SYBASE®
An SAP® Company

プログラマーズ・ガイド

PHP 用 Adaptive Server® Enterprise 拡張モジュール　バージョン 15.7

ドキュメント ID： DC01819-01-1570-02
改訂： 2012 年 6 月

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

PHP 用 Adaptive Server Enterprise 拡張モジュール 1
PHP 用拡張モジュールのインストール 2
設定の概要 3
サンプル PHP スクリプト 4
PHP API リファレンス用拡張モジュール 4
sybase_affected_rows 4
sybase_close 5
sybase_connect 5
sybase_data_seek 6
sybase_fetch_array 6
sybase_fetch_assoc 7
sybase_fetch_field 7
sybase_fetch_object 8
sybase_fetch_row 8
sybase_field_seek 9
sybase_free_result 9
sybase_get_last_message 10
sybase_get_last_status 10
sybase_next_result 10
sybase_num_fields 11
sybase_num_rows 11
sybase_pconnect 12
sybase_query 12
sybase_rpc_bind_param_ex 13
sybase_rpc_execute 14
sybase_rpc_init 14
sybase_select_db 14
sybase_set_message_handler 15
sybase_unbuffered_query 16
sybase_use_result 16
セキュリティ・サービスとディレクトリ・サービス …17

その他のリソース 17
用語解説 19
索引 21

PHP 用 Adaptive Server Enterprise 拡張モジュール

PHP スクリプト言語用の Sybase® Adaptive Server® Enterprise 拡張モジュールを使用すると、PHP 開発者は Adaptive Server のデータベースに対してクエリを実行できます。

PHP 用の拡張モジュールは Adaptive Server データベースに対してクエリを実行するベンダー固有のデータベース・インタフェース・ドライバです。オープンソース・プログラミング言語である PHP (Hypertext Preprocessor) には、共通のデータベースから情報を取得する機能があります。PHP スクリプト言語用の Adaptive Server 拡張モジュールには必要な PHP API が含まれており、これを使用することにより、PHP 開発者は、Adaptive Server のデータベースに対してクエリを実行するためのスタンドアロン・スクリプトを記述することができます。

PHP 用の拡張モジュールはコマンド・ライン PHP 実装に対してテスト済みです。しかし、Web サーバの CGI モジュールなどでの使用については、Sybase ではテストしていません。

PHP 拡張モジュールでのデータ・フロー
次の図に、PHP スクリプトから Adaptive Server データベースへのデータ・フローを示します。

必要なコンポーネント
PHP プログラミング言語を使用して Adaptive Server データベースにアクセスするには、以下のコンポーネントが必要です。

- PHP スクリプト – Adaptive Server データベース・サーバに接続するためのアプリケーション・スクリプト。
- Adaptive Server Enterprise の PHP 用拡張モジュール – Sybase がサポートしているベンダ固有のドライバ。拡張モジュールは PHP スクリプトで呼び出され、API レイヤ DBCAPI に接続します。
- DBCAPI – PHP 拡張モジュールと CT-Library 間の中間の変換レイヤとして動作する関数ライブラリ。

- CT-Library – Adaptive Server にコマンドを送信し、結果を処理し、データを返すために使用できる Sybase の Open Client ライブラリ。

バージョン要件
プラットフォームのサポートの詳細は、使用しているプラットフォームの『Software Developer's Kit/Open Server インストール・ガイド』を参照してください。

- Adaptive Server Enterprise – バージョン 15.7 以上。PHP ドライバはバージョン 15.7 で開発しテスト済みです。しかし、15.7 より前の Adaptive Server バージョンには接続できます。
- PHP のインストール – バージョン 5.3.6
- Open Client SDK の CT-Library – バージョン 15.7
- DBCAPI — 同じ SDK インストールからの DBCAPI ライブラリと PHP ドライバを使用することをおすすめします。

