

SYBASE®

ユーザース・ガイド

**Adaptive Server® Enterprise  
ADO.NET Data Provider**

2.155

[ Microsoft Windows 版 ]

ドキュメント ID : DC00067-01-0215-01

改訂 : 2009 年 10 月

Copyright © 2010 by Sybase, Inc. All rights reserved.

このマニュアルは Sybase ソフトウェアの付属マニュアルであり、新しいマニュアルまたはテクニカル・ノートで特に示されないかぎり、後続のリリースにも付属します。このマニュアルの内容は予告なしに変更されることがあります。このマニュアルに記載されているソフトウェアはライセンス契約に基づいて提供されるものであり、無断で使用することはできません。

このマニュアルの内容を弊社の書面による事前許可を得ずに、電子的、機械的、手作業、光学的、またはその他のいかなる手段によっても、複製、転載、翻訳することを禁じます。

#### マニュアルの注文

マニュアルの注文を承ります。ご希望の方は、サイベース株式会社営業部または代理店までご連絡ください。マニュアルの変更は、弊社の定期的なソフトウェア・リリース時のみ提供されます。

Sybase の商標は、Sybase trademarks ページ (<http://www.sybase.com/detail?id=1011207>) で確認できます。Sybase およびこのリストに掲載されている商標は、米国法人 Sybase, Inc. の商標です。® は、米国における登録商標であることを示します。

Java および Java 関連の商標は、米国およびその他の国における Sun Microsystems, Inc. の商標または登録商標です。

Unicode と Unicode のロゴは、Unicode, Inc. の登録商標です。

このマニュアルに記載されている上記以外の社名および製品名は、当該各社の商標または登録商標の場合があります。

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

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

# 目次

はじめに .....	xi
<b>第 1 章 Adaptive Server Enterprise ADO.NET Data Provider の理解と配備 .....</b>	<b>1</b>
Adaptive Server ADO.NET Data Provider とは .....	1
Adaptive Server ADO.NET Data Provider の配備 .....	2
システム稼働条件 .....	2
必要なファイル .....	2
ADO.NET Data Provider の新しいバージョンへの更新 .....	5
CLR のリダイレクト .....	6
Data Provider の更新の配備 .....	7
サンプル・プロジェクトの実行 .....	9
<b>第 2 章 サンプル・アプリケーションの使用 .....</b>	<b>11</b>
チュートリアル：Simple コード・サンプルの使用 .....	11
Simple サンプル・プロジェクトの理解 .....	13
チュートリアル：Table Viewer コード・サンプルの使用 .....	16
Table Viewer サンプル・プロジェクトの理解 .....	19
チュートリアル：Advanced コード・サンプルの使用 .....	22
Advanced サンプル・プロジェクトの理解 .....	24
<b>第 3 章 アプリケーションの開発 .....</b>	<b>29</b>
Visual Studio .NET プロジェクトでの Data Provider の使用 .....	29
Data Provider アセンブリへの参照の追加 .....	29
Adaptive Server ADO.NET Data Provider クラスの参照 .....	30
データベースへの接続 .....	31
接続プール .....	33
接続ステータスの確認 .....	34
文字セット .....	35
データに対するアクセスと操作 .....	36
AseCommand を使用したデータの取得と操作 .....	36
AseDataAdapter を使用したデータへのアクセスと操作 .....	49
プライマリ・キー値の取得 .....	62
BLOB の処理 .....	68
時刻値の取得 .....	70

	ストアド・プロシージャの使用 .....	72
	トランザクション処理 .....	75
	エラー処理 .....	78
	パフォーマンスの考慮事項 .....	79
	DbType.String と DbType.AnsiString .....	79
<b>第 4 章</b>	<b>Adaptive Server の高度な機能 .....</b>	<b>81</b>
	サポートされている Adaptive Server クラスタ・エディションの機能 .....	81
	ログインのリダイレクト .....	82
	接続マイグレーション .....	82
	接続フェールオーバ .....	82
	分散トランザクションの使用 .....	83
	Enterprise Services を使用するプログラミング .....	83
	ディレクトリ・サービス .....	85
	ディレクトリ・サービスとしての LDAP .....	86
	ディレクトリ・サービスの使用 .....	86
	マイクロ秒の精度の time データ .....	87
	パスワードの暗号化 .....	88
	パスワードの暗号化の有効化 .....	88
	パスワード有効期限の処理 .....	89
	SSL の使用 .....	90
	Adaptive Server ADO.NET Data Provider の SSL .....	91
	証明書によるサーバの検証 .....	92
	SSL 接続の有効化 .....	92
	高可用性システムでのフェールオーバの使用 .....	93
	Kerberos 認証の使用 .....	95
	プロセスの概要 .....	95
	稼働条件 .....	96
	Kerberos 認証の有効化 .....	96
	Key Distribution Center からの初期チケットの取得 .....	97
<b>第 5 章</b>	<b>サポートされている Microsoft ADO.NET の機能 .....</b>	<b>99</b>
	サポートされている Microsoft ADO.NET 2.0 の機能 .....	99
	Adaptive Server 用の Microsoft Enterprise Library DAAB .....	100
	Microsoft ADO.NET Entity Framework と LINQ .....	101
<b>第 6 章</b>	<b>Adaptive Server ADO.NET Data Provider API リファレンス .....</b>	<b>103</b>
	AseBulkCopy クラス .....	104
	AseBulkCopy コンストラクタ .....	104
	Close メソッド .....	104
	Dispose メソッド .....	105
	Finalize メソッド .....	105
	WriteToServer メソッド .....	105
	BatchSize プロパティ .....	105

BulkCopyTimeout プロパティ .....	105
ColumnMappings プロパティ .....	106
DestinationTableName プロパティ .....	106
NotifyAfter プロパティ .....	106
AseRowsCopied イベント .....	106
AseBulkCopyColumnMapping クラス .....	106
AseBulkCopyColumnMapping コンストラクタ .....	106
Equals メソッド .....	107
DestinationColumn プロパティ .....	107
DestinationOrdinal プロパティ .....	107
SourceColumn プロパティ .....	107
SourceOrdinal プロパティ .....	107
AseBulkCopyColumnMappingCollection クラス .....	108
AseBulkCopyColumnMappingCollection コンストラクタ .....	108
Add メソッド .....	108
Contains メソッド .....	108
IndexOf メソッド .....	108
Insert メソッド .....	108
Validate メソッド .....	109
Remove メソッド .....	109
AseBulkCopyOptions 列挙型 .....	109
AseClientFactory クラス .....	110
AseClientFactory コンストラクタ .....	110
Instance フィールド .....	110
CreateCommand メソッド .....	110
CreateCommandBuilder メソッド .....	110
CreateConnection メソッド .....	110
CreateConnectionStringBuilder メソッド .....	111
CreateDataAdapter メソッド .....	111
CreateDataSourceEnumerator メソッド .....	111
CreateParameter メソッド .....	111
CreatePermission メソッド .....	111
CanCreateDataSourceEnumerator プロパティ .....	112
AseClientPermission クラス .....	112
AseClientPermission コンストラクタ .....	112
AseClientPermissionAttribute クラス .....	112
AseClientPermissionAttribute コンストラクタ .....	113
CreatePermission メソッド .....	113
AseCommand クラス .....	113
AseCommand コンストラクタ .....	114
Cancel メソッド .....	114
CommandText プロパティ .....	114
CommandTimeout プロパティ .....	114
CommandType プロパティ .....	115
Connection プロパティ .....	115
CreateParameter メソッド .....	115

---

ExecuteNonQuery メソッド .....	116
ExecuteReader メソッド .....	116
ExecuteScalar メソッド .....	117
ExecuteXmlReader メソッド .....	117
NamedParameters .....	117
Parameters プロパティ .....	118
Prepare メソッド .....	118
Transaction プロパティ .....	119
UpdatedRowSource プロパティ .....	119
AseCommandBuilder クラス .....	119
AseCommandBuilder コンストラクタ .....	120
DataAdapter プロパティ .....	120
DeleteCommand プロパティ .....	120
DeriveParameters メソッド .....	120
Dispose メソッド .....	121
GetDeleteCommand メソッド .....	121
GetInsertCommand メソッド .....	121
GetUpdateCommand メソッド .....	122
InsertCommand プロパティ .....	122
PessimisticUpdate プロパティ .....	123
QuotePrefix プロパティ .....	123
QuoteSuffix プロパティ .....	124
RefreshSchema メソッド .....	124
SelectCommand プロパティ .....	125
UpdateCommand プロパティ .....	125
AseConnection クラス .....	125
AseConnection コンストラクタ .....	126
BeginTransaction メソッド .....	131
ChangeDatabase メソッド .....	131
Close メソッド .....	131
ConnectionString プロパティ .....	132
ConnectionTimeout プロパティ .....	133
CreateCommand メソッド .....	133
Database プロパティ .....	133
InfoMessage イベント .....	134
NamedParameters .....	134
Open メソッド .....	134
State プロパティ .....	134
StateChange イベント .....	135
TraceEnter、TraceExit イベント .....	135
AseConnectionPool クラス .....	135
Available プロパティ .....	135
Size プロパティ .....	136
AseConnectionPoolManager クラス .....	136
AseConnectionPoolManager コンストラクタ .....	136
GetConnectionPool メソッド .....	136

NumberOfOpenConnections プロパティ .....	136
AseDataAdapter クラス .....	137
AseDataAdapter コンストラクタ .....	137
AcceptChangesDuringFill プロパティ .....	138
ContinueUpdateOnError プロパティ .....	138
DeleteCommand プロパティ .....	138
Fill メソッド .....	139
FillError イベント .....	139
FillSchema メソッド .....	140
GetFillParameters .....	140
InsertCommand プロパティ .....	141
MissingMappingAction プロパティ .....	141
MissingSchemaAction プロパティ .....	141
RowUpdated イベント .....	142
RowUpdating イベント .....	142
SelectCommand プロパティ .....	143
TableMappings プロパティ .....	143
Update メソッド .....	143
UpdateCommand プロパティ .....	144
AseDataReader クラス .....	144
Close メソッド .....	145
Depth プロパティ .....	145
Dispose メソッド .....	145
FieldCount プロパティ .....	145
GetBoolean メソッド .....	146
GetByte メソッド .....	146
GetBytes メソッド .....	146
GetChar メソッド .....	147
GetChars メソッド .....	147
GetDataTypeName メソッド .....	148
GetDateTime メソッド .....	148
GetDecimal メソッド .....	149
GetDouble メソッド .....	149
GetFieldType メソッド .....	149
GetFloat メソッド .....	150
GetInt16 メソッド .....	150
GetInt32 メソッド .....	150
GetList メソッド .....	151
GetName メソッド .....	151
GetOrdinal メソッド .....	151
GetSchemaTable メソッド .....	152
GetString メソッド .....	153
GetUInt16 メソッド .....	153
GetUInt32 メソッド .....	153
GetUInt64 メソッド .....	154
GetValue メソッド .....	154

---

GetValues メソッド .....	154
IsClosed プロパティ .....	155
IsDBNull メソッド .....	155
Item プロパティ .....	155
NextResult メソッド .....	156
Read メソッド .....	156
RecordsAffected プロパティ .....	156
AseDbType 列挙型 .....	157
AseError クラス .....	159
ErrorNumber プロパティ .....	159
Message プロパティ .....	159
SqlState プロパティ .....	159
ToString メソッド .....	159
AseErrorCollection クラス .....	161
CopyTo メソッド .....	161
Count プロパティ .....	162
Item プロパティ .....	162
AseException クラス .....	162
Errors プロパティ .....	162
Message プロパティ .....	163
AseFailoverException クラス .....	163
Errors プロパティ .....	163
Message プロパティ .....	163
ToString メソッド .....	164
AseInfoMessageEventArgs クラス .....	164
Errors プロパティ .....	164
Message プロパティ .....	164
AseInfoMessageEventHandler デリゲート .....	164
AseParameter クラス .....	165
AseParameter コンストラクタ .....	165
AseDbType プロパティ .....	165
DbType プロパティ .....	166
Direction プロパティ .....	166
IsNullable プロパティ .....	166
ParameterName プロパティ .....	166
Precision プロパティ .....	167
Scale プロパティ .....	167
Size プロパティ .....	167
SourceColumn プロパティ .....	168
SourceVersion プロパティ .....	168
ToString メソッド .....	169
Value プロパティ .....	169
AseParameterCollection クラス .....	169
Add メソッド .....	170
Clear メソッド .....	170
Contains メソッド .....	170



---

CopyTo メソッド .....	171
Count プロパティ .....	171
IndexOf メソッド .....	171
Insert メソッド .....	171
Item プロパティ .....	172
Remove メソッド .....	172
RemoveAt メソッド .....	172
AseRowsCopiedEventArgs クラス .....	173
AseRowsCopiedEventArgs コンストラクタ .....	173
Abort プロパティ .....	173
RowCopied プロパティ .....	173
AseRowsCopiedEventHandler デリゲート .....	173
AseRowUpdatedEventArgs クラス .....	174
AseRowUpdatedEventArgs コンストラクタ .....	174
Command プロパティ .....	174
Errors プロパティ .....	174
RecordsAffected プロパティ .....	174
Row プロパティ .....	175
StatementType プロパティ .....	175
Status プロパティ .....	175
TableMapping プロパティ .....	175
AseRowUpdatingEventArgs クラス .....	176
AseRowUpdatingEventArgs コンストラクタ .....	176
Command プロパティ .....	176
Errors プロパティ .....	176
Row プロパティ .....	176
StatementType プロパティ .....	177
Status プロパティ .....	177
TableMapping プロパティ .....	177
AseRowUpdatedEventHandler デリゲート .....	177
AseRowUpdatingEventHandler デリゲート .....	178
AseTransaction クラス .....	178
Commit メソッド .....	178
Connection プロパティ .....	178
IsolationLevel プロパティ .....	179
Rollback メソッド .....	179
TraceEnterEventHandler デリゲート .....	179
TraceExitEventHandler デリゲート .....	180
<b>索引 .....</b>	<b>181</b>



# はじめに

## 対象読者

このマニュアルは、サポートされている .NET プログラミング言語を使用して Adaptive Server® Enterprise からデータをアクセスする必要のあるアプリケーション開発者を対象としています。このマニュアルを使用する方は、Microsoft ADO.NET テクノロジーに精通しており、ADO.NET 仕様をコード化できる必要があります。

## このマニュアルの内容

このマニュアルは、次のように構成されています。

- 「[第 1 章 Adaptive Server Enterprise ADO.NET Data Provider の理解と配備](#)」では、Adaptive Server Enterprise ADO.NET Data Provider の概要について説明します。
- 「[第 2 章 サンプル・アプリケーションの使用](#)」では、Adaptive Server ADO.NET Data Provider に付属するサンプル・プロジェクトの使用方法について説明します。
- 「[第 3 章 アプリケーションの開発](#)」 contains information about developing and deploying applications with the Adaptive Server ADO.NET Data Provider.
- 「[第 4 章 Adaptive Server の高度な機能](#)」では、Adaptive Server ADO.NET Data Provider で使用できる Adaptive Server の機能について説明します。
- 「[第 5 章 サポートされている Microsoft ADO.NET の機能](#)」では、Adaptive Server ADO.NET Data Provider でサポートされている Microsoft ADO.NET の機能について説明します。
- 「[第 6 章 Adaptive Server ADO.NET Data Provider API リファレンス](#)」では、Adaptive Server ADO.NET Data Provider API について説明します。

## 関連マニュアル

詳細については、これらのマニュアルを参照できます。

- 『Software Developer’s Kit リリース・ノート Microsoft Windows 版』には、Adaptive Server ADO.NET Data Provider および Software Developer’s Kit (SDK) に関する重要な最新情報が記載されています。
- 『Software Developer’s Kit/Open Server インストール・ガイド』には SDK Adaptive Server ADO.NET Data Provider コンポーネントのインストールについて説明します。
- Adaptive Server Enterprise の『インストール・ガイド』には、Adaptive Server のインストールについて説明します。
- 使用しているプラットフォームの Adaptive Server Enterprise の『リリース・ノート』では、既知の問題および更新の詳細について説明します。

---

## その他の情報

Sybase® Getting Started CD、SyBooks™ CD、Sybase Product Manuals Web サイトを利用すると、製品について詳しく知ることができます。

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイド、SyBooks CD に含まれていないその他のマニュアルや更新情報が収録されています。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です (CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます)。
- SyBooks CD には製品マニュアルが収録されています。この CD は製品のソフトウェアに同梱されています。Eclipse ベースの SyBooks ブラウザを使用すれば、使いやすい HTML 形式のマニュアルにアクセスできます。

一部のマニュアルは PDF 形式で提供されています。これらのマニュアルは SyBooks CD の PDF ディレクトリに収録されています。PDF ファイルを開いたり印刷したりするには、Adobe Acrobat Reader が必要です。

SyBooks をインストールして起動するまでの手順については、Getting Started CD の『SyBooks インストール・ガイド』、または SyBooks CD の『*README.txt*』ファイルを参照してください。

- Sybase Product Manuals Web サイトは、SyBooks CD のオンライン版であり、標準の Web ブラウザを使用してアクセスできます。また、製品マニュアルのほか、EBFs/Updates、Technical Documents、Case Management、Solved Cases、ニュース・グループ、Sybase Developer Network へのリンクもあります。

Technical Library Product Manuals Web サイトにアクセスするには、Product Manuals (<http://www.sybase.com/support/manuals/>) にアクセスしてください。

## Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

### ❖ 製品認定の最新情報にアクセスする

- 1 Web ブラウザで Technical Documents を指定します。  
(<http://www.sybase.com/support/techdocs/>)
- 2 [Partner Certification Report] をクリックします。
- 3 [Partner Certification Report] フィルタで製品、プラットフォーム、時間枠を指定して [Go] をクリックします。
- 4 [Partner Certification Report] のタイトルをクリックして、レポートを表示します。

### ❖ コンポーネント認定の最新情報にアクセスする

- 1 Web ブラウザで Availability and Certification Reports を指定します。  
(<http://certification.sybase.com/>)
- 2 [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。 .
- 3 [Search] をクリックして、入手状況と認定レポートを表示します。

### ❖ Sybase Web サイト ( サポート・ページを含む ) の自分専用のビューを作成する

MySybase プロファイルを設定します。MySybase は無料サービスです。このサービスを使用すると、Sybase Web ページの表示方法を自分専用カスタマイズできます。

- 1 Web ブラウザで Technical Documents を指定します。  
(<http://www.sybase.com/support/techdocs/>)
- 2 [MySybase] をクリックし、MySybase プロファイルを作成します。

## Sybase EBF とソフトウェア・メンテナンス

### ❖ EBF とソフトウェア・メンテナンスの最新情報にアクセスする

- 1 Web ブラウザで Sybase Support Page を指定します。  
(<http://www.sybase.com/support>)
- 2 [EBFs/Maintenance] を選択します。MySybase のユーザ名とパスワードを入力します。
- 3 製品を選択します。
- 4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

鍵のアイコンは、「Technical Support Contact」として登録されていないため、一部の EBF/Maintenance リリースをダウンロードする権限がないことを示しています。未登録でも、Sybase 担当者またはサポート・コンタクトから有効な情報を得ている場合は、[Edit Roles] をクリックして、「Technical Support Contact」の役割を MySybase プロファイルに追加します。

- 5 EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

## 表記規則

このマニュアルで使用されている表記規則は次のとおりです。

- クラス、コマンド名、コマンド・オプション名、メソッド、プログラム名、プログラム・フラグ、プロパティ、キーワード、関数、文、ストアド・プロシージャは次の形式で表記されます。

ExecuteNonQuery メソッドでは、Insert、Update、または Delete 文を使用できます。

- 変数、パラメータ、ユーザが指定する語は、構文内と本文中では次のように斜体で表記されます。

`set password new_passwd` 句では新しいパスワードを指定します。

- データベース、テーブル、カラム、データ型などのデータベース・オブジェクトの名前は、次のように表記されます。

`pubs2` オブジェクトの値。

- コマンドの構文やオプションを示す文は、次のように表記されます。

```
AseDataAdapter adapter
string connectionString
AseCommand selectCommand
```

コマンドの用途を示す例は、次のように表記されます。

```
AseConnection conn = new AseConnection(
    "Data Source='mango';" +
    "Port=5000;" +
    "UID='sa';" +
    "PWD='';" +
    Database='pubs2';");
```

次の表は、構文の表記規則をまとめたものです。

**表 1: 構文の表記規則**

キー	定義
{ }	中カッコは、その中のオプションを1つ以上選択しなければならないことを意味する。コマンドには中カッコは入力しない。
[ ]	角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。
	縦線は、中カッコまたは角カッコの中の複数のオプションのうち1つだけを選択できることを意味する。
,	カンマは、中カッコまたは角カッコの中のオプションをいくつでも選択できることを意味する。複数のオプションを選択する場合には、オプションをカンマで区切る。 カンマは他の構文内容で必須になることもある。
( )	このカッコはコマンドの一部として入力する。
...	省略記号 (...) は、直前の要素を必要な回数だけ繰り返し指定できることを意味する。省略記号はコマンドには入力しない。

**アクセシビリティ機能**

このマニュアルには、アクセシビリティを重視した HTML 版もあります。この HTML 版マニュアルは、スクリーン・リーダーで読み上げる、または画面を拡大表示するなどの方法により、その内容を理解できるよう配慮されています。

Adaptive Server Enterprise ADO.NET Data Provider マニュアルは、連邦リハビリテーション法第 508 条のアクセシビリティ規定に準拠していることがテストにより確認されています。第 508 条に準拠しているマニュアルは通常、World Wide Web Consortium (W3C) の Web サイト用ガイドラインなど、米国以外のアクセシビリティ・ガイドラインにも準拠しています。

---

**注意** アクセシビリティ・ツールを効率的に使用するには、設定が必要な場合もあります。一部のスクリーン・リーダーは、テキストの大文字と小文字を区別して発音します。たとえば、すべて大文字のテキスト (ALL UPPERCASE TEXT など) はイニシャルで発音し、大文字と小文字の混在したテキスト (Mixed Case Text など) は単語として発音します。構文規則を発音するようにツールを設定すると便利かもしれません。詳細については、ツールのマニュアルを参照してください。

---

Sybase のアクセシビリティに対する取り組みについては、[Sybase Accessibility \(http://www.sybase.com/accessibility\)](http://www.sybase.com/accessibility) を参照してください。Sybase Accessibility サイトには、第 508 条と W3C 標準に関する情報へのリンクもあります。

**不明な点があるときは**

Sybase ソフトウェアがインストールされているサイトには、Sybase 製品の保守契約を結んでいるサポート・センタとの連絡担当の方 (コンタクト・パーソン) を決めてあります。マニュアルだけでは解決できない問題があった場合には、担当の方を通して Sybase のサポート・センタまでご連絡ください。





# Adaptive Server Enterprise ADO.NET Data Provider の理解と配備

この章では、Adaptive Server Enterprise ADO.NET Data Provider の概要について説明します。

トピック名	ページ
<a href="#">Adaptive Server ADO.NET Data Provider とは</a>	1
<a href="#">Adaptive Server ADO.NET Data Provider の配備</a>	2
<a href="#">ADO.NET Data Provider の新しいバージョンへの更新</a>	5
<a href="#">サンプル・プロジェクトの実行</a>	9

## Adaptive Server ADO.NET Data Provider とは

Adaptive Server Enterprise ADO.NET Data Provider は、Adaptive Server Enterprise 用の ADO.NET プロバイダです。Adaptive Server バージョン 12.5.x、15.0.x、15.5 でサポートされています。

Adaptive Server ADO.NET Data Provider を使用すると、C#、Visual Basic .NET、マネージ拡張を備えた C++、J# など、.NET でサポートされる任意の言語を使用して Adaptive Server 内のデータにアクセスできます。.NET 共通言語ランタイム (CLR: Common Language Runtime) アセンブリであり、ADO.NET インタフェース全般の機能を提供する、一連の必要なクラスをすべて含んだクラス・ライブラリに相当します。すべてのクラスはマネージ・コードで、任意のマネージ・クライアント・コードからアクセスできます。このような各言語間の通信は、Microsoft .NET Framework によって実現します。

Adaptive Server ADO.NET Data Provider を使用する主な利点としては、次のものが挙げられます。

- Adaptive Server ADO.NET Data Provider は、OLE DB プロバイダよりも高速である。
- .NET 環境において、Adaptive Server ADO.NET Data Provider は Adaptive Server に対するネイティブ・アクセスを提供する。サポートされるその他のプロバイダとは異なり、Adaptive Server と直接通信できるため、ブリッジ技術を必要としない。

## Adaptive Server ADO.NET Data Provider の配備

以降の各項では、Adaptive Server ADO.NET Data Provider を配備するための要件について説明します。サポートされているプラットフォームのリストについては、[Sybase platform certifications ページ](http://certification.sybase.com/ucr/search.do)を参照してください。  
(<http://certification.sybase.com/ucr/search.do>)

### システム稼働条件

Adaptive Server ADO.NET Data Provider を使用するには、コンピュータに次のものがインストールされている必要があります。

- 開発時 – .NET Framework SDK 2.0 と Visual Studio .NET 2005
- 配備時 – .NET Framework 2.0

### 必要なファイル

Adaptive Server ADO.NET Data Provider は、以下のファイルで構成されています。

- *Sybase.AdoNet2.AseClient.dll* は、クライアント・コードで参照されるプロバイダ・アセンブリです。
- *sybdrvado20.dll* には、ユーティリティ・コードが含まれています。
- *sbgse2.dll*、*sybcsi\_certicom\_fips26.dll*、*sybcsi\_core26.dll*、*sybcsi\_profiler26.dll* には、SSL をサポートするためのコードが含まれます。
- *sybdrvkrb.dll* には、Kerberos 認証のためのコードが含まれます。

### グローバル・アセンブリ・キャッシュへの Adaptive Server ADO.NET Data Provider アセンブリの配備

多くの場合、1 台のコンピュータ上の複数のアプリケーションで、Adaptive Server ADO.NET Data Provider アセンブリが共有されています。この結果、アセンブリのコピーが重複して存在することになり、互換性やバージョン管理の問題が生じます。このような状況を回避するため、Adaptive Server ADO.NET Data Provider アセンブリをグローバル・アセンブリ・キャッシュ (GAC: Global Assembly Cache) に配備することをおすすめします。グローバル・アセンブリ・キャッシュはコンピュータ全体を対象範囲とするキャッシュであり、同じコンピュータ上の複数のアプリケーションによって共有されているアセンブリを格納および管理します。このような配備を行うことができない場合は、プロバイダを使用するアプリケーションが実行されるすべてのディレクトリに、Adaptive Server ADO.NET Data Provider アセンブリのコピーをインストールしてください。

アセンブリは Adaptive Server ADO.NET Data Provider のインストール・プログラムによって自動的に GAC に配備されます。インストール・プログラムを使用しない場合は、手動でアセンブリを配備してください。これを行うには、AseGacUtility を実行するか、.NET Framework 構成ツールを使用します。

#### ❖ AseGacUtility を実行してアセンブリを配備する

- 1 AseGacUtility がインストールされているディレクトリに移動します。デフォルトのロケーションは、Adaptive Server ADO.NET Data Provider の 32 ビット版では `C:¥Sybase¥DataAccess¥ADONET¥dll`、Adaptive Server ADO.NET Data Provider の 64 ビット版では `C:¥Sybase¥DataAccess64¥ADONET¥dll` です。
- 2 次のコマンドを実行します。

```
AseGacUtility /i DLL_Name
```

`DLL_Name` には、GAC に配備する DLL を指定します。

たとえば、Sybase.AdoNet2.AseClient.dll を配備するには、次のように入力します。

```
AseGacUtility /i  
c:¥mango¥build¥bin¥Sybase.AdoNet2.AseClient.dll
```

#### ❖ .NET Framework 構成ツールを使用してアセンブリを配備する

- 1 .NET Framework 構成ツールを起動します。構成ツールの起動方法については、各オペレーティング・システムの Microsoft のマニュアルを参照してください。
- 2 左側のツリー・ビューで、[アセンブリ キャッシュ] を選択します。
- 3 [アセンブリ キャッシュにアセンブリを追加する] リンクをクリックします。
- 4 [アセンブリの追加] ダイアログ・ボックスで、インストール・ディレクトリにある Adaptive Server ADO.NET Data Provider アセンブリを検索して、[開く] をクリックします。デフォルトのインストール・ディレクトリは、Adaptive Server ADO.NET Data Provider の 32 ビット版では `C:¥Sybase¥DataAccess¥ADONET¥dll`、Adaptive Server ADO.NET Data Provider の 64 ビット版では `C:¥Sybase¥DataAccess64¥ADONET¥dll` です。

これで、Adaptive Server ADO.NET Data Provider アセンブリが GAC に配備されます。キャッシュ内のアセンブリのリストを確認するには、[アセンブリ キャッシュのアセンブリ一覧の表示] リンクを選択します。

## GAC からのアセンブリの削除

GAC からアセンブリを削除するには、AseGacUtility を実行するか、.NET Framework 構成ツールを使用します。

### ❖ AseGacUtility を実行してアセンブリを削除する

- 1 AseGacUtility がインストールされているディレクトリに移動します。デフォルトのロケーションは、Adaptive Server ADO.NET Data Provider の 32 ビット版では `C:\Sybase\Data.Access\ADONET\dll`、Adaptive Server ADO.NET Data Provider の 64 ビット版では `C:\Sybase\Data.Access64\ADONET\dll` です。
- 2 次のコマンドを実行します。

```
AseGacUtility /u DLL_Name
```

`DLL_Name` には、GAC から削除する DLL を指定します。

たとえば、GAC から `Sybase.AdoNet2.AseClient.dll` を削除するには、次のように入力します。

```
AseGacUtility /u  
c:\mango\build\bin\Sybase.AdoNet2.AseClient.dll
```

### ❖ .NET Framework 構成ツールを使用してアセンブリを削除する

- 1 .NET Framework 構成ツールを起動します。構成ツールの起動方法については、各オペレーティング・システムの Microsoft のマニュアルを参照してください。
- 2 左側のツリー・ビューで、[アセンブリ キャッシュ] を選択します。
- 3 [アセンブリ キャッシュのアセンブリ一覧の表示] リンクをクリックします。
- 4 アセンブリ名のリストで、`Sybase.AdoNet2.AseClient` を探します。システムに複数のバージョンが配備されている場合は、このアセンブリのエントリがバージョンに対応して複数表示される場合もあります。
- 5 削除するアセンブリを 1 つ以上選択します。右クリックして [削除] を選択します。[はい] をクリックして操作を確定します。
- 6 削除したバージョンに対応する発行者ポリシー・ファイルがないか確認し、これらのファイルも削除します。

---

**注意** GAC には、他のアセンブリから特定のアセンブリへの参照も格納されます。この場合、これらの参照が削除されるまで、この参照先のアセンブリは削除できません。これらの参照は、削除コマンドの一部として強制的に削除できません。システムによっては、ユーティリティがアセンブリの削除に失敗して、Windows インストーラで保留中の参照に関するエラーが発生することがあります。これは、レジストリに値が残っているために発生するものです。この問題が発生した場合は、Microsoft のサポートに連絡して解決策を確認してください。

---

## Adaptive Server ADO.NET Data Provider を使用するアプリケーションの配備

以下の手順では、アプリケーションを配備する方法について説明します。

- ❖ **インストール・プログラムと GAC を使用してアプリケーションを配備する**
  - 1 エンド・ユーザのコンピュータのインストール・プログラムを使用して Adaptive Server ADO.NET Data Provider をインストールします。
  - 2 *exe* や *dll* などのアプリケーション固有ファイルを、システムのアプリケーション固有フォルダにコピーします。
- ❖ **GAC を使用してアプリケーションを配備する**
  - 1 Adaptive Server ADO.NET Data Provider を構成する *dll* ファイルを、ターゲット・コンピュータの *C:\\$Sybase\DataAccess\ADONET\dll* などのディレクトリにコピーします。
  - 2 このディレクトリをシステム・パスに追加します。
  - 3 プロバイダ・アセンブリを GAC に配備します。「[グローバル・アセンブリ・キャッシュへの Adaptive Server ADO.NET Data Provider アセンブリの配備](#)」(2 ページ) を参照してください。
  - 4 *exe* や *dll* などのアプリケーション固有ファイルを、システムのアプリケーション固有フォルダにコピーします。
  - 5 アプリケーションを実行します。
- ❖ **GAC を使わずにアプリケーションを配備する**
  - 1 ターゲット・システムで、*exe* や *dll* などのアプリケーション固有ファイルに加えて、Adaptive Server ADO.NET Data Provider を構成する *dll* ファイルもアプリケーション固有フォルダにコピーします。
  - 2 アプリケーションを実行します。

## ADO.NET Data Provider の新しいバージョンへの更新

Adaptive Server ADO.NET Data Provider の更新は、EBF/ESD またはメンテナンス・リリースを通じて入手します。この項では、Data Provider を新しいバージョンに更新する場合の問題点について説明します。更新に関する Microsoft .NET コンセプトの詳細については、Microsoft Developer Network (<http://msdn.microsoft.com>) で公開されている『.NET Framework Deployment Guide』と『.NET Framework Developer's Guide』を参照してください。

新しいバージョンの Adaptive Server Data Provider にアプリケーションを移行するには、次のいずれかを実行してください。

- アプリケーション設定ファイルを作成して、新しいバージョンの Adaptive Server ADO.NET Data Provider を使用するようにアプリケーションをリダイレクトします。「[アプリケーション設定ファイルの使用](#)」(6 ページ) を参照してください。
- 新しいバージョンの Adaptive Server ADO.NET Data Provider に対してアプリケーションを再度ビルドし、配備します。Sybase では、この手順を選択することをおすすめします。

## CLR のリダイレクト

.NET 共通言語ランタイム (CLR) は、アプリケーション・プログラムの実行時、Data Provider などのアセンブリに対する参照を見つけてバインドします。CLR はデフォルトで、アプリケーションの構築に使用された同じバージョンのアセンブリに対する参照をバインドしようとします。そのため、配備しただけで更新バージョンのアセンブリが自動的に使用されるわけではありません。新しいバージョンのアセンブリを使用してアプリケーションを再構築するか、新しいバージョンを使用するように設定ファイルで CLR をリダイレクトする必要があります。

一般的に、Data Provider の EBF/ESD リリースのリリース・レベル (メジャー/マイナー) が同じであれば、前のリリースとの間にバイナリの互換性があります。このような更新ではアプリケーションを再構築をしないことも可能です。Data Provider の更新ごとにアプリケーションの再構築や再配備を行う代わりに、アプリケーション設定や発行者ポリシー・ファイルを使用することもできます。Sybase では通常、ESD/EBF リリースに、適切なリダイレクトを設定した発行者ポリシー・ファイルを収録しています。下位互換性の問題の詳細については、ESD/EDF のドキュメントを参照してください。

## アプリケーション設定ファイルの使用

アプリケーション設定ファイルを使用すると、CLR をリダイレクトして、呼び出し側のアプリケーションのマニフェストに格納されているアセンブリとバージョンの異なるアセンブリをロードできます。

次の例は、以前の Data Provider 1.0.x で構築されたアプリケーションで Data Provider 1.0.159 を使用するように CLR をリダイレクトする方法を示しています。

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Sybase.AdoNet2.AseClient"
          publicKeyToken="95d94fac46c88e1e"
          culture="neutral" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

```
<bindingRedirect oldVersion="1.0.0.0-2.155.999.65535"  
                newVersion="2.155.1000.0"/>  
</dependentAssembly>  
</assemblyBinding>  
</runtime>  
</configuration>
```

設定ファイル・スキーマの詳細については、MSDN Library (<http://msdn2.microsoft.com/en-us/default.aspx>) を参照してください。

---

**注意** アプリケーションごとに独自の設定ファイルが必要です。

---

## 発行者ポリシー・ファイルの使用

アセンブリの発行元は、共有アセンブリの更新とともに発行者ポリシー・ファイルを配布できます。このファイルによって、古いバージョンのアセンブリに対するすべての参照が、新しくインストールされたバージョンにリダイレクトされます。アプリケーション設定ファイルとは異なり、発行者ポリシー・ファイルを機能させるには、グローバル・アセンブリ・キャッシュ (GAC) に配備する必要があります。

発行者ポリシー・ファイルの設定は、アプリケーションまたはアプリケーション設定ファイルのバージョン情報よりも優先されます。ただし、「セーフ・モード」を強制し、特定のアプリケーションで発行者ポリシー・ファイルを無視するように設定することもできます。セーフ・モードを使用するようにアプリケーションを設定する方法については、MSDN ライブラリを参照してください。

一般的に、Adaptive Server ADO.NET Data Provider の更新には、最後にインストールされたバージョンの Data Provider アセンブリにアプリケーションをリダイレクトする発行者ポリシー・ファイルが組み込まれています。これにより、新しいプロバイダ・アセンブリと発行者ポリシー・ファイルが GAC に配備されます。

## Data Provider の更新の配備

以降の各項では、Data Provider の更新の配備に関連する問題点について説明します。

## Data Provider を GAC に配備する

更新された Data Provider アセンブリとポリシー・アセンブリが GAC に配備されると、システム上のすべてのアプリケーションが自動的にこの Provider の使用を開始します。

## 更新された Data Provider を使用しないように特定のアプリケーションを除外する

更新された Data Provider を使用しないように特定のアプリケーションを除外する場合は、そのアプリケーションのアプリケーション設定ファイルを、発行者ポリシー・ファイルを無効にするセーフ・モードを強制するように設定します。

