Adaptive Server Anywhere 8 の暗号化テクノロジ

本書では、データベースの安全管理に役立つ Adaptive Server Anywhere 8 の暗号化テクノロジについて説明します。

安全なデータの管理

本書では、データベースの安全管理に役立つ Adaptive Server Anywhere 8 の暗号化テクノロジについて説明します。データベースには機密の情報や個人的な情報などが含まれている場合があるので、データベースやそこに含まれるデータのセキュリティを考慮した設計になっていることが重要です。
特に、データベースの暗号化、通信の暗号化、パフォーマンスの問題について説明します。特定の暗号化機能の概要についても説明し、詳細情報の参照先も示しています。
データのセキュリティについては、データベース管理者に責任があります。この書で記述されているタスクを実行するには、特に明記されていないかぎり、DBA 権限が必要です。

暗号化テクノロジ概要

Adaptive Server Anywhere は以下の暗号化タイプをサポートしています。

- **データベースの暗号化** データベース暗号化機能を使用する際の、暗号化のレベルを選択します。データベースを安全に管理するために、簡単な暗号化または高度な暗号化のいずれかを選択できます。簡単な暗号化は、難読化と同じです。高度な暗号化にすると、キーなしではデータベースにまったくアクセスできなくなります。
- **通信の暗号化** ネットワーク上の通信のセキュリティを強化するために、簡単な暗号化または高度な暗号化を使用して、TCP/IP ポート経由のクライアント／サーバ通信を暗号化できます。高度な暗号化がサポートされるのは、Solaris、Linux、NetWare、サポートされているすべての Windows オペレーティング・システム (Windows CE を除く) の TCP/IP ポートのみです。

通信の暗号化は別途ライセンスが入手可能なコンポーネントで、インストールする場合はご注文ください。詳細については、SQL Anywhere Studio セキュリティ・オプション を参照してください。

データベースの暗号化

データベース管理者として、データベースの暗号化を使用して第三者によるデータの解読を困難にできます。データベースを安全に管理するために、簡単な暗号化または高度な暗号化のいずれかを選択できます。

簡単な暗号化

簡単な暗号化は、難読化と同じです。これにより第三者は、ディスク・ユーティリティを使用してファイルを表示し、データベースのデータを解読することが困難になります。簡単な暗号化は、難読化と同じです。データは判読できませんが、暗号に関する知識を持ったユーザはデータを解読できます。簡単な暗号化を行うには、KEY 句を指定しないで ENCRYPTED 句を指定します。簡単な暗号化のテクノロジは、バージョン 5.x 以降の すべてのバージョンの SQL Anywhere でサポートされています。
簡単に暗号化されたデータベースを作成するには、次の手順に従います (SQL の場合)。

1. Interactive SQL から既存のデータベースに接続します。

2. ENCRYPTION オプションを含む CREATE DATABASE 文を実行します。たとえば、次の文は、C:¥ ディレクトリにデータベース・ファイル **myencrypteddb.db** を作成します。
   ```
   CREATE DATABASE 'c:¥¥myencrypteddb'
   ENCRYPTED ON
   ```

簡単に暗号化されたデータベースを作成するには、次の手順に従います (コマンド・プロンプトの場合)。

1. コマンド・プロンプトで、**dbinit** ユーティリティを使用してデータベースを作成します。次のオプションを指定してください。
   -e　簡単な暗号化の使用を指定します。

   次のコマンド (すべて 1 行に入力) では、簡単に暗号化されたデータベースが作成されます。
   ```
   dbinit -e myencrypteddb.db
   ```

2. コマンド・プロンプトからデータベースを起動します。
   ```
   dbeng8 myencrypteddb.db
   ```

高度な暗号化

データベースを高度に暗号化すると、キーがなければまったくアクセスできなくなります。他の方法を使っても、データベースを開いたり、データベースやトランザクション・ログ・ファイルを表示したりできません。また、データベースやトランザクション・ログ・ファイルに含まれる情報に手を加えて、ディスク・ユーティリティを使用してファイルを表示してもデータを解読できないようにします。

高度な暗号化の場合は、128 ビット・アルゴリズムとセキュリティ・キーが使用されます。高度に暗号化されたデータベースを作成するには、ENCRYPTED 句とともに KEY 句を指定します。ほとんどのパスワードと同様に、簡単には推測できない KEY 値を選択するのが最善の方法です。KEY には 16 文字以上の値を選択し、大文字と小文字を混在させ、数字、文字、特殊文字を使用することをおすすめします。

