新機能
Adaptive Server® Enterprise 12.5.3a

ドキュメント ID：DC00531-01-1253-01
改訂：2005 年 10 月

このマニュアルでは、Adaptive Server® Enterprise 12.5.3a の新機能について説明します。

1. カラムの暗号化

Adaptive Server の認証とアクセス制御のメカニズムによって、正しく識別され、正しい権限を持つユーザのみがデータにアクセスできるようになります。また、データの暗号化により、ディスク上またはアーカイブ内の機密データが情報漏洩やセキュリティ侵害からさらに保護されます。

Adaptive Server のデータ暗号化では、カラム・レベルでデータを暗号化できます。機密データのみを暗号化して、処理のオーバヘッドを最小化します。

Adaptive Server のカラムの暗号化機能は、中間層やクライアント・アプリケーションでの暗号化よりも簡単に使用できます。sql 文を使用して、暗号化キーを作成し、暗号化するカラムを指定します。Adaptive Server によってキーの生成と格納が処理されます。データの暗号化と復号化は、暗号化カラムのデータを読み書きするときに自動的かつ透過的に行われます。アプリケーションを変更する必要はなく、サードパーティのソフトウェアを購入する必要もありません。

暗号化されたデータは、暗号テキストとして格納されます。暗号化されていないデータはプレーン・テキストとして格納されます。

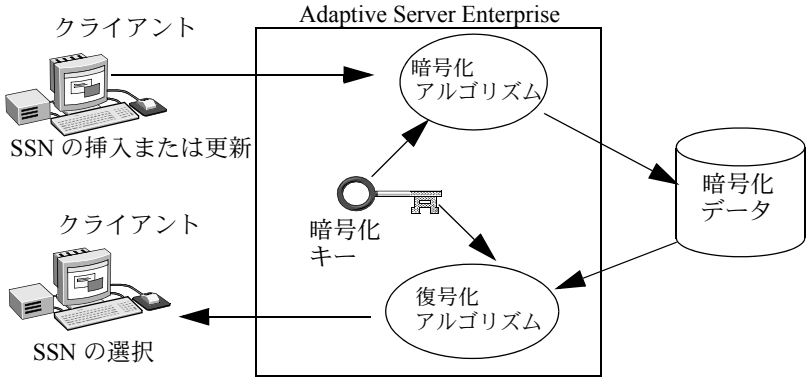
暗号化機能は以下にも含まれています。

- Sybase Central と Adaptive Server プラグイン。詳細については、『Sybase Central User's Manual』を参照してください。
- sybmigrate (マイグレーション・ツール)、**バルク・コピー**、CIS。これらの詳細については、『ASE システム管理ガイド』を参照してください。
- Replication Server。複写時の暗号化の詳細については、『Replication Server 管理ガイド』を参照してください。

1.1 概要

図 1 に、Adaptive Server での暗号化と復号化の概要を示します。この例では、社会保障番号 (SSN) の更新と暗号化が行われています。

図 1: Adaptive Server での暗号化と復号化



暗号化キーを作成するには、create encryption key を使用します。これによって次の処理が実行されます。

- Security Builder Crypto™ API を使用して、指定されたキー長の対称キーが内部的に作成される。
- システム・カタログ *sysencryptkeys* にキーが暗号化された状態で格納される。

Adaptive Server は、特定のカラムの暗号化に使用されたキーを記録しています。カラムの暗号化では対称暗号化アルゴリズムが使用されます。つまり、暗号化と復号化に同じキーが使用されます。

暗号化カラムに対してデータの insert または update を行うとき、Adaptive Server はローを書き込む直前にデータを透過的に暗号化します。暗号化カラムの select を実行するときは、Adaptive Server はデータをローから読み取った後で復号化します。整数と浮動小数点のデータは、次のように正規化形式で暗号化されます。

- 整数データの場合は最上位ビット (MSB) 形式。
- 浮動小数点データの場合は、MSB 形式に対応する IEEE (Institute of Electrical and Electronics Engineers) 浮動小数点標準。

2 つのプラットフォームで同じ文字セットが使用されている場合は、一方のプラットフォームで暗号化されたデータをもう一方のプラットフォームで復号化できます。

Adaptive Server で暗号化されたカラムを使用する場合は、次の手順に従います。

1 ライセンス・オプション ASE_ENCRYPTION をインストールします。詳細については、『ASE インストール・ガイド』を参照してください。

2 次のように Adaptive Server Enterprise の暗号化を有効にします。

```
sp_configure 'enable encrypted columns', 0|1
```

0 － 暗号化を無効にする。

1 － 暗号化を有効にする。

このオプションを設定した後でサーバを再起動します。

暗号化カラムを含むサーバでこのオプションをオフにすると、このようなカラムに対するすべてのコマンドが失敗してエラー・メッセージが生成されます。暗号化カラムを使用するには、設定パラメータとライセンス・オプションの両方が必要です。暗号化カラムを有効にできるのはシステム・セキュリティ担当者だけです。

3 sp_encryption コマンドを使用して、データベースのシステム暗号化パスワードを設定します。詳細については、「システム暗号化パスワードの設定」(4 ページ) を参照してください。

4 カラムの暗号化のためのキーを作成します。詳細については、「暗号化キーの作成」(5 ページ) を参照してください。

5 暗号化するカラムを指定します。「新しいテーブルでの暗号化の指定」(10 ページ) と 「既存テーブルでのデータの暗号化」(11 ページ) を参照してください。

6 データを確認する必要があるユーザに decrypt パーミッションを付与します。詳細については、「復号化のパーミッション」(11 ページ) を参照してください。

1.2 システム暗号化パスワードの設定

システム・セキュリティ担当者は、sp_encryption を使用してシステム暗号化パスワードを設定します。システム・パスワードは sp_encryption が実行されるデータベースに固有であり、暗号化された値はそのデータベースの sysattributes システム・テーブルに格納されます。

```
sp_encryption system_encr_passwd, password
```

パスワードの最大長は 64 バイトです。Adaptive Server で、そのデータベースのすべてのキーを暗号化するために使用されます。一度システムに暗号化パスワードを設定すれば、キーまたはデータにアクセスするためにこのパスワードを指定する必要はありません。

システム暗号化パスワードは、暗号化キーが作成されるすべてのデータベースで設定する必要があります。

システム・セキュリティ担当者は、sp_encryption に古いパスワードを指定して、システム・パスワードを変更できます。

sp_encryption system_encr_passwd, *password* [, *old_password*]

システム・パスワードが変更されると、Adaptive Server は自動的にデータベース内のすべてのキーを新しいパスワードで再暗号化します。

1.3 暗号化キーの作成と管理

Adaptive Server は暗号化キーを作成し、暗号化した状態でデータベースに格納します。名前付きキーの所有者はテーブル所有者に、現在のデータベースのカラムを指定のキーで暗号化するためのパーミッションを付与します。

1.3.1 暗号化キーの作成

キーや暗号化に関連するすべての情報は、create encryption key によってカプセル化されます。この文では、暗号化アルゴリズムとキー・サイズ、キーのデフォルト・プロパティ、暗号化プロセスでの初期化ベクトルまたは埋め込みの使用を指定できます。

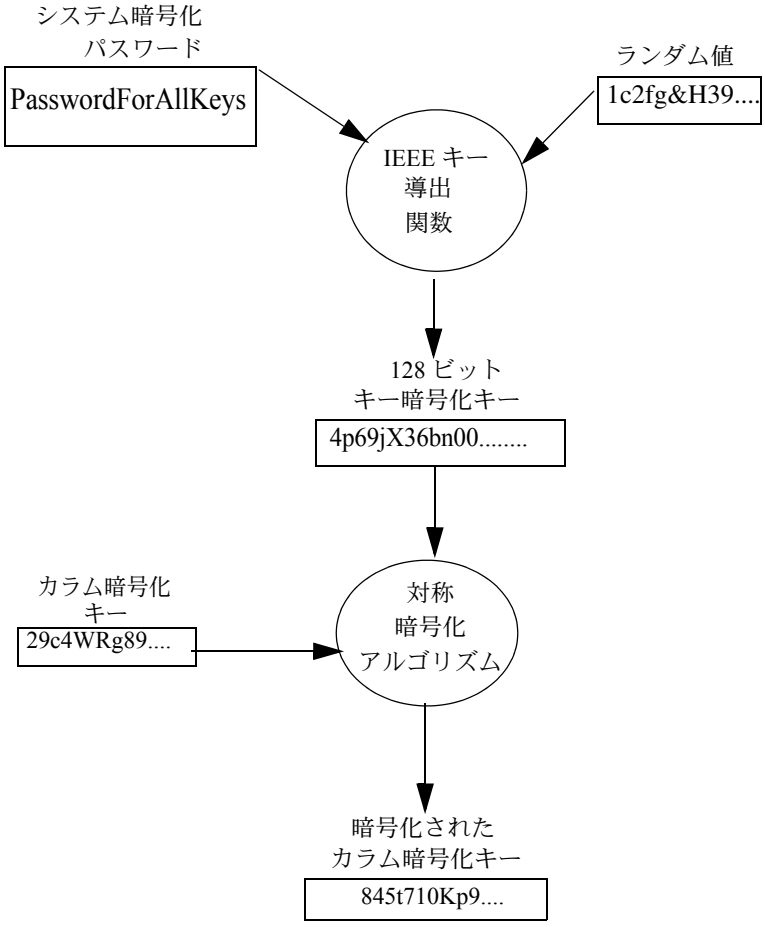
Adaptive Server のカラム暗号化では、AES (Advanced Encryption Standard) 対称キー暗号化アルゴリズムが使用されます。使用可能なキー・サイズは、128、192、256 ビットです。ランダム・キー生成と暗号法の機能は、Security Builder Crypto API によって提供されます。

暗号化カラムごとに別のキーを作成できます。また、キーを複数カラムで共有できますが、各カラムは 1 つのキーのみに対応します。1 つのカラムは初期化ベクトルを使用して暗号化し、もう 1 つのカラムは初期化ベクトルを使用せずに暗号化するには、初期化ベクトルの使用を指定したキーと指定しないキーの 2 つを作成します。