### コア・ファイルのロケーション

Data Provider は、*Sybase.AdoNet2.AseClient.dll* と *sybdrvado20.dll* という 2 つのコア・ファイルで構成されます。GAC には *Sybase.AdoNet2.AseClient.dll* の複数のバージョンをインストールできます。ただし、*sybdrvado20.dll* は GAC にインストールされるのではなく、システムの PATH を使用して実行時に検索されます。このファイルは、Data Provider のインストール・ディレクトリにインストールされます。Sybase は、更新リリースでこの DLL の名前やバージョン文字列を変更する場合があります。たとえば、このファイルの名前はリリース 1.0 では *aseado.dll* でしたが、リリース 2.0 では *sybdrvado20.dll* と呼ばれています。このような更新をインストールするときは、このファイルを使用する Data Provider のバージョンが GAC から削除されるまで、古いバージョンのファイルを削除しないでください。削除すると、古いバージョンのプロバイダを使用するアプリケーションを実行できなくなります。

## GAC がない場合の Data Provider の配備

コンピュータの GAC に Data Provider アセンブリがインストールされていない場合、Data Provider の構成要素であるファイルをアプリケーション・フォルダにコピーしてください。

更新されたバージョンの Data Provider をアプリケーションで使用するには、次のいずれかを実行します。

- 適切な **redirect** を使用してアプリケーション設定ファイルを作成する。
- 新しいバージョンの Data Provider に合わせてアプリケーションを再構築する。
- 発行者ポリシー・ファイルのみを GAC に配備する。これにより、アプリケーションで特に除外されていない限り、コンピュータ上の Data Provider に対するすべての参照で、発行者ポリシー・ファイルの **redirect** が使用されます。



## サンプル・プロジェクトの実行

Adaptive Server ADO.NET Data Provider には次の 3 つのサンプル・プロジェクトが組み込まれています。

- **Simple** – データベースへ接続し、クエリを実行して、返された resultsets を読み込む方法を示すサンプル・プログラム。
- **TableViewer – AseDataAdapter** オブジェクトを使用して結果を DataGrid コントロールにバインドする方法を示すサンプル・プログラム。
- **Advanced** – 入力、出力、および入出力パラメータとともにストアド・プロシージャを呼び出す方法を示すサンプル・プログラム。ストアド・プロシージャの戻り値を読み取り、パラメータを渡すための 2 つのサポートされたメカニズムと、Data Provider のトレース機能を使用します。

Simple サンプルと Table Viewer サンプルを説明するチュートリアルについては、「[第 2 章 サンプル・アプリケーションの使用](#)」を参照してください。

ただし、デフォルトでは、Adaptive Server ADO.NET Data Provider サンプルを実行するのに必要な pub2 データベースは Adaptive Server にインストールされません。pub2 データベースをインストールする方法については、Adaptive Server Enterprise の『インストール・ガイド』を参照してください。



## サンプル・アプリケーションの使用

この章では、Adaptive Server ADO.NET Data Provider に付属するサンプル・プロジェクトの使用方法について説明します。

トピック名	ページ
<a href="#">チュートリアル：Simple コード・サンプルの使用</a>	11
<a href="#">チュートリアル：Table Viewer コード・サンプルの使用</a>	16
<a href="#">チュートリアル：Advanced コード・サンプルの使用</a>	22

**注意** サンプル・プログラムを実行するには、pubs2 サンプル・データベースがインストールされた Adaptive Server にアクセスする必要があります。また、Visual Studio .NET 2005 または .NET Framework 2.0 がインストールされている必要があります。

サンプル・プログラムは、Adaptive Server ADO.NET Data Provider インストール・ディレクトリの次のディレクトリにあります。

- *Samples\CSharp*。C# プログラミング言語で作成された 3 つのサンプルがあります。
- *Samples\VB.NET*。Visual Basic .NET プログラミング言語で作成された 3 つのサンプルがあります。

デフォルトのインストール・ディレクトリは、Adaptive Server ADO.NET Data Provider の 32 ビット版では *C:\Sybase\DataAccess\ADONET*、Adaptive Server ADO.NET Data Provider の 64 ビット版では *C:\Sybase\DataAccess64\ADONET\dll* です。

### チュートリアル：Simple コード・サンプルの使用

Simple プロジェクトでは、次の機能について説明します。

- データベースへの接続
- *AseCommand* オブジェクトを使用したクエリの実行
- *AseDataReader* オブジェクトの使用
- 基本的なエラー処理

サンプルの動作の詳細については、「[Simple サンプル・プロジェクトの理解](#) (13 ページ) を参照してください。

❖ **Visual Studio .NET での Simple コード・サンプルの実行**

- 1 Visual Studio .NET を起動します。
- 2 [ファイル] - [開く] - [プロジェクト] を選択します。
- 3 サンプル・プロジェクトを指定します。  
  
C# の場合は、<インストール・ディレクトリ>%Samples%CSSharp%Simple を参照して、*Simple.csproj* を開きます。  
  
Visual Basic .NET の場合は、<インストール・ディレクトリ>%Samples%VB.NET%Simple を参照して、*Simple.vbproj* を開きます。
- 4 インストール・プログラムを使用して Adaptive Server ADO.NET Data Provider をインストールしている場合は、手順 7 に進みます。
- 5 インストール・プログラムを使用しなかった場合は、プロジェクトの Adaptive Server ADO.NET Data Provider に対する参照を修正する必要があります。これには、まず既存の参照を削除します。
  - a [ソリューション エクスプローラ] ウィンドウで、Simple プロジェクトが展開されていることを確認します。
  - b [参照設定] フォルダを展開します。
  - c *Sybase.AseClient.Data.dll* を右クリックして、[削除] を選択します。
- 6 Adaptive Server ADO.NET Data Provider アセンブリに対する参照を追加します。  
  
詳細については、「[Data Provider アセンブリへの参照の追加](#) (29 ページ) を参照してください。
- 7 Simple サンプルを実行するには、[デバッグ] - [デバッグなしで開始] を選択するか、[Ctrl] キーを押しながら、[F5] キーを押します。  
  
[AseSample] ダイアログ・ボックスが表示されます。
- 8 [AseSample] ダイアログ・ボックスで、サンプルの pubs2 データベースのある Adaptive Server への接続情報を指定して、[接続] をクリックします。  
  
アプリケーションがサンプルの pubs2 データベースに接続し、ダイアログ・ボックスに各作家の姓が表示されます。
- 9 ウィンドウの右上角にある [X] をクリックすると、アプリケーションが終了し、pubs2 データベースとの接続が切断されます。

これでアプリケーションを実行できました。次の項では、アプリケーション・コードについて説明します。

## ❖ Visual Studio を使用しない Simple サンプル・プロジェクトの実行

- 1 DOS プロンプトを開き、< インストール・ディレクトリ >%Samples にある適切なサンプル・ディレクトリに移動します。
- 2 .NET Framework 2.0 バイナリのあるディレクトリをシステム・パスに追加します。
- 3 Adaptive Server ADO.NET Data Provider インストール・ディレクトリにある dll ディレクトリが、システム・パスと LIB 環境変数に含まれていることを確認します。デフォルトのインストール・ディレクトリは、Adaptive Server ADO.NET Data Provider の 32 ビット版では C:%Sybase%DataAccess%ADONET%dll、Adaptive Server ADO.NET Data Provider の 64 ビット版では C:%Sybase%DataAccess64%ADONET%dll です。
- 4 提供されているビルド・スクリプト build.bat を使用してサンプル・プログラムをコンパイルします。
- 5 プログラムを実行するには、次のように入力します。

```
simple.exe
```

[AseSample] ダイアログ・ボックスが表示されます。

- 6 [AseSample] ダイアログ・ボックスで、サンプルの pubs2 データベースのある Adaptive Server への接続情報を指定して、[ 接続 ] をクリックします。アプリケーションがサンプルの pubs2 データベースに接続し、ダイアログ・ボックスに各作家の姓が表示されます。
- 7 ウィンドウの左上角にある [X] をクリックすると、アプリケーションが終了し、pubs2 データベースとの接続が切断されます。

## Simple サンプル・プロジェクトの理解

この項では、Adaptive Server サンプル・データベース pubs2 を使用する Simple コード・サンプルを利用して、Adaptive Server ADO.NET Data Provider の一部の主要機能について説明します。pub2 データベースをインストールする方法については、Adaptive Server Enterprise の『インストール・ガイド』を参照してください。

この項では、コードの一部について説明します。コード全体を参照する場合は、サンプル・プロジェクトを開いてください。

C# の場合：

```
<インストール・ディレクトリ>%Samples%CSsharp%Simple%Simple.csproj
```

Visual Basic .NET の場合：

```
<インストール・ディレクトリ>%Samples%VB.NET%Simple%Simple.vbproj
```

インポートの宣言

プログラムの始めに、Adaptive Server ADO.NET Data Provider 情報をインポートする `import` 文を宣言します。

C# の場合：

```
using Sybase.Data.AseClient;
```

Visual Basic .NET の場合：

```
Imports Sybase.Data.AseClient
```

データベースへの接続

`btnConnect_Click` メソッドは、`new AseConnection` という接続オブジェクトを宣言して初期化します。

C# の場合：

```
AseConnection conn = new AseConnection(  
    "Data Source='" + host +  
    "';Port='" + port +  
    "';UID='" + user +  
    "';PWD='" + pass +  
    "';Database='pubs2';" );
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _  
    "Data Source='" + host + _  
    "';Port='" + port + _  
    "';UID='" + user + _  
    "';PWD='" + pass + _  
    "';Database='pubs2';")
```

`AseConnection` オブジェクトは、接続文字列を使用してサンプル・データベースに接続します。

C# の場合：

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

`AseConnection` オブジェクトの詳細については、「[AseConnection クラス](#)」(125 ページ)を参照してください。

クエリの実行

次のコードは、`Command` オブジェクト (`AseCommand`) によって SQL 文を定義して実行します。その後、`DataReader` オブジェクト (`AseDataReader`) を返します。

C# の場合：

```
AseCommand cmd = new AseCommand( "select au_lname from  
    authors", conn );  
AseDataReader reader = cmd.ExecuteReader();
```

Visual Basic .NET の場合：

```
Dim cmd As New AseCommand( _  
    "select au_lname from authors", conn)  
Dim reader As AseDataReader = cmd.ExecuteReader()
```

**Command** オブジェクトの詳細については、「[AseCommand クラス](#)」(113 ページ)を参照してください。

結果の表示

次のコードは、**AseDataReader** オブジェクトに保持されているローをループして **ListBox** コントロールに追加します。**DataReader** は、`GetString( 0 )` を使用して、ローの最初の値を取得します。

**Read** メソッドが呼び出されるたびに、**DataReader** は結果セットから別のローを取得して返します。読み込まれたそれぞれのローについて、新しい項目が **ListBox** に追加されます。

C# の場合：

```
listAuthors.BeginUpdate();  
while( reader.Read() ) {  
    listAuthors.Items.Add( reader.GetString( 0 ) );  
}  
listAuthors.EndUpdate();
```

Visual Basic .NET の場合：

```
listAuthors.BeginUpdate()  
While reader.Read()  
    listAuthors.Items.Add(reader.GetString(0))  
End While  
listAuthors.EndUpdate()
```

**AseDataReader** オブジェクトの詳細については、「[AseDataReader クラス](#)」(144 ページ)を参照してください。

接続の終了

メソッドの最後にある次のコードで、読み込みオブジェクトと接続オブジェクトをクローズします。

C# の場合：

```
reader.Close();  
conn.Close();
```

Visual Basic .NET の場合：

```
reader.Close()  
conn.Close()
```

### エラー処理

実行時に発生したエラーや Adaptive Server ADO.NET Data Provider オブジェクトのエラーはすべて、メッセージ・ボックスに表示されます。次のコードは、エラーを検出してメッセージを表示します。

C# の場合：

```
catch( AseException ex ) {  
    MessageBox.Show( ex.Message );  
}
```

Visual Basic .NET の場合：

```
Catch ex As AseException  
    MessageBox.Show(ex.Message)  
End Try
```

AseException オブジェクトの詳細については、「[AseException クラス](#)」(162 ページ) を参照してください。

## チュートリアル：Table Viewer コード・サンプルの使用

このチュートリアルは、Adaptive Server ADO.NET Data Provider に付属している Table Viewer プロジェクトに基づいています。アプリケーション全体は、Adaptive Server ADO.NET Data Provider のインストール・ディレクトリにあります。

C# の場合：

```
<インストール・ディレクトリ  
>%Samples%\CSharp\TableViewer\TableViewer.csproj
```

Visual Basic .NET の場合：

```
<インストール・ディレクトリ  
>%Samples%\VB.NET\TableViewer\TableViewer.vbproj
```

Table Viewer プロジェクトは、Simple プロジェクトより複雑です。このサンプルでは、次の機能について説明します。

- データベースへの接続
- AseDataAdapter オブジェクトの使用
- 高度なエラー処理と結果チェック

サンプルの動作の詳細については、「[Table Viewer サンプル・プロジェクトの理解](#)」(19 ページ) を参照してください。



## ❖ Visual Studio .NET での Table Viewer コード・サンプルの実行

- 1 Visual Studio .NET を起動します。
- 2 [ファイル] - [開く] - [プロジェクト] を選択します。
- 3 Adaptive Server ADO.NET Data Provider インストール・ディレクトリにある *Samples* ディレクトリを指定します。*CSharp* または *VB.NET* ディレクトリに移動して、Table Viewer プロジェクトを開きます。
- 4 インストール・プログラムを使用して Adaptive Server ADO.NET Data Provider をインストールしている場合は、手順 7 に進みます。
- 5 インストール・プログラムを使用しなかった場合は、プロジェクトの Adaptive Server ADO.NET Data Provider に対する参照を修正する必要があります。これには、まず既存の参照を削除します。
  - a [ソリューション エクスプローラ] ウィンドウで、Simple プロジェクトが展開されていることを確認します。
  - b [参照設定] フォルダを展開します。
  - c *Sybase.AseClient.Data.dll* を右クリックして、[削除] を選択します。
- 6 Adaptive Server ADO.NET Data Provider アセンブリに対する参照を追加します。

詳細については、[「Data Provider アセンブリへの参照の追加」\(29 ページ\)](#) を参照してください。
- 7 Table Viewer サンプルを実行するには、[デバッグ] - [デバッグなしで開始] を選択するか、[Ctrl] キーを押しながら、[F5] キーを押します。
- 8 [Table Viewer] ダイアログ・ボックスで、pubs2 サンプル・データベースがインストールされた Adaptive Server への接続情報を指定します。[接続] をクリックします。

アプリケーションが Adaptive Server pubs2 サンプル・データベースに接続します。
- 9 [Table Viewer] ダイアログ・ボックスで [実行] をクリックします。

アプリケーションは、サンプル・データベースの authors テーブルからデータを取得して、クエリの結果を Results DataList に入力します。

このアプリケーションで、別の SQL 文を実行することもできます。[SQL 文] ペインに SQL 文を入力して、[実行] をクリックします。
- 10 ウィンドウの右上角にある [X] をクリックすると、アプリケーションが終了し、サンプル・データベースとの接続が切断されます。

❖ **Visual Studio を使用しない Table Viewer サンプル・プロジェクトの実行**

- 1 DOS プロンプトを開いて、< インストール・ディレクトリ >%Samples にある適切なサンプル・ディレクトリに移動します。
- 2 .NET Framework 2.0 バイナリのあるディレクトリをシステム・パスに追加します。
- 3 Adaptive Server ADO.NET Data Provider インストール・ディレクトリにある *dll* ディレクトリが、システム・パスと LIB 環境変数に含まれていることを確認します。デフォルトのインストール・ディレクトリは、Adaptive Server ADO.NET Data Provider の 32 ビット版では `C:%Sybase%DataAccess%ADONET%dll`、Adaptive Server ADO.NET Data Provider の 64 ビット版では `C:%Sybase%DataAccess64%ADONET%dll` です。
- 4 提供されているビルド・スクリプト *build.bat* を使用してサンプル・プログラムをコンパイルします。
- 5 プログラムを実行するには、次のように入力します。

```
tableviewer.exe
```

- 6 [Table Viewer] ダイアログ・ボックスで、**pubs2** サンプル・データベースがインストールされた Adaptive Server への接続情報を指定します。

[ 接続 ] をクリックします。

アプリケーションが Adaptive Server **pubs2** サンプル・データベースに接続します。

- 7 [Table Viewer] ダイアログ・ボックスで [ 実行 ] をクリックします。

アプリケーションは、サンプル・データベースの **authors** テーブルからデータを取得して、クエリの結果を Results DataList に入力します。

このアプリケーションで、別の SQL 文を実行することもできます。[SQL 文] ペインに SQL 文を入力して、[ 実行 ] をクリックします。

- 8 ウィンドウの右上角にある [X] をクリックすると、アプリケーションが終了し、サンプル・データベースとの接続が切断されます。

これでアプリケーションを実行できました。次の項では、アプリケーション・コードについて説明します。

## Table Viewer サンプル・プロジェクトの理解

この項では、Table Viewer コード・サンプルを利用して、Adaptive Server ADO.NET Data Provider の一部の主要機能について説明します。Table Viewer プロジェクトは、Adaptive Server サンプル・データベース pubs2 を使用します。このデータベースは、Adaptive Server インストール・ディレクトリにあるスクリプトからインストールできます。

この項では、数行ずつコードを説明します。コード全体を参照するには、Adaptive Server インストール・ディレクトリのサンプル・プロジェクトを開きます。

C# の場合：

```
<インストール・ディレクトリ> \Samples\CSharp\TableViewer\
TableViewer.csproj
```

Visual Basic .NET の場合：

```
<インストール・ディレクトリ>\Samples\VB.NET\TableViewer
\TableViewer.vbproj
```

### インポートの宣言

プログラムの始めに、Adaptive Server ADO.NET Data Provider 情報をインポートする `import` 文を宣言します。

C# の場合：

```
using Sybase.Data.AseClient;
```

Visual Basic .NET の場合：

```
Imports Sybase.Data.AseClient
```

### インスタンス変数の宣言

`AseConnection` クラスを使用して、`AseConnection` 型のインスタンス変数を宣言します。この接続は、データベースへの初期接続と、[実行] をクリックしてデータベースから結果セットを取得するときに表示されます。

C# の場合：

```
private AseConnection
    _conn;
```

Visual Basic .NET の場合：

```
Private _conn As AseConnection
```

詳細については、「[AseConnection コンストラクタ](#)」(126 ページ) を参照してください。

### データベースへの接続

次のコードは、`Connection String` フィールドにデフォルトで表示される接続文字列のデフォルト値を設定します。

C# の場合：

```
txtConnectionString.Text = "Data Source=" +
    System.Net.Dns.GetHostName() +
    " ;Port='5000';UID='sa';PWD='';Database='pubs2';";
```

Visual Basic .NET の場合：

```
txtConnectionString.Text = "Data Source='" + _  
    System.Net.Dns.GetHostName() + _  
    "';Port='5000';UID='sa';PWD='';Database='pubs2';"
```

**Connection** オブジェクトは、接続文字列を使用してサンプル・データベースに接続します。

C# の場合：

```
_conn = new AseConnection( txtConnectionString.Text );  
_conn.Open();
```

Visual Basic .NET の場合：

```
_conn = New AseConnection(txtConnectionString.Text)  
_conn.Open()
```

詳細については、「[AseConnection クラス](#)」(125 ページ)を参照してください。

#### クエリの定義

次のコードは、SQL Statement フィールドに表示されるデフォルトのクエリを定義します。

C# の場合：

```
this.txtSQLStatement.Text = "SELECT * FROM authors";
```

Visual Basic .NET の場合：

```
Me.txtSQLStatement.Text = "SELECT * FROM authors"
```

#### 結果の表示

アプリケーションは、**Connection** オブジェクトが初期化されているかどうかを確認してから結果セットをフェッチします。初期化されている場合は、接続ステータスがオープンであることを確認します。

C# の場合：

```
if( _conn == null || _conn.State != ConnectionState.Open )  
{  
    MessageBox.Show( "Connect to a database first.", "Not connected" );  
    return;  
}
```

Visual Basic .NET の場合：

```
If ( _conn Is Nothing) OrElse ( _conn.State <> ConnectionState.Open) Then  
    MessageBox.Show("Connect to a database first.", "Not connected")  
    Return  
End If
```

データベースに接続されると、次のコードは **DataAdapter** オブジェクト (**AseDataAdapter**) を使用して SQL 文を実行します。新しい **DataSet** オブジェクトが作成されて、**DataAdapter** オブジェクトの結果が入力されます。最後に、**DataSet** の内容がウィンドウの **DataGrid** コントロールにバインドされます。

C# の場合：

```
using(AseCommand cmd = new AseCommand( txtSQLStatement.Text.Trim(), _conn ))
{
    using(AseDataAdapter da = new AseDataAdapter(cmd))
    {
        DataSet ds = new DataSet();
        da.Fill(ds, "Table");

        dgResults.DataSource = ds.Tables["Table"];
    }
}
```

Visual Basic .NET の場合：

```
Dim cmd As New AseCommand( _
    txtSQLStatement.Text.Trim(), _conn)
Dim da As New AseDataAdapter(cmd)
Dim ds As New DataSet
da.Fill(ds, "Table")
dgResults.DataSource = ds.Tables("Table")
```

グローバル変数を使用して接続を宣言しているため、SQL 文の実行には以前にオープンした接続が再使用されます。

**DataAdapter** オブジェクトの詳細については、「[AseDataAdapter クラス](#)」(137 ページ)を参照してください。

#### エラー処理

アプリケーションがデータベースへの接続を試行しているときにエラーが発生した場合は、次のコードによってエラーが検出され、メッセージが表示されます。

C# の場合：

```
catch( AseException ex )
{
    MessageBox.Show( ex.Source + " :" + ex.Message +
        "(" + ex.ToString() + ")",
        "Failed to connect" );
}
```

Visual Basic .NET の場合：

```
Catch ex As AseException
    MessageBox.Show(ex.Source + " :" + ex.Message + _
        "(" + ex.ToString() + ")" + _
        "Failed to connect")
End Try
```

## チュートリアル：Advanced コード・サンプルの使用

このチュートリアルは、Adaptive Server ADO.NET Data Provider に付属している Advanced プロジェクトに基づいています。アプリケーション全体は、Adaptive Server ADO.NET Data Provider のインストール・ディレクトリにあります。

C# の場合：

```
<インストール・ディレクトリ  
>%Samples%\CSharp\Advanced\Advanced.csproj
```

Visual Basic .NET の場合：

```
<インストール・ディレクトリ  
>%Samples%\VB.NET\Advanced\Advanced.vbproj
```

Advanced プロジェクトでは、次の機能について説明します。

- データベースへの接続
- Adaptive Server ADO.NET Data Provider に対する ADO.NET の呼び出しをトレースするトレース・イベント機能の使用  

トレース・イベント機能を使用すると、実行した ADO.NET の呼び出しをすべてログに記録して、Sybase 製品の保守契約を結んでいるサポート・センタに送る詳細情報の収集やトラブルシューティングに利用できます。
- 名前付きパラメータ (“@param”) の使用
- 次のようなパラメータ・マーカ (“?”) の使用：{? = call sp\_hello(?, ?, ?)}
- 入力パラメータ、入力／出力パラメータ、出力パラメータ、戻り値を使用したストアド・プロシージャの呼び出し。Adaptive Server では次の 2 つの方法でストアド・プロシージャを呼び出すことができます。
  - CommandText としてストアド・プロシージャ名を使用して、AseCommand.CommandType を CommandType.StoredProcedure に設定する。
  - 呼び出し構文を使用する。この構文は、ODBC および JDBC プログラムと互換性があります。

### ❖ Visual Studio .NET での Advanced コード・サンプルの実行

- 1 Visual Studio .NET を起動します。
- 2 [ファイル] - [開く] - [プロジェクト] を選択します。
- 3 Adaptive Server ADO.NET Data Provider インストール・ディレクトリにある Samples ディレクトリを指定します。CSharp または VB.NET ディレクトリに移動して、Advanced プロジェクトを開きます。

- 4 インストール・プログラムを使用して Adaptive Server ADO.NET Data Provider をインストールしている場合は、手順7に進みます。
  - 5 インストール・プログラムを使用しなかった場合は、プロジェクトの Adaptive Server ADO.NET Data Provider に対する参照を修正する必要があります。これには、まず既存の参照を削除します。
    - a [ソリューション エクスプローラ] ウィンドウで、Simple プロジェクトが展開されていることを確認します。
    - b [参照設定] フォルダを展開します。
    - c *Sybase.AseClient.Data.dll* を右クリックして、[削除] を選択します。
  - 6 Adaptive Server ADO.NET Data Provider アセンブリに対する参照を追加します。
  - 7 [デバッグ] – [デバッグなしで開始] を選択して、Advanced プロジェクトを実行します。

[Form1] ダイアログ・ボックスが表示されます。
  - 8 [Form1] ダイアログ・ボックスで [接続] をクリックします。

アプリケーションが Adaptive Server サンプル・データベースに接続します。
  - 9 [Form1] ダイアログボックスで [実行] をクリックします。

アプリケーションはストアド・プロシージャを実行し、入力/出力パラメータ、出力パラメータ、戻り値を返します。
  - 10 ウィンドウの右上角にある [X] をクリックすると、アプリケーションが終了し、サンプル・データベースとの接続が切断されます。
- これでアプリケーションを実行できました。次の項では、アプリケーションについて説明します。

#### ❖ Visual Studio を使用しない Advanced サンプル・プロジェクトの実行

- 1 DOS プロンプトを開いて、<インストール・ディレクトリ>%Samples ディレクトリにある適切なサンプル・ディレクトリに移動します。
- 2 .NET Framework 2.0 バイナリのあるディレクトリをシステム・パスに追加します。
- 3 Adaptive Server ADO.NET Data Provider インストール・ディレクトリにある *dll* ディレクトリが、システム・パスと LIB 環境変数に含まれていることを確認します。デフォルトのインストール・ディレクトリは、Adaptive Server ADO.NET Data Provider の 32 ビット版では *C:%Sybase%DataAccess%ADONET%dll*、Adaptive Server ADO.NET Data Provider の 64 ビット版では *C:%Sybase%DataAccess64%ADONET%dll* です。
- 4 提供されているビルド・スクリプト *build.bat* を使用してサンプル・プログラムをコンパイルします。

- 5 プログラムを実行するには、次のように入力します。

```
advanced.exe
```

- 6 [Form1] ダイアログ・ボックスが表示されます。[ 接続 ] をクリックします。  
アプリケーションが Adaptive Server サンプル・データベースに接続します。
- 7 [Form1] ダイアログボックスで [ 実行 ] をクリックします。  
アプリケーションはストアド・プロシージャを実行し、入力／出力パラメータ、出力パラメータ、戻り値を返します。
- 8 ウィンドウの右上角にある [X] をクリックすると、アプリケーションが終了し、サンプル・データベースとの接続が切断されます。

これでアプリケーションを実行できました。次の項では、アプリケーション・コードについて説明します。

## Advanced サンプル・プロジェクトの理解

この項では、Advanced コード・サンプルを利用して、Adaptive Server ADO.NET Data Provider の一部の主要機能について説明します。Advanced プロジェクトは、Adaptive Server サンプル・データベース pubs2 を使用します。このデータベースは、Adaptive Server の CD からインストールできます。

この項では、数行ずつコードを説明します。コード全体を参照する場合は、サンプル・プロジェクトを開いてください。

C# の場合：

```
<インストール・ディレクトリ>  
¥Samples¥CSharp¥Advanced¥Advanced.csproj
```

Visual Basic .NET の場合：

```
<インストール・ディレクトリ>  
¥Samples¥VB.NET¥Advanced¥Advanced.vbproj
```

トレース・イベント・  
ハンドラの付加

次のコード行は、トレース・イベント・ハンドラを AseConnection に付加します。

C# の場合：

```
_conn.TraceEnter += new  
    TraceEnterEventHandler(TraceEnter);  
_conn.TraceExit += new  
    TraceExitEventHandler(TraceExit);
```

Visual Basic .NET の場合：

```
AddHandler _conn.TraceEnter, AddressOf TraceEnter  
AddHandler _conn.TraceExit, AddressOf TraceExit
```



名前付きパラメータを使用したストアド・プロシージャの呼び出し

メソッド `ExecuteCommandUsingNamedParams()` は、名前付きパラメータを使用して、名前でストアド・プロシージャを呼び出します。

C# の場合 :

```
using(AseCommand cmd = new AseCommand("sp_hello", _conn))
{
    cmd.CommandType = CommandType.StoredProcedure;

    AseParameter inParam = new AseParameter("@inParam", AseDbType.VarChar, 32);
    inParam.Direction = ParameterDirection.Input;
    inParam.Value = textBoxInput.Text;
    cmd.Parameters.Add(inParam);

    AseParameter inoutParam = new AseParameter("@inoutParam",
    AseDbType.VarChar, 64);
    inoutParam.Direction = ParameterDirection.InputOutput;
    inoutParam.Value = textBoxInOut.Text;
    cmd.Parameters.Add(inoutParam);

    AseParameter outParam = new AseParameter("@outParam",
    AseDbType.VarChar, 64);
    outParam.Direction = ParameterDirection.Output;
    cmd.Parameters.Add(outParam);

    AseParameter retValue = new AseParameter("@retValue", AseDbType.Integer);
    retValue.Direction = ParameterDirection.ReturnValue;
    cmd.Parameters.Add(retValue);

    try
    {
        cmd.ExecuteNonQuery();
    }
    catch (AseException ex)
    {
        MessageBox.Show(ex.Source + " : " + ex.Message + " (" + ex.ToString() +
        ")", "Execute Stored Procedure failed.");
    }
}
```

Visual Basic .NET の場合 :

```
Dim cmd As New AseCommand("sp_hello", _conn)
' set command type to stored procedure
cmd.CommandType = CommandType.StoredProcedure

' create the input parameter object and bind it to the command
Dim inParam As New AseParameter("@inParam", AseDbType.VarChar, 32)
inParam.Direction = ParameterDirection.Input
inParam.Value = textBoxInput.Text
cmd.Parameters.Add(inParam)
```

```

' create the inout parameter object and bind it to the command
Dim inoutParam As New AseParameter("@inoutParam", AseDbType.VarChar, 64)
inoutParam.Direction = ParameterDirection.InputOutput
inoutParam.Value = textBoxInOut.Text
cmd.Parameters.Add(inoutParam)

' create the output parameter object and bind it to the command
Dim outParam As New AseParameter("@outParam", AseDbType.VarChar, 64)
outParam.Direction = ParameterDirection.Output
cmd.Parameters.Add(outParam)

' create the return value object and bind it to the command
Dim retValue As New AseParameter("@retValue", AseDbType.Integer)
retValue.Direction = ParameterDirection.ReturnValue
cmd.Parameters.Add(retValue)

' execute the stored procedure
Try
    cmd.ExecuteNonQuery()
Catch ex As AseException
    MessageBox.Show(ex.Source + " :" + ex.Message + " (" + ex.ToString() + ")",
        "Execute Query failed.")
Finally
    ' dispose the command object
    cmd.Dispose()
End Try

```

呼び出し構文とパラメータ・マークを使用したストアド・プロシージャの呼び出し

メソッド `ExecuteCommandUsingParameterMarkers()` は、呼び出し構文とパラメータ・マークを使用してストアド・プロシージャを呼び出します。

C# の場合：

```

using(AseCommand cmd = new AseCommand("{ ?= call sp_hello(?, ?, ?)}", _conn))
{
    cmd.NamedParameters = false;
    AseParameter retValue = new AseParameter(0, AseDbType.Integer);
    retValue.Direction = ParameterDirection.ReturnValue;
    cmd.Parameters.Add(retValue);

    AseParameter inParam = new AseParameter(1, AseDbType.VarChar, 32);
    inParam.Direction = ParameterDirection.Input;
    inParam.Value = textBoxInput.Text;
    cmd.Parameters.Add(inParam);

    AseParameter inoutParam = new AseParameter(2, AseDbType.VarChar, 64);
    inoutParam.Direction = ParameterDirection.InputOutput;
    inoutParam.Value = textBoxInOut.Text;
    cmd.Parameters.Add(inoutParam);

    AseParameter outParam = new AseParameter(3, AseDbType.VarChar, 64);
    outParam.Direction = ParameterDirection.Output;

```

```
cmd.Parameters.Add(outParam);
try
{
    cmd.ExecuteNonQuery();
}
catch (AseException ex)
{
    MessageBox.Show(ex.Source + " :" + ex.Message + " (" + ex.ToString() +
        ")", "Execute Stored Procedure failed.");
}
}
```

### Visual Basic .NET の場合 :

```
Dim cmd As New AseCommand("{ ?= call sp_hello(?, ?, ?)}", _conn)
' need to notify Named Parameters are not being used (which is the default)
cmd.NamedParameters = False

' create the return value object and bind it to the command
Dim retValue As New AseParameter(0, AseDbType.Integer)
retValue.Direction = ParameterDirection.ReturnValue
cmd.Parameters.Add(retValue)

' create the input parameter object and bind it to the command
Dim inParam As New AseParameter(1, AseDbType.VarChar, 32)
inParam.Direction = ParameterDirection.Input
inParam.Value = textBoxInput.Text
cmd.Parameters.Add(inParam)

' create the inout parameter object and bind it to the command
Dim inoutParam As New AseParameter(2, AseDbType.VarChar, 64)
inoutParam.Direction = ParameterDirection.InputOutput
inoutParam.Value = textBoxInOut.Text
cmd.Parameters.Add(inoutParam)

' create the output parameter object and bind it to the command
Dim outParam As New AseParameter(3, AseDbType.VarChar, 64)
outParam.Direction = ParameterDirection.Output
cmd.Parameters.Add(outParam)

' execute the stored procedure
Try
    cmd.ExecuteNonQuery()

    ' get the output, inout and return values and display them
    textBoxReturn.Text = cmd.Parameters(0).Value
    textBoxReturn.ForeColor = Color.Blue

    textBoxInOut.Text = cmd.Parameters(2).V
```

```
        textBoxOutput.Text = cmd.Parameters(3).Value
        textBoxOutput.ForeColor = Color.Blue
Catch ex As AseException
    MessageBox.Show(ex.Source + " :" + ex.Message + " (" + ex.ToString() + ")", _
        "Execute Query Failed")
Finally
    ' dispose the command object
    cmd.Dispose()
End Try
```

この章では、Adaptive Server ADO.NET Data Provider を使用したアプリケーションの開発と配備の方法について説明します。

トピック名	ページ
<a href="#">Visual Studio .NET プロジェクトでの Data Provider の使用</a>	29
<a href="#">データベースへの接続</a>	31
<a href="#">データに対するアクセスと操作</a>	36
<a href="#">ストアド・プロシージャの使用</a>	72
<a href="#">トランザクション処理</a>	75
<a href="#">エラー処理</a>	78
<a href="#">パフォーマンスの考慮事項</a>	79

## Visual Studio .NET プロジェクトでの Data Provider の使用

Adaptive Server ADO.NET Data Provider をインストールしたら、Visual Studio .NET プロジェクトに次の変更を加えて、使用できるようにします。

- Adaptive Server ADO.NET Data Provider アセンブリに対する参照を追加する。
- ソース・コードに Adaptive Server ADO.NET Data Provider クラスを参照する行を追加する。

Adaptive Server ADO.NET Data Provider のインストールと登録の詳細については、「[Adaptive Server ADO.NET Data Provider の配備](#)」(2 ページ) を参照してください。

### Data Provider アセンブリへの参照の追加

参照を追加して、Adaptive Server ADO.NET Data Provider のコードを検索するために含めるアセンブリを Visual Studio .NET に指示します。

❖ **Visual Studio .NET プロジェクトでの Adaptive Server ADO.NET Data Provider に対する参照の追加**

- 1 Visual Studio .NET を起動して、プロジェクトを開きます。
- 2 [ソリューション エクスプローラ] ウィンドウで、[参照設定] フォルダを右クリックし、ポップアップ・メニューから、[参照の追加] を選択します。  
[参照の追加] ダイアログ・ボックスが表示されます。
- 3 [.NET] タブで、**Sybase.AdoNet2.AseClient** コンポーネントが見つかるまでコンポーネントの一覧をスクロールします。このコンポーネントを指定して、[選択] をクリックします。
- 4 [OK] をクリックします。

コンポーネントの一覧に Adaptive Server ADO.NET Data Provider アセンブリがない場合は、[参照] から <インストール・ディレクトリ>¥dll ディレクトリにある *Sybase.AdoNet2.AseClient.dll* を探します。この DLL を選択して [開く] をクリックします。次に、[OK] をクリックします。

---

**注意** デフォルトのロケーションは、Adaptive Server ADO.NET Data Provider の 32 ビット版では *C:¥Sybase¥DataAccess¥ADONET¥dll*、Adaptive Server ADO.NET Data Provider の 64 ビット版では *C:¥Sybase¥DataAccess64¥ADONET¥dll* です。

---

プロジェクトの [ソリューション エクスプローラ] ウィンドウの [参照設定] フォルダにアセンブリが追加されます。

## Adaptive Server ADO.NET Data Provider クラスの参照

Adaptive Server ADO.NET Data Provider を使用するには、Adaptive Server ADO.NET Data Provider を参照する行もソース・コードに追加します。C# と Visual Basic .NET では追加する行が異なります。

❖ **ソース・コードでの Adaptive Server ADO.NET Data Provider クラスの参照**

- 1 Visual Studio .NET を起動して、プロジェクトを開きます。
  - C# の場合は、プロジェクトの先頭にある using ディレクティブの一覧に次の行を追加します。

```
using Sybase.Data.AseClient;
```

- Visual Basic .NET の場合は、プロジェクトの先頭にある行 `Public Class Form1` の前に次の行を追加します。

```
Imports Sybase.Data.AseClient
```

この行は必須ではありません。ただし、これによって Adaptive Server クラスの省略形を使用できるようになります。このコード行がない場合でも、次のコード行を使用できます。

```
Sybase.Data.AseClient.AseConnection conn = new  
Sybase.Data.AseClient.AseConnection();
```