ENCRYPTED 句、KEY 句とともに ALGORITHM 句を使用すると、暗号化アルゴリズムを指定できます。Adaptive Server Anywhere 8 で高度な暗号化を実装するアルゴリズムは 2 つあります。AES は、米国商務省標準技術局 (NIST: National Institute of Standards and Technology) によってブロック暗号のための新しい次世代標準暗号化方式 (AES: Advanced Encryption Standard) として選択されたブロック暗号化アルゴリズムです。MDSR は、Casio が開発した新しいアルゴリズムです。AES または MDSR を選択できます。AES 高度暗号化は Adaptive Server Anywhere に含まれていますが、MDSR は別途入手可能なコンポーネントです。ENCRYPTED 句を使用してもアルゴリズムを指定しない場合、デフォルトは AES です。

警告　KEY は保護してください。キーのコピーは、安全な場所に保管してください。KEY を紛失すると、データベースにまったくアクセスできなくなり、そこからのリカバリも不可能になります。

データベースを高度に暗号化するときは、CREATE DATABASE 文に ENCRYPTION オプションと KEY オプションを指定します。同様に、データベース管理者がデータベースを初期化する際、**dbinit** ユーティリティに高度な暗号化を有効にする各種オプションを指定することもできます。また、Sybase Central の [データベース作成] ウィザードを使用して、高度に暗号化されたデータベースを作成することもできます。**dbinit** ユーティリティに **-ea** オプションを指定すると高度な暗号化が有効になるため、アルゴリズムが AES または MDSR のいずれかに設定されます。**dbinit** ユーティリティに **-ek** オプションまたは **-ep** オプションを指定すると、高度な暗号化が有効になるため、キーをプロンプト・ボックスとコマンドのどちらに入力するかを指定できます。

高度に暗号化されたデータベースを作成するには、次の手順に従います (SQL の場合)。

1. Interactive SQL から既存のデータベースに接続します。

2. ENCRYPTION オプションと KEY オプションを含む CREATE DATABASE 文を実行します。たとえば、次の文は、C:¥ ディレクトリにデータベース・ファイル **myencrypteddb.db** を作成します。
   CREATE DATABASE 'c:¥¥myencrypteddb'
   TRANSACTION LOG ON
   ENCRYPTED ON
   KEY '0kZ2o52AK#'
   ALGORITHM 'AES'

高度に暗号化されたデータベースを作成するには、次の手順に従います (コマンド・プロンプトの場合)。

1. コマンド・プロンプトで、**dbinit** ユーティリティを使用してデータベースを作成します。次のオプションを指定してください。
   暗号化アルゴリズムを指定する (-ea)　このオプションを使用すると、データベースを暗号化するための高度な暗号化アルゴリズムを選択できます。AES または MDSR を選択できます。アルゴリズム名は、大文字と小文字が区別されます。**-ea** オプションを指定する場合は、**-ep** または **-ek** も同時に指定してください。

   暗号化キーを指定する (-ek)　このオプションを使用すると、コマンドに暗号化キーを直接指定することで、高度に暗号化されたデータベースを作成できます。アルゴリズムの指定 (**-ea** オプション) なしで **-ek** オプションを指定した場合は、AES が使用されます。

   暗号化キーを入力するよう要求する (-ep)　このオプションを使用すると、ダイアログ・ボックスに暗号化キーを入力することで、高度に暗号化されたデータベースを作成するように指定できます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。アルゴリズムの指定 (**-ea** オプション) なしで **-ep** オプションを指定した場合は、AES が使用されます。

   次のコマンド (すべて 1 行に入力) では、高度に暗号化されたデータベースが作成され、暗号化キーをコマンドの一部として入力するように指定します。
   dbinit -ea AES -ek "0kZ2o56AK#" myencrypteddb.db

2. コマンド・プロンプトからデータベースを起動します。
   dbeng8 myencrypteddb.db -ek "0kZ2o56AK#"

暗号化キーは、正確に入力されたことを確認するために 2 回入力してください。キーが一致しない場合は、初期化は失敗します。
上記の手順 2 では、コマンド・ラインで暗号化キーを指定するために、-ek サーバ・オプションを使用しています。データベース起動時に暗号化キーを入力するためのダイアログ・ボックスを表示させるには、-ep を次のように使用します。

```
dbeng8 -ep -x tcpip myencrypteddb.db
```