システム・セキュリティ担当者は、create encryption key で as default 句を使用して、データベースのデフォルト暗号化キーを設定します。デフォルト・キーは、create table または alter table で encrypt 修飾子がキー名なしで使用されるときに使用されます。

キー値を保護するために、Adaptive Server はシステム暗号化パスワードを使用して 128 ビットのキー暗号化キーを生成します。このキーは、新たに作成されるキーの暗号化に使用されます。カラム暗号化キーは、暗号化された状態で *sysencryptkeys* システム・テーブルに格納されます。

図 2: ユーザ・キーの暗号化



create encryption key の構文を次に示します。

```
create encryption key keyname [as default] for algorithm
[with [keylength num_bits]
[init_vector [null | random]]
[pad [null | random]]]
```

各パラメータの意味は、次のとおりです。

- *keyname* – 現在のデータベースにおいて、ユーザのテーブル、ビュー、プロシージャのネーム・スペースに対してユニークでなければなりません。

- as default – システム・セキュリティ担当者は、暗号化のためのデータベースのデフォルト・キーを作成できます。これにより、テーブルの作成者が、create table、alter table、select into で keyname を使用せずに暗号化を指定できます。Adaptive Server は同じデータベースのデフォルト・キーを使用します。デフォルト・キーは変更できます。詳細については、「alter encryption key」(34 ページ) を参照してください。
- *algorithm* – AES (Advanced Encryption Standard) のみがアルゴリズムとしてサポートされています。AES では、キー・サイズとして 128 ビット、192 ビット、256 ビット、ブロック・サイズとして 16 バイトがサポートされています。
- keylength *num_bits* – 作成するキーのサイズ (ビット単位)。AES の有効なキー長は 128、192、256 ビットです。デフォルトの keylength は 128 ビットです。
- init_vector random – 暗号化を行うときの初期化ベクトルの使用を指定します。暗号化アルゴリズムで初期化ベクトルを使用すると、まったく同じ 2 つのプレーン・テキストに対して異なる暗号テキストが生成され、暗号分析でデータのパターンが検出されるのを防ぎます。初期化ベクトルを使用すると、データのセキュリティが強化されます。

  初期化ベクトルによってパフォーマンスが影響を受けます。インデックス作成や、ジョインと検索の最適化は、暗号化キーに初期化ベクトルが指定されていないカラムでしか実行できません。詳細については、「パフォーマンスの考慮事項」(18 ページ) を参照してください。
- init_vector null – 暗号化を行うときに初期化ベクトルを使用しません。この指定により、カラムがインデックスに対応できるようになります。

  デフォルトは、初期化ベクトルの使用、つまり init_vector random です。初期化ベクトルの使用は、暗号化の暗号ブロック連鎖 (CBC) モードの使用を意味します。init_vector null の設定は、電子コードブック (ECB) モードを意味します。
- pad null – デフォルトです。データのランダム埋め込みは行われません。

  カラムがインデックスに対応する必要がある場合、埋め込みは使用できません。
- pad random – 暗号化の前に乱数バイトがデータに自動的に埋め込まれます。暗号テキストのランダム化のために初期化ベクトルではなく埋め込みを使用できます。埋め込みに適しているのは、プレーン・テキストの長さがブロック長の半分未満のカラムのみです。AES アルゴリズムのブロック長は 16 バイトです。

たとえば、名前が “safe_key” の 256 ビットのキーをデータベースのデフォルト・キーに指定するには、システム・セキュリティ担当者は次のように入力します。

```
create encryption key safe_key as default for AES with
        keylength 256
```

次の例では、カラムの暗号化にランダム埋め込みを使用する、名前が “salary_key” の 128 ビットのキーが作成されます。

```
create encryption key salary_key for AES with
        init_vector null pad random
```

次の例では、カラムの暗号化に初期化ベクトルを使用する、名前が “mykey” の 192 ビットのキーが作成されます。

```
create encryption key mykey for AES with keylength 192
        init_vector random
```

システム・セキュリティ担当者は、暗号化キーを作成するデフォルトのパーミッションを持っており、そのパーミッションを他のユーザに付与できます。

次に例を示します。

```
grant create encryption key to key_admin_role
```

1.3.2 暗号化キーの使用

暗号化するカラムを指定するときに、同じデータベースまたは別のデータベースの名前付きキーを指定して使用できます。名前付きキーを指定しない場合、カラムは自動的に同じデータベースのデフォルト・キーで暗号化されます。

別のデータベースのキーを使用して暗号化すると、セキュリティ上の大きな利点が得られます。データベース・ダンプが盗まれた場合でも、キーと暗号化データの両方がアクセスから保護されるためです。データにアクセスするには、データを含むデータベース・アーカイブと暗号化キーを含むデータベース・アーカイブにアクセスする必要があります。また、管理者がデータベース・ダンプを別のパスワードで保護すると、不正なアクセスがさらに困難になります。

別のデータベースのキーを使用して暗号化する場合、分散システムでデータとキーの整合性の問題が発生しないように対処する必要があります。データベースのダンプとロードを注意して調整してください。別のデータベースの名前付きキーを指定して使用する場合は次の方法を推奨します。

- 暗号化カラムを含むデータベースをダンプするときは、対応するキーが作成されたデータベースもダンプします。これは、最後のダンプ以降に新しいキーが追加されている場合に必要になります。
- 暗号化キーを含むデータベースをダンプするときは、そのキーで暗号化されたカラムを含むすべてのデータベースをダンプします。これにより、暗号化されたデータと対応するキーの同期が保たれます。

システム・セキュリティ担当者は、sp_encryption を使用して、特定のキーで暗号化されたすべてのカラムを識別できます。詳細については、「sp_encryption」(35 ページ) を参照してください。

1.3.3 キーのパーミッションの付与

キー所有者がキーの select パーミッションを付与しないと、他のユーザは create table 文、alter table 文、select into 文でキーを指定できません。データベース・デフォルト・キーの場合はシステム・セキュリティ担当者が所有者です。キーの select パーミッションの付与は必要な場合にのみ行います。

次の例では、db_admin_role のユーザが create table 文と alter table 文で暗号化を指定するときに、暗号化キー “safe_key” を使用できます。

```
grant select on safe_key to db_admin_role
```

insert、update、delete、select で暗号化カラムを処理するユーザには、暗号化キーの select パーミッションは必要ありません。

1.3.4 キーの変更

情報セキュリティ・ポリシーの一環として、カラムの暗号化に使用するキーを定期的に変更してください。新しいキーは create encryption key を使用して作成し、alter table...modify を使用して新しいキーでカラムを暗号化します。

次の例では、creditcard カラムがすでに暗号化されています。alter table コマンドによって、customer のすべてのローの creditcard の値が、復号化されてから cc_key_new を使用して再暗号化されます。

```
create encryption key cc_key_new for AES

alter table customer modify creditcard encrypt with cc_key_new
```

詳細については、「alter table」(36 ページ) を参照してください。

1.4 データの暗号化

次のデータ型を暗号化できます。

- int、smallint、tinyint
- float4、float8
- decimal および numeric
- char、varchar
- binary、varbinary

暗号化されたデータのディスク上での基本データ型は、varbinary です。varbinary データのサイズの詳細については、「暗号化カラムの長さ」(15 ページ) を参照してください。

NULL 値は暗号化されません。

1.4.1 新しいテーブルでの暗号化の指定

新しいテーブルのカラムを暗号化するには、次のカラム・オプションを create table 文で使用します。

[encrypt [with [*database*.[*owner*].]*keyname*]]

keyname – create encryption key を使用して作成するキーを指定します。テーブルの作成者は、*keyname* の select パーミッションが必要です。*keyname* を指定しないと、Adaptive Server は、create encryption key の as default 句を使用して作成されたデフォルト・キーを探します。create table の完全な構文については、「create table」(39 ページ) を参照してください。

次の例では 2 つのキーが作成されます。init_vector、埋め込み、キー長についてデフォルト値を使用するデータベース・デフォルト・キーと、デフォルト値を使用しない名前付きキー cc_key です。employee テーブルの ssn カラムはデフォルト・キーを使用して暗号化され、customer テーブルの creditcard カラムは cc_key を使用して暗号化されます。

```
create encryption key new_key as default for AES
create encryption key cc_key for AES with
            keylength 256
            init_vector null
            pad random

create table employee_table (ssn char(15) encrypt)

create table customer (creditcard char(20)
            encrypt with cc_key)
```

1.4.2 暗号化カラムのインデックスの作成

暗号化カラムにインデックスを作成できるのは、暗号化キーに初期化ベクトルまたはランダム埋め込みが指定されていない場合です。初期化ベクトルまたはランダム埋め込みが使用されている暗号化カラムに対して create index を実行すると、エラーが発生します。暗号化カラムのインデックスは、等しいか等しくないかを照合するときは役立ちますが、範囲の検索や順序付けには役立ちません。

次の例では、cc_key は初期化ベクトルまたは埋め込みを使用せずに暗号化を指定しています。これにより、cc_key を使用して暗号化されたすべてのカラムにインデックスを作成できます。

```
create encryption key cc_key for AES
            with init_vector null

create table customer(custid int,
            creditcard varchar(16) encrypt with cc_key)

create index cust_idx on customer(creditcard)
```

1.4.3 既存テーブルでのデータの暗号化

既存テーブルのカラムを暗号化するには、次のカラム・オプションを alter table 文で使用します。

[encrypt [with [*database*.[*owner*].]*keyname*

keyname − create encryption key を使用して作成するキーを指定します。テーブルの作成者は、*keyname* の select パーミッションが必要です。*keyname* を指定しないと、Adaptive Server は、create encryption key の as default 句を使用して作成されたデフォルト・キーを探します。alter table の完全な構文については、『ASE リファレンス・マニュアル』を参照してください。

注意 トリガが作成されている既存テーブルのカラムを暗号化すると、エラーが発生して alter table が失敗します。トリガを削除してから、テーブルを暗号化のために変更する必要があります。alter table .. encrypt の後でトリガを再作成してください。