上記のコード行を次のコード行の代わりに使用します。

```
AseConnection conn = new AseConnection();
```

## データベースへの接続

データに対して操作を実行するには、最初にアプリケーションをデータベースに接続します。この項では、Adaptive Server データベースに接続するコードの記述方法について説明します。

詳細については、「[AseConnection クラス](#)」(125 ページ)と「[ConnectionString プロパティ](#)」(132 ページ)を参照してください。

### ❖ Adaptive Server データベースへの接続

#### 1 AseConnection オブジェクトを割り付けます。

次のコードは、“conn” という名前の AseConnection オブジェクトを作成します。

C# の場合：

```
AseConnection conn = new AseConnection();
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection()
```

アプリケーションからデータベースへ複数の接続を設定できます。アプリケーションによっては、Adaptive Server データベースに対する接続を1つだけ使用して、常時この接続をオープンにします。これを実行するには、接続にグローバル変数を宣言します。

C# の場合：

```
private AseConnection_conn;
```

Visual Basic .NET の場合：

```
Private _conn As AseConnection
```

詳細については、<インストール・ディレクトリ>%Samples にある Table Viewer のサンプル・コードと「[Table Viewer サンプル・プロジェクトの理解](#)」(19 ページ)を参照してください。

- 2 データベースへの接続に使用する接続文字列を指定します。

C# の場合：

```
AseConnection conn = new AseConnection(  
    "Data Source='mango';Port=5000;" +  
    "UID='sa';PWD='';" +  
    "Database='pubs2';" );
```

“mango” には、データベース・サーバが実行されているホスト名を指定します。

Visual Basic .NET の場合：

```
Dim conn As New AseConnection(_  
    "Data Source='mango',Port=5000," +_  
    "UID='sa';PWD='';" + _  
    "Database='pubs2';")
```

接続パラメータの完全なリストについては、「[AseConnection コンストラクタ](#)」(126 ページ)を参照してください。

- 3 次のコードを使用して、データベースへの接続をオープンします。

C# の場合：

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

- 4 接続エラーを検出します。

データベースへの接続試行時に発生したすべてのエラーが検出されるようにアプリケーションを設計してください。次のコードは、エラーを検出し、そのメッセージを表示する方法を示しています。

C# の場合：

```
try {  
    _conn = new AseConnection(  
        txtConnectionString.Text );  
    _conn.Open();  
}  
catch( AseException ex ) {  
    MessageBox.Show(  
        ex.Message,  
        "Failed to connect");  
}
```



Visual Basic .NET の場合：

```
Try
    _conn = New AseConnection(_
        txtConnectionString.Text)
    _conn.Open()
Catch ex As AseException
    MessageBox.Show(_
        ex.Message, _
        "Failed to connect")
End Try
```

**AseConnection** オブジェクトの作成時に接続文字列を渡すのではなく、**ConnectionString** プロパティを使用して接続文字列を設定することもできます。

C# の場合：

```
AseConnection conn = new AseConnection();
conn.ConnectionString = "Data Source='mango';" +
    "Port=5000;" +
    "UID='sa';" +
    "PWD='';" +
    "Database='pubs2';" ;
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection()
conn.ConnectionString = "Data Source='mango';" + _
    "Port=5000;" + _
    "UID='sa';" + _
    "PWD='';" + _
    "Database='pubs2';"
```

“mango”には、データベース・サーバ名を指定します。

- 5 データベースへの接続をクローズします。**conn.Close()** メソッドを使用して明示的にクローズするまで、データベースへの接続はオープンしたままになります。

## 接続プール

Adaptive Server Enterprise ADO.NET プロバイダは、アプリケーションがプールからの既存の接続を再使用できる接続プールをサポートします。これにより、データベースへの新しい接続を繰り返し作成する代わりに、接続ハンドルをプールに保存し、接続を再使用できるようになります。接続プールは、デフォルトでオンに設定されています。

プール・サイズの最小値と最大値を指定することもできます。次に例を示します。

```
"Data Source='mango';" +
"Port=5000;" +
"UID='sa';" +
"PWD='';" +
"Database='pubs2';" +
"Max Pool Size=50;" +
"Min Pool Size=5";
```

アプリケーションがデータベースに初めて接続しようとするとき、アプリケーションは、指定した接続パラメータと同じパラメータを使用している既存の接続がプールにあるかどうかをチェックします。一致する接続があった場合は、その接続を使用します。一致する接続がなかった場合は、新しい接続を使用します。接続が切断されると、その接続はプールに戻されて再利用できるようになります。

---

**注意** Max Pool Size が指定されている場合は、オープンできる接続の最大数がこの値に制限されます。制限値に達すると、`AseConnection.Open()` の呼び出しが失敗して `AseException` が発生します。

---

### 接続プールの無効化

接続プールを無効にするには、接続文字列で `Pooling=False` を指定します。

## 接続ステータスの確認

データベースへの接続が確立した後は、接続ステータスをチェックして接続がオープンになっていることを確認してから、データをフェッチして更新できます。接続が失われたり、ビジー状態だったり、別のコマンドを処理中の場合は、それに相当するメッセージを返すことができます。

`AseConnection` クラスには、接続ステータスをチェックする「State プロパティ」があります。使用可能なステータス値は `Open` と `Closed` です。

次のコードは、`Connection` オブジェクトが初期化されているかどうかをチェックし、初期化されている場合は、接続がオープンになっていることを確認します。

C# の場合：

```
if( _conn == null || _conn.State !=
    ConnectionState.Open )
{
    MessageBox.Show( "Connect to a database first",
        "Not connected" );
    return;
}
```

Visual Basic .NET の場合：

```
If (_conn Is Nothing) OrElse (_conn.State <>
ConnectionState.Open) Then
    MessageBox.Show("Connection to a database first",
    "Error")
    Return
End If
```

接続がオープンになっていない場合は、メッセージが返されます。「[State プロパティ](#)」(134 ページ) を参照してください。

## 文字セット

Adaptive Server ADO.NET Data Provider は、ネゴシエートされた文字セットを使用して Adaptive Server と通信します。この文字セットは、Adaptive Server ADO.NET Data Provider のユーザ・インタフェースで ClientCharset または Server Default に設定できます。Server Default がデフォルト設定です。

### 使用法

- ClientCharset を文字セットとして選択する場合は、CodePageType プロパティの値も指定します。有効な値は、ANSI (デフォルト値) および OEM です。
- Other を選択する場合、charset プロパティの値には文字セットの名前を指定します。たとえば、“charset=utf8” という指定では、ネゴシエートされた文字セットは utf8 に設定されます。
- charset プロパティは、できるだけ使用しないでください。.NET 文字列は unicode であり、マルチバイトに変換してから Adaptive Server に char または varchar パラメータとして送信する必要があります。このため、サーバのデフォルト以外の文字セットを使用した場合、パフォーマンスに影響します。
- サポートされていない文字セットを使用すると、次のようなエラーが発生します。

```
[Sybase][driver] Could not load code page for requested
charset
```

このエラーを回避するには、次のいずれかの手順を実行します。

- ドライバのユーザ・インタフェースで、[ 詳細設定 ] タブを開き、ClientCharset を文字セットとして選択します。Other を選択した場合、指定した文字セットがサポートされることを確認してください。
- 接続文字列では、charset プロパティで、サポートされている文字セットが指定されていることを確認してください。

## データに対するアクセスと操作

Adaptive Server ADO.NET Data Provider のデータ・アクセス方法には、**AseCommand** オブジェクトを使用する方法と **AseDataAdapter** オブジェクトを使用する方法の 2 つがあります。

- **AseCommand オブジェクト**： **AseCommand** オブジェクトを使用する方法は、プログラマが接続をより制御しやすくなるため、.NET でデータ・アクセスとデータ操作を行う場合におすすめします。ただし、**AseDataAdapter** を使用すればオフラインでの作業が可能になります。

**AseCommand** オブジェクトを使用すると、SQL 文を実行してデータベースのデータを直接取得したり修正したりできます。また **AseCommand** オブジェクトを使用すると、データベースに対して直接 SQL 文を発行してストアド・プロシージャを呼び出すことができます。

**AseCommand** オブジェクト内では、**AseDataReader** クラスを使用して、クエリまたはストアド・プロシージャから読み込み専用の結果セットを返すことができます。

詳細については、「[AseCommand クラス](#)」(113 ページ)と「[AseDataReader クラス](#)」(144 ページ)を参照してください。

- **AseDataAdapter オブジェクト**： **AseDataAdapter** オブジェクトは、結果セット全体を **DataSet** に取得します。**DataSet** は接続が切断された記憶領域で、データベースから取得されたデータが格納されます。**DataSet** に格納されたデータは編集できます。編集操作が終了すると、**AseDataAdapter** オブジェクトは、**DataSet** に対して行われた変更でデータベースを更新します。**AseDataAdapter** を使用する場合、**DataSet** に取得したローを他のユーザが変更しないように防ぐ方法はありません。そのため、発生する可能性のある競合すべてを解決するロジックをアプリケーションに組み込む必要があります。

詳細については、「[AseDataAdapter を使用する場合の競合の解決](#)」(51 ページ)を参照してください。

**AseDataAdapter** オブジェクトの詳細については、「[AseDataAdapter クラス](#)」(137 ページ)を参照してください。

### AseCommand を使用したデータの取得と操作

以降の各項では、**AseDataReader** を使用したデータの取得と、ローの挿入、更新、削除の方法について説明します。

## AseCommand オブジェクトを使用したデータの取得

AseCommand オブジェクトを使用すると、Adaptive Server データベースに対して SQL 文を発行したり、ストアド・プロシージャを呼び出したりできます。データベースからデータを取得するには、次の種類のコマンドを発行します。

- **ExecuteReader** 単一の結果セットを返すコマンドを発行します。デフォルトではカーソルは使用されません。結果セット全体がクライアント側でフェッチされ、ユーザは1度に1つのローを前方向へのみフェッチできます。カーソルの使用をオンにするには、次の行を **ConnectionString** に追加します。

```
"Use Cursor=true;"
```

これにより、データベース・サーバから結果セット全体をフェッチする代わりに、前方向への読み込み専用カーソルが使用されるようになります。

カーソルを使用すると、クエリによって大規模な **resultset** が返されることが予測される場合にパフォーマンスを改善できますが、クライアントは必然的に **resultset** 全体を使用できなくなります。

いずれの場合も、ユーザは一方方向にのみ結果セットのローをすばやくループできます。

詳細については、「[ExecuteReader メソッド](#)」(116 ページ)を参照してください。

- **ExecuteScalar** 単一の値を返すコマンドを発行します。結果セットの最初のローの最初のカラムや、COUNT や AVG など、集計値を返す SQL 文が返される場合もあります。

詳細については、「[ExecuteScalar メソッド](#)」(117 ページ)を参照してください。

- **ExecuteXmlReader** 単一の結果セットを XML フォーマットで返すコマンドを発行します。通常、このメソッドは FOR XML 句のある **select** 文で使用されます。

詳細については、「[ExecuteXmlReader メソッド](#)」(117 ページ)を参照。

以降の説明では、Adaptive Server ADO.NET Data Provider に含まれている Simple コード・サンプルを使用します。

Simple コード・サンプルの詳細については、「[Simple サンプル・プロジェクトの理解](#)」(13 ページ)を参照してください。

❖ **結果セット全体を返すコマンドの発行**

- 1 **Connection** オブジェクトを宣言して初期化します。

C# の場合：

```
AseConnection conn = new AseConnection(connStr);
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection(connStr)
```

C# の場合：

```
try {  
    conn.Open();  
}  
catch (AseException ex)  
{  
    <error handling>  
}
```

Visual Basic .NET の場合：

```
Try  
    conn.Open()  
Catch ex As AseException  
    <error handling>  
End Try
```

- 2 **SQL** 文を定義して実行する **Command** オブジェクトを追加します。

C# の場合：

```
AseCommand cmd = new AseCommand(_  
    "select au_lname from authors", conn );
```

Visual Basic .NET の場合：

```
Dim cmd As New AseCommand("select au_lname from authors",  
    conn)
```

---

**注意** ストアド・プロシージャを使用してデータベースからデータを取得する場合、ストアド・プロシージャから出力パラメータ値と結果セットの両方が返されると、結果セットはリセットされ、出力パラメータ値が参照されると同時に結果セットのローは参照できなくなります。Sybase では、このような場合、結果セットのローすべてを参照して使い終わるまで、参照側の出力パラメータ値を最後までそのままにしておくことをおすすめします。

---

詳細については、「[ストアド・プロシージャの使用](#)」(72 ページ) と「[AseParameter クラス](#)」(165 ページ) を参照してください。

- 3 **ExecuteReader** メソッドを呼び出して、**DataReader** オブジェクトを返します。

C# の場合 :

```
AseDataReader reader = cmd.ExecuteReader();
```

Visual Basic .NET の場合 :

```
Dim reader as AseDataReader = cmd.ExecuteReader()
```

- 4 結果を表示します。

C# の場合 :

```
listAuthors.BeginUpdate();
while( reader.Read() ) {
    listAuthors.Items.Add( reader.GetString( 0 ) );
}
listAuthors.EndUpdate();
```

Visual Basic .NET の場合 :

```
listAuthors.BeginUpdate()
While reader.Read()
    listAuthors.Items.Add(reader.GetString(0))
End While
listAuthors.EndUpdate()
```

- 5 **DataReader** オブジェクトと **Connection** オブジェクトをクローズします。

C# の場合 :

```
reader.Close();
conn.Close();
```

Visual Basic .NET の場合 :

```
reader.close()
conn.close()
```

#### ❖ 単一の値のみを返すコマンドの発行

- 1 **AseConnection** オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection(
    "Data Source='mango';" +
    "Port=5000;" +
    "UID='sa';" +
    "PWD='';" +
    "Database='pubs2';" );
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _  
    "Data Source='mango';" + _  
    "Port=5000;" + _  
    "UID='sa';" + _  
    "PWD='';" + _  
    "Database='pubs2';")
```

“mango”には、データベース・サーバ名を指定します。

- 2 接続をオープンします。

C# の場合：

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

- 3 SQL 文を定義して実行する **AseCommand** オブジェクトを追加します。

C# の場合：

```
AseCommand cmd = new AseCommand(  
    "select count(*) from authors where state = 'CA'",  
    conn );
```

Visual Basic .NET の場合：

```
Dim cmd As New AseCommand(  
    "select count(*) from authors where state = 'CA'",  
    conn );
```

- 4 **ExecuteScalar** メソッドを呼び出して、値を含むオブジェクトを返します。

C# の場合：

```
int count = (int) cmd.ExecuteScalar();
```

Visual Basic .NET の場合：

```
Dim count As Integer = cmd.ExecuteScalar()
```

- 5 **AseConnection** オブジェクトをクローズします。

C# の場合：

```
conn.Close();
```

Visual Basic .NET の場合：

```
conn.Close()
```

**AseDataReader** には、任意のデータ型で結果を返すことのできる **Get** メソッドがいくつかあります。

詳細については、「[AseDataReader クラス](#)」(144 ページ)を参照してください。



❖ **XmlReader オブジェクトを返すコマンドの発行**

- 1 Connection オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection(connStr);
```

Visual Basic .NET の場合 :

```
Dim conn As New AseConnection(connStr)
```

- 2 接続をオープンします。

C# の場合 :

```
try {  
    conn.Open();  
}  
catch (AseException ex)  
{  
    <error handling>  
}
```

Visual Basic .NET の場合 :

```
Try  
    conn.Open()  
Catch ex As AseException  
    <error handling>  
End Try
```

- 3 SQL 文を定義して実行する Command オブジェクトを追加します。

C# の場合 :

```
AseCommand cmd = new AseCommand(  
    "select * from authors for xml",  
    conn );
```

Visual Basic .NET の場合 :

```
Dim cmd As New AseCommand( _  
    "select au_lname from authors for xml", _  
    conn
```

- 4 ExecuteReader メソッドを呼び出して、DataReader オブジェクトを返します。

C# の場合 :

```
XmlReader reader = cmd.ExecuteReader();
```

Visual Basic .NET の場合 :

```
Dim reader as XmlReader = cmd.ExecuteReader()
```

- XML の結果を使用します。

C# の場合 :

```
reader.read();  
<process xml>
```

Visual Basic .NET の場合 :

```
reader.read()  
<process xml>
```

- DataReader オブジェクトと Connection オブジェクトをクローズします。

C# の場合 :

```
reader.Close();  
conn.Close();
```

Visual Basic .NET の場合 :

```
reader.close()  
conn.close()
```

## AseCommand オブジェクトを使用したローの挿入、更新、削除

AseCommand オブジェクトを使用して、Insert、Update、または Delete 操作を実行するには、ExecuteNonQuery 関数を使用します。ExecuteNonQuery 関数は、結果セットを返さないコマンド (SQL 文またはストアド・プロシージャ) を発行します。

詳細については、「[ExecuteNonQuery メソッド](#)」(116 ページ) を参照してください。

自動インクリメント・プライマリ・キーのプライマリ・キー値を取得する方法については、「[プライマリ・キー値の取得](#)」(62 ページ) を参照してください。

コマンドの独立性レベルを設定する場合は、AseCommand オブジェクトを AseTransaction オブジェクトの一部として使用してください。AseTransaction オブジェクトを使用せずにデータを修正すると、Adaptive Server ADO.NET Data Provider は autocommit モードで動作し、加えられたすべての変更がただちに適用されます。

詳細については、「[トランザクション処理](#)」(75 ページ) を参照してください。

### ❖ ローを挿入するコマンドの発行

- AseConnection オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection( c_connStr );
```

Visual Basic .NET の場合 :

```
Dim conn As New AseConnection(c_connStr)
```

- 2 接続をオープンします。

C# の場合 :

```
conn.Open();
```

Visual Basic .NET の場合 :

```
conn.Open()
```

- 3 Insert 文を定義して実行する **AseCommand** オブジェクトを追加します。

C# の場合 :

```
AseCommand insertCmd = new AseCommand(
    "INSERT INTO publishers " +
    " ( pub_id, pub_name, city, state) " +
    " VALUES( @pub_id, @pub_name, @city, @state )",
    conn);
```

Visual Basic .NET の場合 :

```
Dim insertCmd As new AseCommand( _
    "INSERT INTO publishers " + _
    " ( pub_id, pub_name, city, state) " + _
    " VALUES ( @pub_id, @pub_name, @city, @state )", _
    conn )
```

- 4 **AseCommand** オブジェクトのパラメータを設定します。

次のコードは、それぞれ **dept\_id** カラムと **dept\_name** カラムのパラメータを定義します。

C# の場合 :

```
AseParameter parm = new AseParameter("@pub_id", AseDbType.Char, 4);
insertCmd.Parameters.Add( parm );
parm = new AseParameter("@pub_name", AseDbType.VarChar, 40);
insertCmd.Parameters.Add( parm );
parm = new AseParameter("@city", AseDbType.VarChar, 20);
insertCmd.Parameters.Add( parm );
parm = new AseParameter("@state", AseDbType.Char, 2);
insertCmd.Parameters.Add( parm );
```

Visual Basic .NET の場合 :

```
Dim parm As New AseParameter("@pub_id", AseDbType.Char, 4)
insertCmd.Parameters.Add(parm)
parm = New AseParameter("@pub_name", AseDbType.VarChar, 40)
insertCmd.Parameters.Add(parm)
parm = New AseParameter("@city", AseDbType.VarChar, 20)
insertCmd.Parameters.Add(parm)
parm = New AseParameter("@state", AseDbType.Char, 2)
insertCmd.Parameters.Add(parm)
```

- 5 新しい値を挿入して `ExecuteNonQuery` メソッドを呼び出し、データベースに変更を適用します。

C# の場合 :

```
int recordsAffected = 0;
insertCmd.Parameters[0].Value = "9901";
insertCmd.Parameters[1].Value = "New Publisher";
insertCmd.Parameters[2].Value = "Concord";
insertCmd.Parameters[3].Value = "MA";
recordsAffected = insertCmd.ExecuteNonQuery();
insertCmd.Parameters[0].Value = "9902";
insertCmd.Parameters[1].Value = "My Publisher";
insertCmd.Parameters[2].Value = "Dublin";
insertCmd.Parameters[3].Value = "CA";
recordsAffected = insertCmd.ExecuteNonQuery();
```

Visual Basic .NET の場合 :

```
Dim recordsAffected As Integer
insertCmd.Parameters(0).Value = "9901"
insertCmd.Parameters(1).Value = "New Publisher"
insertCmd.Parameters(2).Value = "Concord"
insertCmd.Parameters(3).Value = "MA"
recordsAffected = insertCmd.ExecuteNonQuery()
insertCmd.Parameters(0).Value = "9902"
insertCmd.Parameters(1).Value = "My Publisher"
insertCmd.Parameters(2).Value = "Dublin"
insertCmd.Parameters(3).Value = "CA"
recordsAffected = insertCmd.ExecuteNonQuery()
```

---

**注意** `Insert`、`Update`、`Delete` 文は、`ExecuteNonQuery` メソッドとともに使用できます。

---

- 6 結果を表示して、ウィンドウのグリッドにバインドします。

C# の場合 :

```
AseCommand selectCmd = new AseCommand("SELECT * FROM publishers", conn );
AseDataReader dr = selectCmd.ExecuteReader();
dataGridView.DataSource = dr;
```

Visual Basic .NET の場合 :

```
Dim selectCmd As New AseCommand("SELECT * FROM publishers", conn)
Dim dr As AseDataReader = selectCmd.ExecuteReader()
DataGridView.DataSource = dr
```

- 7 AseDataReader オブジェクトと AseConnection オブジェクトをクローズします。

C# の場合 :

```
dr.Close();
conn.Close();
```

Visual Basic .NET の場合 :

```
dr.Close()
conn.Close()
```

#### ❖ ローを更新するコマンドの発行

- 1 AseConnection オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection(
    c_connStr );
```

Visual Basic .NET の場合 :

```
Dim conn As New AseConnection(c_connStr)
```

- 2 接続をオープンします。

C# の場合 :

```
conn.Open();
```

Visual Basic .NET の場合 :

```
conn.Open()
```

- 3 Update 文を定義して実行する AseCommand オブジェクトを追加します。

C# の場合 :

```
AseCommand updateCmd = new AseCommand(
    "UPDATE publishers " +
    "SET pub_name = 'My Publisher' " +
    "WHERE pub_id='9901'",
    conn );
```

Visual Basic .NET の場合 :

```
Dim updateCmd As New AseCommand( _
    "UPDATE publishers " + _
    "SET pub_name = 'My Publisher' " + _
    "WHERE pub_id='9901'", _
    conn )
```

詳細については、「ストアド・プロシージャの使用」(72 ページ) と「AseParameter クラス」(165 ページ) を参照してください。

- 4 **ExecuteNonQuery** メソッドを呼び出して、データベースに変更を適用します。

C# の場合 :

```
int recordsAffected = updateCmd.ExecuteNonQuery();
```

Visual Basic .NET の場合 :

```
Dim recordsAffected As Integer =  
updateCmd.ExecuteNonQuery()
```

- 5 結果を表示して、ウィンドウのグリッドにバインドします。

C# の場合 :

```
AseCommand selectCmd = new AseCommand(  
"SELECT * FROM publishers", conn);  
AseDataReader dr = selectCmd.ExecuteReader();  
dataGridView.DataSource = dr;
```

Visual Basic .NET の場合 :

```
Dim selectCmd As New AseCommand(  
"SELECT * FROM publishers", conn)  
Dim dr As AseDataReader = selectCmd.ExecuteReader()  
dataGridView.DataSource = dr
```

- 6 **AseDataReader** オブジェクトと **AseConnection** オブジェクトをクローズします。

C# の場合 :

```
dr.Close();  
conn.Close();
```

Visual Basic .NET の場合 :

```
dr.Close()  
conn.Close()
```

#### ❖ ローを削除するコマンドの発行

- 1 **AseConnection** オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection(c_connStr);
```

Visual Basic .NET の場合 :

```
Dim conn As New AseConnection(c_connStr)
```

- 2 接続をオープンします。

C# の場合 :

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

- 3 Delete 文を定義して実行する `AseCommand` オブジェクトを作成します。

C# の場合：

```
AseCommand updateCmd = new AseCommand  
"DELETE FROM publishers " +  
" WHERE (pub_id > '9900')",  
conn );
```

Visual Basic .NET の場合：

```
Dim updateCmd As New AseCommand(_  
"DELETE FROM publishers " + _  
"WHERE (pub_id > '9900')", _  
conn )
```

- 4 `ExecuteNonQuery` メソッドを呼び出して、データベースに変更を適用します。

C# の場合：

```
int recordsAffected = deleteCmd.ExecuteNonQuery();
```

Visual Basic .NET の場合：

```
Dim recordsAffected As Integer =  
updateCmd.ExecuteNonQuery()
```

- 5 `AseConnection` オブジェクトをクローズします。

C# の場合：

```
conn.Close();
```

Visual Basic .NET の場合：

```
dr.Close()  
conn.Close()
```

## DataReader スキーマ情報の取得

結果セット内のカラムに関するスキーマ情報を取得できます。

`AseDataReader` を使用している場合、`GetSchemaTable` メソッドを使用して結果セットの情報を取得できます。`GetSchemaTable` メソッドは、標準の .NET `DataTable` オブジェクトを返します。このオブジェクトは、カラム・プロパティも含め、結果セット内のすべてのカラムに関する情報を提供します。

`GetSchemaTable` メソッドの詳細については、「[GetSchemaTable メソッド](#)」(152 ページ)を参照してください。

❖ **GetSchemaTable** メソッドを使用した結果セット情報の取得

- 1 Connection オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection(  
    c_connStr );
```

Visual Basic .NET の場合 :

```
Dim conn As New AseConnection( _  
    c_connStr )
```

- 2 接続をオープンします。

C# の場合 :

```
conn.Open();
```

Visual Basic .NET の場合 :

```
conn.Open()
```

- 3 目的の **Select** 文を使用して **AseCommand** オブジェクトを作成します。このクエリの結果セットのスキーマが返されます。

C# の場合 :

```
AseCommand cmd = new AseCommand(  
    "SELECT * FROM authors", conn );
```

Visual Basic .NET の場合 :

```
Dim cmd As New AseCommand( _  
    "SELECT * FROM authors", conn )
```

- 4 **AseDataReader** オブジェクトを作成して、作成した **Command** オブジェクトを実行します。

C# の場合 :

```
AseDataReader dr = cmd.ExecuteReader();
```

Visual Basic .NET の場合 :

```
Dim dr As AseDataReader = cmd.ExecuteReader()
```

- 5 データ・ソースからのスキーマを **DataTable** に書き込みます。

C# の場合 :

```
DataTable  
schema = dr.GetSchemaTable();
```

Visual Basic .NET の場合 :

```
Dim schema As DataTable = _  
dr.GetSchemaTable()
```



- 6 **AseDataReader** オブジェクトと **AseConnection** オブジェクトをクローズします。

C# の場合 :

```
dr.Close();  
conn.Close();
```

Visual Basic .NET の場合 :

```
dr.Close()  
conn.Close()
```

- 7 **DataTable** をウィンドウのグリッドにバインドします。

C# の場合 :

```
dataGrid.DataSource = schema;
```

Visual Basic .NET の場合 :

```
dataGrid.DataSource = schema
```

## AseDataAdapter を使用したデータへのアクセスと操作

以降の各項では、**AseDataAdapter** を使用したデータの取得と、ローの挿入、更新、削除の方法について説明します。

### AseDataAdapter オブジェクトを使用したデータの取得

**AseDataAdapter** は、Fill メソッドを使用してクエリの結果を **DataSet** に書き込み、その **DataSet** を表示グリッドにバインドすることによって結果セット全体を表示します。

**AseDataAdapter** を使用すると、単一の結果セットを返す任意の文字列 (SQL 文またはストアド・プロシージャ) を渡すことができます。**AseDataAdapter** では、デフォルトで、すべてのローが1度の操作でフェッチされます。カーソルを使用するには、接続文字列でプロパティを 'use cursor = true' に設定します。この場合、前方向への読み込み専用カーソルが使用され、すべてのローが読み込まれると、カーソルは自動的にクローズします。**AseDataAdapter** では、**DataSet** に変更を加えることができます。変更操作が完了したら、データベースに再接続して変更を適用します。

**AseDataAdapter** オブジェクトを使用すると、ジョインに基づく結果セットを取得できます。

**AseDataAdapter** の詳細については、「[AseDataAdapter クラス](#)」(137 ページ) を参照してください。

AseDataAdapter の例

次の例は、**AseDataAdapter** を使用して **DataSet** にデータを書き込む方法を示しています。

❖ **AseDataAdapter オブジェクトを使用したデータの取得**

- 1 データベースに接続します。
- 2 新しい DataSet を作成します。ここでは、DataSet の名前を “Results” とします。

C# の場合：

```
DataSet ds =new DataSet ();
```

Visual Basic .NET の場合：

```
Dim ds As New DataSet()
```

- 3 SQL 文を実行して “Results” という DataSet に結果を書き込む、新しい AseDataAdapter オブジェクトを作成します。

C# の場合：

```
AseDataAdapter da=new  
AseDataAdapter(txtSQLStatement.Text, _conn);  
da.Fill(ds, "Results"),
```

Visual Basic .NET の場合：

```
Dim da As New  
AseDataAdapter(txtSQLStatement.Text, conn)  
da.Fill(ds, "Results")
```

- 4 DataSet をウィンドウのグリッドにバインドします。

C# の場合：

```
dgResults.DataSource = ds.Tables["Results"],
```

Visual Basic .NET の場合：

```
dgResults.DataSource = ds.Tables("Results")
```

**AseDataAdapter オブジェクトを使用したローの挿入、更新、削除**

AseDataAdapter オブジェクトは、DataSet に結果セットを取得します。DataSet はテーブルのコレクションであり、テーブル間の関係と制約を保持しています。DataSet は .NET Framework 内に構築され、データベースへの接続に使用される Adaptive Server ADO.NET Data Provider から独立しています。

AseDataAdapter を使用するとき、まだデータベースに接続していない場合は接続がオープンされ、DataSet への書き込みが行われた後、接続が明示的にオープンされたものでない場合は接続がクローズされます。ただし、DataSet に書き込まれたデータは、データベースとの接続が切断している間も変更できます。

変更をデータベースへただちに適用する必要がない場合は、データやスキーマを含め、DataSet を WriteXml メソッドを使用して XML ファイルに書き込むことができます。この場合、後から ReadXml メソッドを使用して DataSet をロードすれば、変更を適用できます。

詳細については、.Net Framework のドキュメントで WriteXml と ReadXml の説明を参照してください。

Update メソッドを呼び出して DataSet からデータベースへ変更を適用するとき、AseDataAdapter は加えられた変更を分析し、必要に応じて Insert、Update、Delete のいずれかのコマンドを呼び出します。

DataSet を使用して変更 (挿入、更新、削除) できるのは、単一のテーブルのデータのみです。ジョインに基づく結果セットは更新できません。

---

**注意** DataSet に対する変更は、すべて接続が切断している間に行われます。これは、データベース内で、これらのローがアプリケーションによってロックされていないことを意味します。変更の適用対象データを他のユーザがすでに変更している場合もあるため、DataSet に加えた変更をデータベースに適用するときに発生する可能性のある競合すべてを解決するように、アプリケーションを設計する必要があります。

---

#### AseDataAdapter を使用する場合の競合の解決

対処する必要のある競合には次のものがあります。

- ユニークなプライマリ・キー – 2人のユーザが1つのテーブルに新しいローを挿入する場合、それぞれのローにユニークなプライマリ・キーを設定する必要があります。自動インクリメント・プライマリ・キーのあるテーブルでは、DataSet の値とデータ・ソースの値とが同期しなくなることがあります。

自動インクリメント・プライマリ・キーのプライマリ・キー値を取得する方法については、「[プライマリ・キー値の取得](#)」(62 ページ) を参照してください。

- 1つの値に対する複数の更新 – 2人のユーザが同じ値を変更するときに、どちらの値が正しいかを判定する論理をアプリケーションに組み込んでください。
- スキーマの変更 – DataSet で更新したテーブルのスキーマを他のユーザが変更した場合、その DataSet に加えた変更をデータベースに適用すると、更新は失敗します。
- データの同時実行性 – 同時実行性のある複数のアプリケーションが一貫したデータを参照できる場合、AseDataAdapter はフェッチしたローをロックしないため、DataSet を取得してオフラインで作業しているときに、2人目のユーザがデータベース内の値を更新できます。

これらの潜在的な問題の多くは、`AseCommand`、`AseDataReader`、`AseTransaction` のオブジェクトを使用してデータベースに変更を適用することで回避できます。トランザクションの独立性レベルを設定し、別のユーザが変更できないようにローをロックするため、Sybase では `AseTransaction` オブジェクトを使用することをおすすめします。

トランザクションを使用して変更をデータベースに適用する方法の詳細については、「[AseCommand オブジェクトを使用したローの挿入、更新、削除](#)」(42 ページ)を参照してください。

競合解決処理を単純化するため、`insert` 文、`update` 文、または `delete` 文をストアド・プロシージャ・コールに設計できます。`Insert`、`Update`、`Delete` 文をストアド・プロシージャに組み込むと、操作が失敗した場合にエラーを検出できます。これらの文に加えて、ストアド・プロシージャにエラー処理論理を追加すると、操作が失敗した場合にエラーをログ・ファイルに記録したり、操作を再試行するなど適切なアクションを実行できます。

### ❖ **AseDataAdapter を使用したテーブルへのローの挿入**

- 1 `AseConnection` オブジェクトを宣言して初期化します。

C# の場合：

```
AseConnection conn = new AseConnection(c_connStr);
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _  
    c_connStr )
```

- 2 接続をオープンします。

C# の場合：

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

- 3 新しい `AseDataAdapter` オブジェクトを作成します。

C# の場合：

```
AseDataAdapter adapter = new AseDataAdapter();  
adapter.MissingMappingAction =  
    MissingMappingAction.Passthrough;  
adapter.MissingSchemaAction =  
    MissingSchemaAction.Add;
```

Visual Basic .NET の場合：

```
Dim adapter As New AseDataAdapter()
adapter.MissingMappingAction = _
    MissingMappingAction.Passthrough
adapter.MissingSchemaAction = _
    MissingSchemaAction.Add
```

- 4 必要な **AseCommand** オブジェクトを作成して、必要なパラメータをすべて定義します。

次のコードは、**Select** コマンドと **Insert** コマンドを作成して、**Insert** コマンドのパラメータを定義します。

C# の場合：

```
adapter.SelectCommand = new AseCommand(
    "SELECT * FROM publishers", conn );
adapter.InsertCommand = new AseCommand(
    "INSERT INTO publishers( pub_id, pub_name, city, state) " +
    "VALUES( @pub_id, @pub_name, @city, @state )", conn);
adapter.InsertCommand.UpdatedRowSource = UpdateRowSource.None;
AseParameter parm = new AseParameter("@pub_id", AseDbType.Char, 4);
parm.SourceColumn = "pub_id";
parm.SourceVersion = DataRowVersion.Current;
adapter.InsertCommand.Parameters.Add( parm );
parm = new AseParameter("@pub_name", AseDbType.VarChar, 40);
parm.SourceColumn = "pub_name";
parm.SourceVersion = DataRowVersion.Current;
adapter.InsertCommand.Parameters.Add( parm );
parm = new AseParameter("@city", AseDbType.VarChar, 20);
parm.SourceColumn = "city";
parm.SourceVersion = DataRowVersion.Current;
adapter.InsertCommand.Parameters.Add( parm );
parm = new AseParameter("@state", AseDbType.Char, 2);
parm.SourceColumn = "state";
parm.SourceVersion = DataRowVersion.Current;
adapter.InsertCommand.Parameters.Add( parm );
```

Visual Basic .NET の場合：

```
adapter.SelectCommand = New AseCommand( _
    "SELECT * FROM publishers", conn )
adapter.InsertCommand = New AseCommand( _
    "INSERT INTO publishers( pub_id, pub_name, city, state) " + _
    " VALUES( @pub_id, @pub_name, @city, @state )", conn)
adapter.InsertCommand.UpdatedRowSource = _
    UpdateRowSource.None
Dim parm As New AseParameter("@pub_id", AseDbType.Char, 4)
parm.SourceColumn = "pub_id"
parm.SourceVersion = DataRowVersion.Current
adapter.InsertCommand.Parameters.Add( parm )
parm = New AseParameter("@pub_name", AseDbType.VarChar, 40)
```

```
parm.SourceColumn = "pub_name"
parm.SourceVersion = DataRowVersion.Current
adapter.InsertCommand.Parameters.Add( parm )
parm = New AseParameter("@city", AseDbType.VarChar, 20)
parm.SourceColumn = "city"
parm.SourceVersion = DataRowVersion.Current
adapter.InsertCommand.Parameters.Add( parm )
parm = New AseParameter("@state", AseDbType.Char, 2)
parm.SourceColumn = "state"
parm.SourceVersion = DataRowVersion.Current
adapter.InsertCommand.Parameters.Add( parm )
```

5 **DataTable** に **Select** 文の結果を書き込みます。

C# の場合：

```
DataTable dataTable = new DataTable( "publishers" );
int rowCount = adapter.Fill( dataTable );
```

Visual Basic .NET の場合：

```
Dim dataTable As New DataTable( "publishers" )
Dim rowCount As Integer = adapter.Fill( dataTable )
```

6 **DataTable** に新しいローを挿入して、変更をデータベースに適用します。

C# の場合：

```
DataRow row1 = dataTable.NewRow();
row1[0] = "9901";
row1[1] = "New Publisher";
row1[2] = "Concord";
row1[3] = "MA";
dataTable.Rows.Add( row1 );
DataRow row2 = dataTable.NewRow();
row2[0] = "9902";
row2[1] = "My Publisher";
row2[2] = "Dublin";
row2[3] = "CA";
dataTable.Rows.Add( row2 );
int recordsAffected = adapter.Update( dataTable );
```

