリメンテーションによって認識されている始動済みアプリケーション・インスタンスを終了します。計測対象アプリケーションが開始済みアプリケーション・インスタンスを終了する際には、通常、このインターフェースを呼び出します。
arm_destroy_application	このルーチンは、登録済みアプリケーションを登録解除します。計測対象アプリケーションがアプリケーションを登録解除する際には、通常、このインターフェースを呼び出します。

ARM インプリメンテーションによって認識されるパラメーター

ARM API を利用すると、ARM インプリメンテーションおよび拡張アカウンティングによって認識される名前付きパラメーターのセットを使用してアプリケーション・トランザクションを記述できます。

ARM 用に設定されたパラメーターによってオペレーティング・システムはアプリケーション・トランザクションを識別でき、拡張アカウンティングはアプリケーション・トランザクションを計測してアカウンティング・データ・ファイルに記録できます。また、アプリケーションは ARM API を使ってトランザクションを記述することにより、サイト固有の情報を提供できます。これらは、一般的に行われる操作です。

拡張アカウンティングは、以下の表に示されるパラメーターを認識します。

表 12. ARM インプリメンテーションおよび拡張アカウンティングによって認識されるパラメーター

パラメーター	説明
アプリケーション名	アプリケーション名は、*app_name* パラメーターによって arm_register_application() インターフェースに対して指定されます。このパラメーターはアプリケーションによって設定され、ユーザーはこれをオーバーライドできません。これはアプリケーション名を指定し、「IBM® DB2 Universal Database™」のような名前が使用されるようになります。

表 12. ARM インプリメンテーションおよび拡張アカウンティングによって認識されるパラメーター (続き)

パラメーター	説明
アプリケーション・グループ名	アプリケーション・グループ名は、*app_group* パラメーターによって arm_start_application() インターフェースに対して指定されます。このパラメーターを使用することで、個々のアプリケーション・インスタンスを集合的に構成してグループ化し、1 つの統合サービスを提供することができます。したがって、「サンプル・サプライ・チェーン・マネージメント」のような識別可能な名前を使用する必要があります。
トランザクション名	トランザクション名は arm_register_transaction() インターフェースを介して指定されます。データ・ファイルを分析して、発生した操作のタイプを判別できるようにするために、記述的な名前を提供する必要があります。
ID 属性	ID 属性は、値が決して変更されない属性を指定するために使用されます。登録済みアプリケーションおよびトランザクションに関して ID 属性を指定して、アプリケーションやトランザクションの不変の性質を記述するためにさまざまなレベルでそれを使用できます。 ID 属性を使って、拡張アカウンティング・サブシステムによって認識される割引率やアカウント・コードを識別できます。プロジェクト・コードと同様に、アカウント・コードは定義済み属性を使って指定されなければなりません。 `EWLM: AIX: Account Class`
コンテキスト属性	コンテキスト属性は、変更される情報を表すために使用されます。コンテキスト属性はアプリケーションおよびトランザクションのインスタンスに関連付けられますが、拡張アカウンティングは後者のアカウンティング・データをキャプチャーしません。トランザクション・インスタンスに関しては、コンテキスト属性の名前だけがキャプチャーされます。また、プロジェクト・コードと同様に、コンテキスト属性を使ってアカウンティング・コードを指定することができます。この場合、定義済み属性を使用します。 `EWLM: AIX: Account Class`

内部的に生成されるトランザクション・アカウンティング・データ

また、拡張アカウンティングは、内部的に生成されるデータもキャプチャーします。ユーザーはこの情報を変更できませんが、アカウンティングではこの情報が必要とされます。

以下の表は、内部的に生成されるデータについて説明しています。

表 13. 内部的に生成されるトランザクション・アカウンティング・データ

パラメーター	説明
ユーザー名と ID	ユーザー名と ID は、トランザクション開始時に指定されるユーザー・サブバッファーから派生します。これは、トランザクションを開始したエージェントを識別します。 IP アドレスまたはマシン名になる場合もあります。必ずしも UNIX のユーザー名やユーザー ID とは限りません。
応答時間とキュー時間	応答時間とキュー時間は、トランザクションに対して提供されたサービス品質を記述します。応答時間とキュー時間は、トランザクションの経過時間、および何らかの実質的アクションが開始されるまでに費やされたアプリケーション時間を識別します。

表 13. 内部的に生成されるトランザクション・アカウンティング・データ (続き)

パラメーター	説明
リソース使用状況	トランザクションによるリソース使用状況は、トランザクションに費やされたプロセッサー時間合計など、リソースの物理的使用状況を記述します。アプリケーションが同時に複数のトランザクションを処理する場合があるため、この統計は常に算出可能であるとは限りません。実際、スレッドを単一のトランザクション用に専用化する arm_bind_thread() インターフェースをアプリケーションが使用しない限り、この統計は生成されません。

間隔アカウンティング

間隔アカウンティングは、指定された間隔ごとにアカウンティング・データを収集します。アクティブ・プロセスに関する中間的なプロセス・レコードを生成するよう、これを構成できます。

間隔アカウンティングのレコードを完成したプロセス・レコードに追加すれば、一定期間内でのユーザーのシステム使用量の合計を反映する、より正確な請求を生成できます。 また、プロセッサー、メモリー、ディスク、ネットワーク・インターフェース、ファイルシステムなどのシステム・リソースに関するアカウンティング・データを定期的にキャプチャーするよう、間隔アカウンティングを構成することもできます。このような情報を使って、パーティションの総使用量を反映した請求を生成できます。

システム・リソースに関する使用情報は、チャージバック以外の用途にも使用できます。プロセッサー、メモリー、ディスク、ネットワーク・インターフェースなどの物理リソースの利用状況を示すため、キャパシティー・プランニングに活用できます。間隔アカウンティングはこれらのリソースの状況を一定時間ごとに表示するため、負荷を判別し、キャパシティー上の決定をデータに基づいて行うことができます。例えば、特定の期間にわたるアイドル状態のプロセッサー・キャパシティーを判別することにより、プロセッサーを追加する必要があるかどうかを判別できます。

間隔アカウンティングが提供するリソース使用状況の履歴ビューを使って、パフォーマンスを分析することができます。例えば、ファイルシステム・レコードを調べることによって、システムへのアクセスの遅延が発生したときにどのファイルシステムがビジー状態であったかを判別できます。このデータは、同じディスク・アダプターによって処理される複数のビジー状態のファイルシステムを識別します。この情報を使って、複数のディスク・アダプター間でファイルシステムのバランスを取ることができます。任意の時点でどのファイルセットがアクセスされているか必ずしも把握できないため、このような情報を含むログの再生機能があると役立ちます。

システム間隔アカウンティング

システム間隔アカウンティングは、システム関連の情報を収集します。

システム間隔アカウンティングは、プロセッサー、ディスク、ネットワーク・アダプター、ファイルシステム、メモリー使用量などのシステム・リソースの使用状況情報を収集します。この情報を使って、システム使用状況の傾向を把握できます。ほとんどの場合、間隔は 1 時間に一度で十分です。

システム間隔は、プロセスのアカウンティング・データを収集しません。

プロセス間隔

プロセス間隔アカウンティングは、プロセス関連の情報を収集します。

プロセス間隔を使用すれば、長期間実行されるジョブ (例えば、何カ月も実行されるデータ・モデリング・アプリケーション) に関する情報をキャプチャーできます。ほとんどのプロセスの場合、標準的なプロセス完了レコードで十分です。このため、プロセス間隔はある程度大きくなり、不必要なデータが生成されません。プロセス間隔は、新しく開始されたジョブを含め、アクティブな各プロセスごとのレコードを書き込みます。大量のレコードが生成される可能性もあるため、1 日 (1,440 分) に一度だけ情報をキャプチャーするようプロセス間隔を設定するのが適切です。ただし、プロセス・レコードがシステムによって自動的に集約される場合には、1 時間に一度のプロセス間隔を設定してください。

間隔アカウンティングのコマンドと高速パス

間隔アカウンティングは、コマンド・ラインまたは SMIT 高速パスを使って管理することができます。

デフォルトでは、間隔アカウンティングは使用不可です。以下の表には、システム・レベルおよびプロセス・レベルの間隔アカウンティングを使用可能/使用不可にするためのコマンドが示されています。

表 14. 間隔アカウンティングのコマンド

タスク	コマンド	SMIT 高速パス
間隔を *time* パラメーター (分単位) で指定してシステム間隔アカウンティングを使用可能にする。または、システム間隔アカウンティングを使用不可にする。	**acctctl isystem** {*time*|off }	**smit system_interval**
time パラメーター (分単位) で指定して、プロセス間隔アカウンティングを使用可能にする。または、プロセス間隔アカウンティングをオフにする。	**acctctl iprocess** {*time*|off}	**smit process_interval**
拡張アカウンティングの状態の照会。	**acctctl**	なし

詳しくは、「*Commands Reference, Volume 1*」を参照してください。

ホストされるアカウンティング・ポリシー

アカウンティング・ポリシーを別のシステムがホストすることによって、アカウンティング・ポリシーを管理するための単一の制御拠点を確立することができます。

制御拠点を 1 つに統合すると、アカウンティング・ポリシーの管理と保守が簡略化されます。この機能を使用すると、エンタープライズ内の複数のマシンにログインしているユーザーを把握することができます。場合によっては、この機能と併せてサーバー固有の請求ストラテジーを定義することもできます。拡張アカウンティング・サブシステムを使用すると、1 つのサーバーに対して複数のポリシーを並行して定義できます。この柔軟性を利用して、業務に最適なソリューションをインプリメントすることができます。