データ型や null の可否などの属性を変更するのと同時にカラムの暗号化プロパティを変更できます。また、alter table を使用して、暗号化されたカラムを追加することもできます。

次に例を示します。

```
alter table customer modify custid encrypt with cc_key
alter table customer add address varchar(50) encrypt      with
cc_key
```

完全な構文については、「alter table」(38 ページ) を参照してください。

1.5 データの復号化

1.5.1 復号化のパーミッション

暗号化カラムからクリア・テキスト・データを選択するか、暗号化カラムで検索またはジョインを実行するには、次の 2 つのパーミッションが必要です。

- カラムの select パーミッション
- ターゲット・リストで使用されるカラムや where、having、order by、update などの句で使用されるカラムの decrypt パーミッション

テーブル所有者は grant decrypt を使用して、テーブルの 1 つ以上のカラムを復号化するための明示的なパーミッションを他のユーザ、グループ、ロールに付与します。プロシージャまたはビューの所有者が次のパーミッションを付与するときに、decrypt パーミッションが暗黙に付与される場合もあります。

- 暗号化カラムを選択するストアド・プロシージャの exec パーミッション (プロシージャの所有者が暗号化カラムを含むテーブルも所有している場合)
- 暗号化カラムを選択するビュー・カラムの decrypt パーミッション (ビューの所有者がテーブルも所有している場合)

どちらの場合も、ベース・テーブルの暗号化カラムに decrypt パーミッションを付与する必要はありません。

構文は次のとおりです。

grant decrypt on [*owner*.] *table*[(*column*[{ ,*column*}])] to *user* | *group* | *role*

テーブル・レベルで decrypt パーミッションを付与すると、テーブルのすべての暗号化カラムの decrypt パーミッションが付与されます。

次のように customer テーブルのすべての暗号化カラムの decrypt パーミッションを付与します。

```
grant decrypt on customer to accounts_role
```

次の例は、ベース・テーブル “employee” の ssn カラムに対する user2 の暗黙の decrypt パーミッションを示します。user1 は、employee テーブルと emp_salary ビュー を次のように設定します。

```
create table employee (ssn varchar(12)encrypt,
        dept_id int, start_date date, salary money)

create view emp_salary as select
            ssn, salary from employee

grant select, decrypt on emp_salary to user2
```

user2 は、次のように emp_salary ビューを選択して、復号化された社会保障番号にアクセスできます。

```
select * from emp_salary
```

1.5.2 復号化パーミッションの取り消し

次の文を使用してユーザの復号化パーミッションを取り消すことができます。

revoke decrypt on [*owner*.] *table*[(*column*[{,*column*}])] from *user* | *group* | *role*

次に例を示します。

```
revoke decrypt on customer from public
```

1.6 暗号化とキーの削除

1.6.1 暗号化の削除

テーブルを所有している場合は、alter table で decrypt オプションを使用して、カラムの暗号化を削除できます。

構文は次のとおりです。

alter table tablename modify columnname decrypt

たとえば、customer テーブルの creditcard カラムの暗号化は次のように削除します。

```
alter table customer modify creditcard decrypt
```

1.6.2 キーの削除

システム・セキュリティ担当者とキーの所有者がキーを削除できます。キーを削除できるのは、そのキーを使用する暗号化カラムがすべてのデータベースにない場合のみです。特定のキーで暗号化されたカラムについて、suspect やオフラインのデータベースを調べることはできません。コマンドを使用すると、使用できないデータベースを知らせる警告メッセージが発行されますが、コマンドは失敗しません。データベースがオンラインになったとき、削除されたキーで暗号化されたカラムを含むテーブルは使用できなくなります。キーをリストアするには、システム管理者が、削除したキーのデータベースのダンプ (キーを削除する前のもの) をロードする必要があります。

次の文を使用して暗号化キーを削除します。

drop encryption key [*database*.[*owner*].]*keyname*

次に例を示します。

```
drop encryption key cust.dbo.cc_key
```

1.7 *select into* コマンド

select into では、デフォルトで暗号化を含まないターゲット・テーブルが作成されます。ソース・テーブルに 1 つ以上の暗号化カラムがある場合でも同じです。select into には、decrypt など、ソース・テーブルのカラム・レベルのパーミッションが必要です。

次の文を使用して新しいテーブルのカラムを暗号化します。

```
select [all|distinct] < column_list>
        into table_name
        [(colname encrypt [with [[database.[owner].]keyname]
                [, colname encrypt
                [with [[ database.]owner].]keyname]])]
         from table_name | view_name
```

ソース・テーブルの対応するデータが暗号化されていない場合でも、ターゲット・テーブルの特定のカラムを暗号化できます。ソース・テーブルのカラムがターゲット・カラムに指定されたのと同じキーで暗号化されている場合、Adaptive Server は、ソース・テーブルでの復号化のステップとターゲット・テーブルでの暗号化のステップを省略します。

ターゲット・テーブルの暗号化のルールは、次の点に関して、ソース・テーブルの create table の encrypt 指定子のルールと同じです。

- 暗号化されるカラムの有効なデータ型
- keyname が省略された場合のデータベース・デフォルト・キーの使用
- ターゲット・カラムの暗号化に使用されるキーの select パーミッションの必要性

たとえば、次のように creditcard カラムを暗号化します。

```
select creditcard, custid, sum(amount) into      #bigspenders
     (creditcard encrypt with cust.dbo.new_cc_key)
     from daily_xacts group by creditcard
     having sum(amount) > $5000
```

1.8 暗号化カラムの長さ

create table や alter table の操作において、Adaptive Server は暗号化カラムの内部での最大長を計算します。スキーマ配置やページ・サイズを決定するために、データベース所有者は暗号化カラムの最大長を把握する必要があります。

AES はブロック暗号化アルゴリズムです。ブロック暗号化アルゴリズムの暗号化データの長さは、暗号化アルゴリズムのブロック・サイズの倍数です。AES のブロック・サイズは 128 ビットすなわち 16 バイトです。このため、暗号化カラムは、少なくとも 16 バイトに次の領域を加えたサイズになります。

- 初期化ベクトル。初期化ベクトルを使用する場合、各暗号化カラムに 16 バイトが追加されます。デフォルトでは、暗号化プロセスで初期化ベクトルが使用されます。init_vector null を create encryption key に指定すると、初期化ベクトルが省略されます。
- プレーン・テキスト・データの長さ。カラムのデータ型が char、varchar、binary、または varbinary の場合、データに 2 バイトのプレフィクスが付けられてから暗号化されます。プレフィクスの 2 バイトが追加されることで暗号テキストがさらに 1 ブロックを必要としないかぎり、暗号化カラムによってこれ以上の領域が使用されることはありません。
- 標識バイト。データベース・システムが末尾の 0 を切り捨てないように暗号テキストに追加される 1 バイトの領域。

表 1: 暗号テキストの長さ

ユーザが指定するカラムのデータ型	入力データ長	初期化ベクトル	カラムの内部データ型	暗号化データ長
tinyint、smallint、int	1、2、または 4	いいえ	varbinary(17)	17
tinyint、smallint、int	1、2、または 4	はい	varbinary(33)	33
tinyint、smallint、int	0 (null)	いいえ	varbinary(17)	0
float、float(4)、real	4	いいえ	varbinary(17)	17
float、float(4)、real	4	はい	varbinary(33)	33
float、float(4)、real	0 (null)	いいえ	varbinary(17)	0
float(8)、double	8	いいえ	varbinary(17)	17
float(8)、double	8	はい	varbinary(33)	33
float(8)、double	0 (null)	いいえ	varbinary(17)	0
numeric(10.2)	3	いいえ	varbinary(17)	17
numeric (10,2)	3	はい	varbinary(33)	33
numeric (38,2)	18	いいえ	varbinary(33)	33
numeric (38,2)	18	はい	varbinary(49)	49
numeric (38,2)	0 (null)	いいえ	varbinary(33)	0
char、varchar (100)	1	いいえ	varbinary(113)	17
char、varchar (100)	14	いいえ	varbinary(113)	17
char、varchar (100)	15	いいえ	varbinary(113)	33
char、varchar (100)	15	はい	varbinary(117)	49
char、varchar (100)	31	はい	varbinary(117)	65

ユーザが指定するカラムのデータ型	入力データ長	初期化ベクトル	カラムの内部データ型	暗号化データ長
char、varchar (100)	0 (null)	はい	varbinary(117)	0
binary、varbinary(100)	1	いいえ	varbinary(113)	17
binary、varbinary(100)	14	いいえ	varbinary(113)	17
binary、varbinary(100)	15	いいえ	varbinary(113)	33
binary、varbinary(100)	15	はい	varbinary(117)	49
binary、varbinary(100)	31	はい	varbinary(117)	65
binary、varbinary(100)	0 (null)	はい	varbinary(117)	0

char と binary は可変長データ型として処理され、ブランクと 0 の埋め込みが削除されてから暗号化されます。データが復号化されるときはブランクと 0 の埋め込みが適用されます。

注意 暗号化カラムではディスク上のカラム長は増加しますが、増加を確認することはできません。たとえば、sp_help にも元のサイズのみが表示されます。

1.9 暗号化カラムの監査

暗号化キーの作成や削除など、暗号化カラムに関連する DDL コマンドを監査できます。また、テーブルを作成するとき、暗号化カラムの名前と対応する暗号化キーが監査レコードに含まれます。データベース全体に対する監査オプションを使用すると、暗号化カラムとキーの監査レコードをグループ化して管理できます。

1.9.1 監査オプション

次の表は、『ASE システム管理ガイド』の抜粋です。既存のイベント・オプションで監査される新しいコマンドと新規イベント・オプションを示します。

表 2: 監査オプション、要件および例

オプション	*login_name*	*object_name*	オプション設定時に使用するデータベース	監査されるコマンドまたはアクセス
encryption_key (データベース固有)	all	監査されるデータベース	すべて	alter encryption key create encryption key drop encryption key create table drop table alter table sp_encryption