PHP 用拡張モジュールのインストール

PHP 用の拡張モジュールは Sybase インストーラでインストールできるコンポーネントです。

PHP 用の拡張モジュールは、インストール・タイプとしてカスタムを選択した場合はオプションでインストールします。選択したインストール・タイプが標準またはフルの場合は、この拡張モジュールはデフォルトでインストールされます。

これは拡張モジュールのインストールの概要です。インストールと設定の完全な手順については、使用しているプラットフォームの『Software Developer's Kit/Open Server インストール・ガイド』を参照してください。

- インストール前の要件：
  - PHP インストール 5.3.6 – 64 ビットのみ。
  - Open Client SDK が必須。DBCAPI ライブラリは Open Client SDK に含まれており、コアの Open Client (CT-Library) とともにインストールされます。

  ドライバの一部として DBCAPI をインストールしないでください。
- PHP 用の拡張モジュールは PHP 拡張モジュールとしてダイナミック・ライブラリにインストールされます。
  Sybase インストーラ は Open Client (CT-Library) ディレクトリに、`sybaseasephp.so` という名前の共有ライブラリとして拡張モジュールをインストールします。
- 以下のファイルは拡張モジュールとともにインストールされます。
  ```
  $SYBASE/$SYBASE_OCS/php/php536_64/lib/sybaseasephp.so
  $SYBASE/$SYBASE_OCS/php/php536_64/devlib/sybaseasephp.so
  $SYBASE/$SYBASE_OCS/sample/php/README
  ```

```
$SYBASE/$SYBASE_OCS/sample/php/firstapp.php
$SYBASE/$SYBASE_OCS/config/generate_php_ini.sh
```

設定の概要

環境設定を行って、拡張モジュールと OCS インストールの場所を特定します。

環境変数

拡張モジュールを正常に使用するには、PHP 実行可能ファイルを実行する環境内で、Sybase 環境変数を設定します。少なくとも、$SYBASE 環境変数と $SYBASE_OCS 環境変数を設定する必要があります。*LD_LIBRARY_PATH* 変数が `$SYBASE/$SYBASE_OCS/lib` ディレクトリの正しい場所を指すように設定します。拡張モジュールはこの場所から DBCAPI ライブラリをロードします。

サンプル・スクリプト

サンプルの `php.ini` ファイルを作成するスクリプトが提供されています。PHP はこれを使用して、インストールされているディレクトリから拡張モジュールを特定します。このスクリプト `generate_php_ini.sh` は `$SYBASE/$SYBASE_OCS/config` にあります。`generate_php_ini.sh` スクリプトが正しい `php.ini` ファイルを生成するように、$SYBASE と $SYBASE_OCS を正しく設定する必要があります。

生成された `php.ini` ファイルを使用して、`firstapp.php` サンプルを実行するか、または拡張モジュールを使用してその他のテストを行うことができます。標準インストールとして、すでに `php.ini` ファイルが存在します。またサンプル `php.ini` の `generate_php_ini.sh` スクリプトからの情報が既存の `php.ini` ファイルにコピーされます。

システム管理者は拡張モジュールを別の拡張モジュールの特定のディレクトリにコピーし、`php.ini` ファイルを変更してそのディレクトリから拡張モジュールをロードできます。たとえば、次のようにします。

```
; Set the default extension directory.
extension_dir = "/usr/local/lib/php/extensions"
;Load the Sybase ASE PHP driver: sybaseasephp
extension="sybaseasephp.so"
```

ファイルを変更したら、`$SYBASE/$SYBASE_OCS/php/php536_64/lib/sybaseasephp.so` または `$SYBASE/$SYBASE_OCS/php/php536_64/devlib/sybaseasephp.so` ライブラリを `/usr/local/lib/php/extensions` にコピーできます。

拡張ライブラリ
`php -m` を実行すると、PHP インストール内のアクティブな拡張ライブラリの中に、`sybaseasephp` 拡張ライブラリが表示されます。