ホストされるアカウンティング・ポリシーをインプリメントするには、Lightweight Directory Access Protocol (LDAP) を使用します。LDAP は、クライアント/サーバー・モデルを使って、ローカルまたはリモート側からディレクトリー (データベース) 内の情報にアクセスし、その情報を更新するための標準メカニズムを定義します。拡張アカウンティング・サブシステムはこのテクノロジーを利用して、アカウンティング・ポリシーおよびプロジェクトを管理し、LDAP クライアントに配布するため単一の制御拠点を提供します。 LDAP サーバー上のアカウンティング・データにアクセスするには、各 LDAP クライアント・システムを構成する必要がありますが、いったん構成をすると、クライアントはポリシーがそのシステムでローカルに定義されたときと同様に動作します。

拡張アカウンティング・サブシステムはユーザー認証と完全に統合されています。新規ユーザーまたはグループを作成するときに、オプションでそのユーザーまたはグループ用のプロジェクト・リストを指定するこ

とができます。このプロジェクト・リストはそのユーザーまたはグループの定義に含まれるようになり、ユーザーがログインするとクライアント・システムに自動的に配信されます。プロジェクト・リストは、もう 1 つのユーザー属性ということです。

通常、プロジェクト定義はユーザーと同じスコープを指定して設定する必要があります。ユーザーが LDAP を介してグローバルに定義されている場合、そのユーザーのプロジェクト定義も LDAP を介してグローバルに定義します。同様に、ユーザーが特定のシステム上でローカルに定義されていれば、そのユーザーのプロジェクト定義も同じシステムでローカルに定義します。こうすると請求ストラテジーは簡略化されますが、拡張アカウンティング・サブシステムでこれが必須というわけではありません。ローカル・ユーザーと LDAP ユーザーのどちらにも対応するように、この両方のプロジェクト・リポジトリーは同時にロードすることができます。ローカル・プロジェクト定義と LDAP プロジェクト定義を設定する方法は重複しない方が望ましいのですが、この推奨事項は拡張アカウンティング・サブシステムで強制されません。プロジェクトはプロジェクト・リポジトリーがロードされる順序に基づいて解決されます。ローカル・プロジェクトを LDAP プロジェクトより優先するには、ローカル・プロジェクト・リポジトリーを LDAP プロジェクト・リポジトリーより先にロードします。また、逆の場合は、逆の順序でロードします。ローカル・プロジェクトか LDAP プロジェクトのどちらか一方のみを使用することも可能です。

LDAP では以下の拡張アカウンティング・データ・セットを使用できます。

- プロジェクト定義
- LDAP ユーザーおよびグループのプロジェクト・リスト (例えば、LDAP ユーザーおよびグループ・ポリシー)
- 管理ポリシー

管理ポリシーも LDAP サーバーに格納できます。管理ポリシーは分類を行うための代替的なメカニズムを提供し、ある意味ではユーザーごとにプロジェクト・リストを指定するより管理が容易です。管理ポリシーは割り当て規則の集合であり、多数の個別のユーザー定義に対して配布されるものではないため、簡単に更新できます。管理ポリシーは 1 つのエンティティーとして管理されます。ローカル・ベース、LDAP ベースの両方の管理ポリシーが定義できます。

以下のポリシーは同時に使用可能にできます。評価の順序は以下のとおりです。

1. ローカル管理ポリシー
2. LDAP 管理ポリシー
3. ユーザー・アカウンティング・ポリシー
4. グループ・アカウンティング・ポリシー

LDAP サーバーは拡張アカウンティングによって認識せず、異なるソフトウェア・レベルで保守可能です。拡張アカウンティング・データを記述するスキーマを LDAP サーバーにアップロードすることによって、プロジェクトとポリシーについての特別の知識がなくても、そのスキーマを使用して、プロジェクトとポリシーについての特別の知識がなくても、アカウンティング・ポリシーおよびプロジェクト定義を配布することができます。

LDAP サーバーおよびクライアント間の接続を構成するには、**mkprojldap** コマンドを使用します。具体的には、このコマンドを使用すると、拡張アカウンティング・データに関連した LDAP スキーマをサーバーにアップロードすることができます。このコマンドは、特定のクライアントに関するデータを格納するサーバー上の場所を定義するときにも使用します。これによって、必要であれば、クライアント・システムごとに別個の管理ポリシーとプロジェクト・リポジトリーをインプリメントすることが可能です。これらの項目は個々に構成可能であるため、最大限の柔軟性を発揮します。異なる管理ポリシーをインプリメントする 1 つの理由は、管理上の必要性です。例えば、ポリシーに特定の許可ユーザーの集合を反映させることなどで

す。または、特定のサーバーで異なる請求ストラテジーを実施することが必要な場合もあります。例えば、サーバー X の使用については常にアカウント Y に課金される、あるいは、サーバー X はプロジェクト W と Z のみに使用される、などの場合があります。

アカウンティングの目的で各クライアント・システムを個別に構成する必要があります。各システムで活動化するポリシー・セットは、**projctl** コマンドを使って指定する必要があります。このコマンドは拡張されており、LDAP ベースのプロジェクトおよび管理ポリシーのアップロードやダウンロードなどの新機能を提供します。一般に、クライアント・システムがセットアップされると、ポリシーとプロジェクト・リポジトリーの場所はエンド・ユーザーから見て透過的になります。

拡張アカウンティング・サブシステムはアカウンティング・データをローカルで生成します。この場合でも、データ・ファイルを定義し、それらのファイルを継続的に管理するには **acctctl** コマンドを使用する必要があります。ただし、これらのファイルをストレージ・エリア・ネットワーク (SAN) などの共用記憶装置サブシステム、または NFS や General Parallel File System (GPFS™) などの分散ファイルシステムに格納して、請求アプリケーションがすべてのデータにアクセスできるようにすることも可能です。

ps コマンドを使用すると、プロセスに割り当てられたプロジェクトの起点を表示することができます。この情報はプロセスのアカウンティング・レコードにも記録されるため、レポートおよび分析ツールは割り当てられたプロジェクトを適切なプロジェクト・リポジトリーに正しくマッチングできます。ただし、これはそのツールが複数のプロジェクト・リポジトリーを認識できることを前提とします。この状態が発生しないようにするには、ローカル・ベースのプロジェクトと LDAP ベースのプロジェクトのプロジェクト範囲がオーバーラップしないように定義します。

マルチシステム・アカウンティング・ポリシー・ファイル

プロジェクト定義ファイルと管理ポリシー・ファイルは LDAP サーバーにアップロード、または LDAP サーバーからダウンロードすることができます。

マルチシステム・アカウンティング・ポリシーには、以下のデータ・セットが含まれます。

- デフォルト LDAP プロジェクト
- デフォルト LDAP 管理ポリシー
- 代替 LDAP 管理ポリシー

これらのデータ・セットは、ローカル・システムの以下のディレクトリーに格納されています。

- `/etc/project/ldap/projdef`
- `/etc/project/ldap/admin`
- `/etc/project/ldap/alter/policy name/admin`

LDAP サーバーに複数の管理ポリシーをロードすることができます。この機能を使用すると、時間ベースのポリシー、例えば、ピーク時およびオフピーク時のポリシーなどをインプリメントできます。

ホスト・アカウンティング・ポリシーに合わせた LDAP サーバーの構成

アカウンティング・ポリシーをクライアント・システムに渡す前に、そのポリシーをホストするよう LDAP サーバーを構成する必要があります。

このセットアップ手順は、LDAP サーバーにアクセスできるように一般的な方法で構成された、任意の LDAP クライアントから実行できます。 LDAP サーバーでこのセットアップ手順を実行する必要はありません。アカウンティング・ポリシーをホストするよう LDAP サーバーを構成するには、最初に拡張アカウンティング・サブシステム・スキーマをアップロードする必要があります。このスキーマは AIX に付属し

て提供されます。このスキーマはアカウンティング・データのレイアウトを記述し、LDAP サーバーが拡張アカウンティングを認識しなくても済むようにします。

LDAP サーバーがクライアントと同じソフトウェア・レベルでなければならないという要件はありません。

LDAP サーバーをセットアップするには、LDAP サーバーごとに次のコマンドを実行します。

```
mkprojldap -u -h hostname -D bindDN -w BindPassword
```

この後に、LDAP サーバー上のどこにアカウンティング・データを格納するかを決める必要があります。各クライアント・システムは特定の場所にあるアカウンティング・データを要求するため、サーバー上のアカウンティング・データのレイアウトを理解することが重要です。サーバーで正しいレイアウトを定義するには、請求ストラテジーを理解しておく必要があります。サーバー固有の請求ポリシーを配置する場合は、管理ポリシーを使用してください。管理ポリシーが特定のマシンをターゲットとして設定できるためです。この場合、管理ポリシーとプロジェクト定義を、そのシステム用に予約済みの LDAP サーバー上の場所に格納してください。

常に同じ方法でユーザーを分類するエンタープライズ・レベルのポリシーを使用する場合は、ユーザーまたはグループ・ポリシーを使用してください。この場合、各クライアントがアクセスできるように、プロジェクト・リポジトリーを LDAP サーバー上のグローバルな場所に定義する必要があります。これ以外のストラテジーも可能です。

アカウンティング・データを格納できるサーバー上の基本の場所を定義するには、次のコマンドを使用します。

```
mkprojldap -s -h hostname -D bindDN -w BindPassword -i InstallPoint
```

例えば、次のように指定します。

```
mkprojlap -s -h ldap.svr.com -D cn=root -w passwd -i -p cn=aixdata,o=ibm -a cn=aixdata,o=ibm
```

このコマンドによって、管理ポリシーとプロジェクト定義を、サーバー上のインストール・ポイントの下に格納できます。このコマンドは、基本の場所ごとに 1 回ずつ root ユーザーとして実行する必要があります。

ホスト・アカウンティング・ポリシーに合わせた LDAP サーバーの構成 (SMIT を使用する場合)