このサーバ・オプションでは、クリア・テキストで暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。

高度な暗号化の制御

Adaptive Server Anywhere では、データベース管理者が管理する高度な暗号化のテクノロジは 4 つあります。それは、高度な暗号化のステータス、暗号化キー、暗号化キーの保護、暗号化アルゴリズムです。

高度な暗号化のステータス

既存のデータベースでは高度な暗号化のオンとオフを簡単に切り替えることはできませんが、高度な暗号化の実装は、3 つのオプションから選択できます。

- 高度な暗号化を指定して新規にデータベースを作成する。
- 下記の手順で **dbunload** と **dbinit** ユーティリティを使用してデータベースを再構築する。
    1. dbunload で -ar オプション (再構築して置き換え) と一緒に -ea、-ek、または -ep オプションを使用して既存のデータベースをアンロードする。
    2. -ea、-ek、または -ep オプションを使用して既存のデータベースをアンロードする。次に dbinit で -ea、-ek、または -ep オプションを使用して新しいデータベースを作成し、そのデータベースにデータをリロードする。

    下記で示すように CREATE ENCRYPTED FILE 文を使用することによって、既存のデータベースを再構築して同時に暗号化ステータスを変更できます。データベースの再構築では、既存のデータベースに含まれるデータとスキーマをアンロードし、新しいデータベースを作成して (ここで高度な暗号化のステータスを含めたさまざまな設定を変更できます)、データを新しいデータベースに再ロードします。高度に暗号化されたデータベースをアンロードするにはキーが必要です。

データベースを暗号化するには、暗号化するデータベースとは別のデータベースに DBA ユーザとして接続しなければなりません。また、暗号化するデータベースは停止している必要があり

ます。

Adaptive Server Anywhere 8.0.2 以降では CREATE ENCRYPTED FILE 文を使って、データベース、トランザクション・ログ、または DB 領域を高度に暗号化できます。
この文は、暗号化されていないデータベース、トランザクション・ログ・ファイル、または DB 領域を取得し、暗号化ファイルを新しく作成します。元のファイルは暗号化しないでください。結果として出力されるファイルは、指定されたアルゴリズムとキーを使用して暗号化されている以外は、元のファイルの正確なコピーです。この文を使用してデータベースを暗号化する場合は、対応するトランザクション・ログ・ファイル (と DB 領域) も同じアルゴリズムとキーを使用して暗号化しなければ、データベースを使用できません。暗号化されたファイルと暗号化されていないファイルを混在させることはできません。また、暗号化されたファイルに異なる暗号化アルゴリズムやキーを混在させることもできません。
リカバリを必要とするデータベースを暗号化する場合は、対応するトランザクション・ログ・ファイルも暗号化します。また、新しいデータベースのリカバリも必要になります。
このプロセスでトランザクション・ログ・ファイルの名前が変わることはありません。したがって、データベースとトランザクション・ログ・ファイルの名前を変更した場合は、復号化後のデータベースに対して **dblog** -t を実行する必要があります。
-an オプションに **-ek** または **-ep** を使用してデータベースのアンロードと再ロードを行うと、既存のデータベースを暗号化したり、既存の暗号化キーを変更したりできます。または、CREATE ENCRYPTED FILE 文と CREATE DECRYPTED FILE 文を合わせて使用しても暗号化キーを変更できます。
詳細については、復号化 を参照してください。
アルゴリズムは AES (明示的に指定されていない場合はデフォルト) または MDSR を指定できます。MDSR は、Win 32 プラットフォームのみでサポートされます。
データベースを復号化し、暗号化されていない新しいデータベースを作成するには、次の手順に従います (SQL の場合)。

1. Interactive SQL から既存のデータベースに接続します。

2. CREATE DECRYPTED FILE 文を実行して、**contacts** データベースを復号化し、暗号化されていない新しいデータベース **contacts2** を作成します。

   ```
   CREATE DECRYPTED FILE 'contacts2.db'
   FROM 'contacts.db'
   KEY 'Sd8f6654*Mnn'
   ```

データベースとログ・ファイルを暗号化し、両ファイルの名前を変更するには、次の手順に従います (SQL の場合)。

1. Interactive SQL から既存のデータベースに接続します。

2. CREATE ENCRYPTED FILE 文を実行して、**contacts** データベースと **contacts** ログ・ファイルを暗号化して、両ファイルの名前を変更します。
   ```
   CREATE ENCRYPTED FILE 'contacts2.db'
   FROM 'contacts.db'
   KEY 'abcd';
   CREATE ENCRYPTED FILE 'contacts2.log'
   FROM 'contacts.log'
   KEY 'abcd';
   ```