Visual Basic .NET の場合：

```
Dim row1 As DataRow = dataTable.NewRow()
row1(0) = "9901"
row1(1) = "New Publisher"
row1(2) = "Concord"
row1(3) = "MA"
dataTable.Rows.Add( row1 )
Dim row2 As DataRow = dataTable.NewRow()
row2(0) = "9902"
row2(1) = "My Publisher"
row2(2) = "Dublin"
```

```
row2(3) = "CA"  
dataTable.Rows.Add( row2 )  
Dim recordsAffected As Integer =_  
    adapter.Update( dataTable )
```

- 7 更新の結果を表示します。

C# の場合 :

```
dataTable.Clear();  
rowCount = adapter.Fill( dataTable );  
dataGridView.DataSource = dataTable;
```

Visual Basic .NET の場合 :

```
dataTable.Clear()  
rowCount = adapter.Fill( dataTable )  
dataGridView.DataSource = dataTable
```

- 8 接続をクローズします。

C# の場合 :

```
conn.Close();
```

Visual Basic .NET の場合 :

```
conn.Close()
```

#### ❖ AseDataAdapter オブジェクトを使用したローの更新

- 1 AseConnection オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection( c_connStr );
```

Visual Basic .NET の場合 :

```
Dim conn As New AseConnection( _  
    c_connStr )
```

- 2 接続をオープンします。

C# の場合 :

```
conn.Open();
```

Visual Basic .NET の場合 :

```
conn.Open()
```

### 3 新しい AseDataAdapter オブジェクトを作成します。

C# の場合 :

```
AseDataAdapter adapter = new AseDataAdapter();
adapter.MissingMappingAction =
    MissingMappingAction.Passthrough;
adapter.MissingSchemaAction =
    MissingSchemaAction.Add;
```

Visual Basic .NET の場合 :

```
Dim adapter As New AseDataAdapter()
adapter.MissingMappingAction = _
    MissingMappingAction.Passthrough
adapter.MissingSchemaAction = _
    MissingSchemaAction.Add
```

### 4 AseCommand オブジェクトを作成して、そのパラメータを定義します。

次のコードは、**Select** コマンドと **Update** コマンドを作成して、**Update** コマンドのパラメータを定義します。

C# の場合 :

```
adapter.SelectCommand = new AseCommand(
    "SELECT * FROM publishers WHERE pub_id > '9900'",
    conn );
adapter.UpdateCommand = new AseCommand(
    "UPDATE publishers SET pub_name = @pub_name, " +
    "city = @city, state = @state " +
    "WHERE pub_id = @pub_id", conn );
adapter.UpdateCommand.UpdatedRowSource =
    UpdateRowSource.None;
AseParameter parm = new AseParameter("@pub_id",
    AseDbType.Char, 4);
parm.SourceColumn = "pub_id";
parm.SourceVersion = DataRowVersion.Current;
adapter.UpdateCommand.Parameters.Add( parm );
parm = new AseParameter("@pub_name",
    AseDbType.VarChar, 40);
parm.SourceColumn = "pub_name";
parm.SourceVersion = DataRowVersion.Current;
adapter.UpdateCommand.Parameters.Add( parm );
parm = new AseParameter("@city",
    AseDbType.VarChar, 20);
parm.SourceColumn = "city";
parm.SourceVersion = DataRowVersion.Current;
adapter.UpdateCommand.Parameters.Add( parm );
parm = new AseParameter("@state",
    AseDbType.Char, 2);
parm.SourceColumn = "state";
parm.SourceVersion = DataRowVersion.Current;
adapter.UpdateCommand.Parameters.Add( parm );
```



Visual Basic .NET の場合：

```

adapter.SelectCommand = New AseCommand( _
    "SELECT * FROM publishers WHERE pub_id > '9900'", _
    conn )
adapter.UpdateCommand = New AseCommand( _
    "UPDATE publishers SET pub_name = @pub_name, " + _
    "city = @city, state = @state " + _
    "WHERE pub_id = @pub_id", conn )
adapter.UpdateCommand.UpdatedRowSource = _
    UpdateRowSource.None
Dim parm As New AseParameter("@pub_id", _
    AseDbType.Char, 4)
parm.SourceColumn = "pub_id"
parm.SourceVersion = DataRowVersion.Current
adapter.UpdateCommand.Parameters.Add( parm )
parm = New AseParameter("@pub_name", _
    AseDbType.VarChar, 40)
parm.SourceColumn = "pub_name"
parm.SourceVersion = DataRowVersion.Current
adapter.UpdateCommand.Parameters.Add( parm )
parm = New AseParameter("@city", _
    AseDbType.VarChar, 20)
parm.SourceColumn = "city"
parm.SourceVersion = DataRowVersion.Current
adapter.UpdateCommand.Parameters.Add( parm )
parm = New AseParameter("@state", _
    AseDbType.Char, 2)
parm.SourceColumn = "state"
parm.SourceVersion = DataRowVersion.Current
adapter.UpdateCommand.Parameters.Add( parm )

```

#### 5 DataTable に Select 文の結果を書き込みます。

C# の場合：

```

DataTable dataTable = new DataTable( "publishers" );
int rowCount = adapter.Fill( dataTable );

```

Visual Basic .NET の場合：

```

Dim dataTable As New DataTable( "publishers" )
Dim rowCount As Integer = adapter.Fill( dataTable )

```

#### 6 ローの更新値で DataTable を更新して、変更をデータベースに適用します。

C# の場合：

```

foreach ( DataRow row in dataTable.Rows )
{
    row[1] = ( string ) row[1] + "_Updated";
}
int recordsAffected = adapter.Update( dataTable );

```

Visual Basic .NET の場合 :

```
Dim row as DataRow
For Each row in dataTable.Rows
    row(1) = row(1) + "_Updated"
Next
Dim recordsAffected As Integer = _
adapter.Update( dataTable )
```

7 結果をウィンドウのグリッドにバインドします。

C# の場合 :

```
dataTable.Clear();
adapter.SelectCommand.CommandText =
    "SELECT * FROM publishers";
rowCount = adapter.Fill( dataTable );
dataGridView.DataSource = dataTable;
```

Visual Basic .NET の場合 :

```
dataTable.Clear()
adapter.SelectCommand.CommandText = _
    "SELECT * FROM publishers";
rowCount = adapter.Fill( dataTable )
dataGridView.DataSource = dataTable
```

8 接続をクローズします。

C# の場合 :

```
conn.Close();
```

Visual Basic .NET の場合 :

```
conn.Close()
```

#### ❖ **AseDataAdapter** オブジェクトを使用したテーブルからのローの削除

1 **AseConnection** オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection( c_connStr );
```

Visual Basic .NET の場合 :

```
Dim conn As New AseConnection( _
    c_connStr )
```

- 2 接続をオープンします。

C# の場合 :

```
conn.Open();
```

Visual Basic .NET の場合 :

```
conn.Open()
```

- 3 **AseDataAdapter** オブジェクトを作成します。

C# の場合 :

```
AseDataAdapter adapter = new AseDataAdapter();  
adapter.MissingMappingAction =  
    MissingMappingAction.Passthrough;  
adapter.MissingSchemaAction =  
    MissingSchemaAction.AddWithKey;
```

Visual Basic .NET の場合 :

```
Dim adapter As New AseDataAdapter()  
adapter.MissingMappingAction = _  
    MissingMappingAction.Passthrough  
adapter.MissingSchemaAction = _  
    MissingSchemaAction.AddWithKey
```

- 4 必要な **AseCommand** オブジェクトを作成して、必要なパラメータをすべて定義します。

次のコードは、**Select** コマンドと **Delete** コマンドを作成して、**Delete** コマンドのパラメータを定義します。

C# の場合 :

```
adapter.SelectCommand = new AseCommand(  
    "SELECT * FROM publishers WHERE pub_id > '9900'",  
    conn );  
adapter.DeleteCommand = new AseCommand(  
    "DELETE FROM publishers WHERE pub_id = @pub_id",  
    conn );  
adapter.DeleteCommand.UpdatedRowSource =  
    UpdateRowSource.None;  
AseParameter parm = new AseParameter("@pub_id",  
    AseDbType.Char, 4);  
parm.SourceColumn = "pub_id";  
parm.SourceVersion = DataRowVersion.Original;  
adapter.DeleteCommand.Parameters.Add( parm );
```

Visual Basic .NET の場合 :

```
adapter.SelectCommand = New AseCommand( _
    "SELECT * FROM publishers WHERE pub_id > '9900'", _
    conn )
adapter.DeleteCommand = New AseCommand( _
    "DELETE FROM publishers WHERE pub_id = @pub_id", conn )
adapter.DeleteCommand.UpdatedRowSource = _
    UpdateRowSource.None
Dim parm As New AseParameter("@pub_id", _
    AseDbType.Char, 4)
parm.SourceColumn = "pub_id"
parm.SourceVersion = DataRowVersion.Original
adapter.DeleteCommand.Parameters.Add( parm )
```

5 **DataTable** に **Select** 文の結果を書き込みます。

C# の場合 :

```
DataTable dataTable = new DataTable( "publishers" );
int rowCount = adapter.Fill( dataTable );
```

Visual Basic .NET の場合 :

```
Dim dataTable As New DataTable( "publishers" )
Dim rowCount As Integer = adapter.Fill( dataTable )
```

6 **DataTable** を変更して、データベースに変更を適用します。

C# の場合 :

```
foreach ( DataRow row in dataTable.Rows )
{
    row.Delete();
}
int recordsAffected = adapter.Update( dataTable );
```

Visual Basic .NET の場合 :

```
Dim row as DataRow
For Each row in dataTable.Rows
    row.Delete()
Next
Dim recordsAffected As Integer = _
    adapter.Update( dataTable )
```

7 結果をウィンドウのグリッドにバインドします。

C# の場合 :

```
dataTable.Clear();
rowCount = adapter.Fill( dataTable );
dataGridView.DataSource = dataTable;
```

Visual Basic .NET の場合：

```
dataTable.Clear()
rowCount = adapter.Fill( dataTable )datagrid.
DataSource = dataTable
```

- 8 接続をクローズします。

C# の場合：

```
conn.Close();
```

Visual Basic .NET の場合：

```
conn.Close()
```

### AseDataAdapter スキーマ情報の取得

AseDataAdapter では、FillSchema メソッドを使用して DataSet 内の結果セットに関するスキーマ情報を取得できます。FillSchema メソッドは、標準の .NET DataTable オブジェクトを返します。このオブジェクトは、結果セット内のカラムすべての名前を提供します。

詳細については、「FillSchema メソッド」(140 ページ)を参照してください。

#### ❖ FillSchema メソッドを使用した DataSet スキーマ情報の取得

- 1 AseConnection オブジェクトを宣言して初期化します。

C# の場合：

```
AseConnection conn = new AseConnection(
    c_connStr );
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _
    c_connStr )
```

- 2 接続をオープンします。

C# の場合：

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

- 3 目的の Select 文を使用して AseDataAdapter を作成します。このクエリの結果セットのスキーマが返されます。

C# の場合：

```
AseDataAdapter adapter = new AseDataAdapter(
    "SELECT * FROM employee", conn );
```

Visual Basic .NET の場合：

```
Dim adapter As New AseDataAdapter( _  
    "SELECT * FROM employee", conn )
```

- 4 スキーマを書き込む新しい **DataTable** オブジェクトを、ここでは “Table” という名前で作成します。

C# の場合：

```
DataTable dataTable = new DataTable( "Table" );
```

Visual Basic .NET の場合：

```
Dim dataTable As New DataTable( "Table" )
```

- 5 データ・ソースからのスキーマを **DataTable** に書き込みます。

C# の場合：

```
adapter.FillSchema( dataTable, SchemaType.Source );
```

Visual Basic .NET の場合：

```
adapter.FillSchema( dataTable, SchemaType.Source )
```

- 6 **AseConnection** オブジェクトをクローズします。

C# の場合：

```
conn.Close();
```

Visual Basic .NET の場合：

```
conn.Close()
```

- 7 **DataSet** をウィンドウのグリッドにバインドします。

C# の場合：

```
dataGrid.DataSource = dataTable;
```

Visual Basic .NET の場合：

```
dataGrid.DataSource = dataTable
```

## プライマリ・キー値の取得

更新対象テーブルに自動インクリメント・プライマリ・キーがある場合、またはプライマリ・キー・プールからプライマリ・キーを取得する場合は、ストアード・プロシージャを使用して、データ・ソースで生成される値を取得できます。

**AseDataAdapter** を使用する場合、この手法を利用して、データ・ソースで生成されるプライマリ・キー値を **DataSet** のカラムに書き込むことができます。**AseCommand** オブジェクトでこの手法を使用するには、パラメータからキー・カラムを取得するか、**DataReader** を再オープンします。

例

次の例では、“adodotnet\_primarykey”という名前のテーブルを使用します。このテーブルには、“id”と“name”という2つのカラムがあります。テーブルのプライマリ・キーは“id”で、自動インクリメント値を含む NUMERIC(8) です。name カラムは CHAR(40) です。

この例では、次のストアド・プロシージャを呼び出して、データベースから自動インクリメント・プライマリ・キー値を取得します。

```
create procedure sp_adodotnet_primarykey
    @p_name char(40),
    @p_id int output
as
begin
    insert into adodotnet_primarykey(name)
        VALUES(@p_name)
    select @p_id = @@identity
END
```

#### ❖ AseCommand オブジェクトを使用した、自動インクリメント・プライマリ・キーを持つ新しいローの挿入

- 1 データベースに接続します。

C# の場合：

```
AseConnection conn = new AseConnection( c_connStr );
conn.Open();
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _
    c_connStr )
conn.Open()
```

- 2 DataTable に新しいローを挿入する AseCommand オブジェクトを新規作成します。次のコードでは、行 int id1 = ( int ) parmId.Value; によってローのプライマリ・キー値を検証します。

C# の場合：

```
AseCommand cmd = conn.CreateCommand();
cmd.CommandText = "sp_adodotnet_primarykey";
cmd.CommandType = CommandType.StoredProcedure;
AseParameter parmId = new AseParameter(
    "@p_id", AseDbType.Integer);
parmId.Direction = ParameterDirection.Output;
cmd.Parameters.Add( parmId );
AseParameter parmName = new AseParameter(
    "@p_name", AseDbType.Char );
parmName.Direction = ParameterDirection.Input;
cmd.Parameters.Add( parmName );
parmName.Value = "R & D --- Command";
cmd.ExecuteNonQuery();
int id1 = ( int ) parmId.Value;
```

```
parmName.Value = "Marketing --- Command";
cmd.ExecuteNonQuery();
int id2 = ( int ) parmId.Value;
parmName.Value = "Sales --- Command";
cmd.ExecuteNonQuery();
int id3 = ( int ) parmId.Value;
parmName.Value = "Shipping --- Command";
cmd.ExecuteNonQuery();
int id4 = ( int ) parmId.Value;
```

Visual Basic .NET の場合 :

```
Dim cmd As AseCommand = conn.CreateCommand()
cmd.CommandText = "sp_adodotnet_primarykey"
cmd.CommandType = CommandType.StoredProcedure
Dim parmId As New AseParameter("@p_id", _
    AseDbType.Integer)
parmId.Direction = ParameterDirection.Output
cmd.Parameters.Add( parmId )
Dim parmName As New AseParameter("@p_name", _
    AseDbType.Char)
parmName.Direction = ParameterDirection.Input
cmd.Parameters.Add(parmName )

parmName.Value = "R & D --- Command"
cmd.ExecuteNonQuery()
Dim id1 As Integer = parmId.Value
parmName.Value = "Marketing --- Command"
cmd.ExecuteNonQuery()
Dim id2 As Integer = parmId.Value
parmName.Value = "Sales --- Command"
cmd.ExecuteNonQuery()
Dim id3 As Integer = parmId.Value
parmName.Value = "Shipping --- Command"
cmd.ExecuteNonQuery()
dim id4 As Integer = parmId.Value
```

- 3 結果をウィンドウのグリッドにバインドして、変更をデータベースに適用します。

C# の場合 :

```
cmd.CommandText = "select * from " +
    "adodotnet_primarykey";
cmd.CommandType = CommandType.Text;
AseDataReader dr = cmd.ExecuteReader();
dataGridView.DataSource = dr;
```



Visual Basic .NET の場合：

```
cmd.CommandText = "select * from " + _
    "adodotnet_primarykey"
cmd.CommandType = CommandType.Text
Dim dr As AseDataReader = cmd.ExecuteReader()
dataGridView.DataSource = dr
```

4 接続をクローズします。

C# の場合：

```
conn.Close();
```

Visual Basic .NET の場合：

```
conn.Close()
```

❖ **AseDataAdapter** オブジェクトを使用した、自動インクリメント・プライマリ・キーを持つ新しいローの挿入

1 新しい AseDataAdapter を作成します。

C# の場合：

```
AseConnection conn = new AseConnection(
    c_connStr );
conn.Open();
DataSet dataSet = new DataSet();
AseDataAdapter adapter = new AseDataAdapter();
adapter.MissingMappingAction =
    MissingMappingAction.Passthrough;
adapter.MissingSchemaAction =
    MissingSchemaAction.AddWithKey;
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _
    c_connStr )
conn.Open()
Dim dataSet As New DataSet()
Dim adapter As New AseDataAdapter()
adapter.MissingMappingAction = _
    MissingMappingAction.Passthrough
adapter.MissingSchemaAction = _
    MissingSchemaAction.AddWithKey
```

- 2 データと DataSet のスキーマを書き込みます。次のコードでは、**AseDataAdapter.Fill** メソッドによって **SelectCommand** を呼び出して、この操作を実行します。既存のレコードが不要な場合は、**Fill** メソッドと **SelectCommand** を使用せずに、**DataSet** を手動で作成することもできます。

C# の場合：

```
adapter.SelectCommand = new AseCommand(  
    "select * from adodotnet_primarykey",  
    conn );
```

Visual Basic .NET の場合：

```
adapter.SelectCommand = New AseCommand( _  
    "select * from adodotnet_primarykey", conn )
```

- 3 新しい **AseCommand** を作成して、データベースからプライマリ・キー値を取得します。

C# の場合：

```
adapter.InsertCommand = new AseCommand(  
    "sp_adodotnet_primarykey", conn );  
adapter.InsertCommand.CommandType =  
    CommandType.StoredProcedure;  
adapter.InsertCommand.UpdatedRowSource =  
    UpdateRowSource.OutputParameters;  
AseParameter parmId = new AseParameter(  
    "@p_id", AseDbType.Integer);  
parmId.Direction = ParameterDirection.Output;  
parmId.SourceColumn = "id";  
parmId.SourceVersion = DataRowVersion.Current;  
adapter.InsertCommand.Parameters.Add( parmId );  
AseParameter parmName = new AseParameter(  
    "@p_name", AseDbType.Char);  
parmName.Direction = ParameterDirection.Input;  
parmName.SourceColumn = "name";  
parmName.SourceVersion = DataRowVersion.Current;  
adapter.InsertCommand.Parameters.Add( parmName );
```

Visual Basic .NET の場合：

```
adapter.InsertCommand = new AseCommand( _  
    "sp_adodotnet_primarykey", conn )  
adapter.InsertCommand.CommandType = _  
    CommandType.StoredProcedure  
adapter.InsertCommand.UpdatedRowSource = _  
    UpdateRowSource.OutputParameters  
Dim parmId As New AseParameter( _  
    "@p_id", AseDbType.Integer)  
parmId.Direction = ParameterDirection.Output  
parmId.SourceColumn = "id"  
parmId.SourceVersion = DataRowVersion.Current  
adapter.InsertCommand.Parameters.Add( parmId )
```

```
Dim parmName As New AseParameter( _  
    "@p_name", AseDbType.Char)  
parmName.Direction = ParameterDirection.Input  
parmName.SourceColumn = "name"  
parmName.SourceVersion = DataRowVersion.Current  
adapter.InsertCommand.Parameters.Add( parmName )
```

#### 4 DataSet に書き込みます。

C# の場合：

```
adapter.Fill( dataSet );
```

Visual Basic .NET の場合：

```
adapter.Fill( dataSet )
```

#### 5 DataSet に新しいローを挿入します。

C# の場合：

```
DataRow row = dataSet.Tables[0].NewRow();  
row[0] = -1;  
row[1] = "R & D --- Adapter";  
dataSet.Tables[0].Rows.Add( row );  
row = dataSet.Tables[0].NewRow();  
row[0] = -2;  
row[1] = "Marketing --- Adapter";  
dataSet.Tables[0].Rows.Add( row );  
row = dataSet.Tables[0].NewRow();  
row[0] = -3;  
row[1] = "Sales --- Adapter";  
dataSet.Tables[0].Rows.Add( row );  
row = dataSet.Tables[0].NewRow();  
row[0] = -4;  
row[1] = "Shipping --- Adapter";  
dataSet.Tables[0].Rows.Add( row );
```

Visual Basic .NET の場合：

```
Dim row As DataRow = dataSet.Tables(0).NewRow()  
row(0) = -1  
row(1) = "R & D --- Adapter"  
dataSet.Tables(0).Rows.Add( row )  
row = dataSet.Tables(0).NewRow()  
row(0) = -2  
row(1) = "Marketing --- Adapter"  
dataSet.Tables(0).Rows.Add( row )  
row = dataSet.Tables(0).NewRow()  
row(0) = -3  
row(1) = "Sales --- Adapter"  
dataSet.Tables(0).Rows.Add( row )  
row = dataSet.Tables(0).NewRow()  
row(0) = -4
```

```
row(1) = "Shipping --- Adapter"  
dataSet.Tables(0).Rows.Add( row )
```

- 6 **DataSet** に加えられた変更をデータベースに適用します。**Update()** メソッドが呼び出されると、プライマリ・キー値が、データベースから取得された値に変わります。

C# の場合：

```
adapter.Update( dataSet );  
dataGrid.DataSource = dataSet.Tables[0];
```

Visual Basic .NET の場合：

```
adapter.Update( dataSet )  
dataGrid.DataSource = dataSet.Tables(0)
```

ユーザが **DataTable** に新しいローを追加して **Update** メソッドを呼び出すと、**AseDataAdapter** は **InsertCommand** を呼び出して、追加された新しいローそれぞれのキー・カラムに出力パラメータをマップします。**Update** メソッドが呼び出されるのは 1 回だけですが、**InsertCommand** は、追加される新しいローそれぞれについて、**Update** メソッドによって必要な回数呼び出されます。

- 7 データベースへの接続をクローズします。

C# の場合：

```
conn.Close();
```

Visual Basic .NET の場合：

```
conn.Close()
```

## BLOB の処理

長い文字列値またはバイナリ・データをフェッチする場合、データを分割してフェッチできるメソッドがあります。バイナリ・データには **GetBytes** メソッドを、文字列データには **GetChars** メソッドを使用します。これらを使用しない場合、BLOB データはデータベースからフェッチされるその他のデータと同様の方法で処理されます。

詳細については、「[GetBytes メソッド](#)」(146 ページ)と「[GetChars メソッド](#)」(147 ページ)を参照してください。

### ❖ **GetChars** メソッドを使用した文字列を返すコマンドの発行

- 1 Connection オブジェクトを宣言して初期化します。
- 2 接続をオープンします。

- SQL 文を定義して実行する Command オブジェクトを追加します。

C# の場合 :

```
AseCommand cmd = new AseCommand(
    "select au_id, copy from blurbs", conn );
```

Visual Basic .NET の場合 :

```
Dim cmd As New AseCommand( _
    "select au_id, copy from blurbs", conn)
```

- ExecuteReader メソッドを呼び出して、DataReader オブジェクトを返します。

C# の場合 :

```
AseDataReader reader = cmd.ExecuteReader();
```

Visual Basic .NET の場合 :

```
Dim reader As AseDataReader = cmd.ExecuteReader()
```

次のコードは、結果セットから2つのカラムを読み込みます。最初のカラムは **varchar** で、2番目のカラムは **Text** です。GetChars は、Text カラムから1度に100文字を読み込みます。

C# の場合 :

```
int length = 100;
char[] buf = new char[ length ];
String au_id;
long dataIndex = 0;
long charsRead = 0;
long blobLength = 0;
while( reader.Read() )
{
    au_id = reader.GetString(0);
    do
    {
        charsRead = reader.GetChars(
            1, dataIndex, buf, 0, length);
        dataIndex += length;
        // do something with the chars read
        //.... some code
        //
        // reinitialize char array
        buf = new char[ length ];
    } while ( charsRead == length );
    blobLength = dataIndex + charsRead;
}
```

Visual Basic .NET の場合 :

```
Dim length As Integer = 100
Dim buf(length) As Char
Dim au_id As String
Dim dataIndex As Long = 0
Dim charsRead As Long = 0
Dim blobLength As Long = 0
While reader.Read()
    au_id = reader.GetString(0)
    Do
        charsRead = reader.GetChars( _
            1, dataIndex, buf, 0, length)
        dataIndex = dataIndex + length
        ' do something with the data read
        '
        ' use code
        '
        ' reinitialize the char array
        ReDim buf(length)
    Loop While (charsRead = length)
    blobLength = dataIndex + charsRead
End While
```

- 5 **DataReader** オブジェクトと **Connection** オブジェクトをクローズします。

C# の場合 :

```
reader.Close();
conn.Close();
```

Visual Basic .NET の場合 :

```
reader.Close()
conn.Close()
```

## 時刻値の取得

.NET Framework には、Time 構造体がありません。Adaptive Server で時刻値をフェッチする場合は、**GetDateTime()** メソッドを使用してください。このメソッドを使用して、.NET Framework の **DateTime** オブジェクトとしてデータを返します。

### ❖ **GetDateTime** メソッドを使用した時刻値の変換

- 1 **Connection** オブジェクトを宣言して初期化します。

C# の場合 :

```
AseConnection conn = new AseConnection(
    c_connStr );
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _  
    c_connStr )
```

- 2 接続をオープンします。

C# の場合：

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

- 3 SQL 文を定義して実行する **Command** オブジェクトを追加します。

C# の場合：

```
AseCommand cmd = new AseCommand(  
    "SELECT title_id, title, pubdate FROM titles",  
    conn );
```

Visual Basic .NET の場合：

```
Dim cmd As New AseCommand( _  
    "SELECT title_id, title, pubdate FROM titles", _  
    conn)
```

- 4 **ExecuteReader** メソッドを呼び出して、**DataReader** オブジェクトを返します。

C# の場合：

```
AseDataReader reader = cmd.ExecuteReader();
```

Visual Basic .NET の場合：

```
Dim reader As AseDataReader = cmd.ExecuteReader()
```

次のコードは、**GetDateTime** メソッドを使用して、**DateTime** 値を返します。

C# の場合：

```
while ( reader.Read() )
{
    String tid = reader.GetString(0);
    String title = reader.GetString(1);
    DateTime time = reader.GetDateTime(2);
    // do something with the data
}
```

Visual Basic .NET の場合：

```
While reader.Read()
    Dim tid As String = reader.GetString(0)
    Dim title As String = reader.GetString(1)
    Dim time As DateTime = reader.GetDateTime(2)
    ' do something with the data...
End While
```

- 5 **DataReader** オブジェクトと **Connection** オブジェクトをクローズします。

C# の場合：

```
reader.Close();
conn.Close();
```

Visual Basic .NET の場合：

```
reader.Close()
conn.Close()
```

## ストアド・プロシージャの使用

Adaptive Server ADO.NET Data Provider ではストアド・プロシージャを使用できます。結果セットを返すストアド・プロシージャを呼び出す場合は、**ExecuteReader** メソッドを使用します。

---

**注意** ストアド・プロシージャを使用してデータベースからデータを取得する場合、ストアド・プロシージャから出力パラメータ値と結果セットの両方が返されると、結果セットはリセットされ、出力パラメータ値が参照されると同時に結果セットのローは参照できなくなります。Sybase では、このような場合、結果セットのローすべてを参照して使い終わるまで、参照側の出力パラメータ値を最後までそのままにしておくことをおすすめします。

---

結果セットを返さないストアド・プロシージャを呼び出す場合は、**ExecuteNonQuery** メソッドを使用します。単一の値のみを返すストアド・プロシージャを呼び出す場合は、**ExecuteScalar** メソッドを使用します。



ストアド・プロシージャでパラメータが必要な場合は、対応する `AseParameter` オブジェクトを作成します。`CommandType` を `StoredProcedure` に指定した場合は、`CommandText` をストアド・プロシージャの名前に設定してください。次に例を示します。

```
sp_producttype
```

`Parameter` オブジェクトの詳細については、「[AseParameter クラス](#)」(165 ページ)を参照してください。

#### ❖ ストアド・プロシージャの実行

- 1 `AseConnection` オブジェクトを宣言して初期化します。

C# の場合：

```
AseConnection conn = new AseConnection(  
    c_connStr );
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _  
    c_connStr )
```

- 2 接続をオープンします。

C# の場合：

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

- 3 SQL 文を定義して実行する `AseCommand` オブジェクトを追加します。次のコードは、`CommandType` プロパティを使用して、コマンドがストアド・プロシージャであることを識別します。

C# の場合：

```
AseCommand cmd = new AseCommand(  
    "titleid_proc", conn );  
cmd.CommandType = CommandType.StoredProcedure;
```

Visual Basic .NET の場合：

```
Dim cmd As New AseCommand( _  
    "titleid_proc", conn )  
cmd.CommandType = CommandType.StoredProcedure
```

- 4 **AseParameter** オブジェクトを追加して、ストアド・プロシージャのパラメータを定義します。ストアド・プロシージャに必要なパラメータそれぞれに、新しく **AseParameter** オブジェクトを作成してください。

C# の場合 :

```
AseParameter param = cmd.CreateParameter();
param.ParameterName = "@title_id";
param.AseDbType = AseDbType.VarChar;
param.Direction = ParameterDirection.Input;
param.Value = "BU";
cmd.Parameters.Add( param );
```

Visual Basic .NET の場合 :

```
Dim param As AseParameter = cmd.CreateParameter()
param.ParameterName = "@title_id"
param.AseDbType = AseDbType.VarChar
param.Direction = ParameterDirection.Input
param.Value = "BU"
cmd.Parameters.Add( param )
```

**Parameter** オブジェクトの詳細については、「[AseParameter クラス](#)」(165 ページ) を参照してください。

- 5 **ExecuteReader** メソッドを呼び出して、**DataReader** オブジェクトを返します。**Get** メソッドを使用して、結果を任意のデータ型で返します。

C# の場合 :

```
AseDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
    string title = reader.GetString(0);
    string id = reader.GetString(1);
    decimal price = reader.GetDecimal(2);
    // do something with the data....
}
```

Visual Basic .NET の場合 :

```
Dim reader As AseDataReader = cmd.ExecuteReader()
While reader.Read()
    Dim title As String = reader.GetString(0)
    Dim id As String = reader.GetString(1)
    Dim price As Decimal = reader.GetDecimal(2)
    ' do something with the data....
End While
```

- 6 **AseDataReader** オブジェクトと **AseConnection** オブジェクトをクローズします。

C# の場合 :

```
reader.Close();
conn.Close();
```

Visual Basic .NET の場合 :

```
reader.Close()
conn.Close()
```

ストアド・プロシージャ  
を呼び出すもう 1 つの  
方法

次のように、呼び出し構文を使用してストアド・プロシージャを呼び出すこともできます。この構文は ODBC および JDBC と互換性があります。次に例を示します。

```
AseCommand cmd = new AseCommand("{ call sp_product_info(?) }",
conn);
```

この場合、コマンドの種類を **CommandType.StoredProcedure** に設定しないでください。この構文は、名前付きパラメータを使用しないで、**AseCommand.NamedParameters** プロパティを“false”に設定している場合に使用できます。

結果セットまたは単一の値を返すストアド・プロシージャの呼び出し方法の詳細については、[「AseCommand オブジェクトを使用したデータの取得」\(37 ページ\)](#)を参照してください。

結果セットを返さないストアド・プロシージャの呼び出し方法の詳細については、[「AseCommand オブジェクトを使用したローの挿入、更新、削除」\(42 ページ\)](#)を参照してください。

## トランザクション処理

Adaptive Server ADO.NET Data Provider では、**AseTransaction** オブジェクトを使用して複数の文をグループ化できます。それぞれのトランザクションは、変更を永続的にデータベースに適用する **COMMIT**、またはトランザクションの操作すべてを取り消す **ROLLBACK** で終了します。トランザクションが完了した後、さらに変更を行う場合は、新しく **AseTransaction** オブジェクトを作成してください。ODBC や Embedded SQL の場合は動作が異なり、トランザクションがクローズされるまで、**COMMIT** や **ROLLBACK** が実行された後もトランザクションが保持されます。

トランザクションを作成しない場合、Adaptive Server ADO.NET Data Provider はデフォルトで **autocommit** モードで動作します。**insert**、**update**、または **delete** が実行されるたびに **Commit** が暗黙的に実行され、操作が完了すると変更がデータベースに反映されます。この場合、変更はロールバックできません。

## トランザクションの独立性レベルの設定

**AseTransaction** オブジェクトの詳細については、「[AseTransaction クラス \(178 ページ\)](#)」を参照してください。

トランザクションの開始時に独立性レベルを指定するように選択できます。この独立性レベルは、そのトランザクション内で実行されるすべてのコマンドに適用されます。

独立性レベルの詳細については、Adaptive Server Enterprise の『パフォーマンス & チューニング・ガイド』を参照してください。

**Select** 文を入力したときに Adaptive Server で使用されるロックは、トランザクションの独立性レベルに応じて異なります。

次の例は、**AseTransaction** オブジェクトを使用して SQL 文を発行してからロールバックを実行します。このトランザクションは独立性レベル 2 (RepeatableRead) を使用し、変更中のローに **Write** ロックを適用して、他のデータベース・ユーザがローを更新できないようにします。

**❖ AseTransaction オブジェクトを使用したコマンドの発行**

- 1 **AseConnection** オブジェクトを宣言して初期化します。

C# の場合：

```
AseConnection conn = new AseConnection(  
    c_connStr );
```

Visual Basic .NET の場合：

```
Dim conn As New AseConnection( _  
    c_connStr )
```

- 2 接続をオープンします。

C# の場合：

```
conn.Open();
```

Visual Basic .NET の場合：

```
conn.Open()
```

- 3 「T シャツ (Tee shirts)」 の価格を変更する SQL 文を発行します。

C# の場合：

```
string stmt = "update product " +  
    " set unit_price = 2000.00 " +  
    " where name = 'Tee shirt'";
```

Visual Basic .NET の場合：

```
Dim stmt As String = "update product " + _  
    " set unit_price = 2000.00 " + _  
    " where name = 'Tee shirt'"
```

- 4 **Command** オブジェクトを使用して SQL 文を発行する **AseTransaction** オブジェクトを作成します。

トランザクションを使用することで、独立性レベルを指定できるようになります。この例では独立性レベル 2 (**RepeatableRead**) を使用して、他のデータベース・ユーザがローを更新できないようにします。

C# の場合：

```
AseTransaction trans = conn.BeginTransaction(  
    IsolationLevel.RepeatableRead );  
AseCommand cmd = new AseCommand( stmt, conn, trans );  
int rows = cmd.ExecuteNonQuery();
```

Visual Basic .NET の場合：

```
Dim trans As AseTransaction = _  
    conn.BeginTransaction( _  
        IsolationLevel.RepeatableRead )  
Dim cmd As New AseCommand( _  
    stmt, conn, trans )  
Dim rows As Integer = cmd.ExecuteNonQuery()
```

- 5 変更をロールバックします。

C# の場合：

```
trans.Rollback();
```

Visual Basic .NET の場合：

```
trans.Rollback()
```

**AseTransaction** オブジェクトを使用すると、データベースに対する変更をコミットしたりロールバックしたりできます。トランザクションを使用しない場合、Adaptive Server ADO.NET Data Provider は **autocommit** モードで動作するため、データベースに加えた変更をロールバックできなくなります。変更を永続的に残すには、次のように指定します。

C# の場合：

```
trans.Commit();
```

Visual Basic .NET の場合：

```
trans.Commit()
```

- 6 **AseConnection** オブジェクトをクローズします。

C# の場合：

```
conn.Close();
```

Visual Basic .NET の場合：

```
conn.Close()
```

## エラー処理

アプリケーションは、ADO.NET エラーも含め、発生するすべてのエラーを処理できるように設計する必要があります。コード内の ADO.NET エラーの処理方法は、アプリケーションの他のエラーを処理する場合と同じです。

実行時にエラーが発生すると、Adaptive Server ADO.NET Data Provider は **AseException** オブジェクトを返します。各 **AseException** オブジェクトは **AseError** オブジェクトのリストで構成され、これらのエラー・オブジェクトにはエラー・メッセージとコードが組み込まれています。この他に、**IndexOutOfRangeException** や **NotSupportedException** などの例外も利用できます。

エラーは、変更をデータベースに適用するときが発生する競合とは異なります。アプリケーションには、競合が発生した場合に正しい値を計算したりログに記録したりする処理が必要です。

Adaptive Server  
ADO.NET Data Provider  
のエラー処理の例

Simple サンプル・プロジェクトの例を次に示します。実行時に発生したエラーや Adaptive Server ADO.NET Data Provider オブジェクトのエラーはすべて、メッセージ・ボックスに表示されます。次のコードは、エラーを検出してメッセージを表示します。

C# の場合：

```
catch( AseException ex )
{
    MessageBox.Show( ex.Message );
}
```

Visual Basic .NET の場合：

```
Catch ex As AseException
    MessageBox.Show(ex.Message)
End Try
```

接続エラー処理の例

Table Viewer サンプル・プロジェクトの例を次に示します。アプリケーションがデータベースへの接続を試行しているときにエラーが発生した場合、次のコードは try-catch ブロックを使用してエラーを検出し、メッセージを表示します。