注意： 同じディレクトリの特定の PHP インストール内にすべての拡張ライブラリを置く必要はありません。

サンプル PHP スクリプト

サンプル・スクリプトは `$SYBASE/$SYBASE_OCS/sample/php/firstapp.php` にインストールされています。

サンプル・スクリプトの実行方法については、同じディレクトリにある README ファイルを参照してください。プラットフォーム・インストール情報については、『Software Developer's Kit/Open Server インストール・ガイド』も参照してください。

PHP API リファレンス用拡張モジュール

拡張モジュール・インタフェース API。

sybase_affected_rows

$conn によって参照されている接続で最後に実行された **INSERT** クエリ、**UPDATE** クエリ、または **DELETE** クエリによって影響を受けているロー数を返します。

この関数は、通常、**INSERT** 文、**UPDATE** 文、または **DELETE** 文で使用されます。**SELECT** 文では、**sybase_num_rows()** 関数の方を使用します。

構文

```
int sybase_affected_rows([resource $conn])
```

パラメータ

- **$conn –** 接続を開く関数によって返される接続リソース。接続リソースが指定されない場合、一番最後に開かれた接続が使用されます。

戻り値

最後の **INSERT** クエリ、**UPDATE** クエリ、または **DELETE** クエリによって影響を受けたロー数。

sybase_close

接続をクローズします。

構文

```
bool sybase_close([resource $conn])
```

パラメータ

- **$conn –** 接続を開く関数によって返される接続リソース。接続リソースが指定されていない場合、一番最後に開かれた接続を閉じます。

戻り値

成功時は TRUE。

失敗時は FALSE。

sybase_connect

Adaptive Server との接続を開きます。

構文

```
resource sybase_connect([string $servername
[, string $username
[, string $password
]]] )
```

パラメータ

- **$servername –** - 関連する Sybase ディレクトリ・サービス内で定義されるサーバ。
- **$username –** - Adaptive Server へのログイン時に使用するユーザ・アカウント。
- **$password –** - Adaptive Server へのログイン時に使用するユーザ・アカウントのパスワード。

戻り値

成功時は、正の接続識別子。

失敗時は FALSE。

sybase_data_seek

指定のロー番号を指すように、結果識別子に関連付けられている結果セット上の内部ロー・ポインタを移動します。

構文

```
bool sybase_data_seek(resource $result, int $row)
```

パラメータ

- **$result – sybase_query().** に対する呼び出しで取得される結果リソース。
- **$row –** 内部ポインタを設定するロー番号 (0 から開始)。

戻り値

TRUE – 内部ポインタの設定に成功した。

FALSE – 内部ポインタを正しく設定できなかった。

sybase_fetch_array

結果ローを関連配列、数値配列、またはその両方としてフェッチします。

構文

```
mixed sybase_fetch_array(resource $result
[, int $result_type ] )
```

パラメータ

- **$result – - sybase_query().** に対する呼び出しで取得される結果リソース。
- **$result_type –** 返される配列のタイプ。以下の値を受け取ります。

  SYB_FETCH_ASSOC – 関連配列

  SYB_FETCH_NUM – 数値配列

  SYB_FETCH_BOTH – (デフォルト) 関連と数値両方のインデックス

戻り値

$result_type が SYB_FETCH_ASSOC または SYB_FETCH_BOTH である場合、関連配列 (**sybase_fetch_assoc()** と同じ)。

$result_type が SYB_FETCH_NUM または SYB_FETCH_BOTH である場合、数値配列 (**sybase_fetch_row()** と同じ)。

FALSE – フェッチするローがなかった場合。

sybase_fetch_assoc

関連配列内で指定された結果識別子に関連付けられた結果セットから、データの 1 つのローをフェッチします。

内部ポインタの位置を結果セット内でロー 1 つ分進めます。その後 **sybase_fetch_assoc()** を呼び出すと、結果セット内の次のローを返します。ローがない場合は FALSE を返します。