3. ログの名前が変更されてもデータベース・ファイルは引き続き古いログ・ファイルを指しているため、以下を実行する必要があります。
   ```
   dblog -ek abcd -t contacts2.log contacts2.db
   ```

4. 元のログ・ファイル名を変更したくない場合は、データベースとログ・ファイルの作成時に新しいパスを指定しなければなりません。この場合は、ファイル名が変更されないので、**dblog** を実行する必要はありません。

データベースの暗号化キーを変更するには、次の手順に従います (SQL の場合)。

1. Interactive SQL から既存のデータベースに接続します。

2. CREATE ENCRYPTED FILE 文を実行して、**contacts** データベースの暗号化キーを変更します。
   ```
   CREATE DECRYPTED FILE 'temp.db'
   FROM 'contacts.db'
   KEY 'oldkey';
   CREATE ENCRYPTED FILE 'contacts.db'
   FROM 'temp.db'
   KEY 'newkey';
   ```

3. contacts.db と temp.db ファイルを削除します。

暗号化アルゴリズム

データベースを高度に暗号化するとき、AES アルゴリズム (明示的に指定されていない場合はデフォルト) または MDSR を選択できます。Adaptive Server Anywhere 9.0 では、MDSR 暗

号化はサポートされないことに注意してください。
AES は、国際評価期間が最近終了し、新しい American Encryption Standard (AES) ブロック暗号化アルゴリズムとして選ばれました。これは、パフォーマンスやサイズの面で Adaptive Server Anywhere データベースの暗号化に役立つ多くのプロパティを備えています。

暗号化キー
ほとんどのパスワードと同様、最善の方法は、簡単には推測できないキー値を選択することです。キーの長さは任意ですが、短いと推測されやすいため、一般的には長い方が適しています。また、数字、文字、特殊文字を組み合わせると、第三者はキーを推測しにくくなります。データベースを起動するたびに、キーを指定してください。キーを忘れた場合はデータベースにまったくアクセスできなくなります。

暗号化キー保護
暗号化キーの入力に、コマンド・プロンプト (デフォルト) またはプロンプト・ボックスのいずれかを選択できます。プロンプト・ボックスでのキー入力を選択すると、キーが表示されないため、さらにセキュリティが強化されます。クライアントでは、データベースを起動するたびにキーを指定してください。データベース管理者がデータベースを起動する場合は、クライアントでキーを使用する必要はありません。

復号化

CREATE DECRYPTED FILE 文は、暗号化されたデータベース、トランザクション・ログ・ファイル、または DB 領域を復号化し、暗号化されていないファイルを新しく作成します。オリジナル・ファイルは、暗号化キーを使用して高度に暗号化してください。復号化後のファイルは、暗号化されたファイルの正確なコピーですが、暗号化されていないため、暗号化キーは必要ありません。
この文を使用してデータベースを復号化する場合は、対応するトランザクション・ログ・ファイル (と DB 領域) も復号化しなければ、データベースを使用できません。
リカバリを必要とするデータベースを復号化する場合は、対応するトランザクション・ログ・ファイルも復号化します。また、新しいデータベースのリカバリも必要になります。
このプロセスでトランザクション・ログ・ファイルの名前が変わることはありません。したがって、データベースとトランザクション・ログ・ファイルの名前を変更した場合は、復号化後のデータベースに対して **dblog** -t を実行する必要があります。
高度な暗号化されたデータベースを復号化し、暗号化されていない新しいデータベースを作成するには、次の手順に従います (SQL の場合)。

1. Interactive SQL から既存のデータベースに接続します。
2. 次に例は、**source** データベースを復号化し、暗号化されていない新しいデータベース **source2** を作成します。

   ```
   CREATE DECRYPTED FILE 'source2.db'
   FROM 'source.db'
   KEY 'Sd8f6654*Mnn'
   ```