C# の場合：

```
try
{
    _conn = new AseConnection(
        txtConnectionString.Text );
    _conn.Open();
}
catch( AseException ex )
{
    MessageBox.Show(ex.Message, "Failed to connect");
}
```

Visual Basic .NET の場合：

```
Try
    Dim _conn As New AseConnection( _
        txtConnectionString.Text )
    conn.Open()
Catch ex As AseException
    MessageBox.Show(ex.Message, "Failed to connect")
End Try
```

その他のエラー処理の例については、「[Simple サンプル・プロジェクトの理解](#) (13 ページ) と 「[Table Viewer サンプル・プロジェクトの理解](#) (19 ページ) を参照してください。

エラー処理の詳細については、「[AseException クラス](#) (162 ページ) と 「[AseError クラス](#) (159 ページ) を参照してください。

## パフォーマンスの考慮事項

この項では、Adaptive Server ADO.NET Data Provider を使用してアプリケーションの開発と展開を行う上で役立つヒントを紹介します。

### DbType.String と DbType.AnsiString

DbType.String と DbType.AnsiString では、ともに文字データが処理されます。ただし、この両データ型の処理方法はそれぞれに異なるので、不適切なデータ型を使用すると、アプリケーションのパフォーマンスに悪影響を及ぼす可能性があります。DbType.String では、パラメータは 2 バイトの Unicode 値として識別されて、サーバに送信されます。DbType.AnsiString では、パラメータはマルチバイトの文字列として送信されます。不要な文字列変換を避けるため、次のように使用します。

- char または varchar のカラムおよびパラメータに対しては、DbType.AnsiString を使用します。
- unichar または univarchar のカラムおよびパラメータに対しては、DbType.String を使用します。





## Adaptive Server の高度な機能

この章では、Adaptive Server ADO.NET Data Provider で使用できる Adaptive Server の機能について説明します。

トピック名	ページ
サポートされている Adaptive Server クラスタ・エディションの機能	81
分散トランザクションの使用	83
ディレクトリ・サービス	85
マイクロ秒の精度の time データ	87
パスワードの暗号化	88
パスワード有効期限の処理	89
SSL の使用	90
高可用性システムでのフェールオーバーの使用	93
Kerberos 認証の使用	95

### サポートされている Adaptive Server クラスタ・エディションの機能

この項では、クラスタ・エディション環境をサポートする ASE ADO.NET Driver の機能について説明します。クラスタ・エディション環境では、複数の Adaptive Server が共有ディスクのセットと高速プライベート相互接続に接続します。この場合、複数の物理ホストと論理ホストを使用して、Adaptive Server を拡張できます。

クラスタ・エディションの詳細については、『Cluster ユーザーズ・ガイド』を参照してください。

## ログインのリダイレクト

クラスタ・エディション環境では一般に、常にサーバ間で処理負荷の不均衡が発生しています。ビジー状態のサーバに対してクライアント・アプリケーションが接続した場合、ログインのリダイレクト機能によって、サーバの負荷バランスが調整されます。具体的には、クラスタ内の負荷が少ない別サーバに対して、クライアント接続がリダイレクトされます。ログインのリダイレクトが発生するのはログイン・シーケンス中であり、リダイレクトが発生したことは、クライアント・アプリケーションには通知されません。ログインのリダイレクト機能をサポートしているサーバに対してクライアント・アプリケーションが接続した時点で、この機能は自動的に有効になります。

---

**注意** クライアントをリダイレクトするように設定されているサーバに対してクライアント・アプリケーションが接続すると、ログインに時間がかかる場合があります。これは、クライアント接続が別サーバにリダイレクトされるたびに、ログイン・プロセスが再開されるからです。

---

## 接続マイグレーション

接続マイグレーション機能を使用すると、クラスタ・エディション環境内のサーバは動的に負荷を分散できます。さらに、既存のクライアント接続とそのコンテキストをクラスタ内の別サーバにシームレスにマイグレートできます。この機能によって、クラスタ・エディション環境では、最適なりソース配分と処理時間の短縮が実現します。サーバ間のマイグレーションはシームレスに行われるので、接続マイグレーション機能は、可用性の高い「ダウン時間ゼロ」の環境を構築する場合にも役立ちます。接続マイグレーション機能をサポートしているサーバに対してクライアント・アプリケーションが接続した時点で、この機能は自動的に有効になります。

---

**注意** 接続マイグレーション中には、コマンドの実行に時間がかかる場合があります。状況に応じて、コマンドのタイムアウト値を増やすことをおすすめします。

---

## 接続フェールオーバー

接続フェールオーバー機能を使用すると、停電やソケットの障害など、予想外の原因でプライマリ・サーバが使用不可になった場合に、クライアント・アプリケーションは接続先を別の Adaptive Server に切り替えることができます。クラスタ環境では、クライアント・アプリケーションは動的なフェールオーバー・アドレスを使用して、複数のサーバに対して何度もフェールオーバーできます。

高可用性に対応したシステムでは、フェールオーバーターゲットの候補をクライアント・アプリケーションにあらかじめ設定しておく必要はありません。Adaptive Server は、クラスタ・メンバシップ、論理クラスタの使用状況、負荷分散などに基づいて、最適なフェールオーバーリストを常にクライアントに提供します。クライアントは、フェールオーバー時にフェールオーバーリストの順序付けを参照して、再接続します。ドライバがサーバに正常に接続した場合は、返されたリストに基づいて、ホスト値のリストが内部的に更新されます。それ以外の場合は、接続失敗例外が発生します。

### クラスタ・エディションの接続フェールオーバーの有効化

クラスタ・エディションの接続フェールオーバーを有効にするには、HASession 接続文字列プロパティを 1 に設定します。次に例を示します。

```
Data Source=server1;Port=port1;User ID=sa;Password=;  
Initial Catalog=sdc;HASession=1;  
AlternateServers=server2:port2,...,serverN:portN;
```

この例では、Data Source はプライマリ・サーバとポートを定義します。Sybase が提供する ADO.NET Provider は、最初にプライマリ・サーバへの接続します。接続に失敗した場合は、AlternateServers に列挙されているサーバへ順番に接続します。接続に成功するか、リストの末尾に達するまで、この処理が繰り返されます。

---

**注意** 接続文字列で指定された代替サーバのリストは、初期接続時にのみ使用されます。使用可能なインスタンスとの接続の確立後、高可用性をサポートしているクライアントは、最適なフェールオーバーターゲットを含む最新のリストをサーバから受信します。この新しいリストは、指定されたリストを上書きします。

---

## 分散トランザクションの使用

Adaptive Server ADO.NET Data Provider を使用し、それを 2 フェーズ・コミット・トランザクションに含めることができます。この機能では、分散トランザクションを管理する .NET Enterprise Services を使用する必要があります。

### Enterprise Services を使用するプログラミング

アンマネージ・コード内のサービスは、COM+ サービスと呼ばれます。COM+ サービス・インフラストラクチャは、マネージ・コードとアンマネージ・コードからアクセスできます。.NET では、これらのサービスは Enterprise Services と呼ばれます。ADO.NET を使用する Enterprise Services 内でトランザクションを扱うのは簡単です。

❖ **Enterprise Services を使用するプログラミング**

- 1 `System.EnterpriseService.ServicedComponent` からコンポーネントを抽出します。
- 2 必要なサービスとそのオプションを指定するための独自の属性 (`Transaction`、`AutoComplete` など) を指定します。属性の全リストについては、Enterprise Services のマニュアルを参照してください。

---

**注意** .NET トランザクション属性の `Timeout Option` には、明示的に `-1` または非常に大きな数を設定する必要があります。.NET のマニュアルでは、ADO.NET トランザクションのデフォルトのタイムアウト値が `0` であることが記載されていますが、これはタイムアウトしないことを意味します。ただし、実際には即時にトランザクションがタイムアウトし、トランザクション全体がロールバックされます。

---

- 3 アセンブリに署名し、ビルドします。
- 4 アセンブリを登録します。

## 分散トランザクションでの接続プロパティのサポート

次に、分散トランザクションのサポート時に使用する接続プロパティを示します。

- 分散トランザクション・プロトコル (`DistributedTransactionProtocol`) – 分散トランザクション、XA インタフェース標準、MS DTC OLE ネイティブ・プロトコルをサポートするために使用するプロトコルを指定するには、接続文字列でプロパティ `DistributedTransactionProtocol=OLE` ネイティブ・プロトコルを設定します。デフォルトのプロトコルは `XA` です。
- 密結合トランザクション (`TightlyCoupledTransaction`) – 2つのリソース・マネージャを使用する分散トランザクションで同一の Adaptive Server サーバを指定すると、「密結合トランザクション」と呼ばれる状態になります。この場合、このプロパティを `1` に設定していないと分散トランザクションが失敗することがあります。

つまり、同一の Adaptive Server サーバに対して2つのデータベース接続をオープンしてから、オープンした接続を同一の分散トランザクションに登録する場合は、`TightlyCoupledTransaction=1` を設定する必要があります。

- 登録 – `AseConnection` オブジェクトは、トランザクションがアクティブであることを特定した場合に、既存の分散トランザクションに自動的に登録します。接続を開くか、接続プールから取得したときに、自動トランザクション登録が行われます。`AseConnection` の接続文字列パラメータとして `Enlist=0` を指定することにより、この自動登録機能を無効にできます。

自動登録が無効である場合、既存トランザクションへの参照である `ITransaction` パラメータとともに `AseConnection` の `EnlistDistributedTransaction` メソッドを呼び出すことにより、既存の分散トランザクションに登録できます。`EnlistDistributedTransaction` を呼び出した後、`AseConnection` のこのインスタンスを使用したすべての更新は、このグローバル・トランザクションの一部として適用されます。したがって、グローバル・トランザクションがコミットまたはロール・バックされると、それに伴ってこのトランザクションもコミットまたはロール・バックされます。

---

**注意** `AseConnection` オブジェクトは、`EnlistDistributedTransaction` を呼び出す前に開いている必要があります。

---

ビジネス・オブジェクトをプールする場合、`EnlistDistributedTransaction` を使用できます。ビジネスオブジェクトが開いている接続とともにプールされている場合、その接続が開いているか、または接続プールから取得される場合に限り、自動トランザクション登録が行われます。プールされたビジネス・オブジェクトにより複数のトランザクションが実行される場合、そのオブジェクトの開いている接続は新しく開始したトランザクションで自動的に登録されません。この場合は、`AseConnection` の自動トランザクション登録を無効にした後、`EnlistDistributedTransaction` を使用して `AseConnection` をトランザクションに登録できます。

---

**警告！** `AseConnection` が `BeginTransaction` を使用するか、または `AseCommand` とともに `BEGIN TRANSACTION` 文を明示的に実行することによりすでにトランザクションを開始している場合、`EnlistDistributedTransaction` は例外を返します。

---

## ディレクトリ・サービス

ディレクトリ・サービスを使用すると、Adaptive Server ADO.NET Data Provider は中央にある LDAP サーバから接続やその他の情報を取得して Adaptive Server サーバに接続できます。ここでは、DSURL (Directory Service URL) を使用して、データを取得する LDAP サーバを示します。

## ディレクトリ・サービスとしての LDAP

LDAP (Lightweight Directory Access Protocol) は、ディレクトリ・サービスへの業界標準のアクセス方法です。ディレクトリ・サービスを使用すると、コンポーネントは LDAP サーバから情報を DN (識別名) で検索できます。LDAP サーバは、企業またはネットワーク上で使用されるサーバ、ユーザ、ソフトウェアの情報を格納したり管理したりします。

LDAP サーバは、Adaptive Server やクライアントが動作しているプラットフォームとは別のプラットフォームに配置できます。LDAP では通信プロトコル、およびクライアント/サーバ間で交換されるメッセージのコンテンツを定義します。LDAP サーバに格納され、取得が可能な情報は、次のとおりです。

- Adaptive Server に関する情報 (IP アドレス、ポート番号、ネットワーク・プロトコルなど)
- セキュリティ・メカニズムとフィルタ
- 高可用性コンパニオン・サーバ名

詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。

LDAP サーバの設定時に、次のアクセス制限を指定できます。

- 匿名認証 - すべてのユーザがあらゆる情報にアクセスできます。
- ユーザ名とパスワードによる認証 - Data Provider は、DSURL または ConnectString の *DSPrincipal* と *DSPassword* プロパティに指定されたユーザ名とパスワードを使用します。

## ディレクトリ・サービスの使用

ディレクトリ・サービスを使用するには、ConnectString に次のプロパティを追加します。

```
DSURL= ldap://SYBLDAP:389/dc=sybase,dc=com??one?sybase
Servername=MANGO
```

URL は LDAP URL で、LDAP ライブラリを使用して URL を解決します。

LDAP サーバの高可用性をサポートするため、DSURL は複数の URL を受け入れます。各 URL は次のようにセミコロンで区切ります。たとえば、次のようになります。

```
DSURL={ldap://SYBLDAP:389/dc=sybase,dc=com??one?sybaseServername=MANGO;
ldap://SYBLDAP1:389/dc=sybase,dc=com??one?sybaseServername=MANGO}
```

DSURL は次のように指定します。

```
ldap://hostport/dn[?attrs[?scope[?filter[?userdn?userpass]]]]
```

パラメータの意味は次のとおりです。

- *hostport* は、次のような、ホスト名とオプションの *portnumber* です。  
SYBLDAP1:389
- *dn* は、*dc=sybase,dc-com* などの検索ベースです。
- *attrs* は、LDAP に要求される属性のカンマ区切りリストです。これはブランクにします。Data Provider はすべての属性を必要とします。
- *scope* は、次の 3 つの文字列のいずれかになります。
  - *base* (デフォルト) – ベースを検索します。
  - *one* – 直下の子を検索します。
  - *sub* – サブ・ツリーを検索します。
- *filter* は検索フィルタです。通常は、*sybaseServername* です。ここをブランクにする場合は、Data Source または *ConnectionString* の *Server Name* プロパティを設定します。
- *userdn* は、ユーザの識別名 (DN: Distinguished Name) です。LDAP サーバが匿名ログインをサポートしていない場合は、ここでユーザの DN を設定するか、*ConnectionString* の *DSPrincipal* プロパティを設定します。
- *userpass* はパスワードです。LDAP サーバが匿名ログインをサポートしていない場合は、ここでパスワードを設定するか、*ConnectionString* の *DSPassword* プロパティを設定します。

## マイクロ秒の精度の time データ

Adaptive Server ADO.NET Data Provider は、SQL データ型の *bigdatetime* と *bigtime* をサポートすることで、マイクロ秒レベルの精度の time データを提供します。

*bigdatetime* と *bigtime* は同様に機能し、SQL データ型の *datetime* および *time* とデータ・マッピングが同じです。

- *bigdatetime* は、Adaptive Server のデータ型 *bigdatetime* に対応し、0000 年 1 月 1 日の 0:00:00.000000 から経過したマイクロ秒数を示します。有効な *bigdatetime* 値の範囲は、0001 年 1 月 1 日の 0:00:00.000000 から 9999 年 12 月 31 日の 23:59:59.999999 までです。
- *bigtime* は、Adaptive Server のデータ型 *bigtime* に対応し、当日の午前 0 時ちょうどから経過したマイクロ秒数を示します。有効な *bigtime* 値の範囲は、00:00:00.000000 から 23:59:59.999999 までです。

### 使用法

- Adaptive Server 15.5 への接続時に、Adaptive Server ADO.NET Data Provider は **bigdatetime** および **bigtime** データ型を使用してデータを転送します。受信した Adaptive Server カラムが **datetime** および **time** として定義されている場合でも同様です。

これは、Adaptive Server は、Adaptive Server カラムに合わせるために、Adaptive Server ADO.NET Data Provider から取得した値を暗黙的にトランケートする可能性があることを意味します。たとえば、**bigtime** の値 23:59:59.999999 は、**time** データ型の Adaptive Server カラムに 23:59:59.996 として保存されます。

- Adaptive Server 15.0.x 以前のバージョンへの接続時には、Adaptive ADO.NET Data Provider は **datetime** および **time** データ型を使用してデータを転送します。

## パスワードの暗号化

Adaptive Server ADO.NET Data Provider はデフォルトで、ネットワークを介してプレーン・テキストのパスワードを Adaptive Server に送信して認証を求めます。ただし、Adaptive Server ADO.NET Data Provider は、パスワードの対称／非対称暗号化もサポートしています。この機能を使用すると、デフォルトの動作を変更し、パスワードを暗号化してからネットワークに送信できます。

対称暗号化メカニズムでは、パスワードの暗号化と復号化に同じキーが使用されます。これに対して、非対称暗号化メカニズムでは、暗号化にはパブリック・キー、復号化には別のプライベート・キーが使用されます。プライベート・キーはネットワークを介して共有されないため、非対称暗号化の方が対称暗号化よりも安全であると考えられます。パスワードの暗号化が有効になっていて、サーバが非対称暗号化をサポートしている場合、非対称暗号化が対称暗号化の代わりに使用されます。

Sybase CSI (Common Security Infrastructure) を使用して、ログイン・パスワードとリモート・パスワードを暗号化できます。CSI 2.6 は、連邦情報処理標準 (FIPS: Federal Information Processing Standard) 140-2 に準拠しています。

## パスワードの暗号化の有効化

パスワードの暗号化を有効にするには、**EncryptPassword** 接続プロパティを設定する必要があります。この接続プロパティでは、パスワードが暗号化フォーマットで転送されるかどうかを指定します。パスワードの暗号化が有効になっていると、ログインがネゴシエートされた場合にのみ、パスワードはネットワークに送信されます。パスワードは最初に暗号化されてから送信されます。**EncryptPassword** の値は次のとおりです。



- 0 – プレーン・テキスト形式のパスワードを使用します。これはデフォルトの値です。
- 1 – 暗号化されたパスワードを使用します。サポートされていない場合、エラー・メッセージを返します。
- 2 – 暗号化されたパスワードを使用します。サポートされていない場合、プレーン・テキスト形式のパスワードを使用します。

---

**注意** 非対称暗号化を使用するには、非対称暗号化をサポートするサーバ (Adaptive Server 15.0.2 など) が必要です。非対称暗号化では、追加の処理時間が必要になり、ログインに若干の遅延が発生する可能性があります。

---

#### 例

この例では、ログインがネゴシエートされ、パスワードの暗号化と送信が行われるまで、ネットワークに sapass は送信されません。

```
AseConnection.ConnectionString=  
"Data Source=MANGO;" +  
  "Port = 5000;" +  
  "Database=pubs2;" +  
  "UID=sa;" +  
  "PWD=sapass;" +  
  "EncryptPassword=1;";
```

## パスワード有効期限の処理

各企業は、自社のデータベース・システム用に独自のパスワード・ポリシーを設定しています。ポリシーに応じて、パスワードは特定の日時で期限切れになります。パスワードがリセットされない限り、データベースに接続した Adaptive Server ドライバはパスワード期限切れエラーをスローし、isql を使用してパスワードを変更するようユーザに要求します。パスワード変更機能を使用すると、別のツールを使用しなくても、期限切れになったパスワードを変更できます。

Adaptive Server ADO.NET Data Provider は、ChangePassword メソッドをサポートしています。これにより、管理者による操作を必要とせずに、期限切れになったパスワードをアプリケーション側で変更できます。詳細については、Microsoft Developer Network (<http://msdn2.microsoft.com/>) で ChangePassword メソッドを検索してください。

## SSL の使用

SSL (Secure Sockets Layer) は、クライアントとサーバ間、およびサーバ同士の接続において、ワイヤ・レベルまたはソケット・レベルで暗号化されたデータを送信する業界標準です。

### SSL ハンドシェイク

サーバとクライアントが、安全な暗号化セッションをネゴシエートして合意してから、SSL 接続が確立されます。これは、「SSL ハンドシェイク」と呼ばれています。クライアント・アプリケーションが接続を要求すると、SSL 対応サーバが証明書を提示し、ID を証明してから、データを送信します。基本的に、SSL ハンドシェイクは次の手順によって構成されています。

- 1 クライアントがサーバに接続要求を送信します。要求には、クライアントがサポートしている SSL (または TLS: Transport Layer Security) オプションが含まれています。
- 2 サーバは、証明書とサポートされている CipherSuite のリストを返します。これには、SSL/TLS サポート・オプション、キー交換で使用されるアルゴリズム、デジタル署名が含まれます。
- 3 クライアントとサーバの両者が1つの CipherSuite について合意すると、安全で暗号化されたセッションが確立されます。

SSL ハンドシェイクと SSL/TLS プロトコルの詳細については、Internet Engineering Task Force Web site (<http://www.ietf.org>) を参照してください。

### パフォーマンス

安全なセッションの確立に必要な、追加のオーバーヘッドが生じます。データを暗号化するとサイズが増え、情報の暗号化と復号化に追加の計算が必要になるからです。一般に、SSL ハンドシェイク中に生じる I/O の増加によって、ユーザ・ログインにかかる時間が 10 ~ 20 倍になることがあります。

### CipherSuite

SSL ハンドシェイク中に、クライアントとサーバは、CipherSuite を介して共通のセキュリティ・プロトコルをネゴシエートします。CipherSuite は、SSL プロトコルで使用されるキー交換アルゴリズム、ハッシュ方式、暗号化方式の優先順位リストです。CipherSuite の詳細については、IETF の Web サイト (IETF organization Web site (<http://www.ietf.org>)) を参照してください。

デフォルトでは、クライアントとサーバの両方がサポートしている最も強力な CipherSuite は、SSL ベースのセッションに使用される CipherSuite です。サーバ接続属性は、接続文字列か、LDAP などのディレクトリ・サービスによって指定されます。

Adaptive Server ADO.NET Data Provider と Adaptive Server は、SSL Plus ライブラリ API と暗号エンジンである Security Builder (両方とも Certicom Corp 製) で使用可能な CipherSuite をサポートしています。

---

**注意** 次に示す CipherSuite のリストは TLS 仕様に準拠しています。TLS は、SSL 3.0 を拡張したものであり、SSL バージョン 3.0 CipherSuite の別名です。

---

Adaptive Server ADO.NET Data Provider でサポートしている CipherSuite は、次のとおりです。

- TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA
- TLS\_RSA\_WITH\_RC4\_128\_SHA
- TLS\_RSA\_WITH\_RC4\_128\_MD5
- TLS\_DHE\_DSS\_WITH\_3DES\_EDE\_CBC\_SHA
- TLS\_DHE\_DSS\_WITH\_RC4\_128\_SHA
- TLS\_DHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA
- TLS\_RSA\_WITH\_DES\_CBC\_SHA
- TLS\_DHE\_DSS\_WITH\_DES\_CBC\_SHA
- TLS\_DHE\_RSA\_WITH\_DES\_CBC\_SHA
- TLS\_RSA\_EXPORT1024\_WITH\_DES\_CBC\_SHA
- TLS\_RSA\_EXPORT1024\_WITH\_RC4\_56\_SHA
- TLS\_DHE\_DSS\_EXPORT1024\_WITH\_RC4\_56\_SHA
- TLS\_DHE\_DSS\_EXPORT1024\_WITH\_DES\_CBC\_SHA
- TLS\_RSA\_EXPORT\_WITH\_RC4\_40\_MD5
- TLS\_RSA\_EXPORT\_WITH\_DES40\_CBC\_SHA
- TLS\_DHE\_DSS\_EXPORT\_WITH\_DES40\_CBC\_SHA
- TLS\_DHE\_RSA\_EXPORT\_WITH\_DES40\_CBC\_SHA

## Adaptive Server ADO.NET Data Provider の SSL

SSL には、次のセキュリティ・レベルがあります。

- SSL セッションが確立されると、ユーザ名とパスワードが暗号化された安全で接続によって送信されます。
- SSL 対応サーバへの接続を確立すると、サーバは接続対象のサーバであることを自己認証し、暗号化された SSL セッションが開始され、データが送信されます。
- サーバ証明書のデジタル署名を比較して、サーバから受信した情報が転送中に変更されたかどうかを判断します。

## 証明書によるサーバの検証

Adaptive Server ADO.NET Data Provider が SSL 対応サーバにクライアント接続する場合、サーバは証明書ファイルが必要です。これは、サーバの証明書と暗号化されたプライベート・キーで構成されます。また、証明書は署名/認証局 (CA: Certification Authority) によってデジタル署名されている必要もあります。Adaptive Server ADO.NET Data Provider のクライアント・アプリケーションが Adaptive Server へのソケット接続を確立する方法は、既存のクライアント接続の確立方法と同じです。ネットワーク・トランスポート・レベルの接続コールがクライアント側で完了し、承認コールがサーバ側で完了すると、ソケット上で SSL ハンドシェイクが行われ、その後でユーザ・データが送信されます。

SSL 対応サーバに正しく接続するには、次のことが必要です。

- 1 クライアント・アプリケーションが接続要求を行った場合、SSL 対応サーバは証明書を提示しなければなりません。
- 2 クライアント・アプリケーションは、証明書に署名した CA を認識しなければなりません。  
「信頼された」CA すべてを含んだリストは、次に示す信頼されたルート・ファイルにあります。

詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

### 信頼されたルート・ファイル

既知で信頼された CA のリストは、信頼されたルート・ファイルに保管されています。エンティティ (クライアント・アプリケーション、サーバ、ネットワーク・リソースなど) に既知の CA の証明書がある以外は、信頼されたルート・ファイルは証明書ファイルのフォーマットと同じです。システム・セキュリティ担当者が、標準 ASCII テキスト・エディタを使って、信頼された CA を追加したり、削除したりします。

アプリケーション・プログラムでは、ConnectionString の `TrustedFile=trusted file path` プロパティを使用して、信頼されたルート・ファイルの位置を指定します。最も一般的に使用される CA (Thawte, Entrust, Baltimore, VeriSign, RSA) が記載された信頼されるルート・ファイルは `%SYBASE%\%ini%trusted.txt` にインストールされています。

## SSL 接続の有効化

Data Provider で SSL を有効化するには、ConnectionString プロパティに `Encryption=ssl; TrustedFile=<信頼されたファイル>` を追加します。これで、AseConnection が Adaptive Server サーバと SSL 接続をネゴシエートするようになります。次に例を示します。

```
AseConnection.ConnectionString=
    "Data Source=MANGO;" +
    "Port = 5000;" +
    "Database=pubs2;" +
    "UID=sa;" +
```

```
"PWD=sapass;" +  
"Encryption=ssl;" +  
"TrustedFile='c:¥sybase¥ini¥trusted.txt';";
```

**注意** SSL を使用するように、Adaptive Server を設定してください。SSL の詳細については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。

## 高可用性システムでのフェールオーバーの使用

高可用性クラスタには、2 つ以上のマシンが含まれます。これらのマシンは、1 つのマシン（またはアプリケーション）がダウンした場合にもう 1 つのマシンが両方のマシンの負荷を処理するように設定されています。このようなマシンのそれぞれを、高可用性クラスタのノードといいます。一般的に、高可用性クラスタはシステムが常に稼働していなければならないような環境で使用します。たとえば、クライアントが 1 年 365 日絶えず接続する銀行のシステムなどです。

フェールオーバーによって、Adaptive Server をアクティブ/アクティブ設定またはアクティブ/パッシブ設定の高可用性クラスタで運用できます。

フェールオーバーが発生すると、プライマリ・コンパニオンに接続していたクライアントは、フェールオーバー・プロパティを使用して、自動的にセカンダリ・コンパニオンへのネットワーク接続を再確立します。フェールオーバーを有効にするには、接続プロパティ `HASession` を “1”（デフォルト値は “0”）に設定します。このプロパティを設定しないと、サーバでフェールオーバーが設定されていても、セッションではフェールオーバーが行われません。`SecondaryServer` プロパティと `SecondaryPort` プロパティも設定します。

使用するシステムの高可用性設定の詳細については、Adaptive Server のマニュアル『高可用性システムにおける Sybase フェールオーバーの使用』を参照してください。

トランザクションでフェールオーバーが発生した場合、フェールオーバー前にデータベースにコミットされた変更のみが保持されます。フェールオーバーが発生すると、プロバイダは、セカンダリ・サーバへの再接続を試行します。セカンダリ・サーバへの接続が確立されると、ADO.NET Data Provider は、フェールオーバーの発生を示すメッセージとともに `AseFailoverException` を返します。クライアントは、新しい接続を使用して、失敗したトランザクションを再適用しなければなりません。セカンダリ・サーバへの接続が確立できない場合は、接続が失われたことを示すメッセージとともに、ADO.NET Data Provider で通常の `AseException` が発生します。次に例を示します。

```
AseConnection.ConnectionString =
    "Data Source='tpsun1';" +
    "Port = 5000;" +
    "Database=pubs2;" +
    "User ID=sa;" +
    "Password=sapass;" +
    "HASession=1;" +
    "Secondary Data Source='tpsun2';" +
    "Secondary Server Port=5000";
```

次のコードは、**AseFailoverException** の検出方法を示しています。

```
....
Open connection
...more code

try
{
    using (AseDataReader rdr =
        selectCmd.ExecuteReader())
    {
        ....
    }
}
catch (AseFailoverException)
{
    //Make sure that you catch AseFailoverException
    //before AseException as AseFailoverException is
    //derived from AseException

    //HA has occurred.The application has successfully
    //connected to the secondary server.All uncommitted
    //transactions have been rolled back.

    //You could retry your transactions or prompt user
    //for a retry operation
}
catch (AseException)
{
    //Either some other problem or the Failover did not
    //successfully connect to the secondary server.Apps.
    //should react accordingly
}
}
```

## Kerberos 認証の使用

Kerberos は、簡単なログイン認証と相互のログイン認証を提供する業界標準のネットワーク認証システムです。Kerberos を使用することで、さまざまなアプリケーションにわたるシングル・サインオンをきわめて安全な環境内で行えます。ネットワークでパスワードを渡す代わりに、Kerberos サーバがユーザのパスワードと使用可能なサービスのパスワードの暗号化されたバージョンを保持します。

さらに Kerberos では、機密性とデータの整合性を維持するために暗号化を使用します。

Adaptive Server と Adaptive Server ADO.NET Data Provider は、Kerberos 接続をサポートします。Adaptive Server ADO.NET Data Provider は特に、MIT、CyberSafe、Active Directory の KDC (Key Distribution Center) をサポートします。

### プロセスの概要

Kerberos 認証プロセスは次のように機能します。

- 1 クライアント・アプリケーションは、特定のサービスにアクセスするための「チケット」を Kerberos サーバに要求します。
- 2 Kerberos サーバは、2つのパケットを含むチケットをクライアントに返します。第1のパケットはユーザ・パスワードにより暗号化されます。第2のパケットはサービス・パスワードにより暗号化されます。これらの各パケット内に「セッション・キー」が含まれます。
- 3 クライアントは、セッション・キーを取得するためにユーザ・パケットを復号化します。
- 4 クライアントは新しい認証パケットを作成し、それをセッション・キーにより暗号化します。
- 5 クライアントは、認証パケットとサービス・パケットをサービスに送信します。
- 6 サービスは、セッション・キーを取得するためにサービス・パケットを復号化し、ユーザ情報を取得するために認証パケットを復号化します。
- 7 サービスは、認証パケットからのユーザ情報と、サービス・パケットにも含まれているユーザ情報を比較します。両者が一致する場合、ユーザは認証済みです。
- 8 サービスは、認証パケットに含まれる検証データに加えてサービス固有の情報を含む確認パケットを作成します。
- 9 サービスは、このデータをセッション・キーとともに暗号化し、それをクライアントに返します。

- 10 クライアントは、パケットを復号化するために Kerberos から受信したユーザ・パケット内のセッション・キーを使用し、サービスがそれ自身の主張に一致しているかどうかを検証します。

こうした方法で、ユーザとサービスは相互に認証されます。以後、クライアントとサービス (この場合は Adaptive Server データベース・サーバ) の間の通信はすべて、セッション・キーにより暗号化されます。これにより、サービスとクライアント間で送信されるすべてのデータが望ましくない閲覧者から正しく保護されます。

## 稼働条件

認証システムとして Kerberos を使用するには、Kerberos に認証を委任するように Adaptive Server Enterprise を設定します。詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。Windows では、クライアント・ライブラリとともに Kerberos クライアント・ライブラリがインストールされます。

Adaptive Server ADO.NET Data Provider で Kerberos を使用するには、Kerberos に対して MIT/CyberSafe Client ライブラリを設定し、Adaptive Server を有効にします。

## Kerberos 認証の有効化

Adaptive Server ADO.NET Data Provider に対して Kerberos を有効にするには、プログラムに次の行を追加します。

```
AuthenticationClient=<one of 'mitkerberos' or  
'cybersafekerberos' or 'activedirectory' and  
ServerPrincipal=<ASE server name>
```

ここで *<ASE server name>* は、KDC (Key Distribution Center) 内で設定された論理名またはプリンシパルです。

Adaptive Server ADO.NET Data Provider はこの情報を使用して、設定された KDC および Adaptive Server サーバと Kerberos 認証をネゴシエートします。Windows では、**activedirectory** を選択することで追加設定を不要にできます。

Kerberos クライアント・ライブラリは、さまざまな KDC 間で互換性を持ちます。たとえば、KDC が Microsoft Active Directory の場合でも、**mitkerberos** と同じ **AuthenticationClient** を設定できます。

Kerberos クライアントで別のキャッシュ内の TGT を検索する場合は、**userprincipal** メソッドを指定できます。

**SQL\_DRIVER\_NOPROMPT** とともに **SQLDriverConnect** を使用する場合、**ConnectionString** は次のようになります。

```
"Driver=Adaptive Server Enterprise;UID=sa;  
PWD='';Server=sampleserver;
```



```
Port=4100;Database=pubs2;
AuthenticationClient=mitkerberos;
ServerPrincipal=MANGO;"
```

## Key Distribution Center からの初期チケットの取得

Kerberos 認証を使用するには、Key Distribution Center から TGT (Ticket Granted Ticket) と呼ばれる初期チケットを生成します。このチケットを取得する手順は、使用する Kerberos ライブラリに応じて異なります。詳細については、ベンダのマニュアルを参照してください。

### ❖ MIT Kerberos クライアント・ライブラリ用の TGT の生成

- 1 コマンド・ラインに次のように入力して kinit ユーティリティを開始します。

```
% kinit
```

- 2 your\_name@YOUR.REALM などの kinit ユーザ名を入力します。

- 3 my\_password など、your\_name@YOUR.REALM のパスワードを入力します。パスワードを入力すると、kinit ユーティリティにより TGT (Ticket Granting Ticket) に対する要求が認証サーバに送信されます。

このパスワードは、キーの計算のために使用されます。そのキーは、応答の一部を復号化するために使用されます。この応答には、セッション・キーに加えて要求の確認が含まれます。パスワードを正しく入力していれば、この段階で TGT が取得されています。

- 4 コマンド・ラインに次のように入力して、TGT が取得されていることを確認します。

```
% klist
```

klist コマンドの結果は次のようになるはずです。

```
Ticket cache: /var/tmp/krb5cc_1234
Default principal: your_name@YOUR.REALM
Valid starting Expires Service principal
24-Jul-95 12:58:02 24-Jul-95 20:58:15 krbtgt/YOUR.REALM@YOUR.REALM
```

#### 結果の説明

**Ticket cache :** チケット・キャッシュ・フィールドにより、どのファイルにクレデンシャル・キャッシュが含まれているかがわかります。

**Default principal :** デフォルトのプリンシパルは、TGT を所有するユーザ (この場合はユーザ自身) のログインです。

**Valid starting/Expires/Service principal :** 以降の出力は、既存のチケットのリストです。これは要求した最初のチケットであるため、1つのチケットのみがリストに含まれています。サービス・プリンシパル (krbtgt/YOUR.REALM@YOUR.REALM) は、このチケットが TGT であることを示しています。このチケットは、約8時間有効であることに注意してください。



## サポートされている Microsoft ADO.NET の機能

この章では、Adaptive Server ADO.NET Data Provider でサポートされている Microsoft ADO.NET の機能について説明します。各機能の詳細については、Microsoft Developer Network (<http://msdn.microsoft.com>) を参照してください。

トピック名	ページ
サポートされている Microsoft ADO.NET 2.0 の機能	99
Adaptive Server 用の Microsoft Enterprise Library DAAB	100
Microsoft ADO.NET Entity Framework と LINQ	101

### サポートされている Microsoft ADO.NET 2.0 の機能

Adaptive Server ADO.NET Data Provider は、以下の Microsoft ADO.NET 2.0 の機能をサポートしています。

- プロバイダ・ファクトリ
- プロバイダ統計情報
- バルク更新
- バルク・コピー
- 非同期コマンド
- プールのクリアに対応する拡張プール・サポート
- 共通ベース・クラス
- データベース・メタデータ

## Adaptive Server 用の Microsoft Enterprise Library DAAB

Adaptive Server ADO.NET Data Provider 2.155 は、Enterprise Library 4.1 DAAB (Data Access Application Block) の機能を拡張して Adaptive Server をサポートします。DAAB はクラスのコレクションであり、データベース・インスタンスの作成やデータベース・レコードの更新など、一般的なデータベース機能を簡素化します。また、DAAB は、データベース固有の機能のカプセル化も行います。これにより、データベースを意識しないアプリケーション設計が可能になります。

Adaptive Server クラス用の DAAB は、Microsoft .NET Framework 3.5 と Microsoft Visual Studio 2008 でサポートされています。Adaptive Server 用の DAAB を使用するには、Microsoft Enterprise Library 4.1 を更新する必要があります。

### ❖ Microsoft Enterprise Library 4.1 の更新

1 DbProviderMapping.cs を更新します。次の行の後に

```
public const string DefaultOracleProviderName =  
    "System.Data.OracleClient";
```

次の行を追加します。

```
/// <summary>  
/// Default name for the Sybase managed provider.  
/// </summary>  
public const string DefaultSybaseProviderName = "Sybase.Data.AseClient";
```