SMIT を使用して、ホスト・マルチシステム・アカウンティング・ポリシーに合わせて LDAP サーバーを構成することができます。

SMIT を使用してホスト・マルチシステム・アカウンティング・ポリシーに合わせて LDAP サーバーを構成するには、次の手順を実行します。

1. root ユーザーとしてログインします。
2. LDAP サーバーのセットアップ用の SMIT メニューにアクセスします。その際の SMIT パスは、**「smitty aacct」 > 「Manage Advanced Accounting Subsystem (拡張アカウンティング・サブシステムの管理)」 > 「Manage LDAP configuration (LDAP 構成の管理)」 > 「LDAP server setup (LDAP サーバーのセットアップ)」**です。
3. 関連のフィールドに、サーバー名、バインド DN、BIND パスワード、およびプロジェクトとポリシーのインストール・ポイントを入力します。
4. Enter を押すと、LDAP サーバーが拡張アカウンティング・サブシステム用に構成されます。

アカウンティング用の LDAP クライアントの構成 (コマンド・ラインを使用する場合)

コマンド・ラインを使用して、アカウンティング用に LDAP クライアントをセットアップすることができます。

これらの指示は、クライアント・システムが LDAP クライアントとして構成されていることを前提としています。

mksecldap コマンドを使用して、LDAP クライアントおよびサーバー間の基本接続を確立します。

mkprojldap コマンドを使用して、LDAP クライアントがアカウンティングで認識されるように設定するための、アカウンティング固有のパラメーターを指定します。

projctl コマンドを使用して、請求ストラテジーに必要なプロジェクトとポリシーを構成します。最後のステップでは、LDAP サーバーが提供するデータにリフレッシュ・ポリシーを指定します。

LDAP クライアントを構成するには、次のステップを実行します。

1. root ユーザーとしてログインします。
2. **mkprojldap –c –D** *bindDN* -w *bindPWD* -a *default-adminDN* –p *default-projectDN* コマンドを実行します。この *default-adminDN* および *default-projectDN* は、クライアントがアカウンティング・データを検索する LDAP サーバー上の基本の場所です。 このコマンドによって、アカウンティング固有の情報が LDAP 構成ファイル (`ldap.cfg`) に追加され、LDAP クライアント・デーモンが再始動されます。 このコマンドは、例えば **`mkprojldap –c -D cn=testroot -w testpwd –a ou=adminpolicy,ou=aacct,cn=aixdata –p ou=projects,ou=aacct,cn=aixdata`** のように入力します。
3. オプション: LDAP サーバーに LDAP プロジェクトまたは管理ポリシーをアップロードする場合は、この時点で実行できます。
4. オプション: ポリシーをロードするときに自動的に LDAP プロジェクトが使用されるように現行システムを構成する場合は、**projctl ldprojs -g -a** コマンドを実行します。 プロジェクトはロードされた順序で解決されます。そのため、ローカル・プロジェクトを優先させるには **projctl ldprojs –a** コマンドを最初に実行します。**–g** フラグは、データが LDAP サーバーから取得されることを示します。ソースを両方とも使用しようとしている場合は、両方のソースを構成する必要があります。
5. オプション: アカウンティングの開始時に自動的に LDAP 管理ポリシーがロードされるように現行システムを構成する場合は、**projctl ldadm –g -a** コマンドを実行します。ローカル管理ポリシーを構成することもできます。ローカル管理ポリシーを構成するには、**projctl ldadm –a** を実行します。 プロジェクトと異なり、管理ポリシー間ではどちらを優先するかを選ぶことはできません。ローカル管理ポリシーが常に LDAP 管理ポリシーより優先されます。
6. **cron** 機能を使用して、LDAP サーバーからロードされるプロジェクトと管理ポリシーを定期的にリフレッシュしてください。 リフレッシュの間隔は、新規ユーザーを受け入れるためのサイト固有のポリシーに応じて、1 時間に 1 回または 1 日 1 回とします。

上記のステップは SMIT を使用して実行することもできます。

LDAP サーバーが提供するアカウンティング機能を使用できるようにクライアントを構成した後は、拡張アカウンティング・サブシステムを管理するために LDAP 固有の知識は必要ありません。ただし、LDAP プロジェクト・リポジトリーに新規プロジェクト定義を追加する場合、または LDAP ベースのアカウンティング・ポリシーまたはプロジェクト定義を変更する場合は例外です。

LDAP サーバー上でのプロジェクト定義

LDAP サーバーへプロジェクト定義をアップロードする場合は、コマンド・ラインまたは SMIT を使用します。

デフォルトでは、プロジェクト定義はデフォルトの LDAP プロジェクト定義パス /etc/project/ldap/projdef からアップロードされます。

プロジェクト定義の LDAP サーバーへのアップロード (コマンド・ラインを使用する場合)

LDAP サーバーをホスト・アカウンティング・データに合わせて構成したら、コマンド・ラインを使用して LDAP サーバーの構成済み DN にプロジェクト定義をアップロードすることができます。

デフォルトでは、プロジェクト定義はデフォルトの LDAP プロジェクト定義パス /etc/project/ldap/projdef からアップロードされます。 **projctl** コマンドに **–d** フラグを使用すると、代替パス名を指定できます。

コマンド・ラインを使用して LDAP サーバーにプロジェクト定義ファイルをアップロードするには、次のステップを実行します。

1. プロジェクト定義を含むファイルに projdef という名前を付けてディレクトリー /etc/project/ldap/ に格納します。ディレクトリー /etc/project/ldap/ は、LDAP プロジェクトおよびポリシーのローカル・リポジトリーとしての役目を果たします。
2. **projctl ldprojs –p –d /etc/project/ldap/** コマンドを実行します。このコマンドによって、projdef ファイルが LDAP サーバー上のデフォルトのプロジェクト DN にアップロードされます。 projdef をデフォルト以外の DN にアップロードするには、**-p** を使用して DN パラメーターを指定します。

プロジェクト定義の LDAP サーバーへのアップロード (SMIT を使用する場合)

LDAP サーバーをホスト・アカウンティング・データに合わせて構成したら、SMIT を使用して LDAP サーバーの構成済み DN にプロジェクト定義をアップロードすることができます。

SMIT を使用して LDAP サーバーにプロジェクト定義ファイルをアップロードするには、次の手順を実行します。

1. root ユーザーとしてログインします。
2. SMIT メニューの「Upload Project Definitions to LDAP Server (プロジェクト定義の LDAP サーバーへのアップロード)」にアクセスします。その際の SMIT パスは、「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Project Definitions (プロジェクト定義)**」 > 「**Upload Project Definitions to LDAP Server (プロジェクト定義の LDAP サーバーへのアップロード)**」です。
3. LDAP サーバー上のローカル・プロジェクト定義ファイル・パスと宛先場所を入力します。
4. Enter を押すと、プロジェクト定義ファイルが LDAP サーバーにアップロードされます。

LDAP サーバー上の管理ポリシー

管理ポリシーを LDAP サーバーにアップロードすることができます。

管理ポリシーを LDAP サーバーにアップロードするには、**projctl ldadm –p** コマンドを使用できます。デフォルトでは、ソースとなる管理ポリシーのファイル場所は /etc/project/ldap/admin です。

別のポリシーをアップロードするように指定するには、**–d** フラグを使用します。 LDAP サーバー上のポリシー用のターゲット場所は、デフォルトで、クライアントのセットアップ時に設定された管理ポリシー

DN によって定義されることになっています。 **-p** フラグを使用して DN を指定すると、LDAP サーバー上の別のターゲット場所にポリシーをアップロードすることができます。 DN はオプショナル・パラメーターです。

管理ポリシーに新規のプロジェクト定義が含まれる場合、または初めてサーバーにポリシーをアップロードするときに、LDAP プロジェクト定義がまだアップロードされていない場合は、プロジェクト定義をアップロードして、ポリシーが別のクライアントにダウンロードされるときにそのポリシーが正しく解決されるようにする必要があります。拡張アカウンティング・サブシステムは、ポリシーのロード時に自動的に LDAP サーバーからプロジェクト定義をダウンロードしますが、ポリシーのアップロード時には自動的にプロジェクトをアップロードしません。このステップはユーザー自身が行う必要があります。サーバー上のプロジェクト・リポジトリーはクリティカル・リソースであり、アップロード操作は置換操作であるためです。

管理ポリシーの LDAP サーバーへのアップロード (コマンド・ラインを使用する場合)

コマンド・ラインを使用して管理ポリシーを LDAP サーバーにアップロードすることができます。

管理ポリシー・ファイルを LDAP サーバーにアップロードするには、次の手順を実行します。

1. 管理ポリシーを含むファイルに `admin` という名前を付けてディレクトリー `/etc/project/ldap/` に格納します。ディレクトリー `/etc/project/ldap/` は、LDAP プロジェクトおよびポリシーのローカル・リポジトリーとしての役目を果たします。
2. root ユーザーとして **projctl ldadm –p –d /etc/project/ldap/** コマンドを実行します。このコマンドによって管理ポリシー・ファイルが LDAP サーバー上のデフォルトのプロジェクト DN にアップロードされます。このファイルをデフォルト以外の DN にアップロードするには、**-p** を使用して DN パラメーターを指定します。

管理ポリシーの LDAP サーバーへのアップロード (SMIT を使用する場合)

SMIT を使用して管理ポリシーを LDAP サーバーにアップロードすることができます。

SMIT を使用して LDAP サーバーに管理ポリシー・ファイルをアップロードするには、次の手順を実行します。

1. root ユーザーとしてログインします。
2. SMIT メニューの「Upload Admin Policy to LDAP Server (管理ポリシーの LDAP サーバーへのアップロード)」にアクセスします。その際の SMIT パスは、「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Automatic Project Assignment (プロジェクトの自動割り当て)**」 > 「**Work with Admin Policies (管理ポリシーの操作)**」 > 「**Upload Admin Policy to LDAP Server (管理ポリシーの LDAP サーバーへのアップロード)**」です。
3. LDAP サーバー上のローカル管理ポリシー・ファイル・パスと宛先場所を入力します。デフォルトでは、ポリシー・ファイルは `default` という名前でアップロードされます。「**Admin Policy Name (管理ポリシー名)**」オプションを使用すると別の名前を指定できます。
4. Enter を押して、管理ポリシーを LDAP サーバーにアップロードします。