クライアント／サーバ通信の暗号化

セキュリティを強化するために、クライアント／サーバのネットワーク通信がネットワークを介するときに暗号化されるようにすることができます。Adaptive Server Anywhere は、Solaris、Linux、NetWare、サポートされるすべての Windows OS (Windows CE を除く) の TCP/IP ポートで証明書による暗号化をサポートするようになりました。高度な暗号化によって、クライアントとサーバ間で交換されるネットワーク・パケットの機密性と整合性が保護されます。この暗号化は、トランスポート・レイヤ・セキュリティ (TLS: Transport Layer Security) とも呼ばれています。

Certicom 暗号化の 2 つのタイプ ECC_TLS と RSA_TLS (RSA_TLS は SQL Anywhere Studio 8.0.2 以降) がサポートされています。

通信の暗号化に関するコンポーネントには別途ライセンスが必要です。インストールする場合はご注文ください。

データベース・サーバを起動するとき、またはクライアント接続パラメータで、クライアント／サーバの暗号化を設定できます。すべてのクライアントとの間で伝送される Adaptive Server Anywhere のすべてのネイティブ・パケット (EmbeddedSQL、ODBC、OLEDB) を暗号化できます。TDS パケット (jConnect 接続と Open Client 接続) は暗号化されません。SQL Anywhere Studio 8.0 のデフォルトでは、Sybase Central や InteractiveSQL からの接続は暗号化されないということです。これらの接続を暗号化したい場合は、iAnywhere JDBC ドライバを使用します。SQL Anywhere Studio 9.0 では、この設定は変更しており、Sybase Central や Interactive SQL からの接続はデフォルトで iAnywhere JDBC ドライバを使うようになっています。

サーバのコマンドで **-ec ECC_TLS** または **RSA_TLS** を指定して高度な暗号化を使用するときは、そのサーバへのすべての接続が TLS ハンドシェイクを実行するようにしてください。ハンドシェイクは偽造できません。Certicom 暗号化により、サーバに被害を及ぼす可能性のある無効なパケットは破棄されます。

サーバからクライアント／サーバ通信を暗号化するには、次の手順に従います。

1. **-ec** オプションを使用して、データベース・サーバを起動します。次に例を示します。

```
dbsrv8 -ec simple,ECC_TLS -x tcpip "c:¥Program Files¥Sybase¥SQL Anywhere 8¥asademo.db"
```

認証に使用する証明書をサーバ・コマンド・ラインで指定できます。次のコマンドはデータベース・サーバを起動し、**rsaserver.crt** 証明書とパスワード **test** の RSA 暗号化を使用した接続のみを受け入れるように指示します。

```
dbsrv8 -ec rsa_tls(certificate=rsaserver.crt; certificate_password=test) -x tcpip asademo.db
```

接続パラメータ(ENC)

この接続パラメータでクライアント･アプリケーションとサーバ間で送信されるパケットを暗号化します。Encryption 値が設定されていない場合、暗号化はサーバ側の設定によって制御されます。デフォルトでは、暗号化は行われません。ネットワーク・パケットのセキュリティが確実でない場合､このパラメータを使用できます｡暗号化はパフォーマンスにほとんど影響しません。**Encryption (ENC)** 接続パラメータには、次の引数を指定できます。

none 暗号化されていない通信パケットを受け入れます｡この値は､Adaptive Server Anywhere バージョン 5.x から 7.x の NO 設定に相当します。

simple すべてのプラットフォームと Adaptive Server Anywhere バージョン 5.x か 8.x でサポートされる簡単な暗号化で暗号化された通信パケットを受け入れます｡この値は､Adaptive Server Anywhere バージョン 5.x から 7.x の YES 設定に相当します。

ECC_TLS 楕円曲線証明書 (Certicom 暗号化テクノロジ) で暗号化された通信パケットを受け入れます。このタイプの暗号化を使用するには、サーバとクライアントの両方が Solaris、Linux、NetWare、この暗号化をサポートするすべての Windows オペレーティング・システム (Windows CE 以外) で実行されている必要があります。また､TCP/IP ポートで接続されていることも必要です。Solaris や Linux 以外の UNIX プラットフォームでは、クライアントまたはサーバの ECC_TLS パラメータは認識されません。

RSA_TLS RSA 暗号化アルゴリズム (Certicom テクノロジ) で暗号化された通信パケットを受け入れます｡このタイプの暗号化を使用するには､サーバとクライアントの両方が Solaris、Linux、NetWare、この暗号化をサポートするすべての Windows オペレーティング・システム (Windows CE 以外) で実行されている必要があります。また､TCP/IP ポートで接続されていることも必要です。Solaris や Linux 以外の UNIX プラットフォームでは、クライアントまたはサーバの **RSA_TLS** パラメータは認識されません。