2 DatabaseConfigurationView.cs に、次の行を追加します。

- 次の行の後に

```
using Microsoft.Practices.EnterpriseLibrary.Data.Sql;
```

次の行を追加します。

```
using Microsoft.Practices.EnterpriseLibrary.Data.Sybase;  
using Sybase.Data.AseClient;
```

- 次の行の後に

```
private static readonly DbProviderMapping defaultGenericMapping = new  
    DbProviderMapping(DbProviderMapping.DefaultGenericProviderName,  
    typeof(GenericDatabase));
```

次の行を追加します。

```
private static readonly DbProviderMapping defaultSybaseMapping = new  
    DbProviderMapping(DbProviderMapping.DefaultSybaseProviderName,  
    typeof(AseDatabase));
```

- 次の行の後に

```
if (OracleClientFactory.Instance == providerFactory)  
    return defaultOracleMapping;
```

次の行を追加します。

```
if (AseClientFactory.Instance == providerFactory)
    return defaultSybaseMapping;
```

- 3 次のアセンブリ参照を Enterprise Library *DataAccessApplicationBlock* の Data ソリューションに追加します。

```
Sybase.AdoNet2.AseClient.dll
Sybase.EnterpriseLibrary.AseClient.dll
```

- 4 Enterprise Library 全体を再構築します。

Enterprise Library DAAP の詳細については、Microsoft Developer Network (<http://msdn.microsoft.com>) を参照してください。

## Microsoft ADO.NET Entity Framework と LINQ

Adaptive Server ADO.NET Data Provider は、Visual Studio LINQ (Language-Integrated Query) と、LINQ-to-SQL コンポーネントを含む、Microsoft ADO.NET Entity Framework をサポートしています。ただし、Microsoft ADO.NET Entity Framework の制限により、次の処理はサポートされていません。

- LINQ Contains 拡張メソッドの使用。Contains は SQL の IN 句に対応します。
- LINQ 拡張メソッドの作成。
- エンティティ・クラス間の関連付けの作成。

ADO.NET Entity Framework と LINQ の利点としては、リレーショナル・ストレージ・スキーマの概念モデルを操作できることが挙げられます。これにより、データ指向アプリケーションの開発および保守に伴う作業を簡素化できます。Microsoft ADO.NET Entity Framework と LINQ を使用するには、*Sybase.AdoNet2.AseClient.dll* への参照を追加します。

ADO.NET Entity Framework と LINQ の詳細については、Microsoft Developer Network (<http://msdn.microsoft.com>) を参照してください。



# Adaptive Server ADO.NET Data Provider API リファレンス

この章では、Adaptive Server ADO.NET Data Provider の API について説明します。最新の API のマニュアルと、Microsoft ADO.NET インタフェースの実装に該当するプロパティとメソッドの詳細については、Adaptive Server ADO.NET のオンライン・ヘルプを参照してください。オンライン・ヘルプにアクセスするには、Microsoft Document Explorer を開いて、[-Sybase ASE ADO.NET Data Provider] を選択します。詳細とその使用例については、Microsoft .NET Framework のドキュメントも参照してください。

トピック名	ページ
<a href="#">AseBulkCopy クラス</a>	104
<a href="#">AseBulkCopyColumnMapping クラス</a>	106
<a href="#">AseBulkCopyColumnMappingCollection クラス</a>	108
<a href="#">AseBulkCopyOptions 列挙型</a>	109
<a href="#">AseClientFactory クラス</a>	110
<a href="#">AseClientPermission クラス</a>	112
<a href="#">AseClientPermissionAttribute クラス</a>	112
<a href="#">AseCommand クラス</a>	113
<a href="#">AseCommandBuilder クラス</a>	119
<a href="#">AseConnection クラス</a>	125
<a href="#">AseConnectionPool クラス</a>	135
<a href="#">AseConnectionPoolManager クラス</a>	136
<a href="#">AseDataAdapter クラス</a>	137
<a href="#">AseDataReader クラス</a>	144
<a href="#">AseDbType 列挙型</a>	157
<a href="#">AseError クラス</a>	159
<a href="#">AseErrorCollection クラス</a>	161
<a href="#">AseException クラス</a>	162
<a href="#">AseFailoverException クラス</a>	163
<a href="#">AseInfoMessageEventArgs クラス</a>	164
<a href="#">AseInfoMessageEventHandler デリゲート</a>	164
<a href="#">AseParameter クラス</a>	165
<a href="#">AseParameterCollection クラス</a>	169
<a href="#">AseRowsCopiedEventArgs クラス</a>	173
<a href="#">AseRowsCopiedEventHandler デリゲート</a>	173
<a href="#">AseRowUpdatedEventArgs クラス</a>	174

トピック名	ページ
<a href="#">AseRowUpdatingEventArgs クラス</a>	176
<a href="#">AseRowUpdatedEventHandler デリゲート</a>	177
<a href="#">AseRowUpdatingEventHandler デリゲート</a>	178
<a href="#">AseTransaction クラス</a>	178
<a href="#">TraceEnterEventHandler デリゲート</a>	179
<a href="#">TraceExitEventHandler デリゲート</a>	180

## AseBulkCopy クラス

説明	データ・ソースから Adaptive Server テーブルにデータをコピーします。
実装	IDisposable

### AseBulkCopy コンストラクタ

説明	AseBulkCopy クラスの新しいインスタンスを初期化します。
構文 1	<code>void AseBulkCopy( AseConnection connection )</code>
構文 2	<code>void AseBulkCopy( string connString )</code>
構文 3	<code>void AseBulkCopy( string connString, AseBulkCopyOptions copyOptions )</code>
構文 4	<code>void AseBulkCopy( AseConnection connection, AseBulkCopyOptions copyOptions, AseTransaction externalTransaction )</code>
パラメータ	<p><b>connection</b> バルク・コピー操作を実行する対象の Adaptive Server 接続。</p> <p><b>connString</b> AseBulkCopy を実行する対象の接続に対する接続文字列。</p> <p><b>copyOptions</b> AseBulkCopy オプション。「<a href="#">AseBulkCopyOptions 列挙型</a>」(109 ページ) を参照してください。</p> <p><b>externalTransaction</b> Adaptive Server 接続で実行するトランザクション。</p>

### Close メソッド

説明	AseBulkCopy オブジェクトによって使用されるリソースを解放します。
構文	<code>void Close()</code>



## Dispose メソッド

説明	AseBulkCopy オブジェクトによって使用されるリソースを解放します。
構文	<code>void Dispose()</code>
実装	<code>IDisposable.Dispose()</code>

## Finalize メソッド

説明	AseBulkCopy オブジェクトによって使用されるリソースを解放します。
構文	<code>void Finalize()</code>

## WriteToServer メソッド

説明	データ・ソースから出力先テーブルにデータを書き込みます。
構文 1	<code>void WriteToServer( DataRow[] rows )</code>
構文 2	<code>void WriteToServer( DataTable table )</code>
構文 3	<code>void WriteToServer( IDataReader reader )</code>
構文 4	<code>void WriteToServer( DataTable table, DataRowState rowState )</code>
パラメータ	<b>reader</b> <code>IDataReader</code> 。インタフェースのデータはデータ・ソースに書き込まれます。 <b>rows</b> <code>DataTable</code> 内のデータ・ロー。データは、データソースに書き込まれます。 <b>rowState</b> コピーの対象にできるデータ・ローの状態を指定します。 <b>table</b> <code>DataTable</code> 。このテーブルのデータはデータ・ソースに書き込まれます。

## BatchSize プロパティ

説明	バッチ内のロー数を表します。
構文	<code>int BatchSize</code>
アクセス	読み込みと書き込み

## BulkCopyTimeout プロパティ

説明	操作がタイムアウトになるまでの実行時間 (秒数) を表します。
構文	<code>int BulkCopyTimeout</code>
アクセス	読み込みと書き込み

## ColumnMappings プロパティ

説明	カラム・マッピングのコレクションを表します。
構文	AseBulkCopyColumnMappingCollection <b>ColumnMappings</b>
アクセス	読み込み専用

## DestinationTableName プロパティ

説明	データの書き込み先のテーブル名を表します。
構文	string <b>DestinationTableName</b>
アクセス	読み込みと書き込み

## NotifyAfter プロパティ

説明	AseRowsCopied イベントをトリガする前に処理するロー数を表します。
構文	int <b>NotifyAfter</b>
アクセス	読み込みと書き込み

## AseRowsCopied イベント

説明	NotifyAfter で指定したロー数が処理されるたびに発生します。
構文	event AseRowsCopiedEventHandler <b>AseRowsCopied</b>

## AseBulkCopyColumnMapping クラス

説明	データ・ソース内のカラムをコピー先テーブル内のカラムにマッピングします。
----	--------------------------------------

## AseBulkCopyColumnMapping コンストラクタ

説明	AseBulkCopyColumnMapping クラスの新しいインスタンスを初期化します。
構文 1	void <b>AseBulkCopyColumnMapping</b> ()
構文 2	void <b>AseBulkCopyColumnMapping</b> ( int <i>sourceIdx</i> , int <i>dbIdx</i> )
構文 3	void <b>AseBulkCopyColumnMapping</b> ( string <i>sourceName</i> , string <i>dbName</i> )

パラメータ	<b>dblNx</b> コピー先テーブル内のカラム・インデックスを表します。
	<b>sourceInx</b> データ・ソース内のカラム・インデックスを表します。
	<b>dbName</b> コピー先テーブル内のカラム名を表します。
	<b>sourceName</b> データ・ソース内のカラム名を表します。

## Equals メソッド

説明	2つのカラム・マッピングが同じであるかどうかを判別します。
構文	<code>bool Equals( Object obj )</code>
パラメータ	<b>obj</b> 比較対象のオブジェクト。
戻り値	オブジェクトが同じ場合は true、それ以外の場合は false。

## DestinationColumn プロパティ

説明	マッピング先のカラム名を表します。
構文	<code>string DestinationColumn</code>
アクセス	読み込みと書き込み

## DestinationOrdinal プロパティ

説明	マッピング先のカラム・インデックスを表します。
構文	<code>int DestinationOrdinal</code>
アクセス	読み込みと書き込み

## SourceColumn プロパティ

説明	データ・ソース内のカラム名を表します。
構文	<code>string SourceColumn</code>
アクセス	読み込みと書き込み

## SourceOrdinal プロパティ

説明	データ・ソース内のカラム・インデックスを表します。
構文	<code>int SourceOrdinal</code>
アクセス	読み込みと書き込み

## AseBulkCopyColumnMappingCollection クラス

説明 カラム・マッピングのコレクションを表します。

### AseBulkCopyColumnMappingCollection コンストラクタ

説明 AseBulkCopyColumnMappingCollection クラスの新しいインスタンスを初期化します。

構文 `void AseBulkCopyColumnMappingCollection()`

### Add メソッド

説明 新しいマッピングをコレクションに追加します。

構文 `int Add( AseBulkCopyColumnMapping value )`

パラメータ **value** 追加対象のカラム・マッピング。

戻り値 カラム・マッピングの追加先のインデックス。

### Contains メソッド

説明 指定されたマッピングに対応するコレクションを検索します。

構文 `bool Contains( AseBulkCopyColumnMapping value )`

パラメータ **value** 検索対象のカラム・マッピング。

戻り値 指定されたマッピングがコレクションに含まれている場合は true、それ以外の場合は false。

### IndexOf メソッド

説明 指定されたマッピングのインデックスを取得します。

構文 `int IndexOf( AseBulkCopyColumnMapping value )`

パラメータ **value** インデックスの取得元のカラム・マッピング。

戻り値 指定されたマッピングのインデックス。

### Insert メソッド

説明 指定されたインデックスでマッピングを挿入します。

構文 `void Insert( int index, AseBulkCopyColumnMapping value )`

パラメータ **index** 新しいマッピングの挿入先となるコレクションの位置。  
**value** 追加対象のカラム・マッピング。

## Validate メソッド

説明 カスタム・プロセスを実行して、指定されたオブジェクトをコレクションに追加できることを検証します。

構文 `void OnValidate( Object value )`

パラメータ **value** 検証対象のオブジェクト。

## Remove メソッド

説明 コレクションからマッピングを削除します。

構文 `void Remove( AseBulkCopyColumnMapping value )`

パラメータ **value** 削除対象のカラム・マッピング。

## AseBulkCopyOptions 列挙型

説明 AseBulkCopy クラスで使用可能な動作オプションを指定します。

構文 `enum AseBulkCopyOptions`

メンバ

メンバ名	AseBulkCopy の動作
CheckConstraints	挿入操作時の制約をチェックします。
Default	デフォルトでは、すべてのオプションがオフになっています。
FireTriggers	挿入操作時にトリガを起動するようにサーバに指示します。
KeepIdentity	データ・ソースの ID 値を使用します。KeepIdentity が選択されていない場合、Adaptive Server が ID を割り当てます。
KeepNulls	null 値を挿入します。KeepNulls が選択されていない場合、デフォルト値で null 値が置き換えられます。
TableLock	バルク・コピー操作時にテーブル全体をロックします。
UseInternalTransaction	設定した場合、バルク・コピーの各バッチはトランザクション内に配置されます。

## AseClientFactory クラス

説明	Adaptive Server ADO.NET Data Provider データ・ソース・クラスのインスタンスを作成する場合に使用できる一連のメソッドを表します。
基本クラス	DbProviderFactory

## AseClientFactory コンストラクタ

説明	AseClientFactory クラスの新しいインスタンスを初期化します。
構文	<code>void AseClientFactory()</code>

## Instance フィールド

説明	厳密に型指定されたデータ・オブジェクトを取得する場合に使用できる AseClientFactory のインスタンスを表します。
----	---

## CreateCommand メソッド

説明	System.Data.Common.DbCommand クラスを実装する Adaptive Server ADO.NET Data Provider クラスの新しいインスタンスを返します。
構文	<code>DbCommand CreateCommand()</code>
戻り値	System.Data.Common.DbCommand の新しいインスタンス。

## CreateCommandBuilder メソッド

説明	System.Data.Common.DbCommandBuilder クラスを実装する Adaptive Server ADO.NET Data Provider クラスの新しいインスタンスを返します。
構文	<code>DbCommandBuilder CreateCommandBuilder()</code>
戻り値	System.Data.Common.DbCommandBuilder の新しいインスタンス。

## CreateConnection メソッド

説明	System.Data.Common.DbConnection クラスを実装する Adaptive Server ADO.NET Data Provider クラスの新しいインスタンスを返します。
構文	<code>DbConnection CreateConnection()</code>
戻り値	System.Data.Common.DbConnection の新しいインスタンス。

## CreateConnectionStringBuilder メソッド

説明	System.Data.Common.DbConnectionStringBuilder クラスを実装する Adaptive Server ADO.NET Data Provider クラスの新しいインスタンスを返します。
構文	DbConnectionStringBuilder <b>CreateConnectionStringBuilder()</b>
戻り値	System.Data.Common.DbConnectionStringBuilder の新しいインスタンス。

## CreateDataAdapter メソッド

説明	System.Data.Common.DbDataAdapter クラスを実装する Adaptive Server ADO.NET Data Provider クラスの新しいインスタンスを返します。
構文	DbDataAdapter <b>CreateDataAdapter()</b>
戻り値	System.Data.Common.DbDataAdapter の新しいインスタンス。

## CreateDataSourceEnumerator メソッド

説明	System.Data.Common.DbDataSourceEnumerator クラスを実装する Adaptive Server ADO.NET Data Provider クラスの新しいインスタンスを返します。
構文	DbDataSourceEnumerator <b>CreateDataSourceEnumerator()</b>
戻り値	System.Data.Common.DbDataSourceEnumerator の新しいインスタンス。

## CreateParameter メソッド

説明	System.Data.Common.DbParameter クラスを実装する Adaptive Server ADO.NET Data Provider クラスの新しいインスタンスを返します。
構文	DbParameter <b>CreateParameter()</b>
戻り値	System.Data.Common.DbParameter の新しいインスタンス。

## CreatePermission メソッド

説明	System.Data.Common.CreatePermission クラスを実装する Adaptive Server ADO.NET Data Provider クラスの新しいインスタンスを返します。
構文	CodeAccessPermission <b>CreatePermission( PermissionState state )</b>
パラメータ	<b>state</b> パーミッションにリソースへのアクセス権を設定するかどうかを指定します。
戻り値	System.Data.Common.CreatePermission の新しいインスタンス。

## CanCreateDataSourceEnumerator プロパティ

説明	特定の System.Data.Common.DbProviderFactory が System.Data.Common.DbDataSourceEnumerator クラスをサポートするかどうかを指定します。
構文	bool <b>CanCreateDataSourceEnumerator</b>
アクセス	読み込み専用
プロパティ値	System.Data.Common.DbProviderFactory のインスタンスが System.Data.Common.DbDataSourceEnumerator クラスをサポートする場合は true、それ以外の場合は false。

## AseClientPermission クラス

説明	Adaptive Server ADO.NET Data Provider で、ユーザが Adaptive Server Enterprise データ・ソースに適切なセキュリティ・レベルでアクセスできるようにします。
基本クラス	DBDataPermission

## AseClientPermission コンストラクタ

説明	AseClientPermission クラスの新しいインスタンスを初期化します。
構文 1	void <b>AseClientPermission</b> ( )
構文 2	void <b>AseClientPermission</b> ( PermissionState state )
構文 3	void <b>AseClientPermission</b> ( PermissionState state, bool allowBlankPassword )
パラメータ	<b>state</b> PermissionState 値の 1 つ。 <b>allowBlankPassword</b> ブランクのパスワードが許可されるかどうかを示す。

## AseClientPermissionAttribute クラス

説明	カスタム・セキュリティ属性にセキュリティの操作を関連付けます。
基本クラス	DBDataPermissionAttribute



## AseClientPermissionAttribute コンストラクタ

説明	AseClientPermissionAttribute クラスの新しいインスタンスを初期化します。
構文	<code>void AseClientPermissionAttribute( SecurityAction action )</code>
パラメータ	<b>action</b> 宣言されているセキュリティを使用して実行可能な操作を表す SecurityAction 値の 1 つ。
戻り値	AsePermissionAttribute オブジェクト。

## CreatePermission メソッド

説明	属性のプロパティに応じて設定される AsePermission オブジェクトを返します。
構文	<code>IPermission CreatePermission( )</code>

## AseCommand クラス

説明	Adaptive Server に対して実行するコマンドを表し、動的 SQL 文またはストアド・プロシージャをカプセル化します。Adaptive Server データベースでコマンドを実行する次のメソッドがあります。 <ul style="list-style-type: none"><li>• <code>ExecuteNonQuery</code> - 結果セットを返さない <code>Execute</code> コマンド。</li><li>• <code>ExecuteScalar</code> - 結果セットを返さない <code>Execute</code> コマンド。</li><li>• <code>ExecuteReader</code> - 単一の値を返す <code>Execute</code> コマンド。</li><li>• <code>ExecuteXmlReader</code> - XML を返す <code>Execute</code> コマンド。</li></ul>
基本クラス	Component
実装	IDbCommand、IDisposable
参照	<a href="#">「AseCommand を使用したデータの取得と操作」 (36 ページ)</a> 、 <a href="#">「データに対するアクセスと操作」 (36 ページ)</a>

---

**注意** パラメータを取るストアド・プロシージャを呼び出す場合は、ストアド・プロシージャに対してパラメータを指定してください。詳細については、「[ストアド・プロシージャの使用](#)」(72 ページ)と「[AseParameter クラス](#)」(165 ページ)を参照してください。

---

## AseCommand コンストラクタ

説明	AseCommand オブジェクトを初期化します。
構文 1	<code>void AseCommand( )</code>
構文 2	<code>void AseCommand( string cmdText )</code>
構文 3	<code>void AseCommand( string cmdText, AseConnection connection )</code>
構文 4	<code>void AseCommand( string cmdText, AseConnection connection, AseTransaction transaction )</code>
パラメータ	<b>cmdText</b> : SQL 文またはストアド・プロシージャのテキスト。 <b>connection</b> : 現在の接続。 <b>transaction</b> : AseConnection を実行する AseTransaction。

## Cancel メソッド

説明	AseCommand オブジェクトの実行をキャンセルします。
構文	<code>void Cancel( )</code>
使用法	キャンセルする対象がない場合は、何も実行されません。処理中のコマンドがあり、キャンセル操作に失敗した場合、例外は生成されません。
実装	<code>IDbCommand.Cancel</code>

## CommandText プロパティ

説明	SQL 文またはストアド・プロシージャのテキストを表します。
構文	<code>string CommandText</code>
アクセス	読み込みと書き込み
プロパティ値	実行する SQL 文またはストアド・プロシージャの名前。デフォルトは空の文字列です。
実装	<code>IDbCommand.CommandText</code>
参照	<a href="#">「AseCommand コンストラクタ」(114 ページ)</a>

## CommandTimeout プロパティ

説明	コマンド実行の試行を終了してエラーを生成するまでの待機時間を秒単位で表します。
構文	<code>int CommandTimeout</code>
アクセス	読み込みと書き込み
実装	<code>IDbCommand.CommandTimeout</code>

デフォルト	30 秒
使用法	0 を指定すると、待機時間は制限されません。コマンド実行の試行が永久的に待機されるため、0 は指定しないでください。

## CommandType プロパティ

説明	<b>AseCommand</b> によって示されるコマンドの種類を表します。
構文	CommandType <b>CommandType</b>
アクセス	読み込みと書き込み
実装	IDbCommand.CommandType
使用法	サポートされるコマンドの種類は次のとおりです。 <ul style="list-style-type: none"><li>• <b>CommandType.StoredProcedure</b> : この CommandType を指定したときは、コマンド・テキストをストアド・プロシージャの名前にして、必要な引数を AseParameter オブジェクトとして指定します。</li><li>• <b>CommandType.Text</b> : これはデフォルトの値です。</li></ul> CommandType プロパティを StoredProcedure に設定したときは、CommandText プロパティにストアド・プロシージャの名前を設定してください。Execute メソッドの 1 つを呼び出すと、このストアド・プロシージャが実行されます。

## Connection プロパティ

説明	AseCommand オブジェクトで使用する接続オブジェクトを表します。
構文	AseConnection <b>Connection</b>
アクセス	読み込みと書き込み
デフォルト	デフォルト値は null 参照です。Visual Basic の場合は “Nothing” です。

## CreateParameter メソッド

説明	AseCommand オブジェクトにパラメータを渡す AseParameter オブジェクトを作成します。
構文	AseParameter <b>CreateParameter()</b>
戻り値	AseParameter オブジェクトとしての新しいパラメータ。

使用法	<ul style="list-style-type: none"><li>• ストアド・プロシージャや一部の SQL 文では、<b>@name</b> パラメータによって文のテキストに示されたパラメータを取ることができます。</li><li>• <b>CreateParameter</b> メソッドは <b>AseParameter</b> オブジェクトを作成します。<b>AseParameter</b> のプロパティを設定することで、パラメータの値やデータ型などを指定できます。</li></ul>
参照	<a href="#">「AseParameter クラス」(165 ページ)</a>

## ExecuteNonQuery メソッド

説明	Insert、Update、Delete、データ定義文など、結果セットを返さない文を実行します。
構文	<code>int ExecuteNonQuery( )</code>
戻り値	影響を受けたローの数。
実装	<code> IDbCommand.ExecuteNonQuery</code>
使用法	<ul style="list-style-type: none"><li>• <b>ExecuteNonQuery</b> では、<b>DataSet</b> を使わずにデータベース内のデータを変更できます。データを変更するには、<b>Update</b>、<b>Insert</b>、または <b>Delete</b> 文を実行します。</li><li>• <b>ExecuteNonQuery</b> はローを返しません、出力パラメータ、またはパラメータにマッピングされた戻り値にはデータが格納されます。</li><li>• <b>Update</b>、<b>Insert</b>、<b>Delete</b> 文では、戻り値はコマンドによって影響を受けたローの数です。他の種類の文では、戻り値は -1 になります。</li></ul>

## ExecuteReader メソッド

説明	結果セットを返す SQL 文を実行します。
構文 1	<code>AseDataReader ExecuteReader( )</code>
構文 2	<code>AseDataReader ExecuteReader( CommandBehavior behavior )</code>
パラメータ	<b>behavior</b> : <code>CloseConnection</code> 、 <code>Default</code> 、 <code>KeyInfo</code> 、 <code>SchemaOnly</code> 、 <code>SequentialAccess</code> 、 <code>SingleResult</code> 、または <code>SingleRow</code> のいずれか。  デフォルト値は <code>CommandBehavior.Default</code> です。このパラメータの詳細については、.NET Framework のドキュメントで <code>CommandBehavior Enumeration</code> ( <code>CommandBehavior</code> 列挙体) に関する説明を参照してください。
戻り値	<code>AseDataReader</code> オブジェクトとしての結果セット。
使用法	必要に応じて <code>CommandText</code> や <code>Parameters</code> を設定した、現在の <code>AseCommand</code> オブジェクトを実行します。 <code>AseDataReader</code> オブジェクトは、結果セットの前方への読み込みのみ可能です。変更可能な結果セットが必要な場合は、 <code>AseDataAdapter</code> クラスを使用してください。

参照 [「AseDataAdapter クラス」\(137 ページ\)](#)、[「AseDataReader クラス」\(144 ページ\)](#)

## ExecuteScalar メソッド

**説明** 単一の値を返す文を実行します。複数のローとカラムを返すクエリに対してこのメソッドを呼び出すと、最初のローの最初のカラムだけが返されます。

**構文** `object ExecuteScalar()`

**戻り値** 結果セットの最初のローの最初のカラム。結果セットが空の場合は、null 参照が返されます。

**実装** `IDbCommand.ExecuteScalar`

## ExecuteXmlReader メソッド

**説明** 有効な FOR XML 句を持つ SQL 文を実行し、結果セットを返します。XML が格納された 1 つのテキスト・カラムを返す文を使用して、ExecuteXmlReader を呼び出すこともできます。

**構文 1** `XmlReader ExecuteXmlReader()`

**パラメータ** なし

**戻り値** XmlReader オブジェクトとしての結果セット。

**使用法** 必要に応じて CommandText や Parameters を設定した、現在の AseCommand オブジェクトを実行します。

**参照** Microsoft .NET ドキュメントの XmlReader。

## NamedParameters

**説明** このプロパティは、この接続に関連付けられている AseCommand オブジェクトのデフォルトの動作を決定します。

**構文** `bool NamedParameters`

**プロパティ値** デフォルト値は、関連付けられている AseConnection オブジェクトで設定されている値と同じ。このプロパティが“true”(AseConnection のデフォルト)の場合、Provider は、パラメータ・マーカ(?)ではなくパラメータ名が使われていると想定します。次に例を示します。

```
select total_sales from titles where title_id = @title_id
```

パラメータ・マーカを使用する場合は、このプロパティを false に設定します。これは、ODBC および JDBC と互換性があります。

次に例を示します。

```
select total_sales from titles where title_id = ?
```

アクセス 読み込みと書き込み

## Parameters プロパティ

説明	現在の文のパラメータ・コレクションを表します。パラメータを記述するには、CommandText に <b>@name</b> パラメータまたは疑問符を使用します。
構文	AseParameterCollection <b>Parameters</b>
アクセス	読み込み専用
プロパティ値	SQL 文またはストア・プロシージャのパラメータ。デフォルト値は空のコレクションです。
使用方法	<ul style="list-style-type: none"><li>• CommandType が Text に設定されている場合は、<b>@name</b> パラメータを使用します。次に例を示します。<pre>SELECT * FROM Customers WHERE CustomerID = @name</pre><b>NamedParameters</b> が “false” に設定されている場合は、? パラメータ・マーカーを使用します。次に例を示します。<pre>SELECT * FROM Customers WHERE CustomerID = ?</pre></li><li>• 疑問符のプレースホルダを使用する場合、<b>AseParameter</b> オブジェクトが <b>AseParameterCollection</b> に追加される順序と、コマンド・テキスト内のパラメータの疑問符プレースホルダの位置を完全に一致させてください。</li><li>• コレクション内のパラメータと実行されるクエリの要件が一致しない場合は、エラーが発生するか、例外が返されます。</li><li>• 出力パラメータには、<b>AseDbType</b> を指定してください ( 準備のありなしに関係なく ) 。</li></ul>
参照	<a href="#">「AseParameterCollection クラス」 (169 ページ)</a>

## Prepare メソッド

説明	データ・ソースに対する <b>AseCommand</b> を準備またはコンパイルします。
構文	void <b>Prepare()</b>
実装	IDbCommand.Prepare
使用方法	<ul style="list-style-type: none"><li>• <b>Prepare</b> を呼び出す前に、準備対象となる文の各パラメータのデータ型を指定します。</li><li>• <b>Prepare</b> の呼び出し後に <b>Execute</b> メソッドを呼び出すと、<b>Size</b> プロパティに指定した値よりも大きいパラメータ値は、最初に指定したパラメータのサイズに自動的にトランケートされます。このとき、トランケート・エラーは返されません。</li></ul>

## Transaction プロパティ

説明	現在のコマンドをトランザクションに関連付けます。
構文	<b>AseTransaction Transaction</b>
アクセス	読み込みと書き込み
使用法	デフォルト値は null 参照です。Visual Basic の場合は Nothing です。  Transaction プロパティに特定の値がすでに設定されていて、コマンドが実行中の場合、このプロパティは設定できません。AseCommand オブジェクトと同じ AseConnection に接続されていない AseTransaction オブジェクトをトランザクション・プロパティに設定すると、次回、文を実行する際に例外が返されます。

## UpdatedRowSource プロパティ

説明	AseDataAdapter の Update メソッドで使用されるときに DataRow に適用されるコマンドの結果を表します。
構文	UpdateRowSource <b>UpdatedRowSource</b>
アクセス	読み込みと書き込み
実装	IDbCommand.UpdatedRowSource
プロパティ値	UpdatedRowSource 値の 1 つ。コマンドが自動的に生成される場合、デフォルトは None です。コマンドが自動的に生成されない場合、デフォルトは “Both” になります。

## AseCommandBuilder クラス

説明	SQL Select 文に基づいてシングルテーブルの SQL Insert、Update、Delete 文を自動的に作成します。
基本クラス	Component
実装	IDisposable
参照	<a href="#">「AseDataAdapter クラス」 (137 ページ)</a>

## AseCommandBuilder コンストラクタ

説明	AseCommandBuilder オブジェクトを初期化します。
構文 1	<code>void AseCommandBuilder( )</code>
構文 2	<code>void AseCommandBuilder( AseDataAdapter adapter )</code>
パラメータ	<b>adapter</b> 調整のための文を生成する対象となる AseDataAdapter オブジェクトです。

## DataAdapter プロパティ

説明	文を生成する対象となる AseDataAdapter を表します。
構文	<code>AseDataAdapter DataAdapter</code>
アクセス	読み込みと書き込み
プロパティ値	AseDataAdapter オブジェクトです。
使用法	AseCommandBuilder の新しいインスタンスを作成すると、この AseDataAdapter に関連するすべての既存 AseCommandBuilder が解放されます。

## DeleteCommand プロパティ

説明	Update() が呼び出されたとき、DataSet 内の削除されたローに対応するデータベース内のローを削除するためにデータベースに対して実行される AseCommand オブジェクトを表します。
構文	<code>AseCommand DeleteCommand</code>
アクセス	読み込みと書き込み
使用法	既存の AseCommand オブジェクトに DeleteCommand を割り当てた場合、AseCommand オブジェクトのクローンは作成されません。DeleteCommand は、既存の AseCommand に対する参照を維持します。

## DeriveParameters メソッド

説明	指定した AseCommand オブジェクトの Parameters コレクションを作成します。これは、AseCommand で指定したストアド・プロシージャで使用されます。
構文	<code>void DeriveParameters( AseCommand command )</code>
パラメータ	<b>コマンド</b> パラメータを抽出する対象となる AseCommand オブジェクトです。



- 使用法
- `DeriveParameters` は、`AseCommand` のすべての既存パラメータ情報を上書きします。
  - `DeriveParameters` では、データベース・サーバへの追加呼び出しが必要になります。パラメータ情報が事前にわかっている場合は、情報を明示的に設定することによって `Parameters` コレクションを作成する方が効率的です。

## Dispose メソッド

説明 オブジェクトに関連付けられたリソースを解放します。

構文 `void Dispose( )`

## GetDeleteCommand メソッド

説明 生成された `AseCommand` オブジェクトは、`AseDataAdapter.Update()` が呼び出されたときに、データベースに対して `Delete` オペレーションを実行します。

構文 `AseCommand GetDeleteCommand( )`

戻り値 削除を実行するために必要な自動生成された `AseCommand` オブジェクト。

- 使用法
- `GetDeleteCommand` メソッドは、実行対象の `AseCommand` オブジェクトを返します。これは、情報入手またはトラブルシューティングの目的で役立つ場合があります。
  - 修正するコマンドのベースとして `GetDeleteCommand` を使用することもできます。たとえば、`GetDeleteCommand` を呼び出し、`CommandTimeout` 値を変更したうえで、その値を明示的に `AseDataAdapter` に設定できます。
  - アプリケーションが `Update` または `GetDeleteCommand` を呼び出すと、最初の SQL 文が生成されます。最初の SQL 文が生成された後は、アプリケーションは、何らかの方法で SQL 文を変更する場合に `RefreshSchema` を呼び出す必要があります。呼び出しを行わない場合、`GetDeleteCommand` は以前の文の情報を引き続き使用します。

## GetInsertCommand メソッド

説明 生成された `AseCommand` オブジェクトは、`AseDataAdapter.Update()` が呼び出されたときに、データベースに対して `Insert` オペレーションを実行します。

構文 `AseCommand GetInsertCommand( )`

戻り値 挿入を実行するために必要な自動生成された `AseCommand` オブジェクト。

- 使用法
- **GetInsertCommand** メソッドは、実行対象の **AseCommand** オブジェクトを返します。これは、情報入手またはトラブルシューティングの目的で役立つ場合があります。
  - 修正するコマンドのベースとして **GetInsertCommand** を使用することもできます。たとえば、**GetInsertCommand** を呼び出し、**CommandTimeout** 値を変更したうえで、その値を明示的に **AseDataAdapter** に設定できます。
  - アプリケーションが **Update** または **GetInsertCommand** のいずれかを呼び出すと、SQL 文が生成されます。最初の SQL 文が生成された後は、アプリケーションは、何らかの方法で SQL 文を変更する場合に **RefreshSchema** を呼び出す必要があります。呼び出しを行わない場合、**GetInsertCommand** は引き続き、以前の文の情報 (正しくないことがあります) を使用します。

## GetUpdateCommand メソッド

説明 生成された **AseCommand** オブジェクトは、**AseDataAdapter.Update()** が呼び出されたときに、データベースに対して **Update** 処理を実行します。

構文 **AseCommand GetUpdateCommand( )**

戻り値 更新を実行するために必要な自動生成された **AseCommand** オブジェクト。

- 使用法
- **GetUpdateCommand** メソッドは、実行対象の **AseCommand** オブジェクトを返します。これは、情報入手またはトラブルシューティングの目的で役立つ場合があります。
  - 修正するコマンドのベースとして **GetUpdateCommand** を使用することもできます。たとえば、**GetUpdateCommand** を呼び出し、**CommandTimeout** 値を変更したうえで、その値を明示的に **AseDataAdapter** に設定できます。
  - アプリケーションが **Update** または **GetUpdateCommand** を呼び出すと、最初の SQL 文が生成されます。最初の SQL 文が生成された後は、アプリケーションは、何らかの方法で SQL 文を変更する場合に **RefreshSchema** を呼び出す必要があります。呼び出しを行わない場合、**GetUpdateCommand** は、以前の文の情報 (正しくないことがあります) を引き続き使用します。

## InsertCommand プロパティ

説明 **Update()** が呼び出されたとき、**DataSet** 内の挿入されたローに対応するデータベース内のローを追加するためにデータベースに対して実行される **AseCommand** を表します。

構文 **AseCommand InsertCommand**

アクセス 読み込みと書き込み

使用法	<p>既存の <code>AseCommand</code> オブジェクトに <code>InsertCommand</code> を割り当てた場合、<code>AseCommand</code> のクローンは作成されません。<code>InsertCommand</code> は、既存の <code>AseCommand</code> に対する参照を維持します。</p> <p>このコマンドによって複数のローが返された場合、<code>AseCommand</code> オブジェクトの <code>UpdatedRowSource</code> プロパティの設定に応じて、それらのローが <code>DataSet</code> に追加されます。</p>
参照	<a href="#">「Update メソッド」 (143 ページ)</a>

## PessimisticUpdate プロパティ

説明	ペシミスティック更新またはオプティミスティック更新のどちらを実装するかを示します。
構文	<code>public bool PessimisticUpdate</code>
アクセス	読み込みと書き込み
プロパティ値	ペシミスティック更新の場合は <code>true</code> 。オプティミスティック更新の場合は <code>false</code> 。
使用法	<ul style="list-style-type: none"><li>ペシミスティック更新では、レコードがロックされます。ロックされたレコードは、すべてのユーザが表示できますが、編集できるユーザは 1 人だけに制限されます。</li><li>オプティミスティック更新では、複数のユーザが同じレコードを編集できます。</li></ul>

## QuotePrefix プロパティ

説明	スペースなどの文字を含むデータベース・オブジェクト名を指定する場合に使用する先頭文字 (複数可) です。
構文	<code>string QuotePrefix</code>
アクセス	読み込みと書き込み
プロパティ値	使用する先頭文字 (複数可)。角かっこを指定できます。また、ASE <code>QUOTED_IDENTIFIER</code> オプションが “off” に設定されている場合は、二重引用符も使用できます。デフォルトは空の文字列です。
使用法	<ul style="list-style-type: none"><li>ASE オブジェクトには、スペース、カンマ、セミコロンなどの文字を含めることができます。<code>QuotePrefix</code> プロパティと <code>QuoteSuffix</code> プロパティには、オブジェクト名をカプセル化するためのデリミタを指定します。</li><li><code>Insert</code>、<code>Update</code>、または <code>Delete</code> オペレーションの後に <code>QuotePrefix</code> プロパティまたは <code>QuoteSuffix</code> プロパティを変更することはできませんが、<code>DataAdapter</code> の <code>Update</code> メソッドを呼び出した後でその設定を変更できます。</li></ul>