構文

```
array sybase_fetch_assoc(resource $result)
```

パラメータ

- **$result –** - **sybase_query()** に対する呼び出しで取得される結果リソース。

戻り値

成功時は、結果セットの次のローを返します。

FALSE – フェッチするローがなかった場合。

sybase_fetch_field

フィールド情報を含むオブジェクトを返します。

この関数は、提供されるクエリ結果からフィールド情報を取得するのに使用できます。

構文

```
object sybase_fetch_field(resource $result [, int $field_offset ])
```

パラメータ

- **$result –** - **sybase_query()** に対する呼び出しで取得される結果リソース。
- **$field_offset –** 情報を取得するフィールド番号 (0 から開始)。フィールドオフセットが指定されていない場合、この関数によってまだ取得されていない次のフィールドが使用されます。

戻り値

プロパティとして次のフィールド情報を含むオブジェクトを返します。

フィールド名	フィールド・タイプ	フィールドの説明
name	string	カラム名。
table	string	カラムが取得されたテーブル。
max_length	int	カラムの最大長。
type	string	cspublic.h で定義されているカラムのデータ型。

sybase_fetch_object

オブジェクトとしてローをフェッチします。

この関数は、**sybase_fetch_assoc()** と似ていますが、配列ではなくオブジェクトを返します。内部ポインタの位置を結果セット内でロー 1 つ分進めます。

構文

```
object sybase_fetch_object(resource $result [, mixed $object ])
```

パラメータ

- **$result –** - **sybase_query()** に対する呼び出しで取得される結果リソース。
- **$object –** 返すオブジェクトのタイプを指定します (オプション)。デフォルトのオブジェクト・タイプは stdClass です。

戻り値

フェッチされたローのフィールド名に対応するプロパティを含むオブジェクトを返します。

FALSE – フェッチするローがなかった場合。

sybase_fetch_row

数値配列内の指定結果識別子に関連付けられている結果セットから、1 つのロー分のデータをフェッチします。

内部ポインタの位置を結果セット内でロー 1 つ分進めます。

その後 **sybase_fetch_row()** を呼び出すと、結果セット内の次のローを返します。ローがない場合は FALSE を返します。

構文

```
array sybase_fetch_row(resource $result)
```

パラメータ

- **$result –** - **sybase_query()** に対する呼び出しで取得される結果リソース。

戻り値

数値配列 (0 から開始) – **sybase_fetch_array($result, SYB_FETCH_NUM)** と同じ

FALSE – フェッチするローがない。

sybase_field_seek

要求したフィールドオフセットに内部ポインタを設定します。

sybase_fetch_field() の次回呼び出しでフィールドオフセットが指定されない場合、フィールド内部ポインタが使用されます。

構文

```
bool sybase_field_seek(resource $result, int $field_offset)
```

パラメータ

- **$result –** - **sybase_query()** に対する呼び出しで取得される結果リソース。
- **$field_offset –** フィールドオフセット (0 から開始)。