クライアント接続で ECC_TLS または RSA_TLS が指定されている場合、ソフトウェアが、クライアントの持つ認証局の証明書でサーバの証明書を検証します｡また､次の引数を使用して、クライアントについて指定した値とサーバの証明書の値が一致することを検証します。

- **trusted_certificates** クライアントがサーバを認証するときに使用する証明書ファイルを指定します。

- **certificate_company** 組織フィールドの値を指定します。サーバの値とクライアントの値を一致させます。

- **certificate_unit** 組織単位フィールドの値を指定します。サーバの値とクライアントの値を一致させます。

- **certificate_name** 証明書の共通名を指定します。サーバの値とクライアントの値を一致させます。

警告:
SQL Anywhere Studio に含まれているサンプル証明書は、テスト以外には使用しないでください。サンプル証明書とそれに対応するパスワードが Sybase ソフトウェアとともに広く配布されているため、サンプル証明書を配備しても安全性を確保できません。システムを保護するには、独自の証明書を作成してください。
証明書の詳細については、『Mobile Link 同期ユーザーズ・ガイド』 の第 13 章 トランスポート・レイヤ・セキュリティ「自己署名証明書」を参照してください。
特定のクライアントからクライアント／サーバ通信を暗号化するには、次の手順に従います。

1. 接続文字列に **Encryption (ENC)** 接続パラメータを追加します。
   dbsrv8 -ec simple,ECC_TLS -x tcpip "c:¥Program Files¥Sybase¥SQL Anywhere 8¥asademo.db"

connection_property システム関数を使用して、現在の接続の暗号化設定を取り出せます。使用されている暗号化のタイプに応じて、この関数は none、simple、または Certicom の 3 つの値のうちいずれか 1 つを返します。

connection_property システム関数の使用方法の詳細については、『Adaptive Server Anywhere SQL リファレンス・マニュアル』の第 3 章 SQL 関数「CONNECTION_PROPERTY 関数」を参照してください。
RSA 暗号化を使用して、TCP/IP リンクを通じてデータベース・サーバに接続するには、次の手順に従ってください。

1. 次の接続文字列フラグメントは、RSA 暗号化とサンプルの信用された証明書を使用し

て、TCP/IP リンクを通じてデータベース・サーバ **myeng** に接続します。

```
"ENG=myeng; LINKS=tcpip;
Encryption=RSA_TLS (trusted_certificates=rsaserver.crt)"
```

[接続] ダイアログの [詳細] タブや、[ODBC データ・ソース] ダイアログの [ネットワーク] タブでもこのパラメータを設定できます。

暗号化設定の取り出し

connection_property システム関数を使用して、現在の接続の暗号化設定を取り出せます。使用されている暗号化のタイプに応じて、この関数は none、simple、ECC_TLS、または RSA_TLS の 4 つの値のうちいずれか 1 つを返します。

```
SELECT connection_property ( 'encryption' )
```

connection_property システム関数の使用方法の詳細については、『Adaptive Server Anywhere SQL リファレンス・マニュアル』の第 3 章 SQL 関数「CONNECTION_PROPERTY 関数」を参照してください。

パフォーマンスの問題

Adaptive Server Anywhere のパフォーマンスは、データベースが暗号化されている場合にはある程度低下します。パフォーマンスの影響は、ディスクとのページの読み取りや書き込みの頻度によって異なります。また、サーバが使用するキャッシュ・サイズを適切に設定することによって影響を最小限にできます。

キャッシュの初期サイズを増やすには、サーバの起動時に **-c** オプションで指定します。キャッシュの動的なサイズ変更がサポートされているオペレーティング・システムでは、使用されるキャッシュ・サイズが、使用可能なメモリの容量によって制限される場合があります。そのため、キャッシュ・サイズを増加するには、使用可能なメモリを増加します。

SQL Anywhere Studio セキュリティ・オプション

Adaptive Server Anywhere サーバまたは Mobile Link 同期サーバとクライアント間のデータ暗号化 (トランスポート・レイヤ・セキュリティ) と、MDSR データベース・ファイル暗号化用のソフトウェアは、別途注文してください。AES データベース・ファイル暗号化用ソフトウェアは、基本パッケージに含まれています。

詳細については、SQL Anywhere Studio パッケージに同梱されているカードを参照してください。