カーネルのプロジェクトとポリシー

LDAP プロジェクトとポリシーを使用する場合は、最初にその LDAP プロジェクトとポリシーをカーネルにロードする必要があります。プロジェクトとポリシーをカーネルにロードする場合は、コマンド・ラインまたは SMIT を使用します。

LDAP プロジェクトのカーネルへのロード (コマンド・ラインを使用する場合)

LDAP プロジェクトとポリシーを使用する場合は、最初にその LDAP プロジェクトとポリシーをカーネルにロードする必要があります。コマンド・ラインを使用すると、LDAP プロジェクトをカーネルにロードすることができます。

カーネルにロードする LDAP プロジェクトは、LDAP サーバー上のデフォルトのプロジェクト DN 上に存在する必要があります。

LDAP プロジェクトをカーネルにロードするには、次の手順を実行します。

1. root ユーザーとしてログインします。
2. **projctl ldprojs –g** コマンドを実行します。このコマンドによって、`projdef` ファイルが LDAP サーバーから `/etc/project/ldap/` ディレクトリーにダウンロードされた後、そのファイル内のプロジェクト定義がカーネルにロードされます。

LDAP プロジェクトのカーネルへのロード (SMIT を使用する場合)

LDAP プロジェクトとポリシーを使用する場合は、最初にその LDAP プロジェクトとポリシーをカーネルにロードする必要があります。 SMIT を使用すると、LDAP プロジェクトをカーネルにロードすることができます。

カーネルにロードする LDAP プロジェクトは、LDAP サーバー上のデフォルトのプロジェクト DN 上に存在する必要があります。

SMIT を使用して LDAP プロジェクトをカーネルにロードするには、次の手順を実行します。

1. root ユーザーとしてログインします。
2. SMIT メニューの「`Load/Re-load Project Definitions (プロジェクト定義のロード/再ロード)`」にアクセスします。その際の SMIT パスは、「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Project Definitions (プロジェクト定義)**」 > 「**Load/Re-load Project Definitions (プロジェクト定義のロード/再ロード)**」です。
3. プロジェクト・リポジトリーとして「`LDAP`」を選択します。
4. Enter を押すと、LDAP プロジェクト定義がカーネルにロードされます。

LDAP 管理ポリシーのカーネルへのロード (コマンド・ラインを使用する場合)

projctl ldadm –g コマンドを使用すると、LDAP 管理ポリシーをカーネルにロードすることができます。

LDAP プロジェクトをカーネルにロードするには、その管理ポリシー・ファイルが LDAP サーバー上のデフォルトの管理者 DN になければなりません。 LDAP プロジェクトをカーネルにロードするには、次の手順を実行します。

1. root ユーザーとしてログインします。
2. **projctl ldprojs –g** コマンドを実行します。このコマンドによって、`projdef` ファイルが LDAP サーバーから `/etc/project/ldap/` ディレクトリーにダウンロードされた後、そのファイル内のプロジェクト定義がカーネルにロードされます。

LDAP 管理ポリシーのカーネルへのロード (SMIT を使用する場合)

SMIT を使用すると、LDAP 管理ポリシーをカーネルにロードすることができます。

SMIT を使用して LDAP 管理ポリシーをカーネルにロードするには、以下の手順を実行します。

1. root ユーザーとしてログインします。

2. SMIT メニューの「Load/Re-load Admin Policy (管理ポリシーのロード/再ロード)」にアクセスします。その際の SMIT パスは、「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Automatic Project Assignment (プロジェクトの自動割り当て)**」 > 「**Work with Admin Policies (管理ポリシーの操作)**」 > 「**Load/Re-load Admin Policy (管理ポリシーのロード/再ロード)**」です。
3. LDAP を「**Current focus is on (現行フォーカスがオンです)**」として選択します。
4. Enter を押すと、LDAP 管理ポリシーがカーネルにロードされます。

ローカル・ファイルのプロジェクトとポリシー

プロジェクトとポリシーをローカル・ファイルへダウンロードする場合は、コマンド・ラインまたは SMIT を使用します。

LDAP プロジェクト定義のローカル・ファイルへのダウンロード (コマンド・ラインを使用する場合)

プロジェクト定義を LDAP サーバーからクライアントにダウンロードすることによって、新規プロジェクト定義を追加、または使用されていないプロジェクト定義を削除して、その後にプロジェクト定義ファイルをアップロードすることができます。プロジェクト定義ファイルは直接編集できるため、**projctl** コマンドを使ってそれらの作業を行う必要はありません。

LDAP プロジェクト定義ファイルをローカル・ディレクトリーにダウンロードする場合は、**projctl** コマンドを使用できます。このコマンドによって、プロジェクト定義ファイルが LDAP サーバー上のデフォルトのプロジェクト DN から指定のローカル・ディレクトリーにダウンロードされます。プロジェクト定義ファイルを特定の識別名 (DN) からダウンロードするには、**-g** フラグを使用して DN パラメーターを指定する必要があります。

拡張アカウンティング・サブシステムは、アカウンティング・ポリシーのロード時に自動的に LDAP サーバーからプロジェクト定義をダウンロードします (ただし、クライアント・システムがプロジェクト・リポジトリーとして LDAP を使用するように構成されている場合)。明示的に LDAP サーバーからプロジェクト定義をロードし、そのプロジェクト定義が自動的に構成されることを指定すると、プロジェクト定義はクライアントのセットアップ・フェーズ中にダウンロードされます。これを行うには、**projctl ldprojs -g -a** コマンドを使用します。

LDAP プロジェクトは各クライアント・システムにキャッシュされます。定期的にアカウンティング・データをサーバーからダウンロードするために、リフレッシュ・ポリシーを定義する必要があります。ポリシーによるリフレッシュの頻度を指定できるようにする SMIT メニュー項目があります。プロジェクトがクライアント・システムに自動的にダウンロードされる前に、新規ユーザーにそのシステムへのアクセスが許可されると、この遅延によって、その新規ユーザーを正しく分類するためのクライアント機能に影響が出る可能性があります。この場合、その新規ユーザーはシステムを使用できますが、正しく分類されません。ユーザーが正しく分類されていないことを示す SYSLOG メッセージが自動的に生成されます。リフレッシュ・コマンドは、ユーザーが作業の実行を予定しているクライアント・システムから手動で呼び出すことができます。リフレッシュ・コマンドは **projctl ldall -r** です。

LDAP プロジェクト定義のローカル・ファイルへのダウンロード (SMIT を使用する場合)

プロジェクト定義を LDAP サーバーからクライアントにダウンロードすることによって、新規プロジェクト定義を追加、または使用されていないプロジェクト定義を削除して、その後にプロジェクト定義ファイルをアップロードすることができます。プロジェクト定義ファイルは直接編集できるため、**projctl** コマンドを使ってそれらの作業を行う必要はありません。

SMIT を使用して LDAP プロジェクト定義をダウンロードするには、次の手順を実行します。

1. root ユーザーとしてログインします。
2. SMIT メニューの「Load/Re-load Project Definitions (プロジェクト定義のロード/再ロード)」にアクセスします。その際の SMIT パスは、「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Project Definitions (プロジェクト定義)**」 > 「**Download Project Definitions from LDAP server (プロジェクト定義の LDAP サーバーからのダウンロード)**」です。
3. LDAP サーバー上のローカル・プロジェクト定義ファイル・パスおよび場所を入力します。
4. Enter を押すと、プロジェクト定義ファイルがダウンロードされます。

LDAP 管理ポリシーのローカル・ファイルへのダウンロード (コマンド・ラインを使用する場合)

コマンド・ラインを使用すると、LDAP 管理ポリシーのローカル・コピーをクライアントにダウンロードすることができます。これによって、アップロードする前に、LDAP 管理ポリシー・ファイルのローカル・コピーに対して必要な更新をすべて適用できます。

LDAP 管理ポリシー・ファイルをローカル・ディレクトリーにダウンロードするには、**projctl ldadm –g –d local-dir** コマンドを使用します。このコマンドによって、管理ポリシー・ファイルが関連の別名およびプロジェクト定義ファイルと一緒に、LDAP サーバー上のデフォルトの管理者 DN およびプロジェクト DN から指定のローカル・ディレクトリーにダウンロードされます。特定の DN から管理ポリシー・ファイルをダウンロードするには、**–g** フラグを使用して DN パラメーターを指定します。特定の DN から管理ポリシーがダウンロードされると、プロジェクト定義ファイルはダウンロードされません。

LDAP 管理ポリシーのローカル・ファイルへのダウンロード (SMIT を使用する場合)

SMIT を使用すると、LDAP 管理ポリシーのローカル・コピーをクライアントにダウンロードすることができます。これによって、アップロードする前に、LDAP 管理ポリシー・ファイルのローカル・コピーに対して必要な更新をすべて適用できます。