戻り値

TRUE – 内部ポインタが正しく設定された。

FALSE – 内部ポインタが正しく設定されなかった。

sybase_free_result

結果セットに関連付けられているすべてのメモリを解放します。

スクリプトが終了すると、結果メモリは自動的に解放されます。しかしプログラミングの習慣として、必要でなくなったメモリはユーザ自身が解放することをおすすめします。

構文

```
bool sybase_free_result(resource $result)
```

パラメータ

- **$result –** - **sybase_query()** に対する呼び出しで取得される結果リソース。

戻り値

TRUE – メモリが正常に解放されたとき。

FALSE – メモリの開放に失敗したとき。

sybase_get_last_message

Server により返された最後のメッセージ。

構文

```
string sybase_get_last_message(void)
```

パラメータ

なし

戻り値

サーバにより返された最後のメッセージ。

FALSE – 最後のサーバ・メッセージの取得に失敗した。

sybase_get_last_status

接続 $conn で送信された最新ステータスの結果を返します。

構文

```
int sybase_get_last_status(resource $conn)
```

パラメータ

- **$conn –** 接続を開く関数によって返される接続リソース。

戻り値

接続 $conn で送信された最新ステータスの結果。

sybase_next_result

接続 $conn 上の次の結果セットを指す結果セット識別子を返します。

構文

```
mixed sybase_next_result(resource $conn)
```

パラメータ

- **$conn –** 接続を開く関数によって返される接続リソース。

戻り値

成功時は、正の Sybase 結果セット識別子。

FALSE – 接続上に結果セットがない。

sybase_num_fields

結果セット内のフィールド数を返します。

構文

```
int sybase_num_fields(resource $result)
```

パラメータ

- **$result –** - **sybase_query()** に対する呼び出しで取得される結果リソース。

戻り値

結果セット内のフィールド数。

FALSE – フィールド数の取得に失敗した。

sybase_num_rows

SELECT 文の結果セット内のロー数を返します。

sybase_num_rows() は、完全な結果セットが読み込まれているときの正しいロー数を返します。

構文

```
int sybase_num_rows(resource $result)
```

パラメータ

- **$result –** - **sybase_query()** に対する呼び出しで取得される結果リソース。

戻り値

結果セット内のロー数。

FALSE – ロー数の取得に失敗した。

sybase_pconnect

Adaptive Server との永続的な接続を開きます。

以前、この呼び出しと同じ引数で永続的な接続を開いたことがある場合、新しい接続が開くのではなく、既存の接続の識別子が返されます。

構文

```
resource sybase_pconnect([string $servername
[, string $username
[, string $password
]]] )
```

パラメータ

- **$servername –** 関連する Sybase ディレクトリ・サービス内で定義されているサーバの名前。たとえば、interfaces ファイル。
- **$username –** Adaptive Server へのログインに使用されるユーザ・アカウントの名前。
- **$password –** Adaptive Server へのログインに使用されるユーザ・アカウントのパスワード。

戻り値

成功時は、正の接続識別子。

失敗時は FALSE。

sybase_query

接続にクエリを送信します。

構文

```
mixed sybase_query(string $query [, resource $conn])
```

パラメータ

- **$query –** - Adaptive Server に送信するクエリを含む文字列。
- **$conn –** 接続を開く関数によって返される接続リソース。接続リソースが指定されていない場合、一番最後に開かれた接続を使用します。

戻り値

成功時は、正の Sybase 結果セット識別子。

TRUE – クエリは成功したが、結果セットは返されなかった。

FALSE – クエリは失敗。

sybase_rpc_bind_param_ex

PHP 変数をリモート・プロシージャ・パラメータにバインドします。

構文

```
bool sybase_rpc_bind_param_ex(resource $stmt,
int $param_id,
mixed $var,
string $type
[, bool $is_null
[, int $direction]] )
```

パラメータ

- **$stmt – sybase_rpc_init()** 呼び出しによって返される文識別子リソース。
- **$param_id –** バインドするストアド・プロシージャ・パラメータの位置を示すインデックス。最初のパラメータは 0 になります。
- **$var –** バインドする PHP 変数の参照 (アドレス)。
- **$type –** バインドする PHP 変数のデータ型。以下のいずれかです。

  'd' - double

  'i' - integer

  'b' - binary

  's' - string
- **$is_null –** 変数が NULL を含むか含まないかを示すオプションのブール値。
- **$direction –** 次のいずれか (オプション)。

  SASE_D_INPUT – 入力パラメータ用 (デフォルト)