- 参照
- 識別子の詳細については、『ASE リファレンス・マニュアル』を参照してください。
  - QUOTED IDENTIFIER オプションの詳細については、『ASE リファレンス・マニュアル』を参照してください。

## QuoteSuffix プロパティ

- 説明
- スペースなどの文字を含むデータベース・オブジェクト名を指定する場合に使用する末尾文字 (複数可) です。
- 構文
- string **QuoteSuffix**
- アクセス
- 読み込みと書き込み
- プロパティ値
- 使用する末尾文字 (複数可)。角かっこを指定できます。また、ASE QUOTED\_IDENTIFIER オプションが off に設定されている場合は、二重引用符も使用できます。デフォルトは空の文字列です。
- 使用法
- ASE オブジェクトには、スペース、カンマ、セミコロンなどの文字を含めることができます。QuotePrefix プロパティと QuoteSuffix プロパティには、オブジェクト名をカプセル化するためのデリミタを指定します。
  - Insert、Update、または Delete オペレーションの後に QuotePrefix プロパティまたは QuoteSuffix プロパティを変更することはできませんが、DataAdapter の Update メソッドを呼び出した後でその設定を変更できます。
- 参照
- 識別子の詳細については、『ASE リファレンス・マニュアル』を参照してください。
  - QUOTED IDENTIFIER オプションの詳細については、『ASE リファレンス・マニュアル』を参照してください。

## RefreshSchema メソッド

- 説明
- Insert 文、Update 文、または Delete 文の生成で使用されるデータベース・スキーマ情報を更新します。
- 構文
- void **RefreshSchema**( )
- 使用法
- 関連する AseDataAdapter の SelectCommand の値が変更された場合は、必ず RefreshSchema を呼び出してください。

## SelectCommand プロパティ

説明	Fill または FillSchema で、DataSet にコピーする結果セットをデータベースから取得するために使用される AseCommand を表します。
構文	AseCommand <b>SelectCommand</b>
アクセス	読み込みと書き込み
使用法	<ul style="list-style-type: none"><li>• 事前に作成された AseCommand に SelectCommand を割り当てた場合、AseCommand のクローンは作成されません。SelectCommand は、その AseCommand オブジェクトに対する参照を維持します。</li><li>• SelectCommand がローを返さない場合は、DataSet にテーブルが追加されず、例外も発生しません。</li><li>• Select 文は、AseDataAdapter コンストラクタ内でも指定できます。</li></ul>

## UpdateCommand プロパティ

説明	Update() が呼び出されたとき、DataSet 内の更新されたローに対応するデータベース内のローを更新するためにデータベースに対して実行される AseCommand を表します。
構文	AseCommand <b>UpdateCommand</b>
アクセス	読み込みと書き込み
使用法	<ul style="list-style-type: none"><li>• 事前に作成された AseCommand に UpdateCommand を割り当てた場合、AseCommand のクローンは作成されません。UpdateCommand は、その AseCommand オブジェクトに対する参照を維持します。</li><li>• このコマンドによって複数のローが返された場合、AseCommand オブジェクトの UpdatedRowSource プロパティの設定に応じて、それらのローが DataSet にマージされます。</li></ul>
参照	<a href="#">「Update メソッド」 (143 ページ)</a>

## AseConnection クラス

説明	Adaptive Server データベースへの接続を表します。
基本クラス	Component
実装	IDbConnection、IDisposable
参照	<a href="#">「データベースへの接続」 (31 ページ)</a>

## AseConnection コンストラクタ

**説明** AseConnection オブジェクトを初期化します。データベースに対して操作を実行するには、最初に接続をオープンします。

**構文 1** `AseConnection()`

**構文 2** `AseConnection( string connectionString )`

**パラメータ** **connectionString** : Adaptive Server 接続文字列。接続文字列は、一連のキーワードと値のペアをセミコロンで区切ったものです。

**表 6-1: 接続文字列パラメータ**

プロパティ名	説明	必須	デフォルト値
UID、UserID、User ID、User	Adaptive Server サーバへの接続に必要なユーザ ID。大文字と小文字を区別する。	はい	空
PWD、Password	Adaptive Server サーバへの接続に使用するパスワード。大文字と小文字を区別する。	いいえ ( ユーザ名にパスワードが不要な場合 )	空
Server、Data Source、DataSource、Address、Addr、Network Address、Server Name	Adaptive Server サーバの名前または IP アドレス。	はい	空
Port、Server Port	Adaptive Server サーバのポート番号。	はい ( ポート番号が Datasource に指定されていない場合 )	空
AnsiNull	ODBC に厳格に準拠し、“= NULL” を使用不可とする。代わりに “Is Null” を使用する。デフォルトの動作を変更する場合は 1 に設定する。	いいえ	0
ApplicationName、Application Name	クライアント・アプリケーションを識別するために Adaptive Server が使用する名前。	いいえ	空
BufferCacheSize	入力／出力バッファをプール内に保持する。結果セットが非常に大きい場合、この値を大きくするとパフォーマンスが向上する。	いいえ	20
CharSet	指定されている文字セット。Adaptive Server ADO.NET Data Provider はデフォルトで Adaptive Server と同じデフォルトの文字セットをネゴシエートする。デフォルトの文字セットは ServerDefault。	いいえ	ServerDefault の値
ClientHostName	サーバへのログイン・レコードで渡されるクライアント・ホストの名前。たとえば、ClientHostName='MANGO'。	いいえ	空
ClientHostProc	サーバへのログイン・レコードで渡される、このホスト・マシン上のクライアント・プロセスを識別する語。たとえば、ClientHostProc='MANGO-PROC'。	いいえ	空

プロパティ名	説明	必須	デフォルト値
CodePageType	使用する文字コードの種類を指定する。有効な値は、ANSI と OEM。	いいえ	ANSI
ConnectionIdleTimeout, Connection IdleTimeout, Connection Idle Timeout	ドライバが接続をクローズする前に、接続が接続プール内でアイドル状態を維持できる時間 (秒数)。ゼロを指定すると、接続は無期限にアイドル状態を維持できる。	いいえ	0
Connection Lifetime	接続がオープン状態を維持できる時間 (秒数)。デフォルトの Connection Lifetime に到達 (超過) した接続をクライアントがクローズしたときに、ドライバは接続を接続プールに返すのではなくクローズする。アイドル状態の接続は、デフォルトの Connection Lifetime に到達した時点でクローズされ接続プールから削除される。  Connection Lifetime のデフォルト値は 0 であり、接続は無期限にオープン状態のままであることを示す。	いいえ	0
CumulativeRecordCount, Cumulative Record Count, CRC	ストアド・プロシージャで複数の update 文が実行されたとき、ドライバ (ADO.NET のプロバイダを使用する場合) はデフォルトで、更新されたレコード総数を返す。このカウントには、update や insert に設定されたトリガで実行されるすべての更新操作が含まれる。ドライバが最後の更新カウントだけを返すようにする場合は、このプロパティを 0 に設定する。	いいえ	1
Database, Initial Catalog	接続するデータベース。	いいえ	空
DSPassword, Directory Service Password	LDAP サーバで匿名アクセスが許可されない場合に、LDAP サーバでの認証に使用するパスワード。パスワードは DSURL でも指定可能。	いいえ	空
DSPrincipal, Directory Service Principal	LDAP サーバで匿名アクセスが許可されない場合に、LDAP サーバでの認証に使用するユーザ名。プリンシパルは DSURL でも指定可能。	いいえ	空
DSURL, Directory Service URL	LDAP サーバへの URL。	いいえ	空
DTCProtocol (Windows のみ)	分散トランザクションを使用する場合には、ドライバで XA プロトコルまたは OleNative プロトコルのどちらかを使用することを許可する。詳細については、「第 4 章 Adaptive Server の高度な機能」の「分散トランザクションの使用」(83 ページ)を参照。	いいえ	XA

プロパティ名	説明	必須	デフォルト値
EnableServerPacketSize	最適なパケット・サイズを選択するために、Adaptive Server サーバ・バージョン 15.0 以降を許可する。	いいえ	1
EncryptPassword、EncryptPwd、Encrypt Password	パスワードの暗号化を有効にするかどうかを指定する。0 はパスワードの暗号化を無効にし、1 はパスワードの暗号化を有効にする。	いいえ	0
Encryption	指定されている暗号化方式。指定できる値：ssl、none。	いいえ	空
FetchArraySize	サーバから結果をフェッチする場合にドライバが取得するロー数を指定する。	いいえ	25
HASession	高可用性を有効にするかどうかを指定する。0 は高可用性を無効にし、1 は高可用性を有効にする。	いいえ	0
IgnoreErrorsIfRS Pending	エラー・メッセージが表示された場合に、ドライバの処理を続行するかどうかを指定する。1 に設定した場合、ドライバでエラーが無視され、サーバからさらに結果が取得可能な場合は結果の処理が続行される。0 に設定した場合、ドライバではエラーが発生した場合に保留中の結果があっても結果の処理を停止する。	いいえ	0
Language	Adaptive Server が返すエラー・メッセージの言語。	いいえ	空 - デフォルトでは英語を使用。
LoginTimeOut、Connect Timeout、Connection Timeout	アプリケーションへ戻る前に、ログインの試行を待機する秒数。0 に設定するとタイムアウトが無効になり、接続の試行を永久的に待機する。	いいえ	15
min pool size	Adaptive Server への接続を強制的にクローズし、オープンしている接続総数が最小プール・サイズに達しないようにできる。プール内の接続数が最小プール・サイズと等しくなると、AseConnection.Close() で接続がクローズされる。	いいえ	20
max pool size	指定した最大プール・サイズよりも接続プールが大きくなるように制限できる。この制限に達すると、AseConnection.Open() で AseException が返される。	いいえ	100



プロパティ名	説明	必須	デフォルト値
NamedParameters	名前付きパラメータではなくパラメータ・マーカを使用する場合は、false に設定する。名前付きパラメータを使用する SQL は、SELECT * from titles where title_id = @title_id のようになる。 同じ SQL にパラメータ・マーカを使用すると、SELECT * from titles where title_id = ? のようになる。	いいえ	true
PacketSize、 Packet Size	Adaptive Server とクライアント間で送受信されるネットワーク・パケット 1 つあたりのバイト数。	いいえ	512
Ping Server	接続プールにある接続を使用する前に、その接続が有効であることを確認する必要がある場合は、false に設定する。	いいえ	true
pooling	接続プールを無効にする場合は、false に設定する。	いいえ	true
QuotedIdentifier	Adaptive Server で二重引用符で囲まれた文字列を識別子として処理するかどうかを指定する。0 は引用符付き識別子を無効にし、1 は引用符付き識別子を有効にする。	いいえ	0
RestrictMaximum PacketSize	EnableServerPacketSize に 1 を設定した場合にメモリに制約があるときは、このプロパティに 512 の倍数 (最大 65536) の int 値を設定する。	いいえ	0
SecondaryPort、 Secondary Port、 Secondary Server Port	アクティブ/アクティブ設定またはアクティブ/パッシブ設定でフェールオーバー・サーバとして機能する Adaptive Server サーバのポート番号。	はい (HASession が 1 に設定されていて、ポート番号が Secondary DataSource に指定されていない場合)	空
SecondaryServer、 Secondary Data Source、 Secondary DataSource、 Secondary Server、 Secondary Address、 Secondary Addr、 Secondary Network Address	アクティブ/アクティブ設定またはアクティブ/パッシブ設定でフェールオーバー・サーバとして機能する Adaptive Server サーバの名前または IP アドレス。 「セカンダリ・データ・ソース」は、SecondaryServer:SecondaryPort または SecondaryServer, SecondaryPort で指定可能。	はい (HASession が 1 に設定されている場合)	空
SQL Initialization String、 SQLInitializationString、 SQL Init String、 SQLInitString、 Initialization String、 InitializationString	データベース・サーバに渡されて、接続の直後に実行されるコマンドをスペースで区切って定義する。指定するコマンドは、結果の問い合わせには使用されない SQL コマンドであること。	いいえ	空

プロパティ名	説明	必須	デフォルト値
TextSize	Adaptive Server との間で送受信されるバイナリまたはテキスト・データのバイト単位での最大サイズ。たとえば、TextSize=64000 では、最大サイズが 64 KB に制限される。	いいえ	空。Adaptive Server のデフォルトは 32 KB。
TightlyCoupled Transaction (Windows のみ)	分散トランザクションを使用するときに、同一の Adaptive Server サーバに接続している 2 つの DSN を使用している場合は、このプロパティを 1 に設定する。 <a href="#">「第 4 章 Adaptive Server の高度な機能」の「分散トランザクションの使用」(83 ページ)</a> を参照。	いいえ	0
TrustedFile	Encryption を ssl に設定した場合は、このプロパティに信頼されたファイルへのパスを設定する。	いいえ	空
UseCursor、 Use Cursor	ドライバでカーソルを使用するかどうかを指定する。0 はカーソルを使用しない。1 はカーソルを使用する。	いいえ	0

**注意** DataSource、Data Source、Secondary Data Source、Secondary DataSource は特殊なキーワードです。これらは、サーバ名を指定するだけでなく、次の形式でも使用できます。

```
DataSource=servername,port
```

または

```
DataSource=servername:port
```

たとえば、DataSource=gamg:4100 では、サーバ名が“gamg”に、ポートが“4100”に設定されます。この場合、Port キーワードは接続文字列で不要になります。

#### 例

次の文は、“HR-001”という名前の Adaptive Server データベース・サーバで実行されている“policies”という名前のデータベースへの接続に使用する AseConnection オブジェクトを初期化します。この接続では、ユーザ ID に“admin”、パスワードに“money”を使用します。

```
"Data Source='HR-001';
Port=5000; UID='admin';
PWD='money';
Database='policies';"
```

## BeginTransaction メソッド

**説明** トランザクション・オブジェクトを返します。1つのトランザクション・オブジェクトに関連付けられているすべてのコマンドは、単一のトランザクションとして実行されます。トランザクションは、Commit() または Rollback() によって終了します。

**構文 1** `AseTransaction BeginTransaction( )`

**構文 2** `AseTransaction BeginTransaction( IsolationLevel isolationLevel )`

**パラメータ** **isolationLevel** : IsolationLevel 列挙体のメンバ。デフォルト値は ReadCommitted です。

**戻り値** 新しいトランザクションを表すオブジェクト。

**使用法** トランザクション・オブジェクトにコマンドを関連付けるには、AseCommand.Transaction プロパティを使用します。

**例**

```
AseTransaction tx = conn.BeginTransaction(IsolationLevel.ReadUncommitted );
```

**参照** [「Commit メソッド」\(178 ページ\)](#)、[「Rollback メソッド」\(179 ページ\)](#)、[「トランザクション処理」\(75 ページ\)](#)。一貫性の詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。

## ChangeDatabase メソッド

**説明** オープンしている AseConnection に対する現在のデータベースを変更します。

**構文** `void ChangeDatabase( string database )`

**パラメータ** **database** : 現在のデータベースの代わりに使用するデータベースの名前。

**実装** `IDbConnection.ChangeDatabase`

## Close メソッド

**説明** データベース接続をクローズします。

**構文** `void Close( )`

**実装** `IDbConnection.Close`

**使用法** Close メソッドは、保留中のすべてのトランザクションをロールバックします。次に、接続を接続プールに解放します。接続プールが無効の場合は、接続をクローズします。StateChange イベントの処理中に Close を呼び出すと、以降の StateChange イベントは起動されません。1つのアプリケーションで Close を複数回呼び出すことができます。

## ConnectionString プロパティ

説明	データベースへの接続文字列を表します。
構文	<code>string ConnectionString</code>
アクセス	読み込みと書き込み
実装	<code>IDbConnection.ConnectionString</code>
使用法	<ul style="list-style-type: none"><li>ConnectionString は ODBC 接続文字列のフォーマットとできるかぎり同じになるように設計されています。</li><li>ConnectionString プロパティを設定できるのは、接続がクローズされているときだけです。接続文字列値の多くには、対応する読み込み専用プロパティがあります。接続文字列を設定すると、これらのプロパティのすべてが更新されます。ただし、エラーが検出された場合は、どのプロパティも更新されません。AseConnection プロパティは、ConnectionString に含まれている設定だけを返します。</li><li>クローズされている接続に対して ConnectionString をリセットすると、パスワードを含むすべての接続文字列値と関連するプロパティがリセットされます。</li><li>プロパティを設定すると、接続文字列の予備検証が実行されます。アプリケーションが Open メソッドを呼び出すと、接続文字列は完全に検証されます。接続文字列に無効またはサポートされないプロパティが検出された場合は、ランタイム例外が生成されます。</li><li>値は一重引用符または二重引用符で囲みます。一重引用符または二重引用符を接続文字列の値の一部として使用する場合は、もう一方の引用符で値を囲みます。たとえば、<code>name="value's"</code> または <code>name= 'value's'</code> は有効ですが、<code>name='value's'</code> または <code>name= ""value""</code> は無効です。 ブランク文字は、値または引用符の中で使用されていないかぎり無視されます。 キーワードと値の各ペアはセミコロンで区切ります。セミコロンが値の一部の場合は、引用符も使用して区切ってください。 エスケープ・シーケンスはサポートされず、値の型は無関係です。 名前では、大文字と小文字を区別しません。接続文字列内に同じプロパティ名が複数回、記述されている場合は、最後の記述に指定されている値が使用されます。</li><li>ユーザ ID やパスワードをダイアログ・ボックスから取得して接続文字列に挿入する場合など、ユーザ入力に基づいて接続文字列を構築するときは注意が必要です。ユーザが規定外の接続文字列をこれらの値に組み込むことを、アプリケーションは禁止する必要があります。</li></ul>

例 次の文は、サーバ `mango` で実行されている `pubs2` という名前の Adaptive Server データベースに接続する接続文字列を設定し、その接続をオープンします。

```
AseConnection conn = new AseConnection( "Data Source=mango;  
Port=5000;  
UID=sa;  
PWD='';  
Database='pubs2';  
" );  
conn.Open();
```

## ConnectionTimeout プロパティ

説明 接続要求がタイムアウトし、エラーが発生するまでの秒数を表します。

構文 `int ConnectionTimeout`

アクセス 読み込み専用

デフォルト 15 秒

実装 `IDbConnection.ConnectionTimeout`

例 次の文は、`ConnectionTimeout` を 30 秒に変更します。

```
conn.ConnectionTimeout = 30;
```

## CreateCommand メソッド

説明 `AseCommand` オブジェクトを初期化します。`AseCommand` のプロパティを使用して、その動作を制御できます。

構文 `AseCommand CreateCommand()`

戻り値 `AseCommand` オブジェクト。

使用法 `Command` オブジェクトは `AseConnection` に関連付けられます。

## Database プロパティ

説明 接続がオープンされた後に使用される現在のデータベースの名前を表します。

構文 `string Database`

アクセス 読み込み専用

実装 `IDbConnection.Database`

使用法 `if (conn.Database != "pubs2") conn.ChangeDatabase("pubs2");`

## InfoMessage イベント

説明	Adaptive Server ADO.NET Data Provider が警告メッセージまたは情報メッセージを送信したときに発生します。
構文	<code>event AseInfoMessageEventHandler InfoMessage</code>
使用法	イベント・ハンドラが、このイベントに関連するデータを含む <code>AseInfoMessageEventArgs</code> 型の引数を受け取ります。 <code>Errors</code> および <code>Message</code> プロパティに、このイベントに固有の情報が記載されます。

## NamedParameters

説明	この接続に関連付けられている <code>AseCommand</code> オブジェクトのデフォルトの動作を決定します。詳細については、 <code>AseCommand</code> クラス ( <code>NamedParameter</code> プロパティ) を参照してください。
構文	<code>bool NamedParameters</code>
プロパティ値	プロパティ値は <code>ConnectionString (NamedParameters='true'/'false')</code> によって設定されます。または、ユーザが <code>AseConnection</code> のインスタンスを介して直接設定できます。
アクセス	読み込みと書き込み

## Open メソッド

説明	事前に指定された接続文字列を使用して、データベースへの接続をオープンします。
構文	<code>void Open( )</code>
実装	<code>IDbConnection.Open</code>
使用法	<ul style="list-style-type: none"><li>接続プール内にあるオープンされている接続を使用できる場合、<code>AseConnection</code> はその接続を使用します。使用できない場合は、データ・ソースへの新しい接続を確立します。</li><li>スコープ外にある <code>AseConnection</code> はクローズされません。そのため、<code>Close</code> または <code>Dispose</code> を呼び出して、明示的に接続をクローズしてください。</li></ul>

## State プロパティ

説明	接続の現在のステータスを表します。
構文	<code>ConnectionState State</code>
アクセス	読み込み専用
デフォルト	<code>Closed</code>

実装	IDbConnection.State
参照	<a href="#">「接続ステータスの確認」(34 ページ)</a>

## StateChange イベント

説明	接続のステータスが変ったときに発生します。
構文	event StateChangeEventHandler <b>StateChange</b>
使用法	イベント・ハンドラが、このイベントに関連するデータを含む <b>StateChangeEventArgs</b> 型の引数を受け取ります。 <b>StateChangeEventArgs</b> の 2 つのプロパティ、 <b>CurrentState</b> と <b>OriginalState</b> に、このイベントに固有の情報が記載されます。

## TraceEnter、TraceExit イベント

説明	デバッグ目的で、アプリケーション内でのデータベース・アクティビティをトレースします。
構文	<pre>public delegate void <b>TraceEnterEventHandler</b>(AseConnection connection, object source, string method, object[] parameters); public delegate void <b>TraceExitEventHandler</b>(AseConnection connection, object source, string method, object returnValue);</pre>
使用法	<b>TraceEnter</b> と <b>TraceExit</b> イベントは、独自のトレース方法を組み込むときに使用します。このイベントは、個々の接続インスタンスに固有です。これによって、接続ごとのログをそれぞれ異なるファイルに記録できます。このイベントは無視することも、他のトレース用にプログラムすることもできます。また .NET イベントを使用することで、1 つの接続オブジェクトに対して複数のイベント・ハンドラを設定できます。これによって、ウィンドウとファイルの両方に、同時にイベントのログを出力できます。

## AseConnectionPool クラス

説明	同じ接続のプールを管理します。
----	-----------------

## Available プロパティ

説明	プール内で使用可能な接続数を表します。
構文	int <b>Available</b>
アクセス	読み込み専用

## Size プロパティ

説明	プール内の接続数を表します。
構文	<code>int Size</code>
アクセス	読み込み専用

## AseConnectionPoolManager クラス

説明	すべての接続プールを管理します。
----	------------------

## AseConnectionPoolManager コンストラクタ

説明	AseConnectionPoolManager クラスの新しいインスタンスを初期化します。
構文	<code>void AseConnectionPoolManager()</code>

## GetConnectionPool メソッド

説明	指定された接続文字列の接続を管理する接続プールを取得します。
構文	<code>AseConnectionPool GetConnectionPool( string connectionString )</code>
パラメータ	<b>connectionString</b> : 取得対象のプールを識別する接続文字列。
戻り値	<code>connectionString</code> を管理する接続プール。

## NumberOfOpenConnections プロパティ

説明	すべての接続プール内でオープンしている接続の数を表します。
構文	<code>int NumberOfOpenConnections</code>
アクセス	読み込み専用



## AseDataAdapter クラス

説明	<p>DataSet クラスと Adaptive Server の間でリンクの機能を果たします。それには 2 つの重要なメソッドが使用されます。</p> <ul style="list-style-type: none"> <li>Fill() - サーバから取得したデータで DataSet を更新する。</li> <li>Update() - DataSet に対して行った変更をサーバに適用する。</li> </ul> <p>AseDataAdapter Fill または Update メソッドを使用するときには接続がオープンされていない場合は、このメソッドで接続をオープンできます。</p>
基本クラス	Component
実装	IDbDataAdapter、IDisposable
使用法	DataSet を使用すると、データをオフラインで操作できます。AseDataAdapter は、DataSet に一連の SQL 文を関連付けるメソッドを提供します。
参照	<p><a href="#">「AseDataAdapter を使用したデータへのアクセスと操作」 (49 ページ)</a>、<a href="#">「データに対するアクセスと操作」 (36 ページ)</a></p>

## AseDataAdapter コンストラクタ

説明	AseDataAdapter オブジェクトを初期化します。
構文 1	<b>AseDataAdapter( )</b>
構文 2	<b>AseDataAdapter( AseCommand selectCommand )</b>
構文 3	<b>AseDataAdapter( string selectCommandText, AseConnection selectConnection )</b>
構文 4	<b>AseDataAdapter( string selectCommandText, string selectConnectionString )</b>
パラメータ	<p><b>selectCommand</b> : Fill 中に、DataSet に格納するレコードをデータ・ソースから選択するために使用される AseCommand オブジェクト。</p> <p><b>selectCommandText</b> : AseDataAdapter の SelectCommand プロパティで使用する Select 文またはストアド・プロシージャ。</p> <p><b>selectConnection</b> : データベースへの接続を定義する AseConnection オブジェクト。</p> <p><b>selectConnectionString</b> : Adaptive Server データベースへの接続文字列。</p>
例	<p>次のコードは AseDataAdapter オブジェクトを初期化します。</p> <pre>AseDataAdapter da = new AseDataAdapter( "SELECT emp_id, emp_lname FROM employee," conn );</pre>

## AcceptChangesDuringFill プロパティ

説明	DataTable に DataRow が追加された後で、そのローに対して AcceptChanges を呼び出すかどうかを指定する値を表します。
構文	bool <b>AcceptChangesDuringFill</b>
アクセス	読み込みと書き込み
使用法	このプロパティを“true”に設定すると、DataAdapter は DataRow に対して AcceptChanges 関数を呼び出します。“false”に設定すると、AcceptChanges は呼び出されず、新しく追加されたローは挿入対象のローとして扱われます。デフォルトは“true”です。

## ContinueUpdateOnError プロパティ

説明	ローの更新中にエラーが発生したときに、例外を生成するかどうかを指定する値を表します。
構文	bool <b>ContinueUpdateOnError</b>
アクセス	読み込みと書き込み
使用法	<ul style="list-style-type: none"><li>デフォルトは“false”です。このプロパティを“true”に設定すると、例外を生成せずに更新が続行されます。</li><li>ContinueUpdateOnError が“true”の場合は、ローの更新中にエラーが発生しても例外は返されません。ローの更新がスキップされ、そのローの RowError プロパティにエラー情報が記録されます。DataAdapter は後続のローの更新を継続します。</li><li>ContinueUpdateOnError が“false”の場合は、エラーの発生時に例外が返されます。</li></ul>

## DeleteCommand プロパティ

説明	Update() が呼び出されたとき、DataSet 内の削除されたローに対応するデータベース内のローを削除するためにデータベースに対して実行される AseCommand オブジェクトを表します。
構文	AseCommand <b>DeleteCommand</b>
アクセス	読み込みと書き込み
使用法	既存の AseCommand オブジェクトに DeleteCommand を割り当てた場合、AseCommand オブジェクトのクローンは作成されません。DeleteCommand は、既存の AseCommand に対する参照を維持します。

## Fill メソッド

説明	データベースから取得したデータで、 <b>DataSet</b> または <b>DataTable</b> オブジェクトのローを追加またはリフレッシュします。
構文 1	<code>int Fill( DataSet dataSet )</code>
構文 2	<code>int Fill( DataSet dataSet, string srcTable )</code>
構文 3	<code>int Fill( DataSet dataSet, int startRecord, int maxRecords, string srcTable )</code>
構文 4	<code>int Fill( DataTable dataTable )</code>
パラメータ	<b>dataSet</b> : レコードとスキーマ ( オプション ) が書き込まれる DataSet。 <b>srcTable</b> : テーブル・マッピングに使用するソース・テーブルの名前。 <b>startRecord</b> : 処理を開始するレコード番号 ( 0 から始まる )。 <b>maxRecords</b> : DataSet に読み込まれるレコードの最大数。 <b>dataTable</b> : レコードとスキーマ ( オプション ) が書き込まれる DataTable。
戻り値	DataSet での追加またはリフレッシュが成功したローの数。
使用法	<ul style="list-style-type: none"><li>• <b>StartRecord</b> 引数を使用して、<b>DataSet</b> にコピーされるレコード数を制限している場合でも、<b>AseDataAdapter</b> クエリでは、すべての該当レコードがデータベースからクライアントにフェッチされます。結果セットが大きい場合は、パフォーマンスに重大な影響を与える可能性があります。</li><li>• 結果セットの前方向への読み込み専用で問題ない場合は、代替方法として、変更を実行する SQL 文 (<b>ExecuteNonQuery</b>) などとともに <b>AseDataReader</b> を使用できます。別の代替方法として、必要な結果セットだけを返すストアド・プロシージャを記述することもできます。</li><li>• <b>SelectCommand</b> がローを返さない場合は、<b>DataSet</b> にテーブルは追加されず、例外も発生しません。</li></ul>
参照	サポートされている fill メソッドの一覧については、.NET Framework のドキュメントで <b>DdDataAdapter.fill()</b> の説明を参照してください。

## FillError イベント

説明	fill 操作中にエラーが発生したときに返されます。
構文	<code>event FillErrorHandler FillError</code>

- 使用法
- FillError イベントを使用すると、エラーの発生後に Fill 操作を続行するかどうかをユーザが指定できます。FillError イベントは、次のような場合に発生する可能性があります。
- DataSet に追加するデータを共通言語ランタイム型に変換することで、データの精度が失われる場合。
  - 追加するローに含まれるデータが、DataSet 内の DataColumn に強制される制約に違反している場合。

## FillSchema メソッド

- 説明
- 1 つ以上の DataTable を DataSet に追加して、データ・ソース内のスキーマと一致するようにスキーマを設定します。
- 構文 1
- ```
DataTable[] FillSchema( DataSet dataSet, SchemaType schemaType )
```
- 構文 2
- ```
DataTable[] FillSchema( DataSet dataSet, SchemaType schemaType, string srcTable )
```
- 構文 3
- ```
DataTable FillSchema( DataTable dataTable, SchemaType schemaType )
```
- パラメータ
- dataSet** : レコードとスキーマ ( オプション ) が書き込まれる DataSet。
- schemaType** : スキーマの挿入方法を指定する SchemaType 値の 1 つ。
- srcTable** : テーブル・マッピングに使用するソース・テーブルの名前。
- dataTable** : DataTable。
- 戻り値
- 構文 1 と 2 の場合、戻り値は DataSet に追加された DataTable オブジェクトのコレクションに対する参照です。構文 3 の場合、戻り値は 1 つの DataTable に対する参照です。
- 参照
- 「[AseDataAdapter スキーマ情報の取得](http://msdn.microsoft.com/)」 (61 ページ)、.NET Framework help (<http://msdn.microsoft.com/>)

## GetFillParameters

- 説明
- Select 文の実行時にユーザによって設定されるパラメータを取得します。
- 構文
- ```
AseParameter[] GetFillParameters( )
```
- 戻り値
- ユーザが設定したパラメータを含む IDataParameter オブジェクトの配列。
- 実装
- ```
IDataAdapter.GetFillParameters
```

## InsertCommand プロパティ

|      |                                                                                                                                                                                                                                                                              |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明   | Update() が呼び出されたとき、DataSet 内の挿入されたローに対応するデータベース内のローを追加するためにデータベースに対して実行される AseCommand を表します。                                                                                                                                                                                 |
| 構文   | AseCommand <b>InsertCommand</b>                                                                                                                                                                                                                                              |
| アクセス | 読み込みと書き込み                                                                                                                                                                                                                                                                    |
| 使用法  | <ul style="list-style-type: none"><li>既存の AseCommand オブジェクトに InsertCommand を割り当てた場合、AseCommand のクローンは作成されません。InsertCommand は、既存の AseCommand に対する参照を維持します。</li><li>このコマンドによって複数のローが返された場合、AseCommand オブジェクトの UpdatedRowSource プロパティの設定に応じて、それらのローが DataSet に追加されます。</li></ul> |
| 参照   | <a href="#">「Update メソッド」</a> (143 ページ)、 <a href="#">「AseCommand オブジェクトを使用したローの挿入、更新、削除」</a> (42 ページ)、 <a href="#">「AseDataAdapter オブジェクトを使用したローの挿入、更新、削除」</a> (50 ページ)                                                                                                      |

## MissingMappingAction プロパティ

|        |                                                    |
|--------|----------------------------------------------------|
| 説明     | 一致するテーブルまたはカラムが受信データにない場合に実行するアクションを決定します。         |
| 構文     | MissingMappingAction <b>MissingMappingAction</b>   |
| アクセス   | 読み込みと書き込み                                          |
| プロパティ値 | MissingMappingAction 値の 1 つ。デフォルトは Passthrough です。 |
| 実装     | IDataAdapter.MissingMappingAction                  |

## MissingSchemaAction プロパティ

|        |                                                 |
|--------|-------------------------------------------------|
| 説明     | 既存の DataSet スキーマが受信データと一致しないときに実行するアクションを決定します。 |
| 構文     | MissingSchemaAction <b>MissingSchemaAction</b>  |
| アクセス   | 読み込みと書き込み                                       |
| プロパティ値 | MissingSchemaAction 値の 1 つ。デフォルトは Add です。       |
| 実装     | IDataAdapter.MissingSchemaAction                |

## RowUpdated イベント

**説明** 更新時、データ・ソースに対してコマンドが実行された後に発生します。更新が試行され、イベントが初期化されます。

**構文** `event AseRowUpdatedEventHandler RowUpdated`

**使用法** イベント・ハンドラが、このイベントに関連するデータを含む `AseRowUpdatedEventArgs` 型の引数を受け取ります。次の `AseRowUpdatedEventArgs` プロパティには、このイベントに固有の情報が記載されます。

- `Command`
- `Errors`
- `RecordsAffected`
- `Row`
- `StatementType`
- `Status`
- `TableMapping`

詳細については、.NET Framework のドキュメントで `OleDbDataAdapter.RowUpdated Event` の説明を参照してください。

## RowUpdating イベント

**説明** 更新時、データ・ソースに対してコマンドが実行される前に発生します。更新が試行され、イベントが初期化されます。

**構文** `event AseRowUpdatingEventHandler RowUpdating`

**使用法** イベント・ハンドラが、このイベントに関連するデータを含む `AseRowUpdatingEventArgs` 型の引数を受け取ります。次の `AseRowUpdatingEventArgs` プロパティには、このイベントに固有の情報が記載されます。

- `Command`
- `Errors`
- `Row`
- `StatementType`
- `Status`
- `TableMapping`

詳細については、.NET Framework のドキュメントで `OleDbDataAdapter.RowUpdating Event` の説明を参照してください。

## SelectCommand プロパティ

|      |                                                                                                                                                                                                                                                                                                       |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明   | Fill または FillSchema で、DataSet にコピーする結果セットをデータベースから取得するために使用される AseCommand を表します。                                                                                                                                                                                                                      |
| 構文   | AseCommand <b>SelectCommand</b>                                                                                                                                                                                                                                                                       |
| アクセス | 読み込みと書き込み                                                                                                                                                                                                                                                                                             |
| 使用法  | <ul style="list-style-type: none"> <li>• 事前に作成された AseCommand に SelectCommand を割り当てた場合、AseCommand のクローンは作成されません。SelectCommand は、その AseCommand オブジェクトに対する参照を維持します。</li> <li>• SelectCommand がローを返さない場合は、DataSet にテーブルが追加されず、例外も発生しません。</li> <li>• Select 文は、AseDataAdapter コンストラクタ内でも指定できます。</li> </ul> |

## TableMappings プロパティ

|      |                                                                                                                                                                                                  |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明   | ソース・テーブルと DataTable とのマスタ・マッピングを提供するコレクションを表します。                                                                                                                                                 |
| 構文   | DataTableMappingCollection <b>TableMappings</b>                                                                                                                                                  |
| アクセス | 読み込み専用                                                                                                                                                                                           |
| 使用法  | <ul style="list-style-type: none"> <li>• デフォルト値は空のコレクションです。</li> <li>• 変更を調整するときに、AseDataAdapter は DataTableMappingCollection コレクションを使用して、データ・ソースで使われるカラム名と DataSet で使われるカラム名を関連付けます。</li> </ul> |

## Update メソッド

|       |                                                                                                                                                                                                                                              |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | データベース内のテーブルを、DataSet に対して行われた変更で更新します。                                                                                                                                                                                                      |
| 構文 1  | int <b>Update</b> ( DataSet <i>dataSet</i> )                                                                                                                                                                                                 |
| 構文 2  | int <b>Update</b> ( DataSet <i>dataSet</i> , string <i>srcTable</i> )                                                                                                                                                                        |
| 構文 3  | int <b>Update</b> ( DataTable <i>dataTable</i> )                                                                                                                                                                                             |
| 構文 4  | int <b>Update</b> ( DataRow[] <i>dataRows</i> )                                                                                                                                                                                              |
| パラメータ | <p><b>dataSet</b> : 更新するレコードとスキーマ ( オプション ) を持つ DataSet。</p> <p><b>srcTable</b> : テーブル・マッピングに使用するソース・テーブルの名前。</p> <p><b>dataTable</b> : レコードとスキーマ ( オプション ) で更新する DataTable。</p> <p><b>dataRows</b> : データ・ソースの更新に使用する DataRow オブジェクトの配列。</p> |
| 戻り値   | DataSet からの更新に成功したローの数。                                                                                                                                                                                                                      |

|     |                                                                                                                                                                                                                                            |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 使用法 | Update は、データ・セット内にある挿入、更新、削除された各ローに対して、それぞれ <code>InsertCommand</code> 、 <code>UpdateCommand</code> 、 <code>DeleteCommand</code> プロパティを使用して実行されます。                                                                                         |
| 参照  | <a href="#">「DeleteCommand プロパティ」</a> (138 ページ)、 <a href="#">「InsertCommand プロパティ」</a> (141 ページ)、 <a href="#">「UpdateCommand プロパティ」</a> (144 ページ)、 <a href="#">「AseDataAdapter オブジェクトを使用したローの挿入、更新、削除」</a> (50 ページ)、.NET Framework のドキュメント |