SMIT を使用して LDAP 管理ポリシー・ファイルをダウンロードするには、次の手順を実行します。

1. root ユーザーとしてログインします。
2. SMIT メニューの「Load/Re-load Admin Policy (管理ポリシーのロード/再ロード)」にアクセスします。その際の SMIT パスは、「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Automatic Project Assignment (プロジェクトの自動割り当て)**」 > 「**Work with Admin Policies (管理ポリシーの操作)**」 > 「**Download Admin Policy from LDAP Server (管理ポリシーの LDAP サーバーからのダウンロード)**」です。
3. LDAP サーバーのフィールドに、ローカル管理ポリシー・ファイル・パスおよび場所を入力します。管理ポリシー・ファイル名を指定しない場合、デフォルトのポリシー・ファイルがダウンロードされます。管理ポリシー・ファイル名を指定してダウンロードするには、「**Admin Policy Name (管理ポリシー名)**」フィールドに入力します。
4. 管理ポリシーをダウンロードするには、Enter を押してください。

LDAP プロジェクトの更新

クライアント・マシンに LDAP プロジェクトをダウンロードしなくても、それらのプロジェクトに必要な更新を行うことができます。

LDAP プロジェクトを更新するには、**projctl** のサブコマンド **add**、**rm**、および **chattr** を使用します。

LDAP サーバーのプロジェクト・リポジトリーを更新するには、**-p** フラグを使用します。

LDAP プロジェクトの更新 (コマンド・ラインを使用する場合)

コマンド・ラインを使用して、新規プロジェクト定義を LDAP プロジェクト定義ファイルに追加、またはそこから削除することができます。

新規プロジェクト定義を LDAP プロジェクト定義ファイルに追加するには、**projctl add projname projnumber –p** コマンドを使用します。このコマンドによって、新規プロジェクト定義がデフォルトの LDAP プロジェクト DN 上のプロジェクト定義ファイルに追加されます。

プロジェクト定義を LDAP プロジェクト定義ファイルから除去するには、**projctl rm projname –p** コマンドを使用します。このコマンドによって、指定したプロジェクト定義がデフォルトのプロジェクト DN 上にあるプロジェクト定義ファイルから除去されます。

LDAP プロジェクト定義ファイル内のプロジェクト定義の集約属性を使用可能にするには、**projctl chattr agg projname –s –p** コマンドを使用します。このコマンドによって、LDAP プロジェクト定義ファイルに指定されているプロジェクト定義の集約属性が使用可能になります。

上記のいずれの操作でも、特定の LDAP DN のプロジェクト定義を変更する必要がある場合は、**–p** フラグを使用して DN パラメーターを指定します。

LDAP プロジェクトの更新 (SMIT を使用する場合)

SMIT を使用して、新規プロジェクト定義を LDAP プロジェクト定義ファイルに追加、またはそこから削除することができます。

SMIT パスを使用して新規プロジェクト定義を LDAP プロジェクト定義ファイルに追加する場合、SMIT パスとして「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Project Definitions (プロジェクト定義)**」 > 「**Add Project Definition (プロジェクト定義の追加)**」を使用します。

SMIT を使用してプロジェクト定義を LDAP プロジェクト定義ファイルから除去する場合、SMIT パスとして「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Project Definitions (プロジェクト定義)**」 > 「**Remove Project Definition (プロジェクト定義の除去)**」を使用します。

SMIT を使用して LDAP プロジェクト定義ファイル内のプロジェクト定義の集約属性を使用可能にする場合は、SMIT パスとして「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Project Definitions (プロジェクト定義)**」 > 「**Show/Change Project Definitions (プロジェクト定義の表示/変更)**」を使用します。

上記のいずれの操作でも、特定の LDAP DN のプロジェクト定義を変更する必要がある場合は、**–p** フラグを使用して DN パラメーターを指定します。

プロジェクトとポリシーの除去およびアンロード

プロジェクトとポリシーをアンロードおよび除去する場合は、コマンド・ラインまたは SMIT を使用します。

LDAP プロジェクトのカーネルからのアンロード

LDAP プロジェクトをカーネルからアンロードすることができます。

LDAP プロジェクトをカーネルからアンロードするには、**projctl unldprojs –g** コマンドを使用します。このコマンドでは、LDAP リポジトリーからロードされたプロジェクト定義のみが除去されます。

SMIT を使用して LDAP プロジェクトをカーネルからアンロードする場合、SMIT パスとして「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Project Definitions (プロジェクト定義)**」 > 「**Unload Active Project Definitions (アクティブ・プロジェクト定義のアンロード)**」を使用します。

LDAP ポリシーのカーネルからのアンロード

LDAP 管理ポリシーをカーネルからアンロードすることができます。

LDAP 管理ポリシーをカーネルからアンロードするには、**projctl unldadm –g** コマンドを使用します。このコマンドでは、LDAP リポジトリーからロードされた管理ポリシーのみが除去されます。

SMIT を使用して LDAP 管理ポリシーをカーネルからアンロードする場合は、SMIT パスとして「**smitty aacct**」 > 「**Manage Project Definitions and Assignments (プロジェクト定義および割り当ての管理)**」 > 「**Automatic Project Assignment (プロジェクトの自動割り当て)**」 > 「**Work with Admin Policies (管理ポリシーの操作)**」 > 「**Unload Admin Policy (管理ポリシーのアンロード)**」を使用します。

LDAP プロジェクトのサーバーからの削除

LDAP プロジェクトをサーバーから削除することができます。

LDAP プロジェクト・リポジトリーを LDAP サーバーから削除するには、**projctl unldprojs –p** コマンドを使用します。このコマンドを使用して、**-p** フラグへの引数としてデフォルト DN を指定することにより、クライアント・システム用に構成されたデフォルト DN に配置されないプロジェクトを削除することもできます。

LDAP 管理ポリシーのサーバーからの削除

LDAP 管理ポリシーをサーバーから削除することができます。

LDAP 管理ポリシーを LDAP サーバーから削除するには、**projctl unldadm –p** コマンドを使用します。このコマンドでは、クライアント用に構成されたデフォルト管理ポリシーがデフォルト DN から削除されます。代替の管理ポリシーを削除するには、削除するポリシーの名前に **-p** フラグを付けて指定する必要があります。このコマンドを使用して、デフォルトの DN を明示的に指定することにより、その DN に配置されない管理ポリシーを削除することもできます。これは、次の方法で代替ポリシー名に結合できます。**-p** *name*:*DN*。名前および DN は、コロンで区切られる必要があります。コロンは、ポリシーの名前または DN では使用できません。

データ集約

データ集約は、データを累算して 1 つのアカウンティング・レコードに入れる方法であり、それ以外の場合は、複数のレコードを使用して提示されます。

集約されたデータは、19 ページの『間隔アカウンティング』で説明されている間隔に応じて、定期的に書き込まれます。

注:

1. データ集約を使用するためには、間隔アカウンティングを使用可能に設定する必要があります。
2. プロセス間隔およびシステム間隔を 60 分に設定します。

アカウンティング・データは、アプリケーションまたはミドルウェアに影響せずに、カーネル内部で合計されます。データは、間隔アカウンティングによって使用可能になります。これは、それらのレコードを定期的に **acct** ファイルに書き込むカーネル機能です。カーネルは、レコードを集約する際に、それらのレコードを、内部で管理される集約レコードのセットにマップします。これらのレコードは、システムに対して入力されたが、拡張アカウンティング内部で記録メカニズムにコミットされていないという意味では処理待ち状態です。間隔アカウンティングは、集計済みレコードが **acct** ファイルに書き込まれるようにそれらをコミットするのに使用されます。

集約データは、さまざまなデータ構造を使用して記録されるため、請求アプリケーションがそれらの構造を認識していることを確認する必要があります。ご使用の請求アプリケーションがデータ集約をサポートしているかどうかを判別するには、その請求アプリケーションに付属の資料をご覧ください。データ集約は、システム・レベルまたはプロジェクト・レベルで使用可能または使用不可の設定が可能です。

システム・レベルのデータ集約コマンドと高速パス

システム・レベルのデータ集約は、コマンド・ラインまたは SMIT 高速パスを使って管理することができます。

システム・レベルのデータ集約コマンドを使用して、プロセス・データ集約、カーネル・エクステンション・データ集約、および ARM トランザクション・データ集約を使用可能/使用不可にすることができます。以下の表には、プロセス・データ集約、カーネル・エクステンション・データ集約、および ARM トランザクション・データ集約を使用可能/使用不可にするためのコマンドがリストされています。

表 15. システム・レベルのデータ集約コマンド

タスク	コマンド	SMIT 高速パス
システム全体にわたるプロセス集約の使用可能化/使用不可化	**acctctl agproc {on|off}**	**smit system_paggr**
システム全体にわたるサード・パーティー・カーネル・エクステンションの集約の使用可能化/使用不可化	**acctctl agke {on|off}**	**smit system_kaggr**
システム全体にわたる ARM トランザクションの集約の使用可能化/使用不可化	**acctctl agarm {on|off}**	**smit system_aaggr**
全般的なアカウンティング状態の照会	**acctctl**	なし

詳しくは、「*Commands Reference, Volume 1*」を参照してください。

プロジェクト・レベルのデータ集約コマンド

プロジェクト・レベルのデータ集約は、コマンド・ラインまたは SMIT 高速パスを使って管理することができます。

以下の表には、個々のプロジェクトの集約を使用可能/使用不可にするためのコマンドが示されています。

表 16. プロジェクト・レベルのデータ集約コマンド

タスク	コマンド	SMIT 高速パス
(*projname* パラメーターによって指定された) プロジェクトの集約を使用可能/使用不可にする	**projctl chattr agg** *projname* {**-sl-u**} [**-d** *projpath*]	なし
すべてのプロジェクトの集約状態の照会	**projctl qprojs** [**-n**]	なし
指定したプロジェクトの集約状態の照会	**projctl qproj** [*projectname*]	なし