オプション	*login_name*	*object_name*	オプション設定時に使用するデータベース	監査されるコマンドまたはアクセス
	例 `sp_audit "encryption_key", "all", "pubs2", "on"`			
	(pubs2 データベースで指定したすべてのコマンドを監査する。)			

1.9.2 値の監査

表 3 は、event カラムに表示される値を sp_audit のオプション順にリストにしたものです。「extrainfo 出力の情報」の欄では、監査テーブルの extrainfo カラムに表示される情報を、表 3 に示すカテゴリに基づいて説明しています。

表 3: event カラムと extrainfo カラムの値

監査オプション	監査されるコマンド	イベント	extrainfo 出力の情報

alter	alter table	3	キーワードまたはオプション：
			`ADD/DROP/MODIFY COLUMNS`
			`REPLACE COLUMN`
			`ADD CONSTRAINT`
			`DROP CONSTRAINT`
			1 つ以上の暗号化カラムが追加される場合、キーワードの内容は次のようになります。
			`ADD/DROP/MODIFY COLUMNS column1/keyname1, [,column2/keyname2]`
			このとき、*keyname* はキーの完全修飾名です。

create	create table	10	暗号化カラムでは、キーワードにはカラム名とキー名が含まれます。 `EK column1/keyname1[,column2 keyname2]` このとき、EK は、後続の情報が暗号化キーを参照することを示すプレフィクスです。また、*keyname* はキーの完全修飾名です。
encryption_key	sp_encryption	106	キーワードには、パスワードが最初に設定される場合は `ENCR_ADMIN system_encr_passwd password` ******** が含まれます。パスワードが後で変更される場合は、`ENCR_ADMIN system_encr_passwd password` ******** ******** が含まれます。
	create encryption key	107	キーワードの内容は次のとおりです。 algorithm Name-bitlength/IV [RANDOM|NULL]/PAD [RANDOM|NULL] 例：`AES-128/IV RANDOM/PAD NULL`

監査オプション	監査されるコマンド	イベント	extrainfo 出力の情報
	alter encryption key	108	キーワードの内容は次のとおりです。 NOT DEFAULT キーがデフォルト・キーでなくなった場合 DEFAULT キーがデフォルト・キーになった場合
	drop encryption key	109	

1.9.3 新しいイベントの名前と番号

特定の監査イベントの監査証跡を問い合わせることができます。audit_event_name にパラメータとして *event id* を指定して使用します。

```
audit_event_name(event_id)
```

表 4 は、新しいイベントの番号と名前を示します。

表 4: 新しいイベント番号

イベント番号	出力されるイベント名
106	Encrypted Column Administration
107	Create Encryption Key
108	Alter Encryption Key
109	Drop Encryption Key

1.10 パフォーマンスの考慮事項

暗号化は CPU 集約操作であるため、CPU 使用率や暗号化カラムを使用するコマンドの実行時間の面で、アプリケーションにパフォーマンス・オーバヘッドをもたらす場合があります。オーバヘッドは、CPU と Adaptive Server エンジンの数、システムの負荷、暗号化データにアクセスする同時セッション数、クエリで参照される暗号化カラムの数によって異なります。暗号化キーのサイズと暗号化データ長も要因になります。一般に、キー・サイズが大きくデータ長が長いほど、暗号化操作での CPU 使用率が高くなります。

この項では、暗号化カラムの検索におけるパフォーマンスの影響や、Adaptive Server Enterprise が暗号化データの処理を最適化して暗号化と復号化の操作回数を最小限に抑える方法について説明します。

1.10.1 暗号化カラムのインデックス

暗号化カラムにインデックスを作成できるのは、そのカラムの暗号化キーで初期化ベクトルとランダム埋め込みの使用が指定されていない場合です。初期化ベクトルまたはランダム埋め込みを使用すると、同一のデータが異なるパターンの暗号テキストに暗号化されるため、インデックスによってユニークであることを示すことができません。

暗号化データのインデックスは、データが等しいか等しくないかを照合するときは役立ちますが、データの順序付け、範囲の検索、または最小値と最大値の検出には役立ちません。Adaptive Server が暗号化カラムで順序に依存する検索を実行している場合、暗号化データに対してインデックス・ルックアップを実行できません。代わりに各ローの暗号化カラムを復号化してから検索する必要があります。このプロセスのためにデータの処理が遅くなります。

1.10.2 暗号化カラムでのジョイン

Adaptive Server は、次の条件が適用される場合に、暗号テキストの比較を実行して 2 つの暗号化カラムのジョインを最適化します。

- ジョインするカラムが同じデータ型である。char と varchar および binary と varbinary はそれぞれ同じデータ型とみなされる。
- int 型と float 型でカラムの長さが同じ。numeric 型と decimal 型でカラムの精度と位取りが同じ。
- ジョインするカラムが同じキーで暗号化されている。
- ジョインするカラムが式の一部ではない。たとえば、t.encr_col1 = s.encr_col1 +1 というジョインでは暗号テキストのジョインを実行できない。
- init_vector と pad が NULL に設定されて暗号キーが作成された。
- ジョイン演算子が ‘=’ または ‘<>’。
- データのソート順がデフォルトである。

たとえば、次の例では、暗号テキストに対してジョインを行うスキーマが設定されます。

```
create encryption key new_cc_key for AES
        with init_vector NULL
create table customer
        (custid int,
        creditcard char(16) encrypt with new cc_key)
create table daily_xacts
       (cust_id int, creditcard char(16) encrypt with
        new_cc_key, amount money........)
```

次のように、ジョインするカラムにインデックスも設定できます。

```
create index cust_cc on customer(creditcard)

create index daily_cc on daily_xacts(creditcard)
```

Adaptive Server は、次の select 文を実行して、特定のクレジット・カードで顧客の毎日の請求額を集計します。このとき、customer テーブルまたは daily_xacts テーブルの creditcard カラムは復号化されません。

```
select sum(d.amount) from daily_xacts d, customer c
     where d.creditcard = c.creditcard and
     c.custid = 17936
```

1.10.3 定数値の探索引数と暗号化カラム

暗号化カラムと定数値が等しいか等しくないかを比較する場合、Adaptive Server はカラムのスキャンを最適化するために、テーブルの各ローの暗号化カラムを復号化するのではなく、定数値を 1 回暗号化します。「暗号化カラムでのジョイン」(19 ページ) と同じ制約が適用されます。

Adaptive Server は、暗号化カラムの範囲検索を実行するときにはインデックスを利用できません。各ローを復号化してからデータ比較を実行する必要があります。クエリに他の述部が含まれる場合、Adaptive Server は最も効率のよいジョイン順序を選択します。このため、多くの場合は、最小のデータセットで暗号化カラムの検索が行われます。

クエリに複数の範囲検索があり、有効なインデックスがない場合は、暗号化カラムに対する範囲検索が最後になるようにクエリを作成します。たとえば、次のクエリは、ロード・アイランドの納税者で所得が $100, 000 を超える人の社会保障番号を検索します。zipcode カラムの範囲検索は、暗号化された調整総所得のカラムに対する範囲検索よりも前に指定します。

```
select ss_num from taxpayers
       where zipcode like '02%' and
       agi_enc > 100000
```

1.10.4 暗号テキストとしての暗号化データの移動

Adaptive Server が暗号化データをコピーするときは、データを復号化してから再暗号化するのではなく、できるかぎり暗号テキストをコピーすることで処理を最適化します。これは、select into、バルク・コピー、複写に適用されます。

1.11 システム・テーブル

1.11.1 *syscolumns*

syscolumns システム・テーブルの次のカラムで暗号化のプロパティが説明されます。

フィールド	タイプ	値	説明
encrtype	int	NULL	暗号化された状態でのデータ型。
encrlen	int	NULL	暗号化データの長さ。
encrkeyid	int	NULL	キーのオブジェクト id。
encrkeydb	varchar (30)	NULL	暗号化キーが格納されているデータベースの名前。暗号化カラムと同じデータベースにキーが格納される場合は NULL。
encrdate	datetime	NULL	作成日付。sysobjects.crdate からコピーされる。

1.11.2 *sysobjects*

sysobjects には、タイプが EK (暗号化キー) の各キーのエントリが含まれます。

データベース間でのキー参照に関しては、syscolumns.encrdate と sysobjects.crdate が一致します。

sysencryptkeys の encrkeyid は、*sysobjects* の id カラムと一致します。

1.11.3 *sysencryptkeys*

デフォルト・キーも含め、データベースに作成される各キーには、データベース固有のシステム・カタログ sysencryptkeys にエントリがあります。

表 5: sysencryptkeys

フィールド	タイプ	説明
id	int	暗号化キー ID。
ekalgorithm	int	暗号化アルゴリズム。
type	smallint	キー・タイプを示す。値は EK_SYMMETRIC と EK_DEFAULT。
status	int	内部ステータス情報。
eklen	smallint	ユーザが指定したキーの長さ。
value	varbinary(1282)	キーの暗号化された値。キーの対称暗号を含む。キーを暗号化するために、Adaptive Server は AES でシステム暗号化パスワードの 128 ビット・キーを使用する。
uid	int null	未使用。
eksalt	varbinary(20)	暗号化キーの復号化の検証に使用されるランダムな salt を含む。

フィールド	タイプ	説明
ekpairid	int null	未使用。
pwdate	datetime null	未使用。
expdate	int null	未使用。
ekpwdwarn	int null	未使用。

1.12 *ddlgen* ユーティリティの変更

ddlgen は、暗号化キー用の DDL 文の生成をサポートしています。キーを指定するには次の構文を使用します。

<dbName>.<owner>.<keyName>

暗号化キー用の新しいタイプ EK は、暗号化キーを作成してそのパーミッションを付与する DDL を生成するためのものです。ddlgen を使用すると、暗号化カラム情報と grant decrypt 文がテーブル用の DDL と一緒に生成されます。

次の例は、ポート 1955 を使用するマシン “HARBOR” の データベース “accounts” のすべての暗号化キーに対する DLL を生成します。

```
ddlgen -Uroy -Proy123 -SHARBOR:1955 -TEK
-Naccounts.dbo.%
```