  SASE_D_OUTPUT – 出力パラメータ用

戻り値

TRUE – PHP 変数が正常にバインドされた。

FALSE – PHP 変数のバインドに失敗した。

sybase_rpc_execute

sybase_rpc_init() で初期化された $stmt 内のリモート・プロシージャ・コールを実行します。

構文

```
mixed sybase_rpc_execute(resource $stmt)
```

パラメータ

- **$stmt – sybase_rpc_init()** 呼び出しによって返される文識別子リソース。

戻り値

成功時は、正の Sybase 結果セット識別子。

FALSE – RPC の実行に失敗した。

sybase_rpc_init

接続 $conn で $procedure に対して初期化されている文を指す文識別子を返します。

構文

```
mixed sybase_rpc_init(resource $conn, string $procedure)
```

パラメータ

- **$conn –** 接続を開く関数によって返される接続リソース。
- **$procedure – sybase_rpc_execute()** で実行されるリモート (ストアド)・プロシージャの名前。

戻り値

成功時は、正の Sybase 文識別子。

FALSE – RPC 文の初期化に失敗した。

sybase_select_db

接続リソースにより参照されているサーバ上の現在アクティブなデータベースを設定します。

以降の **sybase_query()** 呼び出しはすべて、現在の接続リソースのアクティブなデータベースに対して行われます。

構文

```
bool sybase_select_db(string $database_name [, resource $conn])
```

パラメータ

- **$database_name –** 使用するデータベースの名前。
- **$conn –** 接続を開く関数によって返される接続リソース。接続リソースが指定されていない場合、一番最後に開かれた接続を使用します。

戻り値

TRUE – 現在のデータベースが正しく設定された。

FALSE – 現在のデータベースの設定に失敗した。

sybase_set_message_handler

クライアントまたはサーバのメッセージが受信されると呼び出されるユーザ定義済みのコールバック関数を設定します。

構文

```
bool sybase_set_message_handler(callback $handler, int $msg_type
[, resource $conn])
```

パラメータ

- **$handler –** コールバック・ハンドラには以下の引数があります。

  int – message_number

  int – severity

  int – state

  int – line_number

  string – description
- **$msg_type –** 下記のいずれか

  SYB_CLIENTMSG_CB – クライアント・メッセージ・コールバック

  SYB_SERVERMSG_CB – サーバ・メッセージ・コールバック

  注意： **$msg_type** は必須ですが、インストールされているメッセージ・ハンドラがクライアント・メッセージとサーバ・メッセージの両方について呼び出されることから、現在は無視されています。
- **$conn –** 接続を開く関数によって返される接続リソース。接続リソースが指定されていない場合、一番最後に開かれた接続を使用します。

戻り値

TRUE – コールバック関数のインストールに成功した。

FALSE – コールバック関数のインストールに失敗した。

sybase_unbuffered_query

$conn によって参照されている接続にクエリを送信します。

完全結果セットが **sybase_query()** のように自動的にフェッチされ、バッファされることがありません。これにより、特に結果セットが大きい場合、パフォーマンスの向上につながります。

sybase_fetch_array() や類似の関数を使用して、必要に応じて多くのローを読み込み、**sybase_data_seek()** を使用してターゲット・ローにジャンプします。

完全な結果セットが読み込まれているときに正しいロー数を返すには、**sybase_num_rows()** を使用します。

構文

```
mixed sybase_unbuffered_query(string $query [, resource $conn])
```

パラメータ

- **$query –** Adaptive Server に送信するクエリを含む文字列。
- **$conn –** 接続を開く関数によって返される接続リソース。