詳しくは、「*Commands Reference, Volume 4*」を参照してください。

レポートおよび分析

拡張アカウンティング・サブシステムは、通常チャージバック・メカニズムに含まれているさまざまなリソースに関するアカウンティング・データを提供します。

アカウンティング・データ・ファイルのフォーマット、および個々のアカウンティング・レコードのフォーマットは、sys/aacct.h ヘッダー・ファイルの中に記載されています。アカウンティング・データ・ファイルを解析し、アカウンティング・レポートを生成するには、**acctrpt** コマンドを使用します。このコマンドを使用すると、プロセス・アカウンティング・レポート、LPAR アカウンティング・レポート、トランザクション・アカウンティング・レポート、およびワークロード・パーティション・アカウンティング・レポートの 4 つのタイプのアカウンティング・レポートを生成することができます。

acctrpt コマンドと共に、AIX にはアカウンティング・データ・ファイルを解析するサンプル・プログラムが用意されています。そのサンプル・プログラムは **readaacct** です。このプログラムを使用して、アカウンティング・データを分析し、アカウンティング・データをスプレッドシートにインポートすることができます。このコマンドの構文は次のとおりです。

```
/usr/samples/aacct/readaacct [-F file] [-t trid] [-b begin_time] [-e end_time] [-c] [-h] [-@ <wpar>]
```

このコマンドには以下のフラグがあります。

- **-c** フラグを使用すると、コロンで区切られたフォーマットで情報が表示されます。
- **-h** フラグを使用すると、ファイルに関する情報 (ホスト名とマシン・モデル、データが生成された場所のシリアル番号など) が表示されます。
- **-b** および **-e** フラグを使用すると、時間に基づいて情報が表示されます。
- **-t** フラグを使用すると、レコードに基づいて情報が表示されます。
- **-@** フラグはワークロード・パーティションに固有のレコードを提供します。

以下は、**readaacct -F /tmp/afile -h** コマンドを実行した後に表示される出力の例です。

```
# readaacct -F /tmp/afile -h
File Name=/tmp/a
Version=0
Flags=0
Offset=3084288
File Size=3145728
State=2
ID=1
First Time=1087266596
Last Time=1087301336
System ID=IBM,01025990A
System Model=IBM,7040-681
Host Name=bigboylp9
Partition Name=bigboylp9
Partition Number=9
```

```
------------------------------------------------------------
Transaction ID=1
Flags=f1
Transaction Project=0
Sub project ID=0
Transaction start time=6-14-2004 21:29:56
UID=0
GID=0
PID=335912
eWLM Service Class=0
Flags=1
Command Name=acctctl
Controlling Terminal's Device Number=23,5
Process Start Time=1087266596
WLM Class key=7770295601810996315
Incrementing Statistics:
Elapsed process time=0.032406 seconds
Elapsed thread time=0.032406 seconds
Process CPU time=0.015238 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=5 seconds
Elapsed Page seconds of virtual memory=5 seconds
Bytes of local file I/O=0
Bytes of other file I/O=0
Bytes of local sockets=0
Bytes of remote sockets=0
```

プロセスのアカウンティング・レポートの例

プロセスのアカウンティング・レポートを生成することができます。

プロセスのアカウンティング・レポート

以下は、プロセスのアカウンティング・レポートの例です。

```
 # /usr/bin/acctrpt -f /var/aacct/acctdata
                                                                                (C) PELAPSE   TELAPS   CPU   (sec)
                                                                                (M) VMEM      DMEM     PMEM  (pg)
                                                                                (F) LFILE     DFILE          (MB)
TIMESTAMP  PROJID   UID      GID      PID      CMD          STARTED  EXITED (S) LSOCKET   DSOCKET        (MB)
---------  ------   ---      ---      ---      ---          -------  ------ --- -------   -------  ------
11151936   System   root     system   524386   acctctl      11151936 E       C: 0.3824    0.3824   0.2394
                                                                               M: 25        27       0
                                                                               F: 0.00      0.00
                                                                               S: 0.00      0.00
11151936   System   root     system   524388   nfssync_kpro 11151936 E       C: 0.0008    0.0008   0.0003
                                                                               M: 0         0        0
                                                                               F: 0.00      0.00
                                                                               S: 0.00      0.00
```

集約されたプロセスのアカウンティング・レポート

以下は、集約されたプロセスのアカウンティング・レポートの例です。 **acctrpt** コマンドのフラグ **-P**、**-U**、**-G**、および **-C** は、集約されたプロセスのアカウンティング・レポートの準備を容易にします。これらのフラグはそれぞれ、プロジェクト ID、ユーザー ID、グループ ID、およびコマンド名に対応します。管理者は、これらのフラグの順序を変更してデータの表示順序を制御することができます。以下に、すべてのユーザー ID に基づく集約されたプロセスのアカウンティング・レポートの例を示します。

```
 # /usr/bin/acctrpt -U ALL -f /var/aacct/acctdata
```

```
                                                  (C) PELAPSE  TELAPSE  CPU      (sec)
                                                  (M) VMEM     DMEM     PMEM     (pg)
                                                  (F) LFILE    DFILE             (MB)
PROJID   UID      GID      CMD          CNT       (S) LSOCKET  RSOCKET           (MB)
         ---                                      --- -------- -------- --------
-        root     -        -            590       C:  54880.8  281458.2 60.3
                                                  M:  6        14100294 14098216
                                                  F:  32.3     1.4
                                                  S:  0.0      19.8
-        104      -        -            28        C:  3.3      3.3      1.0
                                                  M:  186      299      708
                                                  F:  8.3      0.0
                                                  S:  0.0      0.0
```

LPAR アカウンティング・レポートの例

LPAR アカウンティング・レポートを生成することができます。

LPAR アカウンティング・レポート

以下は、LPAR アカウンティング・レポートの例です。 **acctrpt** コマンドの **-L** フラグを使用して、特定の LPAR リソースに対するアカウンティング・レポートを生成します。以下に、ファイルシステム統計の例を示します。

```
# /usr/bin/acctrpt -L FILESYS -f /var/aacct/acctdata

File Systems Accounting Report
------------------------------
CNT    DEVNAME            MOUNTPT    FSTYPE RDWR    OPEN    CREATE  LOCKS   XFERS(MBs)
7      specfs             specfs     16     48853   1066    0       0       1.2
7      pipefs             pipefs     16     3827    0       0       0       3.3
7      /export/sp1n1fs2u /farm/cwu   18     4917    4510    750     0       28.0
```

トランザクションのアカウンティング・レポートの例

トランザクションのアカウンティング・レポートを生成することができます。

トランザクションのアカウンティング・レポート

以下は、トランザクションのアカウンティング・レポートの例です。 ARM トランザクションのアカウンティング・レポートを生成するには、**acctrpt** コマンドと一緒に **-T** フラグを使用します。

```
# /usr/bin/acctrpt -T -f /var/aacct/acctdata

                    (A)  CLASS           NAME            USER     GROUP    TRANSACTION
PROJID   CNT        (T)  RESPONSE        QUEUED          CPU                            (sec)
------   ---        ---  --------        -------         ----    -------   ------------
System   144        A:  -               WebSphere        -       server1   URI
                    T:  0.00            0.00           0.00
System   32         A:  -               IBM Webserving   -       IBM_SERV  Apache/1.3.28(Unix)
                    T:  0.00            0.00           67.01
```

アカウンティング・レコード

拡張アカウンティングが生成するレコードには複数のタイプがあります。

拡張アカウンティングが生成するアカウンティング・レコードは、sys/aacct.h ファイルの中で定義されています。以下の表は、これらのレコードについて説明しています。

表 17. アカウンティング・レコード

アカウンティング・レコード	説明
パッド・レコード (タイプ 0)	このレコードには、意味のあるアカウンティング・データが含まれません。レポート・ツールおよび分析ツールは、このレコードをスキップします。これは、単に位置合わせのために生成されるレコードです。
プロセス・レコード (タイプ 1)	このレコードは、プロセスが終了したとき、プロセスが再分類されたとき (**setUser ID()**、**chproj()**、**exec()**)、およびシステムが再分類されたときに書き込まれます。このレコードはプロセス間隔ごとに書き込まれます。 このレコードには、以下の情報が含まれます。 • ユーザー ID • グループ ID • プロセス ID • プロセス・フラグ (終了、コア、シグナルによって強制終了、チェックポイントによって強制終了) • 基本コマンド名 • WLM クラス • 制御端末 • プロセスの開始時間 (秒、EPOCH から) • プロセスの経過時間 (マイクロ秒) • 結合スレッドの経過時間 (マイクロ秒) • プロセス (結合スレッド) のプロセッサー時間 (マイクロ秒) • ディスク・ページの経過ページ秒 • 実ページの経過ページ秒 • 仮想メモリーの経過ページ秒 • ローカル論理ファイルの入出力 (JFS、J2、バイト単位) • 他の論理ファイルの入出力 (NFS、DFS、バイト単位) • ローカル・ソケットの入出力 (UNIX ドメインおよびループバック、バイト単位) • リモート・ソケットの入出力 (バイト単位) プロセスの開始時間およびプロセス ID を使用して、特定のプロセスの間隔レコードを相互に関係させることができます。終了フラグを使用して、間隔レコードと終了レコードを区別することができます。

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
集約されたプロセス・レコード (タイプ 2)	このレコードは、プロセス・レコードから派生します。プロジェクトによって各ユーザーごとに異なるレコードが生成されます。 このレコードはプロセス間隔ごとに生成され、以下の情報を含んでいます。 • ユーザー ID • 最初のレコードが集約された時間 (秒、EPOCH から) • 集約されたプロセスの数 • 集約されたプロセスの経過時間 (マイクロ秒) • 集約されたスレッドの経過時間 (マイクロ秒) • 集約されたプロセス (結合スレッド) のプロセッサー時間 (マイクロ秒) • ディスク・ページの集約された経過ページ秒 • 実ページの集約された経過ページ秒 • 仮想メモリーの集約された経過ページ秒 • 集約されたローカル論理ファイル入出力 (JFS、J2、バイト単位) • 集約された他の論理ファイル入出力 (NFS、DFS、バイト単位) • 集約されたローカル・ソケット入出力 (UNIX ドメインおよびループバック、バイト単位) • 集約されたリモート・ソケット入出力 (バイト単位)