または、次のように、-D オプションを使用してデータベース名を指定することもできます。

```
    ddlgen -Uroy -Proy134 -SHARBOR:1955 -TEK -Ndbo.%
     -Daccounts
-----------------------------------------------------------
----------------
-- DDL for EncryptedKey 'ssn_key'
-----------------------------------------------------------
----------------
      print 'ssn_key'

create encryption key accounts.dbo.ssn_key
      for AES
      with keylength 128
      init vector random
go


-----------------------------------------------------------
----------------
-- DDL for EncryptedKey 'ek1'
-----------------------------------------------------------
----------------
print 'ek1'

create encryption key accounts.dbo.ek1 as default
     for AES
```

```
        with keylength 192
        init vector NULL
go

use accounts
go

grant select on accounts.dbo.ek1 to acctmgr_role
go
```

ddlgen には、create encryption key を生成する拡張オプションもあります。この文では、*sysencryptkeys* として暗号化されたキーの値が指定されます。このオプション -XOD は、データ移動のためにサーバ間で暗号化キーを同期する必要がある場合に使用できます。たとえば、サーバ “PACIFIC” の cc_key をサーバ “ATLANTIC” でも使用できるようにするには、次のように “PACIFIC” で -XOD を使用して ddlgen を実行します。

```
ddlgen -Sfred -Pget2work -SPACIFIC:8532 -TEK -Nsales.dbo.cc_key -XOD
```

ddlgen の出力は次のとおりです。

```
------------------------------------------------------------------------------
-- DDL for EncryptedKey 'cc_key'
------------------------------------------------------------------------------
print 'cc_key'

create encryption key sales.dbo.cc_key
      for AES
with keylength 128
passwd 0x0000E1D8235FEBEB118901
init_vector NULL
keyvalue 0xF772B99CE547D2932A12E0A83F2114848BD93F38016C068D720DDEBAC4DF8AA001
keystatus 32
go
```

次に、create encryption key で生成されたキーを変更して、“ATLANTIC” のターゲット・データベースを指定し、ターゲット・サーバでコマンドを実行します。これで “PACIFIC” から “ATLANTIC” に移動したデータを復号化するために、サーバ “ATLANTIC” で cc_key が使用できるようになります。

ddlgen 構文オプションの詳細については、『ASE ユーティリティ・ガイド』を参照してください。複写されたデータベースでの ddlgen の使用例については、『Replication Server 管理ガイド』を参照してください。

1.13 暗号化データの複写

サイトがスキーマの変更を複写する場合、次の DDL 文が複写されます。

- alter encryption key
- 暗号化の拡張指定を含む create table と alter table
- create encryption key
- grant create encryption key と revoke create encryption key
- キーの grant select と revoke select
- カラムの grant decrypt と revoke decrypt
- sp_encryption system_encr_passwd
- drop encryption key

キーは暗号化された状態で複写されます。

システムによって DDL が複写されない場合は、複写サイトの暗号化キーを手動で同期する必要があります。ddlgen では、キーの値を複写するための特殊な形式の create encryption key をサポートしています。

insert と update では、暗号化された状態で暗号化カラムが複写されます。これにより、Replication Server がディスクのステーブル・キューで複写データを処理するときに、データが保護されます。

複写時の暗号化の詳細については、『Replication Server 管理ガイド』を参照してください。

1.14 バルク・コピー (*bcp*)

bcp によって、暗号化データがデータベースの内外でプレーン・テキストまたは暗号テキストの形式で転送されます。デフォルトでは、bcp はプレーン・テキスト・データをコピーします。bcp は、プレーン・テキスト・ファイルを次のように処理します。

- bcp in を実行すると、データは Adaptive Server によって自動的に暗号化されてから挿入されます。低速 bcp が使用されます。ユーザには、すべてのカラムの insert パーミッションと select パーミッションが必要です。
- bcp out を実行すると、データは Adaptive Server によって自動的に復号化されます。すべてのカラムの select パーミッションが必要です。また、暗号化カラムの decrypt パーミッションも必要です。

次の例では、“customer” テーブルがプレーン・テキスト・データとしてコンピュータのネイティブ・フォーマットでコピーされます。

```
bcp uksales.dbo.customer out uk_customers -n -Uroy
-Proy123
```

bcp の -C オプションを使用すると、データは暗号テキストとしてコピーされます。暗号テキストをコピーするときは、異なるオペレーティング・システム間でデータをコピーできます。文字データを暗号テキストとしてコピーする場合は、両方のプラットフォームが同じ文字セットをサポートしている必要があります。

bcp の -C オプションでは、管理者がデータの decrypt パーミッションを持たない場合でも bcp を実行できます。-C オプションを使用すると、bcp では次のようにデータが処理されます。

- bcp in の実行時にデータは暗号テキスト形式であるとみなされ、Adaptive Server は暗号化を行いません。bcp in で -C オプションを使用できるのは、Adaptive Server にコピーされるファイルが、bcp out の -C オプションを使用して作成された場合のみです。暗号テキストは、カラム属性がまったく同じカラムからコピーされ、コピー先のカラムと同じキーで暗号化されているはずです。高速 bcp が使用されます。ユーザには、insert パーミッションと select パーミッションが必要です。
- bcp out では、データは復号化されずに Adaptive Server からコピーされます。暗号テキスト・データは 16 進数形式です。ユーザには、すべてのカラムの select パーミッションが必要です。暗号テキストをコピーするときは、暗号化カラムの decrypt パーミッションは必要ありません。
- 暗号化された char データまたは varchar データは、暗号化時に Adaptive Server によって使用された文字セットを維持します。データが暗号テキスト形式で別のサーバにコピーされる場合、そのターゲット・サーバで使用される文字セットは、ソースからコピーされる暗号化データの文字セットと一致する必要があります。暗号化時にソース・サーバでデータに関連付けられていた文字セットは、暗号化データには格納されないため、ターゲット・サーバでは認識されず変換も行われません。

  システム管理者は、ソース・サーバとターゲット・サーバの文字セットが一致することを確認する必要があります。また、-C オプションなしでも bcp を実行できます。文字セットの問題は検出されません。

  文字セット変換のための -J オプションは、-C オプションと一緒には使用できません。

次の例では、“customer” テーブルがコピーされます。cc_card カラムは暗号テキストとしてコピーされます。その他のカラムは文字形式でコピーされます。ユーザ “roy” には customer cc_card の decrypt パーミッションは必要ありません。

```
bcp uksales.dbo.customer out uk_customers -C -c -Uroy -Proy123
```

警告！ ビューに対する bcp out で -C フラグを使用できるのは、ビューの条件で暗号化カラムが検索されない場合のみです。

1.15 コンポーネント統合サービス (CIS)

デフォルトでは、暗号化と復号化はリモート Adaptive Server で処理されます。CIS は、リモート Adaptive Server で暗号化カラムを 1 回チェックします。リモート Adaptive Server で暗号化がサポートされる場合、CIS によって、暗号化カラムに関連するメタデータでローカルの *syscolumns* カタログが更新されます。

- create proxy_table を実行すると、リモート・テーブルの任意の暗号化カラムの情報によって *syscolumns* が自動的に更新されます。
- create existing table を実行すると、リモート・テーブルの任意の暗号化カラム・メタデータによって *syscolumns* が自動的に更新されます。encrypt キーワードは create existing table の *columnlist* では使用できません。CIS は、リモート・テーブルで暗号化カラムを検出すると、そのカラムが暗号化されていることを自動的にマークします。
- 暗号化カラムがある場所では create table は使用できません。
- alter table は、プロキシ・テーブルの暗号化カラムでは使用できません。
- select into existing を実行すると、ソースからプレーン・テキストが取得され、ターゲットのテーブルに挿入されます。その後、ローカル Adaptive Server がプレーン・テキストを暗号化してから、暗号化カラムに挿入します。

次のカラムは、リモート・サーバの *syscolumns* カタログによって更新されます。

- encrtype − ディスク上のデータ型
- encrlen − 暗号化データの長さ。
- status2 − カラムが暗号化されていることを示すステータス・ビット

1.16 データベースの *load* と *dump*

dump と load は、暗号化カラムの暗号テキストに対して実行されます。この動作により、ディスク上で暗号化カラムのデータが暗号化された状態を保つことが保証されます。dump と load はデータベース全体に対応します。デフォルト・キーと、同じデータベースに作成されているその他のキーは、それらが対応するデータと一緒にダンプおよびロードされます。

ロードするデータベースに、他のデータベースで使用される暗号化キーが含まれる場合、新しい構文 with override を使用しないと load が正常に終了しません。

```
load database key_db from "/tmp/key_db.dat" with override
```

キーとそのキーで暗号化するカラムが別のデータベースにある場合は、次の方法を推奨します。

- 暗号化カラムを含むデータベースをダンプするときは、対応するキーが作成されたデータベースもダンプします。これは、最後のダンプ以降に新しいキーが追加されている場合に必要です。

- 暗号化キーを含むデータベースをダンプするときは、そのキーで暗号化されたカラムを含むすべてのデータベースをダンプします。これにより、暗号化されたデータと対応するキーの同期が保たれます。
- 暗号化キーを含むデータベースと、暗号化カラムを含むデータベースをロードした後で、両方のデータベースを同時にオンラインにします。

キーを含むデータベースを別の名前のデータベースにロードした場合、他のデータベースにある暗号化カラムにアクセスするときにエラーが発生します。キーがあるデータベースの名前を変更する場合は、次の手順に従います。

- 暗号化カラムを含むデータベースをダンプする前に、alter table を使用してデータを復号化します。
- キーと暗号化カラムを含むデータベースをダンプします。
- データベースをロードした後で、alter table を使用して、名前を変更したデータベースのキーでデータを再暗号化します。

暗号化キーと暗号化カラムの一貫性の問題は、データベース間の参照整合性の問題と似ています。詳細については、『ASE システム管理ガイド』の「データベース間の制約とデータベースのロード」を参照してください。