  接続リソースが指定されていない場合、一番最後に開かれた接続を使用します。

戻り値

成功時は、正の Sybase 結果セット識別子。

TRUE – クエリは成功したが、結果セットは返されなかった。

FALSE – クエリは失敗。

sybase_use_result

接続 $conn に前回バッファされているクエリの結果セットを格納して、この格納された結果セットを指す結果セット識別子を返します。

構文

```
mixed sybase_use_result(resource $conn)
```

パラメータ

- **$conn –** 接続を開く関数によって返される接続リソース。

戻り値

成功時は、正の Sybase 結果セット識別子。

FALSE – 接続上に格納する結果セットがない。

セキュリティ・サービスとディレクトリ・サービス

`ocs.cfg` ファイルと `libtcl.cfg` ファイルを使用してセキュリティ・オプションを設定します。

セキュリティを有効にする拡張モジュールのオプションは、現在サポートされていません。

接続については、`ocs.cfg` を使用してディレクトリとセキュリティ・プロパティを設定します。`libtcl.cfg` を編集して、セキュリティ・ドライバとディレクトリ・サービス・ドライバをロードします。

詳細については、『Open Client/Server 設定ガイド UNIX 版』の「設定ファイル」を参照してください。

その他のリソース

拡張モジュールのインストールと設定に関する追加情報は以下を参照してください。

- Open Client/Open Server の設定情報のマニュアル：
  『Open Client/Server 設定ガイド UNIX 版』の「設定ファイル」
- すべての Open Client/Server 製品についてのプラットフォームに関連した問題：
  『Open Client/Server プログラマーズ・ガイド補足 UNIX 版』
- Open Client/Server ランタイム設定ファイルの使用方法：
  『Open Client Client-Library/C リファレンス・マニュアル』の「ランタイム設定ファイルの使い方」の「Open Client/Server ランタイム設定ファイルの構文」
- プラットフォームのサポート：
  使用しているプラットフォームの『Software Developer’s Kit/Open Server インストール・ガイド』

用語解説

スクリプト言語に特有の用語集

- **Client-Library –** Open Client の一部で、クライアント・アプリケーションを記述するためのルーチンの集まり。Client-Library は、Sybase 製品ラインのカーソルや他の高度な機能を取り込むように設計されています。
- **CS-Library –** Client-Library と Server-Library のアプリケーションの両方で役立つユーティリティ・ルーチンの集まり。Open Client および Open Server の両方に含まれています。
- **CT-Library –** (CT-Lib API) は Open Client スイートの一部であり、スクリプト・アプリケーションで Adaptive Server に接続するために必要です。
- **DBD –** ベンダ固有のデータベース・ドライバで、DBI データベース API 呼び出しをターゲット・データベース SDK が理解できる形式に変換します。
- **PHP –** Hypertext Preprocessor の自己参照的な略。
- **スレッド (thread) –** Open Server アプリケーションからライブラリ・コードまでの実行のパス。また、スタック領域、ステータス情報およびイベント・ハンドラに対応するパス。
- **Transact-SQL –** データベース言語 SQL の機能拡張バージョン。アプリケーションは、Transact-SQL を使用して、Adaptive Server Enterprise と通信できます。

索引

D

DBCAPI 1

い

インストール
    インストール・ファイル 2
    概要 2
    要件 2

か

拡張モジュール
    PHP API リファレンス 4
拡張モジュール API リファレンス
    sybase_affected_rows 4
    sybase_close 5
    sybase_connect 5
    sybase_data_seek 6
    sybase_fetch_array 6
    sybase_fetch_assoc 7
    sybase_fetch_field 7
    sybase_fetch_object 8
    sybase_fetch_row 8
    sybase_field_seek 9
    sybase_free_result 9
    sybase_get_last_message 10
    sybase_get_last_status 10
    sybase_next_result 10
    sybase_num_fields 11
    sybase_num_rows 11
    sybase_pconnect 12
    sybase_query 12
    sybase_rpc_bind_param_ex 13
    sybase_rpc_execute 14
    sybase_rpc_init 14
    sybase_select_db 14
    sybase_set_message_handler 15
    sybase_unbuffered_query 16
    sybase_use_result 16
環境変数 3

こ

コンポーネント
    説明 1
    必要な 1

さ

サンプル・スクリプト 4

そ

その他のリソース 17

て

データ・フロー図 1

は

バージョン要件 1

よ

用語解説 19