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
集約されたアプリケーション・レコード (タイプ 3)	このレコードは、プロセス・レコードから派生します。レコードは、ユーザー・レベル、プロジェクト・レベル、およびアプリケーション・レベルで生成されます。このレコードは集約されたプロセス・レコードと似ていますが、アプリケーションには名前が付いている点が異なります。このレコードは、(管理ポリシーのみによってサポートされる) アプリケーション固有の規則を使ってプロセスが分類されたときに生成されます。 このレコードはプロセス間隔ごとに生成され、以下の情報を含んでいます。 • ユーザー ID • 最初のレコードが集約された時間 (秒、EPOCH から) • プロジェクト分類を生成したコマンドの i ノード • プロジェクト分類を生成したコマンドのデバイス番号 • 集約されたアプリケーションの数 • 集約されたプロセスの経過時間 (マイクロ秒) • 集約されたスレッドの経過時間 (マイクロ秒) • 集約されたプロセス (結合スレッド) のプロセッサー時間 (マイクロ秒) • ディスク・ページの集約された経過ページ秒 • 実ページの集約された経過ページ秒 • 仮想メモリーの集約された経過ページ秒 • 集約されたローカル論理ファイル入出力 (JFS、J2、バイト単位) • 集約された他の論理ファイル入出力 (NFS、DFS、バイト単位) • 集約されたローカル・ソケット入出力 (UNIX ドメインおよびループバック、バイト単位) • 集約されたリモート・ソケット入出力 (バイト単位)

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
プロセッサーとメモリーの使用状況レコード (タイプ 4)	このレコードは、プロセッサーの使用状況、およびラージ・ページ・プールのサイズが変更されたときの情報を提供します。このレコードも、事前マイグレーションおよび事後マイグレーションの間にシステム間隔ごとに生成されます。 このレコードには、以下の情報が含まれます。 • レコードが生成された理由 • オンライン論理プロセッサーの数 • パーティションのライセンス済みプロセッサー・キャパシティー • アイドル時間の合計 (ミリ秒) • 入出力待ち時間の合計 (ミリ秒) • カーネル・プロセス時間の合計 (ミリ秒) • ユーザー・プロセス時間の合計 (ミリ秒) • 割り込み時間の合計 (ミリ秒) • 物理メモリーのサイズ (MB) • ラージ・ページ・プールのサイズ (MB) • 使用中のラージ・ページ (MB) • ページング・スペースからのページ・イン数 • ページング・スペースへのページ・アウト数 • スタート I/O の数 • ページ・スチールの数 • 間隔の開始時点からの経過時間 (ミリ秒) • パーティションに割り当てられる物理メモリーの量 (ページ秒) • パーティションに割り当てられるライセンス済みメモリーの量 (4K ページで報告) • パーティションが使用中のライセンス済み入出力メモリーの量 (ページ秒)
ポリシー・レコード (タイプ 5)	このレコードは、ポリシー・ファイルがロードまたはアンロードされたときに書き込まれます。これは、単に通知のために提供されるレコードです。 このレコードには、以下の情報が含まれます。 • ポリシーのタイプ: 管理、ユーザー、またはグループ • ロードまたはアンロード

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
ファイルシステム・アクティビティー・レコード (タイプ 6)	このレコードは、システム・レベルでのファイルシステムの使用状況を示します。マウントされたファイルシステムごとに別個のレコードが生成されます。 このレコードはシステム間隔ごとに生成され、以下の情報を含んでいます。 • 読み取り/書き込みによって転送された合計バイト数 • 読み取り/書き込み要求の総数 • オープンの総数 • 作成の総数 • ロックの総数 • ファイルシステム・タイプ • デバイス名 • マウント・ポイント
ネットワーク・インターフェース入出力レコード (タイプ 7)	このレコードは、システム・レベルでのネットワーク・インターフェースの使用状況に関する情報を提供します。 このレコードはシステム間隔ごとに生成され、以下の情報を含んでいます。 • ネットワーク・インターフェースの論理名 • 入出力の数 • 転送された合計バイト数
ディスク入出力レコード (タイプ 8)	このレコードは、システム・レベルでのディスクの使用状況に関する情報を提供します。それぞれの論理ディスク装置ごとに別個のレコードが書き込まれます。 このレコードはシステム間隔ごとに生成され、以下の情報を含んでいます。 • ディスクの論理名 • ディスク転送の総数 • 読み取りの総数 • 書き込みの総数 • ディスク転送のブロック・サイズ
損失データ・レコード (タイプ 9)	このレコードは、拡張アカウンティングによって記録できなかったために削除されたアカウンティング・レコードに関する情報を提供します。すべてのアカウンティング・データ・ファイルがいっぱいのとき、これが発生します。新しいアカウンティング・レコードを書き込むことが再び可能になると、拡張アカウンティングは障害を示す損失データ・レコードを生成します。 このレコードには、以下の情報が含まれます。 • 損失レコードの数 • プロセス・レコードに関連した損失したプロセッサー時間 (マイクロ秒) • データ損失が始まった時間 (マイクロ秒、EPOCH から) • サード・パーティー・カーネル・エクステンション・レコードに関連した損失したプロセッサー時間 (マイクロ秒)

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
サーバー VIO レコード (タイプ 10)	このレコードは、ホスト・パーティションで生成されます。クライアント・パーティションと共有されるそれぞれの論理デバイスごとに別個のレコードが生成されます。定期的にこのレコードを生成するために、システム間隔を使用することができます。 このレコードには、以下の情報が含まれます。 • クライアント・パーティション番号 • サーバーの装置 ID • デバイスの論理装置 ID (LUN) • 受信バイト数 • 送信バイト数
クライアント VIO レコード (タイプ 11)	このレコードは、クライアント・パーティションで生成されます。クライアント・パーティションでの仮想デバイスの使用状況を示します。仮想デバイスのインスタンスごとに、別個のレコードが記録されます。定期的にこのレコードを生成するために、システム間隔を使用することができます。 このレコードには、以下の情報が含まれます。 • サーバー・パーティション番号 • サーバーの装置 ID • デバイスの論理装置 ID • 受信バイト数 • 送信バイト数
サード・パーティー・カーネル・エクステンション共通の集約レコード (タイプ 12)	このレコードは、指定したアカウンティング・レコードのアカウンティング情報を提供します。サード・パーティー・カーネル・エクステンションによって生成される集約アカウンティング・レコードから派生します。このレコードは、システム間隔ごとに拡張アカウンティングに書き込まれます。 このレコードには、以下の情報が含まれます。 • カーネル・エクステンションのコマンド名 (u ブロックから) • サード・パーティー・カーネル・エクステンションのトランザクション ID (129 から 256 の範囲) • 集約されたアカウンティング・レコードの数 • このトランザクション・クラスのリソース使用状況または累積プロセッサー時間 • 最初のレコードが集約された時間 (秒、EPOCH から)

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
ARM アプリケーション環境レコード (タイプ 13)	このレコードは、アプリケーション環境インスタンスを示します。 **arm_register_application()** システム・コールおよび **arm_start_application()** システム・コールを介してオペレーティング・システムに渡されるデータから作成されます。このレコードは可変長です。すべてのオフセットは、レコードの開始位置から相対的に計算されます。 このレコードには、以下の情報が含まれます。 • このレコードのデータの記録に使われた文字セット • アプリケーション環境 ID • アプリケーション名のオフセット • アプリケーション・グループのオフセット • アプリケーション ID 属性のオフセット • アプリケーション・コンテキスト属性のオフセット オペレーティング・システムはアプリケーション環境の内容をそれぞれのアカウンティング・データ・ファイルに記録しようとします。このため、各アカウンティング・データ・ファイルを独立した項目として後処理することが可能です。これにより、アカウンティング・データ・ファイルの相互依存を防ぐことができます。
ARM トランザクション環境レコード (タイプ 14)	このレコードは、トランザクション環境インスタンスを示します。 **arm_register_transaction()** システム・コールを介してオペレーティング・システムに渡されるデータから作成されます。このレコードは可変長です。すべてのオフセットは、レコードの開始位置から相対的に計算されます。 このレコードには、以下の情報が含まれます。 • このレコードのデータの記録に使われた文字セット • トランザクション環境 ID • トランザクション名のオフセット • アプリケーション ID 属性のオフセット • アプリケーション・コンテキスト属性のオフセット (名前のみ) オペレーティング・システムはトランザクション環境の内容をそれぞれのアカウンティング・データ・ファイルに記録しようとします (必ずしも保障されてはいません)。このため、各アカウンティング・データ・ファイルを独立した項目として後処理することが可能です。これにより、アカウンティング・データ・ファイルの相互依存を防ぐことができます。

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
ARM トランザクション・インスタンス・レコード (タイプ 15)	このレコードは ARM トランザクション・インスタンスを示します。 **arm_start_transaction()** および **arm_stop_transaction()** システム・コールを介してオペレーティング・システムに渡されるデータから作成されます。これは可変長です。すべてのオフセットは、レコードの開始位置から相対的に計算されます。 このレコードには、以下の情報が含まれます。 • トランザクションの完了状況 • アプリケーション環境 ID • トランザクション環境 ID • ユーザー識別 (User ID ではない) のオフセット • ユーザー名 (uname ではない) のオフセット • アカウンティング・コードのオフセット • 応答時間 (ミリ秒) • キュー時間 (ミリ秒) • リソース使用状況 アプリケーションおよびトランザクションの環境 ID は、それぞれアプリケーションおよびトランザクションの環境レコードで定義されます。これらのレコードは、アプリケーション名、アプリケーション・グループ、トランザクション名、および属性をトランザクション・インスタンスと関連付けるために使用される必要があります。
集約された ARM トランザクション・インスタンス・レコード (タイプ 16)	このレコードは、ARM トランザクションの集約が使用可能になっている場合に ARM トランザクション・インスタンス・レコード (タイプ 15) の代わりに生成されます。 このレコードには、以下の情報が含まれます。 • トランザクションの完了状況 • 最初のレコードが集約された時間 (秒、EPOCH から) • アプリケーション環境 ID • トランザクション環境 ID • ユーザー識別 (User ID ではない) のオフセット • ユーザー名 (uname ではない) のオフセット • アカウンティング・コードのオフセット • 集約された応答時間 (ミリ秒) • 集約されたキュー時間 (ミリ秒) • 集約されたリソース使用状況