以前のバージョンの Adaptive Server には暗号化データを含むダンプをロードしないでください。まず Adaptive Server バージョン 12.5.3a にデータベースをロードして、暗号化を削除します。ダンプを実行してから、そのデータベースを以前のバージョンの Adaptive Server にロードしてください。詳細については、「ダウングレード手順」(30 ページ) を参照してください。

キーの詳細については、「暗号化キーの作成と管理」(5 ページ) を参照してください。

1.17 *unmount database*

カラムが別のデータベースのキーで暗号化されているときは、関連するすべてのデータベースをまとめてマウント解除します。暗号化カラムを含むデータベースとキーを含むデータベースの依存関係は、参照整合性を使用するデータベース間の依存関係と似ています。

override オプションを使用して、別のデータベースのキーで暗号化されているカラムを含むデータベースに unmount を実行します。

次のコマンドの場合、key_db に作成されている暗号化キーが、col_db のカラムを暗号化するために使用されています。これらのコマンドによって、指定したデータベースが正常にマウント解除されます。

```
unmount database key_db, col_db
unmount database key_db with override
unmount database col_db with override
```

次のコマンドは、override がないためエラー・メッセージが生成されて失敗します。

```
unmount database key_db

unmount database col_db
```

1.18 *quiesce database*

データベースに暗号化キーが含まれるときに quiesce database を使用できます。

別のデータベースのキーで暗号化されたカラムを含むデータベースに対して quiesce を実行するときは、with override を使用する必要があります。

quiesce database ***key_db, col_db*** を実行できます。このとき、*key_db* は暗号化キーを含むデータベース、*col_db* は *key_db* のキーで暗号化されたカラムを含むテーブルがあるデータベースです。

たとえば、col_db のカラムの暗号化に使用された暗号化キーが key_db に含まれる場合、次のコマンドは正常に終了します。

```
quiesce database key_tag hold key_db for external
         dump to "/tmp/keydb.dat"

quiesce database encr_tag hold col_db for external dump
      to "/tmp/col.dat" with override

quiesce database col_tag hold key_db, col_db for
    external dump to "/tmp/col.dat"
```

1.19 drop database

キーを誤って削除しないように、他のデータベースのカラムの暗号化に現在使用されているキーを含むデータベースに対しては drop database が失敗するように Adaptive Server が制御しています。そのような暗号化キーを含むデータベースを削除するには、まず暗号化を削除するか、暗号化カラムを含むデータベースを削除する必要があります。

次の例では、key_db は暗号化キーを含むデータベース、col_db は暗号化カラムを含むデータベースです。

```
drop database key_db, col_db
```

この例では、Adaptive Server によってエラーが生成され、key_db の削除が失敗します。col_db の削除は成功します。両方のデータベースを削除するには、次のように col_db を最初に削除します。

```
drop database col_db, key_db
```

1.20 *sybmigrate*

sybmigrate は、サーバ間でデータをマイグレートするために使用するマイグレーション・ツールです。

sybmigrate のデフォルトでは、暗号化カラムは暗号テキスト形式でマイグレートされます。このため、ソースでのデータの復号化とターゲットでの暗号化によるオーバヘッドが回避されます。場合によっては、sybmigrate でマイグレート方法として reencrypt が選択され、ソースでのデータの復号化とターゲットでの暗号化が行われることもあります。

暗号化カラムが含まれるデータベースでは、sybmigrate によって次の処理が実行されます。

1 システム暗号化パスワードをマイグレートします。システム暗号化パスワードをマイグレートしないように指定すると、sybmigrate は、暗号テキストを直接マイグレートする代わりに、reencrypt 方法を使用して暗号化カラムをマイグレートします。

2 暗号化キーをマイグレートします。マイグレートするキーをユーザが選択できます。また、現在のデータベースのカラムの暗号化に使用されたキーは、sybmigrate によって自動的に選択されます。ユーザがシステム暗号化パスワードのマイグレーションを選択した場合は、sybmigrate が暗号化キーを実際の値でマイグレートします。*sysencryptkeys* システム・テーブルのキー値は、システム暗号化パスワードを使用して暗号化されています。このような値がマイグレートされます。ユーザがシステム暗号化パスワードをマイグレートしなかった場合、sybmigrate はキーを名前でマイグレートします。これは、ターゲットで適切に復号化できないキーのマイグレートを回避するためです。キーを名前でマイグレートすると、ソースとは異なるキー値のキーがターゲットで作成されます。

3 データをマイグレートします。デフォルトでは、データは暗号テキスト形式で転送されます。暗号テキスト・データは、異なるオペレーティング・システムにマイグレートできます。文字データの場合は、ターゲット・サーバーでソース・サーバと同じ文字セットが使用されている必要があります。

sybmigrate は、データベースを作業の単位として実行されます。ソース・サーバのデータベースのデータが、別のデータベースのキーで暗号化されている場合は、キーのデータベースを最初にマイグレートします。

sybmigrate で、マイグレートしたデータの再暗号化が選択されるのは次の場合です。

- 現在のデータベースのいずれかのキーがマイグレーションの対象として選択されていない、またはいずれかのキーがすでにターゲット・サーバに存在している場合。ターゲットのキーがソースのキーと同一である保証はありません、そのためマイグレートするデータを再暗号化する必要があります。

- システム・パスワードがマイグレーションの対象として選択されなかった場合。ターゲットのシステム・パスワードがソースと異なるときは、キーを値でマイグレートできません。また、データも暗号テキストとしてマイグレートできません。
- ユーザが次のフラグを使用する場合。

```
sybmigrate -T 'ALWAYS_REENCRYPT'
```

データの再暗号化によってパフォーマンスが低下する場合があります。再暗号化モードでマイグレーションを実行すると、この影響に関するメッセージがマイグレーション・ログ・ファイルに書き込まれます。

暗号化カラムをマイグレートするには、sa_role と sso_role の両方を有効にしてください。

1.21 ダウングレード手順

サーバで enable encrypted columns を一度も設定していない場合は、旧バージョンの Adaptive Server で 12.5.3a のデータベースを使用する際に何も行う必要はありません。暗号化カラムを設定していないことを確認するには、システム・テーブル *sysencryptkeys* がすべてのデータベースに存在しないことを調べてください。

ダウングレード手順の前には、すべてのデータベースをバックアップする必要があります。

暗号化カラムを設定していたサーバをダウングレードする前に、暗号化カラムを含むテーブルを削除または変更して、暗号化を削除する必要があります。次に sp_encryption remove_catalog を実行します。これにより、各データベースに暗号化カラムがないことが確認され、システム・テーブル *sysencryptkeys* が削除されます。12.5.3a に対応して *syscolumns* に追加された新しいカラムは、古いバイナリでは無視されるため、削除する必要はありません。

12.5.3a サーバを以前のバージョン 12.5.x にダウングレードする場合は、次の手順に従います。

1 暗号化カラムが現在有効になっていない場合は、システム・セキュリティ担当者が次のコマンドを実行します。

```
sp_configure 'enable encrypted columns',1
```

2 drop または alter を使用して、すべてのデータベースの暗号化カラムを含むすべてのテーブルを復号化します。システム・セキュリティ担当者は、暗号化キーが作成されていた各データベースで次のコマンドを実行して、それらのデータベースで作成されたすべての暗号化キーをリストします。

```
sp_encryption help
```

リストされたキーごとに、システム・セキュリティ担当者は次のコマンドを実行して、そのキーで暗号化されたカラムのリストを表示します。

sp_encryption help, <*keyname*>, 'display_cols'

暗号化カラムごとに、次のいずれかを実行する必要があります。

- 暗号化カラムを復号化する alter table
- 暗号化カラムを削除する alter table
- 暗号化カラムを含むテーブルに対する drop
- 暗号化キーの削除

3 システム・テーブルが削除されている間は Adaptive Server に他のユーザが一切アクセスできないようにするために、サーバをシングルユーザ・モードで再起動します。詳細については、『ASE ユーティリティ・ガイド』を参照してください。

4 sso_role や sa_role を持つユーザは、次のシステム・ストアド・プロシージャを実行する必要があります。これによって、各データベースから *sysencryptkeys* カタログが削除されます。

```
sp_encryption remove_catalog
```

処理できないデータベースがある場合、コマンドはエラーを出力して終了します。*sysencryptkeys* のキーで暗号化されたカラムが存在する場合、このコマンドでは *sysencryptkeys* は削除されません。エラーまたは警告が出力されて、次のデータベースの処理が続行されます。

sp_encryption で *sysencryptkeys* の削除が成功すると、各データベースの *sysattributes* から次のローも削除されます。

- *sysencryptkeys* を追加したアップグレード項目のレコード
- データベースのシステム暗号化パスワード

5 *sybsystemprocs* データベースからシステム・ストアド・プロシージャ sp_encryption を削除します。

6 サーバを停止します。これで、12.5.3a 以前のバージョンの 12.5.x Adaptive Server バイナリを使用できます。

カラムの暗号化を再び有効にするには、ダウングレードした 12.5.3a サーバをロールフォワードして 12.5.3a に戻すときに、enable encrypted columns を設定します。12.5.3a サーバの再起動時に、*sysencryptkeys* システム・テーブルが各データベースにインストールされます。

1.21.1 ダウングレードにおける複写の問題

暗号化データを含むデータベースで複写が有効になっているサーバをダウングレードするときは、ダウングレード手順を開始する前に次のいずれかを行う必要があります。

1 プライマリ・データベースのトランザクション・ログのすべての複写データが、スタンバイすなわち複写データベースに正常に転送されたことを確認します。確認のプロセスはアプリケーションによって異なります。

2 プライマリ・データベースのトランザクション・ログをトランケートし、Replication Server でそのデータベースの RS ロケータを 0 に設定します。次のコマンドを使用します。

プライマリ・データベースで次のコマンドを実行します。

```
sp_stop_rep_agent primary_dbname
        dbcc settrunc ('ltm', 'ignore')
        dump tran  primary_dbname with truncate_only
        dbcc settruc ('ltm', 'valid')
```

Replication Server をシャットダウンします。Replication Server の RSSD で次のコマンドを実行します。

```
rs_zeroltm primary_servername, primary_dbname
```

1.22 新しいコマンド

1.22.1 *create encryption key*

キーや暗号化に関連するすべての情報は、create encryption key によってカプセル化されます。この文では、暗号化アルゴリズムとキー・サイズ、キーのデフォルト・プロパティ、暗号化プロセスでの初期化ベクトルまたは埋め込みの使用を指定できます。

Adaptive Server では、キーの生成と暗号化に Security Builder Crypto が使用されます。

システム・セキュリティ担当者は、暗号化キーを作成するデフォルトのパーミッションを持っており、そのパーミッションを他のユーザに付与できます。

構文

create encryption key [[*database.[owner].]keyname* [as default] for *algorithm*
[with [keylength *num_bits*]
[init_vector [NULL | random]]
[pad [NULL | random]]]

- *keyname* – 現在のデータベースにおいて、ユーザのテーブル、ビュー、プロシージャのネーム・スペースに対してユニークでなければなりません。

- as default – システム・セキュリティ担当者は、暗号化のためのデータベースのデフォルト・キーを作成できます。これにより、テーブルの作成者が、create table、alter table、select into で keyname を使用せずに暗号化を指定できます。Adaptive Server は同じデータベースのデフォルト・キーを使用します。デフォルト・キーは変更できます。詳細については、「alter encryption key」(34 ページ) を参照してください。
- *algorithm* – AES (Advanced Encryption Standard) のみがアルゴリズムとしてサポートされています。AES では、キー・サイズとして 128 ビット、192 ビット、256 ビット、ブロック・サイズとして 16 バイトがサポートされています。
- keylength *num_bits* – 作成するキーのサイズ (ビット単位)。AES の有効なキー長は 128、192、256 ビットです。デフォルトの keylength は 128 ビットです。
- init_vector random – 暗号化を行うときの初期化ベクトルの使用を指定します。暗号化アルゴリズムで初期化ベクトルを使用すると、まったく同じ 2 つのプレーン・テキストに対して異なる暗号テキストが生成され、暗号分析でデータのパターンが検出されるのを防ぎます。初期化ベクトルを使用すると、データのセキュリティが強化されます。

  初期化ベクトルによってパフォーマンスが影響を受けます。インデックス作成や、ジョインと検索の最適化は、暗号化キーに初期化ベクトルが指定されていないカラムでしか実行できません。詳細については、「パフォーマンスの考慮事項」(18 ページ) を参照してください。
- init_vector null – 暗号化を行うときに初期化ベクトルを使用しません。この指定により、カラムがインデックスに対応できるようになります。

  デフォルトは、初期化ベクトルの使用、つまり init_vector random です。初期化ベクトルの使用は、暗号化の暗号ブロック連鎖 (CBC) モードの使用を意味します。init_vector null の設定は、電子コードブック (ECB) モードを意味します。
- pad null – デフォルトです。データのランダム埋め込みは行われません。

  カラムがインデックスに対応する必要がある場合、埋め込みは使用できません。
- pad random – 暗号化の前に乱数バイトがデータに自動的に埋め込まれます。暗号テキストのランダム化のために初期化ベクトルではなく埋め込みを使用できます。埋め込みに適しているのは、プレーン・テキストの長さがブロック長の半分未満のカラムのみです。AES アルゴリズムのブロック長は 16 バイトです。

たとえば、名前が “safe_key” の 256 ビットのキーをデータベースのデフォルト・キーに指定するには、システム・セキュリティ担当者は次のように入力します。

```
create encryption key safe_key as default for AES with
       keylength 256
```

次の例では、カラムの暗号化にランダム埋め込みを使用する、名前が “salary_key” の 128 ビットのキーが作成されます。

```
create encryption key salary_key for AES with
        init_vector null pad random
```

次の例では、カラムの暗号化に初期化ベクトルを使用する、名前が “mykey” の 192 ビットのキーが作成されます。

```
create encryption key mykey for AES with keylength 192
        init_vector random
```

1.22.2 *alter encryption key*

デフォルト暗号化キーを変更するには、次のように入力します。

alter encryption key *key1* as default

デフォルト・キーがすでに存在する場合は、そのキーのデフォルト・プロパティが失われます。*key1* がデフォルト・キーになります。

key1 がデフォルト・キーである場合は、次のように *key1* のデフォルト指定を削除できます。

alter encryption key *key1* as not default

key1 がデフォルト・キーでない場合は、コマンドによってエラーが返されます。

alter encryption key as default または alter encryption key as not default を実行できるのはシステム・セキュリティ担当者のみです。他のユーザには付与できません。

1.22.3 *drop encryption key*

キーの所有者とシステム・セキュリティ担当者が暗号化キーを削除できます。いずれかのデータベースのいずれかのカラムが、削除するキーで暗号化されている場合、コマンドは失敗します。

構文 drop encryption key [*database*.[*owner*].]*keyname*

1.22.4 *grant create encryption key*

暗号化キーを作成するパーミッションは、システム・セキュリティ担当者から付与されます。

構文 grant create encryption key to *user* | *role*| *group*

1.22.5 *revoke create encryption key*

システム・セキュリティ担当者は、他のユーザ、グループ、ロールの暗号化キーを作成するパーミッションを取り消すことができます。

構文

revoke create encryption key from *user* | *role* | *group*

1.22.6 *grant decrypt*

テーブル所有者またはシステム・セキュリティ担当者は、テーブルまたはテーブル内のカラム・リストに対して decrypt パーミッションを付与します。

構文

grant decrypt on [*owner.*]*tablename*[(*columnname* [{,*columname*}])]
to *user* | *group* | *role*

注意 テーブルまたはカラムに対する grant all では、decrypt パーミッションは付与されません。

1.22.7 *revoke decrypt*

テーブル所有者またはシステム・セキュリティ担当者は、テーブルまたはテーブル内のカラム・リストの decrypt パーミッションを取り消します。

構文

revoke decrypt on [owner.] *tablename*[(*columnname* [{,*columname*}])] from *user* | *group* | *role*

1.23 *sp_encryption*

システム・セキュリティ担当者は、sp_encryption を使用してシステム暗号化パスワードを設定します。システム・パスワードは sp_encryption が実行されるデータベースに固有であり、暗号化された値はそのデータベースの sysattributes システム・テーブルに格納されます。

sp_encryption system_encr_passwd, '*password*'

sp_encryption を使用して指定されるパスワードの最大長は 64 バイトです。Adaptive Server でそのデータベースのすべてのキーを暗号化するために使用されます。キーまたはデータにアクセスするためにこのパスワードを指定する必要はありません。

システム暗号化パスワードは、暗号化キーが作成されるすべてのデータベースで設定する必要があります。

システム・セキュリティ担当者は、sp_encryption に古いパスワードを指定して、システム・パスワードを変更できます。

sp_encryption system_encr_passwd, *'password* '[, *'old_password*']

システム・パスワードが変更されると、Adaptive Server は自動的にデータベース内のすべてのキーを新しいパスワードで再暗号化します。

1.23.1 *sp_encryption help*

sp_encryption help を実行すると、キーの名前、所有者、サイズ、暗号化アルゴリズムが表示されます。キーがデータベースのデフォルト・キーとして指定されているかどうか、このキーによる暗号化でランダム埋め込みか初期化ベクトルが使用されているかどうかも表示されます。

```
sp_encryption help [, keyname [, display_cols]]
```

sp_encryption help を sso_role を持つユーザが実行すると、データベース内のすべてのキーのキー・プロパティが表示されます。sso_role のないユーザが実行すると、そのユーザがそのデータベース内で select パーミッションを持っているキーのみのキー・プロパティが表示されます。

sp_encryption help, *keyname* を実行すると、keyname に指定したキーのプロパティが表示されます。このコマンドを sso_role のないユーザが実行する場合、そのユーザにはキーの select パーミッションが必要です。

sp_encryption help, keyname, display_cols を実行できるのは、sso_role を持つユーザのみです。keyname のキーで暗号化されたカラムがリストされます。

1.24 コマンド構文の変更

この項では、カラムの暗号化機能が加えられたことによって変更または追加されたコマンドの構文について説明します。

1.24.1 alter table

alter table を使用して、既存のデータを暗号化または復号化するか、暗号化カラムをテーブルに追加します。

構文

カラムを暗号化する場合

```
alter table tablename add column_name
        encrypt [with [database.[owner].]keyname
```

既存のカラムを復号化する場合

```
[decrypt [with [database.[owner].]keyname]]
```

keyname − create encryption key を使用して作成するキーを指定します。テーブルの作成者は、*keyname* の select パーミッションが必要です。*keyname* を指定しないと、Adaptive Server は、create encryption key または alter encryption key を使用してデフォルトとして作成されたデフォルト・キーを探します。

例

暗号化キーを作成し、既存の “employee” テーブルの ssn カラムを暗号化します。

```
alter table employee modify ssn
         encrypt with ssn_key
grant decrypt on employee(ssn) to hr_manager_role,
         hr_director_role
```

alter table を使用して暗号化キーを変更します。すでに暗号化されているカラムに対して encrypt 修飾子を使用すると、Adaptive Server はそのカラムを復号化してから新しいキーで再暗号化します。テーブルに多数のローが含まれる場合、このオペレーションにはかなり時間がかかることがあります。

1.24.2 create table

encrypt 修飾子を使用して、テーブルのカラムの暗号化を設定します。

構文

```
create table tablename (colname datatype [default_clause]
        [encrypt [with [database.[owner].]keyname]])
```

keyname − create encryption key を使用して作成するキーを指定します。テーブルの作成者は、*keyname* の select パーミッションが必要です。*keyname* を指定しないと、Adaptive Server は、create encryption key または alter encryption key の as default 句を使用して作成されたデフォルト・キーを探します。

例

暗号化を含む employee テーブルを作成します。

```
create table employee_table (ssn char(15) null encrypt)
```