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
プロジェクト定義レコード (タイプ 17)	このレコードは、プロジェクト定義のリストを示します。これは、プロジェクト定義ファイルのロード時に書き込まれます。すべてのプロジェクト定義を記録するために、複数のレコードが必要になる場合があります。このレコードを使って完全なプロジェクト情報がそれぞれのデータ・ファイル内に作成されるため、各データ・ファイルを独立したエンティティーとして扱うことが可能です。請求アプリケーションの性質によっては、これが必須でないこともあります。この機能を使用不可にするには、プロジェクト定義アカウンティング・レコードを使用不可にします。 このレコードは可変長で、以下の情報を含んでいます。 • プロジェクトの数 • プロジェクト定義領域のバイト数 • プロジェクト定義領域
WPAR プロセス・レコード (タイプ 33)	このレコードは WPAR アカウンティングがグローバル WPAR で使用可能になっている場合のみ作成されます。このレコードは、アプリケーション WPAR のプロセスが終了したとき、プロセスが再分類されたとき (**setUser ID()**、**chproj()**、**exec()**)、およびシステムが再分類されたときに書き込まれます。このレコードはプロセス間隔ごとに書き込まれます。 このレコードには、以下の情報が含まれます。 • ユーザー ID • グループ ID • プロセス ID • プロセス・フラグ (終了、コア、シグナルによって強制終了、チェックポイントによって強制終了) • 基本コマンド名 • WLM クラス • 制御端末 • プロセスの開始時間 (秒、EPOCH から) • プロセスの経過時間 (マイクロ秒) • 結合スレッドの経過時間 (マイクロ秒) • プロセス (結合スレッド) のプロセッサー時間 (マイクロ秒) • ディスク・ページの経過ページ秒 • 実ページの経過ページ秒 • 仮想メモリーの経過ページ秒 • ローカル論理ファイルの入出力 (JFS、J2、バイト単位) • 他の論理ファイルの入出力 (NFS、DFS、バイト単位) • ローカル・ソケットの入出力 (UNIX ドメインおよびループバック、バイト単位) • リモート・ソケットの入出力 (バイト単位) • WPAR 名 プロセスの開始時間およびプロセス ID を使用して、特定のプロセスの間隔レコードを相互に関係させることができます。終了フラグを使用して、間隔レコードと終了レコードを区別することができます。

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
WPAR の集約されたプロセス・レコード (タイプ 34)	このレコードは WPAR アカウンティングがグローバル WPAR で使用可能になっている場合のみ作成されます。このレコードは、アプリケーション WPAR の WPAR プロセス・レコードから派生します。プロジェクトによって各ユーザーごとに異なるレコードが生成されます。 このレコードはプロセス間隔ごとに生成され、以下の情報を含んでいます。 • ユーザー ID • 最初のレコードが集約された時間 (秒、EPOCH から) • 集約されたプロセスの数 • 集約されたプロセスの経過時間 (マイクロ秒) • 集約されたスレッドの経過時間 (マイクロ秒) • 集約されたプロセス (結合スレッド) のプロセッサー時間 (マイクロ秒) • ディスク・ページの集約された経過ページ秒 • 実ページの集約された経過ページ秒 • 仮想メモリーの集約された経過ページ秒 • 集約されたローカル論理ファイル入出力 (JFS、J2、バイト単位) • 集約された他の論理ファイル入出力 (NFS、DFS、バイト単位) • 集約されたローカル・ソケット入出力 (UNIX ドメインおよびループバック、バイト単位) • 集約されたリモート・ソケット入出力 (バイト単位) • WPAR 名

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
WPAR の集約されたアプリケーション・レコード (タイプ 35)	このレコードは WPAR アカウンティングがグローバル WPAR で使用可能になっている場合のみ作成されます。このレコードは、アプリケーション WPAR の WPAR プロセス・レコードから派生します。レコードは、ユーザー・レベル、プロジェクト・レベル、およびアプリケーション・レベルで生成されます。このレコードは集約されたプロセス・レコードと似ていますが、アプリケーションには名前が付いている点が異なります。このレコードは、(管理ポリシーのみによってサポートされる) アプリケーション固有の規則を使ってプロセスが分類されたときに生成されます。 このレコードはプロセス間隔ごとに生成され、以下の情報を含んでいます。 • ユーザー ID • 最初のレコードが集約された時間 (秒、EPOCH から) • プロジェクト分類を生成したコマンドの i ノード • プロジェクト分類を生成したコマンドのデバイス番号 • 集約されたアプリケーションの数 • 集約されたプロセスの経過時間 (マイクロ秒) • 集約されたスレッドの経過時間 (マイクロ秒) • 集約されたプロセス (結合スレッド) のプロセッサー時間 (マイクロ秒) • ディスク・ページの集約された経過ページ秒 • 実ページの集約された経過ページ秒 • 仮想メモリーの集約された経過ページ秒 • 集約されたローカル論理ファイル入出力 (JFS、J2、バイト単位) • 集約された他の論理ファイル入出力 (NFS、DFS、バイト単位) • 集約されたローカル・ソケット入出力 (UNIX ドメインおよびループバック、バイト単位) • 集約されたリモート・ソケット入出力 (バイト単位) • WPAR 名

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
WPAR プロセッサーとメモリーの使用状況レコード (タイプ 36)	このレコードは WPAR アカウンティングがグローバル WPAR で使用可能になっている場合のみ作成されます。このレコードは、システム/アプリケーション WPAR のプロセッサー使用状況に関する情報、およびラージ・ページ・プールのサイズが変更されたときの情報を提供します。このレコードも、事前マイグレーションおよび事後マイグレーションの間にシステム間隔ごとに生成されます。 このレコードには、以下の情報が含まれます。 • レコードが生成された理由 • オンライン論理プロセッサーの数 • パーティションのライセンス済みプロセッサー・キャパシティー • アイドル時間の合計 (ミリ秒) • 入出力待ち時間の合計 (ミリ秒) • カーネル・プロセス時間の合計 (ミリ秒) • ユーザー・プロセス時間の合計 (ミリ秒) • 割り込み時間の合計 (ミリ秒) • 物理メモリーのサイズ (MB) • ラージ・ページ・プールのサイズ (MB) • 使用中のラージ・ページ (MB) • ページング・スペースからのページ・イン数 • ページング・スペースへのページ・アウト数 • スタート I/O の数 • ページ・スチールの数 • 間隔の開始時点からの経過時間 (ミリ秒) • WPAR 名
WPAR ファイルシステム・アクティビティー・レコード (タイプ 38)	このレコードは WPAR アカウンティングがグローバル WPAR で使用可能になっている場合のみ作成されます。このレコードは、システム・レベルでのシステム/アプリケーション WPAR に固有のファイルシステムの使用状況を示します。マウントされたファイルシステムごとに別個のレコードが生成されます。 このレコードはシステム間隔ごとに生成され、以下の情報を含んでいます。 • 読み取り/書き込みによって転送された合計バイト数 • 読み取り/書き込み要求の総数 • オープンの総数 • 作成の総数 • ロックの総数 • ファイルシステム・タイプ • デバイス名 • マウント・ポイント • WPAR 名

表 17. アカウンティング・レコード (続き)

アカウンティング・レコード	説明
WPAR ネットワーク・インターフェース入出力レコード (タイプ 39)	このレコードは WPAR アカウンティングがグローバル WPAR で使用可能になっている場合のみ作成されます。このレコードは、システム・レベルでのシステム/アプリケーション WPAR によるネットワーク・インターフェースの使用状況に関する情報を提供します。 このレコードはシステム間隔ごとに生成され、以下の情報を含んでいます。 • ネットワーク・インターフェースの論理名 • 入出力の数 • 転送された合計バイト数 • WPAR 名
WPAR サード・パーティー・カーネル・エクステンション共通の集約レコード (タイプ 44)	このレコードは WPAR アカウンティングがグローバル WPAR で使用可能になっている場合のみ作成されます。このレコードは、指定したアカウンティング・レコードのアカウンティング情報を提供します。サード・パーティー・カーネル・エクステンションによって生成されるアプリケーション WPAR の集約アカウンティング・レコードから派生します。このレコードは、システム間隔ごとに拡張アカウンティングに書き込まれます。 このレコードには、以下の情報が含まれます。 • カーネル・エクステンションのコマンド名 (u ブロックから) • サード・パーティー・カーネル・エクステンションのトランザクション ID (129 から 256 の範囲) • 集約されたアカウンティング・レコードの数 • このトランザクション・クラスのリソース使用状況または累積プロセッサー時間 • 最初のレコードが集約された時間 (秒、EPOCH から) • WPAR 名

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Dept. LRAS/Bldg. 903
11501 Burnet Road
Austin, TX 78758-3400
USA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらのCookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie などの各種テクノロジーの使用について詳しくは、『IBM オンラインでのプライバシー・ステートメントのハイライト』(http://www.ibm.com/privacy/jp/ja/)、『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』というタイトルのセクション、および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。 現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[カ行]

[タ行]

[ハ行]

[ラ行]

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21