1.24.3 *enable encrypted columns* 設定パラメータ

暗号化機能を使用するには、設定パラメータ enable encrypted columns を 1 に設定する必要があります。暗号化カラムを含むサーバでこの設定オプションをオフにすると、それらのカラムに対するすべてのコマンドが失敗してエラーが生成されます。暗号化を有効にするには、設定パラメータとライセンス・オプションの両方が必要です。

```
sp_configure 'enable encrypted columns', 1
```

1.24.4 *load database*

ロードするデータベースに、他のデータベースで使用される暗号化キーが含まれる場合、with override を使用しないと load が正常に終了しません。

```
load database key_db from "/tmp/key_db.dat" with override
```

1.24.5 *select into*

select into には、decrypt など、ソース・テーブルのカラム・レベルのパーミッションが必要です。

構文

次の構文を使用して新しいテーブルのカラムを暗号化します。

```
select [all|distinct] < column_list>
        into target_table
        [(colname encrypt [with [database.[owner].]keyname]
        [,colname encrypt
        [with [database.[owner].]keyname]])]
    from tabname | viewname
```

例

テーブルの creditcard カラムを暗号化します。

```
select creditcard, custid, sum(amount) into
       #bigspenders
       (creditcard encrypt with
          cust.database.new_cc_key)
       from daily_xacts group by creditcard
       having sum(amount) > $5000
```

1.24.6 *dbcc*

dbcc checkcatalog には、次のような追加の一貫性チェックが含まれます。

1 *sysobjects* の暗号化キーのローごとに、キーを定義するローの存在が *sysencryptkeys* でチェックされます。

2 *syscolumns* で暗号化されているとしてマークされているカラムごとに、キーの存在が *sysobjects* と *sysencryptkeys* でチェックされます。

1.25 コマンドの完全な構文

この項では、このマニュアルに記載されているコマンドの完全な構文を示します。

1.25.1 *alter encryption key*

alter encryption key *key1* as default | not default

1.25.2 alter table

alter table [[*database*.[*owner*].*table_name*
{ add *column_name datatype*
[default {*constant_expression* | user | null}]
{identity | null | not null}
[off row | in row]
[[constraint *constraint_name*]
{ { unique | primary key }
[clustered | nonclustered]
[asc | desc]
[with { fillfactor = *pct*,
max_rows_per_page = *num_rows*,
reservepagegap = *num_pages* }]
[on *segment_name*]
| references [[*database*.]*owner*.]*ref_table*
[(*ref_column*)]
[match full]
| check (*search_condition*)] ... }
[encrypt [with [*database* .] *owner*] .]*keyname*
[, *next_column*]...
| add {[constraint *constraint_name*]
{ unique | primary key}
[clustered | nonclustered]
(*column_name* [asc | desc]
[, *column_name* [asc | desc]...])
[with { fillfactor = *pct*,

```
                    max_rows_per_page = num_rows,
                    reservepagegap = num_pages}]
        [on segment_name]
| foreign key (column_name [{, column_name}...])
        references [[database.]owner.]ref_table
        [(ref_column [{, ref_column}...])]
        [match full]
| check (search_condition)}
| drop {column_name [, column_name]...
        | constraint constraint_name }
| modify column_name datatype [null | not null]
            [encrypt | [with [database .]owner].] keyname
| modify column_name datatype [null | not null
            [encrypt | [with [database .]owner].] keyname]
                |decrypt
            [, next_column]...
| replace column_name
        default { constant_expression | user | null}
        | partition number_of_partitions
| unpartition| { enable | disable } trigger
| lock {allpages | datarows | datapages } }
| with exp_row_size=num_bytes
| [ alter_partition_clause ]
[ partition_clause ]
```

1.25.3 *create table*

```
create table [database .[owner ].]table_name (column_name datatype)
    [default {constant_expression  | user | null}]
    {[{identity | null | not null}]
        [off row | [ in row [ (size_in_bytes) ] ]
    [[constraint constraint_name ]
        {{unique | primary key}
        [clustered | nonclustered] [asc | desc]
        [with { fillfactor = pct,
                    max_rows_per_page = num_rows, }
                    reservepagegap = num_pages }]
        [on segment_name]
        | references [[database .]owner .]ref_table
            [(ref_column )]
            [match full]
            | check (search_condition)}]}
            [match full]...
    [encrypt [with [[ database.] owner] . ] keyname]
    | [constraint constraint_name]
        {{unique | primary key}
            [clustered | nonclustered]
            (column_name [asc | desc]
                [{, column_name [asc | desc]}...])
            [with { fillfactor = pct
                max_rows_per_page = num_rows ,
                reservepagegap = num_pages } ]
            [on segment_name]
        |foreign key (column_name [{,column_name}...])
            references [[database.]owner.]ref_table
                [(ref_column [{, ref_column}...])]
                [match full]
        | check (search_condition) ... }
    [{, {next_column | next_constraint}}...])
    [lock {datarows | datapages | allpages }]
    [with { max_rows_per_page = num_rows,
            exp_row_size = num_bytes,
```

```
                reservepagegap = num_pages,
                identity_gap = value } ]
    [ table_lob_clause ]
    [ on segment_name ]
    [ [ external table ] at pathname ]
    [ partition_clause ]
```

1.25.4 *select*

```
into_clause ::=
    into [[database.]owner.]table_name
    [(colname encrypt [with [database.] owner].] keyname
                [, colname encrypt [with [database.]owner] . ]
                    keyname])]
    [ lock {datarows | datapages | allpages } ]
    [ with into_option [, into_option] ...]

    into_option ::=
        | max_rows_per_page = num_rows
        | exp_row_size = num_bytes
        | reservepagegap = num_pages
        | identity_gap = gap
        [existing table table_name]
        [[external type] at "path_name"
        [column delimiter delimiter]]
```

2. IPv6 (Internet Protocol version 6)

Adaptive Server を IPv6-aware (認識) にするには、トレース・フラグ 7841 を使用して Adaptive Server を起動します。これによって、Adaptive Server が IPv6 の可用性を判別できるようになり、IPv6-aware (認識) になります。

IPv6 は 32 ビット版と 64 ビット版の Sun Solaris プラットフォームで使用できます。

- 32 ビット版と 64 ビット版の Sun Solaris プラットフォーム
- Windows
- 32 ビット版と 64 ビット版の HP プラットフォーム

使用しているプラットフォームでの IPv6 を有効化しているネットワークの設定と管理の方法については、オペレーティング・システムのマニュアルを参照してください。

3. リアルタイム・メッセージング

Adaptive Server バージョン 12.5.3a の Real Time Messaging Services では、TIBCO JMS と IBM WebSphere MQSeries の両方がサポートされています。

4. AIX における 64 ビット版 Adaptive Server の PAM サポート

AIX 64 ビット版の Adaptive Server バージョン 12.5.3a は、プラグ可能認証モジュールに基づくユーザ認証 (PAMUA) をサポートしています。12.5.3a 64 ビット版 Adaptive Server は AIX 5.1 用にリリースされていますが、IBM では AIX 5.2 のみで 64 ビット・アプリケーションの PAM をサポートしています。この機能を使用するために AIX 5.2 へのアップグレードをおすすめします。OS サポート担当者に問い合わせて、ご使用の IBM ホストに対応する PAM の最新パッチを入手してください。

AIX 5.1 に付属の 64 ビット版 PAM ライブラリには、`/usr/lib/security/64` から 64 ビット・ライブラリを自動的にロードする機能はありません。AIX 5.1 上の 64 ビット版 ASE 12.5.3a で PAMUA を有効にするには、*/etc/pam.conf* ファイルに PAM モジュールのフル・パス名を指定する必要があります。次の例は、OS モジュール pam_aix を AIX 5.1 マシンに指定する方法を示します。

Adaptive Server 認証に必要なフル・パス名：`/usr/lib/security/64/pam_aix`

5. 機能一覧

この一覧表では、プラットフォームごとに Adaptive Server Enterprise バージョン 12.5.3a で提供されるさまざまな機能を示します。

図 3: 一覧

Operating System	Sol32	Sol64	HP32	HP64	AIX64	Linux x86	Windows x86
Options							
Encrypted Columns	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a
High Availability	*	*	*	*	*	*	*
Distributed Transaction	*	*	*	*	*	*	*
XML Management	*	*	*	*	*	*	*
Java Option	*	*	*	*	*	*	*
Native XML	*	*	*	*	*	*	*
Java Based XML	*	*	*	*	*	*	*
Web Services	*	*	*	*	*	*	*
Security & Dir Services	*	*	*	*	*	*	*
LDAP Server Directory	*	*	*	*	new for 12.5.3a	*	*
LDAP User Authentication	*	*	*	*	new for 12.5.3a	*	*
Secure Socket Layer	*	*	*	*	*	*	*
Cybersafe Kerberos	*	*					*
MIT Kerberos	*	*		new for 12.5.3a	new for 12.5.3a	*	
Platform Native Kerberos	*	*					
Fine Grained Access Control	*	*	*	*	*	*	*
Pluggable Authentication Modu	*	*			new for 12.5.3a	*	
Content Management	*	*	*	*	*	*	*
Enhanced Full Text Search	*	*	*	*	*	*	*
Real Time Messaging	*	*		*	*	*	*
JMS support	*	*		*	*	*	*
Websphere MQ support	new for 12.5.3a	new for 12.5.3a		new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	
Disaster Recovery	*		*			*	*
Features Included with ASE							
IPv6	*	*	new for 12.5.3a	new for 12.5.3a			new for 12.5.3a
Cross Platform Dump and Loa	*	*	*	*	*	*	*
Job Scheduler	*	*	*	*	*	*	*
ASE Replicator	*	*	*	*	*	*	*

- * は機能がサポートされていることを示します。空欄は、そのプラットフォームでは機能が使用できないことを示します。
- 高可用性のサポートはアクティブ／アクティブのサポートを意味します。

  高可用性のアクティブ／パッシブのサポートは、Solaris/SPARC プラットフォームに限られます。
- オプションが使用できるかどうかはバージョンによって異なります。各バージョンでの違いの詳細は、Adaptive Server Enterprise データ・シートを参照してください。