### UpdateCommand プロパティ

|      |                                                                                                                                                                                                                                                                                   |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明   | Update() が呼び出されたとき、DataSet 内の更新されたローに対応するデータベース内のローを更新するためにデータベースに対して実行される AseCommand を表します。                                                                                                                                                                                      |
| 構文   | AseCommand <b>UpdateCommand</b>                                                                                                                                                                                                                                                   |
| アクセス | 読み込みと書き込み                                                                                                                                                                                                                                                                         |
| 使用法  | <ul style="list-style-type: none"><li>事前に作成された AseCommand に UpdateCommand を割り当てた場合、AseCommand のクローンは作成されません。UpdateCommand は、その AseCommand オブジェクトに対する参照を維持します。</li><li>このコマンドによって複数のローが返された場合、AseCommand オブジェクトの UpdatedRowSource プロパティの設定に応じて、それらのローが DataSet にマージされます。</li></ul> |
| 参照   | <a href="#">「Update メソッド」</a> (143 ページ)                                                                                                                                                                                                                                           |

## AseDataReader クラス

|       |                                                                                                                                                                                                                                                                                                                                                                                                |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | クエリまたはストアド・プロシージャで、前方向への読み込み専用の結果セットが取得されます。                                                                                                                                                                                                                                                                                                                                                   |
| 基本クラス | MarshalByRefObject                                                                                                                                                                                                                                                                                                                                                                             |
| 実装    | IDataReader、IDisposable、IDataRecord、IEnumerable、IListSource                                                                                                                                                                                                                                                                                                                                    |
| 使用法   | <ul style="list-style-type: none"><li>AseDataReader には、コンストラクタはありません。AseDataReader オブジェクトを取得するには、次のように AseCommand を実行します。</li></ul> <pre>AseCommand cmd = new AseCommand("Select emp_id from employee", conn ); AseDataReader reader = cmd.ExecuteReader();</pre> <ul style="list-style-type: none"><li>AseDataReader では、前方向にのみ移動できます。より柔軟な方法で結果を操作する必要がある場合は、AseDataAdapter を使用します。</li></ul> |



- カーソルの使用時、`AseDataReader` は必要なだけのローを取得します。詳細については、「[AseConnection コンストラクタ](#)」(126 ページ) の `ConnectionString` プロパティの `UseCursor` パラメータを参照してください。

参照 [「ExecuteReader メソッド」](#) (116 ページ)、[「データに対するアクセスと操作」](#) (36 ページ)

## Close メソッド

**説明** `AseDataReader` クラスをクローズします。

**構文** `void Close()`

**実装** `IDataReader.Close`

**使用法** `AseDataReader` を使用した後は、`Close` メソッドを明示的に呼び出してください。

## Depth プロパティ

**説明** 現在のローのネストの深さを示す値を表します。一番外側のテーブルの深さはゼロです。

**構文** `int Depth`

**アクセス** 読み込み専用

**プロパティ値** 現在のローのネストの深さ。

**実装** `IDataReader.Depth`

## Dispose メソッド

**説明** オブジェクトに関連付けられたリソースを解放します。

**構文** `void Dispose()`

## FieldCount プロパティ

**説明** 結果セット内のカラム数を表します。

**構文** `int FieldCount`

**アクセス** 読み込み専用

**プロパティ値** 有効なレコード・セットに位置していない場合は 0、それ以外の場合は現在のレコードのカラム数。デフォルトは -1 です。

**実装** `IDataRecord.FieldCount`

**使用法** 有効なレコード・セットに位置していない場合、このプロパティの値は 0 です。それ以外の場合は、現在のレコードのカラム数になります。デフォルトは -1 です。

## GetBoolean メソッド

**説明** 指定したカラムの値を Boolean として取得します。

**構文** `bool GetBoolean( int ordinal )`

**パラメータ** **ordinal** : 値を取得するカラムを表す序数。番号付けは 0 から始まります。

**戻り値** カラムの値。

**実装** `IDataRecord.GetBoolean`

**使用法** 変換は実行されません。bit 型のカラムからデータを取得するときに、このメソッドを使用してください。

## GetByte メソッド

**説明** 指定したカラムの値を Byte として取得します。

**構文** `byte GetByte( int ordinal )`

**パラメータ** **ordinal** : 値を取得するカラムを表す序数。番号付けは 0 から始まります。

**戻り値** カラムの値。

**実装** `IDataRecord.GetByte`

**使用法** 変換は実行されません。tinyint 型のカラムからデータを取得するときに、このメソッドを使用してください。

## GetBytes メソッド

**説明** 指定したカラム・オフセットからのバイト・ストリームを、配列としてバッファに読み込みます。読み込みは、指定したバッファ・オフセットから開始されます。

**構文** `long GetBytes( int ordinal, long dataIndex, byte[] buffer, int bufferIndex, int length )`

**パラメータ** **ordinal** : 値を取得するカラムを表す序数。番号付けは 0 から始まります。

**dataIndex** : バイトの読み込みを開始するカラム値内のインデックス。

**buffer** : データの格納先となる配列。

**bufferIndex** : データのコピーを開始する配列内のインデックス。

**length** : 指定したバッファにコピーする最大長。

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 戻り値 | 読み込まれたバイト数。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 実装  | <code>IDataRecord.GetBytes</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 使用法 | <ul style="list-style-type: none"> <li>• <code>GetBytes</code> は、フィールド内の使用可能なバイト数を返します。ほとんどの場合、これは正確なフィールド長になります。ただし、<code>GetBytes</code> を使用してフィールドからバイトをすでに取得している場合は、返されるバイト数がフィールドの実際の長さよりも小さくなる場合があります。この現象は、<code>AseDataReader</code> クラスが大規模なデータ構造体をバッファに読み込んでいるときなどに起こります。</li> <li>• <code>null</code> 参照 (Visual Basic の場合は “Nothing”) のバッファを渡すと、<code>GetBytes</code> はフィールド長をバイト単位で返します。</li> <li>• 変換は実行されません。 <code>image</code>、<code>binary</code>、<code>timestamp</code>、<code>varbinary</code> 型のカラムからデータを取得するときに、このメソッドを使用してください。</li> </ul> |

## GetChar メソッド

|       |                                                                                                                                                                                                                                                                           |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | 指定したカラムの値を文字として取得します。                                                                                                                                                                                                                                                     |
| 構文    | <code>char GetChar( int ordinal )</code>                                                                                                                                                                                                                                  |
| パラメータ | <b>ordinal</b> : 値を取得するカラムを表す序数。番号付けは 0 から始まります。                                                                                                                                                                                                                          |
| 戻り値   | カラムの値。                                                                                                                                                                                                                                                                    |
| 実装    | <code>IDataRecord.GetChar</code>                                                                                                                                                                                                                                          |
| 使用法   | <ul style="list-style-type: none"> <li>• 変換は実行されません。 <code>tinyint</code>、<code>char(1)</code>、<code>varchar(1)</code> 型のカラムからデータを取得するときに、このメソッドを使用してください。</li> <li>• このメソッドを呼び出す前に、<code>AseDataReader.IsDBNull</code> を呼び出して <code>null</code> 値をチェックしてください。</li> </ul> |
| 参照    | <a href="#">「IsDBNull メソッド」 (155 ページ)</a>                                                                                                                                                                                                                                 |

## GetChars メソッド

|       |                                                                                                                                                                                                                                                  |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | 指定したカラム・オフセットからの文字ストリームを、配列としてバッファに読み込みます。読み込みは、指定したバッファ・オフセットから開始されます。                                                                                                                                                                          |
| 構文    | <code>long GetChars( int ordinal, long dataIndex, char[] buffer, int bufferSize, int length )</code>                                                                                                                                             |
| パラメータ | <p><b>ordinal</b> : カラムを表す、0 から始まる序数。</p> <p><b>dataIndex</b> : <code>read</code> 操作を開始するロー内のインデックス。</p> <p><b>buffer</b> : データのコピー先のバッファ。</p> <p><b>bufferIndex</b> : <code>read</code> 操作を開始するバッファのインデックス。</p> <p><b>length</b> : 読み込む文字数。</p> |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 戻り値 | 実際に読み込まれた文字数。                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 実装  | <code>IDataRecord.GetChars</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 使用法 | <p><code>GetChars</code> は、フィールド内の使用可能な文字数を返します。ほとんどの場合、これは正確なフィールド長になります。ただし、<code>GetChars</code> を使用してフィールドから文字をすでに取得している場合は、返される文字数がフィールドの実際の長さよりも小さくなる場合があります。この現象は、<code>AseDataReader</code> が大規模なデータ構造体をバッファに読み込んでいるときなどに起こります。</p> <p><code>null</code> 参照 (Visual Basic の場合は <code>Nothing</code>) のバッファを渡すと、<code>GetChars</code> はフィールド長を文字単位で返します。</p> <p>変換は実行されません。<code>text</code>、<code>char</code>、<code>varchar</code> 型のカラムからデータを取得するときに、このメソッドを使用してください。</p> |
| 参照  | <a href="#">「BLOB の処理」 (68 ページ)</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                |

## GetDataTypeName メソッド

|       |                                                  |
|-------|--------------------------------------------------|
| 説明    | ソースのデータ型の名前を取得します。                               |
| 構文    | <code>string GetDataTypeName( int index )</code> |
| パラメータ | <b>index</b> : カラムを表す、0 から始まる序数。                 |
| 戻り値   | バックエンドのデータ型の名前。                                  |
| 実装    | <code>IDataRecord.GetDataTypeName</code>         |

## GetDateTime メソッド

|       |                                                                                                                                                                                                                                                                                                                                                                        |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | 指定したカラムの値を <code>DateTime</code> オブジェクトとして取得します。                                                                                                                                                                                                                                                                                                                       |
| 構文    | <code>DateTime GetDateTime( int ordinal )</code>                                                                                                                                                                                                                                                                                                                       |
| パラメータ | <b>ordinal</b> : カラムを表す、0 から始まる序数。                                                                                                                                                                                                                                                                                                                                     |
| 戻り値   | 指定されたカラムの値。                                                                                                                                                                                                                                                                                                                                                            |
| 実装    | <code>IDataRecord.GetDateTime</code>                                                                                                                                                                                                                                                                                                                                   |
| 使用法   | <ul style="list-style-type: none"><li>変換は実行されません。そのため、取得するデータは <code>DateTime</code> オブジェクトである必要があります。</li><li>このメソッドを呼び出す前に、<code>AseDataReader.IsDBNull</code> を呼び出して <code>null</code> 値をチェックしてください。</li><li>データベース内の対応する Adaptive Server 型が <code>datetime</code>、<code>smalldatetime</code>、<code>date</code>、<code>time</code> である場合は、このメソッドを使用してください。</li></ul> |
| 参照    | <a href="#">「IsDBNull メソッド」 (155 ページ)</a>                                                                                                                                                                                                                                                                                                                              |

## GetDecimal メソッド

|       |                                                                                                                                                                                                                                                                                      |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | 指定したカラムの値を <b>Decimal</b> オブジェクトとして取得します。                                                                                                                                                                                                                                            |
| 構文    | <code>decimal GetDecimal( int ordinal )</code>                                                                                                                                                                                                                                       |
| パラメータ | <b>ordinal</b> : 値を取得するカラムを表す序数。番号付けは 0 から始まります。                                                                                                                                                                                                                                     |
| 戻り値   | 指定されたカラムの値。                                                                                                                                                                                                                                                                          |
| 実装    | <code>IDataRecord.GetDecimal</code>                                                                                                                                                                                                                                                  |
| 使用法   | <ul style="list-style-type: none"><li>変換は実行されません。<code>decimal</code>、<code>numeric</code>、<code>smallmoney</code>、<code>money</code> 型のカラムからデータを取得するときに、このメソッドを使用してください。</li><li>このメソッドを呼び出す前に、<code>AseDataReader.IsDBNull</code> を呼び出して <code>null</code> 値をチェックしてください。</li></ul> |
| 参照    | <a href="#">「IsDBNull メソッド」(155 ページ)</a>                                                                                                                                                                                                                                             |

## GetDouble メソッド

|       |                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | 指定したカラムの値を倍精度浮動小数点数として取得します。                                                                                                                                                                                                                                                                                                                                                                            |
| 構文    | <code>double GetDouble( int ordinal )</code>                                                                                                                                                                                                                                                                                                                                                            |
| パラメータ | <b>ordinal</b> : 値を取得するカラムを表す序数。番号付けは 0 から始まります。                                                                                                                                                                                                                                                                                                                                                        |
| 戻り値   | 指定されたカラムの値。                                                                                                                                                                                                                                                                                                                                                                                             |
| 実装    | <code>IDataRecord.GetDouble</code>                                                                                                                                                                                                                                                                                                                                                                      |
| 使用法   | <ul style="list-style-type: none"><li>このメソッドを呼び出す前に、<code>AseDataReader.IsDBNull</code> を呼び出して <code>null</code> 値をチェックしてください。</li><li>変換は実行されません。そのため、取得するデータは倍精度浮動小数点数である必要があります。</li><li>精度が 16 以上の Adaptive Server データ型 <code>float</code> には <code>GetDouble</code> を使用してください。精度が 16 未満の Adaptive Server データ型 <code>real</code> と <code>float</code> には <code>GetFloat</code> を使用してください。</li></ul> |
| 参照    | <a href="#">「IsDBNull メソッド」(155 ページ)</a>                                                                                                                                                                                                                                                                                                                                                                |

## GetFieldType メソッド

|       |                                             |
|-------|---------------------------------------------|
| 説明    | オブジェクトのデータ型を取得します。                          |
| 構文    | <code>Type GetFieldType( int index )</code> |
| パラメータ | <b>index</b> : カラムを表す、0 から始まる序数。            |
| 戻り値   | オブジェクトのデータ型を取得します。                          |
| 実装    | <code>IDataRecord.GetFieldType</code>       |

## GetFloat メソッド

|       |                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | 指定したカラムの値を単精度浮動小数点数として取得します。                                                                                                                                                                                                                                                                                                                                                               |
| 構文    | <code>float GetFloat( int ordinal )</code>                                                                                                                                                                                                                                                                                                                                                 |
| パラメータ | <b>ordinal</b> : 値を取得するカラムを表す序数。番号付けは 0 から始まります。                                                                                                                                                                                                                                                                                                                                           |
| 戻り値   | 指定されたカラムの値。                                                                                                                                                                                                                                                                                                                                                                                |
| 実装    | <code>IDataRecord.GetFloat</code>                                                                                                                                                                                                                                                                                                                                                          |
| 使用法   | <ul style="list-style-type: none"><li>変換は実行されません。そのため、取得するデータは単精度浮動小数点数である必要があります。</li><li>このメソッドを呼び出す前に、<code>AseDataReader.IsDBNull</code> を呼び出して null 値をチェックしてください。</li><li>精度が 16 未満の Adaptive Server データ型 <code>real</code> と <code>float</code> には <code>GetFloat</code> を使用してください。精度が 16 以上の Adaptive Server データ型 <code>float</code> には <code>GetDouble</code> を使用してください。</li></ul> |
| 参照    | <a href="#">「IsDBNull メソッド」(155 ページ)</a>                                                                                                                                                                                                                                                                                                                                                   |

## GetInt16 メソッド

|       |                                                         |
|-------|---------------------------------------------------------|
| 説明    | 指定したカラムの値を、16 ビットの符号付き整数として取得します。                       |
| 構文    | <code>short GetInt16( int ordinal )</code>              |
| パラメータ | <b>ordinal</b> : 値を取得するカラムを表す序数。番号付けは 0 から始まります。        |
| 戻り値   | 指定されたカラムの値。                                             |
| 実装    | <code>IDataRecord.GetInt16</code>                       |
| 使用法   | 変換は実行されません。smallint 型のカラムからデータを取得するときに、このメソッドを使用してください。 |

## GetInt32 メソッド

|       |                                                          |
|-------|----------------------------------------------------------|
| 説明    | 指定したカラムの値を、32 ビットの符号付き整数として取得します。                        |
| 構文    | <code>int GetInt32( int ordinal )</code>                 |
| パラメータ | <b>ordinal</b> : 値を取得するカラムを表す序数。番号付けは 0 から始まります。         |
| 戻り値   | 指定されたカラムの値。                                              |
| 実装    | <code>IDataRecord.GetInt32</code>                        |
| 使用法   | 変換は実行されません。int[eger] 型のカラムからデータを取得するときに、このメソッドを使用してください。 |

## GetList メソッド

|     |                                                                                                                                                                                                 |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明  | ICollection を実装します。                                                                                                                                                                             |
| 構文  | ICollection GetList();                                                                                                                                                                          |
| 実装  | ICollection                                                                                                                                                                                     |
| 使用法 | .NET DataGrid オブジェクトの DataSource プロパティを AseDataReader に設定できます。次に、グリッドはこのメソッドを使用して、AseDataReader の結果をそのセルに直接バインドします。<br><br>通常、このメソッドを直接使用することはありません。AseDataReader はこのメソッドを実装しているため、次のように実行できます。 |

```
using(AseCommand cmd = new AseCommand(select total_sales from titles
    where title_id = 'BU1032', conn)
{
    using(AseDataReader rdr = cmd.ExecuteReader())
    {
        MyGrid.DataSource = rdr;
    }
}
```

## GetName メソッド

|       |                                   |
|-------|-----------------------------------|
| 説明    | 指定したカラムの名前を取得します。                 |
| 構文    | string GetName( int index )       |
| パラメータ | <b>index</b> : 0 から始まるカラム・インデックス。 |
| 戻り値   | 指定されたカラムの名前。                      |
| 実装    | IDataRecord.GetName               |

## GetOrdinal メソッド

|       |                               |
|-------|-------------------------------|
| 説明    | 名前を指定したカラムの序数を取得します。          |
| 構文    | int GetOrdinal( string name ) |
| パラメータ | <b>name</b> : カラム名。           |
| 戻り値   | カラムを表す、0 から始まる序数。             |
| 実装    | IDataRecord.GetOrdinal        |





## GetString メソッド

|       |                                                                                                                            |
|-------|----------------------------------------------------------------------------------------------------------------------------|
| 説明    | 指定したカラムの値を文字列として取得します。                                                                                                     |
| 構文    | <code>string GetString( int ordinal )</code>                                                                               |
| パラメータ | <b>ordinal</b> : 値を取得するカラムを表す序数。番号付けは 0 から始まります。                                                                           |
| 戻り値   | 指定されたカラムの値。                                                                                                                |
| 実装    | <code>IDataRecord.GetString</code>                                                                                         |
| 使用法   | 変換は実行されません。そのため、取得するデータは文字列である必要があります。<br><br>このメソッドを呼び出す前に、 <code>AseDataReader.IsDBNull</code> を呼び出して null 値をチェックしてください。 |
| 参照    | <a href="#">「IsDBNull メソッド」(155 ページ)</a>                                                                                   |

## GetUInt16 メソッド

|       |                                                |
|-------|------------------------------------------------|
| 説明    | 指定したカラムの値を、16 ビットの符号なし整数として取得します。              |
| 構文    | <code>UInt16 GetUInt16( int ordinal )</code>   |
| パラメータ | <b>ordinal</b> 値を取得するカラムを表す序数。番号付けは 0 から始まります。 |
| 戻り値   | 指定されたカラムの値。                                    |
| 使用法   | 変換は実行されません。そのため、取得するデータは 16 ビットの整数である必要があります。  |

## GetUInt32 メソッド

|       |                                                |
|-------|------------------------------------------------|
| 説明    | 指定したカラムの値を、32 ビットの符号なし整数として取得します。              |
| 構文    | <code>UInt32 GetUInt32( int ordinal )</code>   |
| パラメータ | <b>ordinal</b> 値を取得するカラムを表す序数。番号付けは 0 から始まります。 |
| 戻り値   | 指定されたカラムの値。                                    |
| 使用法   | 変換は実行されません。そのため、取得するデータは 32 ビットの整数である必要があります。  |

## GetUInt64 メソッド

|       |                                                |
|-------|------------------------------------------------|
| 説明    | 指定したカラムの値を、64 ビットの符号なし整数として取得します。              |
| 構文    | <code>UInt64 GetUInt64( int ordinal )</code>   |
| パラメータ | <b>ordinal</b> 値を取得するカラムを表す序数。番号付けは 0 から始まります。 |
| 戻り値   | 指定されたカラムの値。                                    |

## GetValue メソッド

|       |                                                  |
|-------|--------------------------------------------------|
| 説明    | 指定した序数に対応するカラムの値を、ネイティブ・フォーマットで取得します。            |
| 構文    | <code>object GetValue( int ordinal )</code>      |
| パラメータ | <b>ordinal</b> : 値を取得するカラムを表す序数。番号付けは 0 から始まります。 |
| 戻り値   | 対応するカラムの値。                                       |
| 実装    | <code>IDataRecord.GetValue</code>                |
| 使用法   | このメソッドでは、null のデータベース・カラムに対して DBNull が返されます。     |

## GetValues メソッド

|       |                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | 現在のローのすべての属性カラムを取得します。                                                                                                                                                                                                                                                                                                                                                                     |
| 構文    | <code>int GetValues( object[ ] values )</code>                                                                                                                                                                                                                                                                                                                                             |
| パラメータ | <b>values</b> : 結果セット内のすべてのローを保持するオブジェクトの配列。                                                                                                                                                                                                                                                                                                                                               |
| 戻り値   | 配列内のオブジェクトの数。                                                                                                                                                                                                                                                                                                                                                                              |
| 実装    | <code>IDataRecord GetValues</code>                                                                                                                                                                                                                                                                                                                                                         |
| 使用法   | <ul style="list-style-type: none"><li>ほとんどのアプリケーションでは、各カラムを個別に取得するよりも、<b>GetValues</b> メソッドを使用してすべてのカラムを取得する方が効率的です。</li><li>対象のローに含まれるカラム数よりも配列長が短い <b>Object</b> 配列を渡すことができます。<b>Object</b> 配列が保持しているデータ量だけが、配列にコピーされます。対象のローに含まれるカラム数よりも配列長が長い <b>Object</b> 配列も渡すことができます。</li><li>このメソッドでは、null のデータベース・カラムに対して DBNull が返されます。</li><li>指定した序数に対応するカラムの値を、ネイティブ・フォーマットで取得します。</li></ul> |

## IsClosed プロパティ

|        |                                                                           |
|--------|---------------------------------------------------------------------------|
| 説明     | AseDataReader がクローズされている場合は “true” を、それ以外の場合は “false” を返します。              |
| 構文     | <code>bool IsClosed</code>                                                |
| アクセス   | 読み込み専用                                                                    |
| プロパティ値 | AseDataReader がクローズされている場合は “true”、それ以外の場合は “false”。                      |
| 実装     | <code>IDataReader.IsClosed</code>                                         |
| 使用法    | AseDataReader をクローズした後で呼び出すことができるのは、IsClosed と RecordsAffected プロパティだけです。 |

## IsDBNull メソッド

|       |                                                                                                |
|-------|------------------------------------------------------------------------------------------------|
| 説明    | カラムに null 値が含まれるかどうかを示す値を返します。                                                                 |
| 構文    | <code>bool IsDBNull( int ordinal )</code>                                                      |
| パラメータ | <b>ordinal</b> : カラムを表す、0 から始まる序数。                                                             |
| 戻り値   | 指定したカラムの値が DBNull と等価の場合は “true”、それ以外の場合は “false”。                                             |
| 実装    | <code>IDataRecord.IsDBNull</code>                                                              |
| 使用法   | 型指定された Get メソッド (GetByte、GetChar など) を呼び出す前に、例外の発生を防止する目的で、このメソッドを呼び出して null のカラム値をチェックしてください。 |

## Item プロパティ

|       |                                                                      |
|-------|----------------------------------------------------------------------|
| 説明    | ネイティブ・フォーマットでのカラムの値を表します。C# で、このプロパティは AseDataReader クラスのインデクサになります。 |
| 構文 1  | <code>object this[ int index ]</code>                                |
| 構文 2  | <code>object this[ string name ]</code>                              |
| パラメータ | <b>index</b> : カラムの序数。<br><b>name</b> : カラム名。                        |
| アクセス  | 読み込み専用                                                               |
| 実装    | <code>IDataRecord.Item</code>                                        |

## NextResult メソッド

|     |                                                                           |
|-----|---------------------------------------------------------------------------|
| 説明  | バッチ SQL 文の結果を読み込むときに、 <b>AseDataReader</b> を次の結果へ進めます。                    |
| 構文  | <b>bool NextResult( )</b>                                                 |
| 戻り値 | さらに結果セットがある場合は“true”。それ以外の場合は“false”。                                     |
| 実装  | <b>IDataReader.NextResult</b>                                             |
| 使用法 | バッチ SQL 文の実行によって生成される複数の結果を処理する目的で使用します。<br>デフォルトでは、データ・リーダーは最初の結果に置かれます。 |

## Read メソッド

|     |                                                                                             |
|-----|---------------------------------------------------------------------------------------------|
| 説明  | 結果セットの次のローを読み込み、 <b>AseDataReader</b> をそのローに進めます。                                           |
| 構文  | <b>bool Read( )</b>                                                                         |
| 戻り値 | さらにローがある場合は“true”を返します。それ以外の場合は“false”を返します。                                                |
| 実装  | <b>IDataReader.Read</b>                                                                     |
| 使用法 | <b>AseDataReader</b> のデフォルトの位置は、最初のレコードの前です。そのため、データへのアクセスを開始するには <b>Read</b> を呼び出す必要があります。 |
| 例   | 次のコードは、結果内の 1 つのカラムの値をリスト・ボックスに書き込みます。                                                      |

```
while( reader.Read() )
{
    listResults.Items.Add( reader.GetValue( 0 ).ToString() );
}
listResults.EndUpdate();
reader.Close();
```

## RecordsAffected プロパティ

|        |                                                                               |
|--------|-------------------------------------------------------------------------------|
| 説明     | SQL 文の実行によって変更、挿入、削除されたローの数を表します。                                             |
| 構文     | <b>int RecordsAffected</b>                                                    |
| アクセス   | 読み込み専用                                                                        |
| プロパティ値 | 変更、挿入、または削除されたローの数。影響を受けたローがない場合または文が失敗した場合は 0、 <b>Select</b> 文の場合は -1 になります。 |
| 実装     | <b>IDataReader.RecordsAffected</b>                                            |

- 使用法
- 変更、挿入、または削除されたローの数。影響を受けたローがない場合または文が失敗した場合、値は 0 です。Select 文の場合、値は -1 です。
  - このプロパティの値は累積です。たとえば、バッチ・モードで 2 つのレコードが挿入された場合、RecordsAffected の値は 2 になります。
  - AseDataReader をクローズした後で呼び出すことができるのは、IsClosed と RecordsAffected プロパティだけです。

## AseDbType 列挙型

Adaptive Server のデータ型を指定します。データ型のマッピングの詳細については、表 6-2 を参照してください。

|     |               |
|-----|---------------|
| メンバ | Binary        |
|     | Bit           |
|     | Char          |
|     | Date          |
|     | DateTime      |
|     | Decimal       |
|     | Double        |
|     | Float         |
|     | Integer       |
|     | Image         |
|     | LongVarChar   |
|     | Money         |
|     | Nchar         |
|     | Numeric       |
|     | NVarChar      |
|     | Real          |
|     | SmallDateTime |
|     | SmallMoney    |
|     | Text          |
|     | Time          |
|     | TimeStamp     |
|     | TinyInt       |
|     | UniChar       |
|     | UniVarChar    |
|     | VarBinary     |
|     | VarChar       |

**注意** Numeric と Decimal の精度は、Adaptive Server の精度の 38 ではなく 26 に制限されています。

データ型のマッピング

次の表は、Adaptive Server ADO.NET Data Provider でのデータ型のマッピングを示します。

**表 6-2: Adaptive Server ADO.NET でのデータ型のマッピング**

| Adaptive Server データベース型 | AseDbType 列挙型    | .NET DbType 列挙型       | .Net クラス名 |
|-------------------------|------------------|-----------------------|-----------|
| binary                  | Binary           | Binary                | Byte[]    |
| bigint                  | BigInt           | Int64                 | Int64     |
| bit                     | Bit              | Boolean               | Boolean   |
| char                    | Char             | AnsiStringFixedLength | String    |
| date                    | Date             | Date                  | DateTime  |
| datetime                | DateTime         | DateTime              | DateTime  |
| decimal                 | Decimal          | Decimal               | Decimal   |
| double                  | Double           | Double                | Double    |
| float (<16)             | Real             | Single                | Single    |
| float (>=16)            | Double           | Double                | Double    |
| image                   | Image            | Binary                | Byte[]    |
| int[eger]               | Integer          | Int32                 | Int32     |
| money                   | Money            | Currency              | Decimal   |
| nchar                   | NChar            | AnsiStringFixedLength | String    |
| nvarchar                | NVarChar         | AnsiString            | String    |
| numeric                 | Numeric          | VarNumeric            | Decimal   |
| real                    | Real             | Single                | Single    |
| smalldatetime           | SmallDateTime    | DateTime              | DateTime  |
| smallint                | SmallInt         | Int16                 | Int16     |
| smallmoney              | SmallMoney       | Currency              | Decimal   |
| text                    | text             | AnsiString            | String    |
| time                    | time             | time                  | DateTime  |
| timestamp               | TimeStamp        | Binary                | Byte[]    |
| tinyint                 | TinyInt          | Byte                  | Byte      |
| unichar                 | UniChar          | StringFixedLength     | String    |
| unitext                 | Unitext          | String                | String    |
| univarchar              | UniVarChar       | String                | String    |
| unsignedbigint          | UnsignedBigInt   | UInt64                | UInt64    |
| unsignedint             | UnsignedInt      | UInt32                | UInt32    |
| unsignedsmallint        | UnsignedSmallInt | UInt16                | UInt16    |
| varbinary               | VarBinary        | Binary                | Byte[]    |
| varchar                 | VarChar          | AnsiString            | String    |

## AseError クラス

|       |                                      |
|-------|--------------------------------------|
| 説明    | データ・ソースから返された警告またはエラーに関する情報を収集します。   |
| 基本クラス | Object<br>AseError には、コンストラクタはありません。 |
| 参照    | <a href="#">「エラー処理」(78 ページ)</a>      |

## ErrorNumber プロパティ

|      |                                              |
|------|----------------------------------------------|
| 説明   | エラー・メッセージの番号を表します。                           |
| 構文   | <code>public int <b>MessageNumber</b></code> |
| アクセス | 読み込み専用                                       |

## Message プロパティ

|      |                                           |
|------|-------------------------------------------|
| 説明   | エラーの簡単な説明を表します。                           |
| 構文   | <code>public string <b>Message</b></code> |
| アクセス | 読み込み専用                                    |

## SqlState プロパティ

|      |                                                                                                           |
|------|-----------------------------------------------------------------------------------------------------------|
| 説明   | ANSI SQL 標準に従った、Adaptive Server 固有の 5 文字の SQL ステータスを表します。エラーが複数の場所で発生している場合、5 文字のエラー・コードはエラーの発生元を明らかにします。 |
| 構文   | <code>public string <b>SqlState</b></code>                                                                |
| アクセス | 読み込み専用                                                                                                    |

## ToString メソッド

|     |                                                                                                       |
|-----|-------------------------------------------------------------------------------------------------------|
| 説明  | エラー・メッセージの完全なテキストを取得します。                                                                              |
| 構文  | <code>public string <b>ToString()</b></code>                                                          |
| 使用法 | 戻り値は、“AseError:” の後にメッセージが続く形式の文字列です。次に例を示します。<br><code>AseError:UserId or Password not valid.</code> |
| 説明  | メッセージのステータスを表します。MsgNumber に対する変更子として使用されます。                                                          |
| 構文  | <code>public int <b>State</b></code>                                                                  |

|     |                                                                                                                                                                 |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明  | メッセージの重大度を表します。                                                                                                                                                 |
| 構文  | <code>public int Severity</code>                                                                                                                                |
| 説明  | メッセージを送信しているサーバの名前を表します。                                                                                                                                        |
| 構文  | <code>public string ServerName</code>                                                                                                                           |
| 説明  | メッセージの発生元であるストアド・プロシージャまたはリモート・プロシージャ・コール (RPC) の名前を表します。                                                                                                       |
| 構文  | <code>public string ProcName</code>                                                                                                                             |
| 説明  | コマンド・バッチまたはストアド・プロシージャ内でエラーが発生した箇所の行番号を表します ( 該当する場合 ) 。                                                                                                        |
| 構文  | <code>public int LineNum</code>                                                                                                                                 |
| 説明  | 拡張メッセージに関連付けられます。                                                                                                                                               |
| 構文  | <code>public int Status</code>                                                                                                                                  |
| 説明  | このダイアログでアクティブな任意のトランザクションの現在のステータスを表します。                                                                                                                        |
| 構文  | <code>public int TranState</code>                                                                                                                               |
| 説明  | Adaptive Server サーバから送信されたエラー・メッセージであるかどうかを表します。                                                                                                                |
| 構文  | <code>bool IsFromServer</code>                                                                                                                                  |
| 使用法 | 戻り値は “true” または “false” です。<br><pre>if (ex.Errors[0].IsInformation )     MessageBox.Show("ASE has reported the following error:" + ex.Errors[0].Message);</pre> |
| 説明  | Adaptive Server ADO.NET Data Provider から送信されたエラー・メッセージであるかどうかを表します。                                                                                             |
| 構文  | <code>bool IsFromClient</code>                                                                                                                                  |
| 使用法 | 戻り値は “true” または “false” です。<br><pre>if (ex.Errors[0].IsInformation )     MessageBox.Show("ASE has reported the following error:" + ex.Errors[0].Message);</pre> |
| 説明  | メッセージがエラーを表すかどうか判断されます。                                                                                                                                         |
| 構文  | <code>bool IsError</code>                                                                                                                                       |
| 使用法 | 戻り値は “true” または “false” です。<br><pre>if ( !ex.Errors[0].IsInformation )     MessageBox.Show("Error:" + ex.Errors[0].Message):</pre>                              |
| 説明  | メッセージが、状況が正常でない可能性があることを示す警告であるかどうかを表します。                                                                                                                       |



|     |                                                                                                                                |
|-----|--------------------------------------------------------------------------------------------------------------------------------|
| 構文  | <b>bool IsWarning</b>                                                                                                          |
| 使用法 | 戻り値は“true”または“false”です。<br><pre>if ( !ex.Errors[0].IsInformation )     MessageBox.Show("Error:" + ex.Errors[0].Message);</pre> |
| 説明  | アクティブなカタログが変更されたなどの情報を知らせるメッセージであるかどうかを表します。                                                                                   |
| 構文  | <b>bool IsInformation</b>                                                                                                      |
| 使用法 | 戻り値は“true”または“false”です。<br><pre>if ( !ex.Errors[0].IsInformation )     MessageBox.Show("Error:" + ex.Errors[0].Message);</pre> |

## AseErrorCollection クラス

|       |                                                                                                                                               |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | Adaptive Server ADO.NET Data Provider によって生成されたすべてのエラーを収集します。                                                                                 |
| 基本クラス | オブジェクト                                                                                                                                        |
| 実装    | ICollection、IEnumerable<br>AseErrorCollection には、コンストラクタはありません。通常、AseErrorCollection は AseException.Errors または InfoMessageArgs プロパティから取得されます。 |
| 参照    | <a href="#">「Errors プロパティ」 (162 ページ)</a> 、 <a href="#">「エラー処理」 (78 ページ)</a>                                                                   |

## CopyTo メソッド

|       |                                                              |
|-------|--------------------------------------------------------------|
| 説明    | AseErrorCollection の要素を配列にコピーします。配列内の指定したインデックスからコピーが開始されます。 |
| 構文    | <b>void CopyTo( Array array, int index )</b>                 |
| パラメータ | <b>array</b> : 要素のコピー先となる配列。<br><b>index</b> : 配列の開始インデックス。  |
| 実装    | ICollection.CopyTo                                           |

## Count プロパティ

|      |                                |
|------|--------------------------------|
| 説明   | コレクション内のエラーの数を表します。            |
| 構文   | <code>int Count</code>         |
| アクセス | 読み込み専用                         |
| 実装   | <code>ICollection.Count</code> |

## Item プロパティ

|        |                                                   |
|--------|---------------------------------------------------|
| 説明     | 指定したインデックスの位置にあるエラーを表します。                         |
| 構文     | <code>AseError this[ int index ]</code>           |
| パラメータ  | <b>index</b> : 取得するエラーの 0 から始まるインデックス。            |
| プロパティ値 | 指定したインデックスの位置にあるエラーが記録された <code>AseError</code> 。 |
| アクセス   | 読み込み専用                                            |

## AseException クラス

|       |                                                           |
|-------|-----------------------------------------------------------|
| 説明    | <code>Adaptive Server</code> が警告またはエラーを返したときに発生する例外を表します。 |
| 基本クラス | <code>SystemException</code>                              |

`AseException` には、コンストラクタはありません。通常、`AseException` オブジェクトは `catch` 内で宣言されます。次に例を示します。

```
...
catch (AseException ex )
{
    MessageBox.Show(ex.Message, "Error" );
}
```

|    |                                  |
|----|----------------------------------|
| 参照 | <a href="#">「エラー処理」 (78 ページ)</a> |
|----|----------------------------------|

## Errors プロパティ

|      |                                                                                        |
|------|----------------------------------------------------------------------------------------|
| 説明   | 1 つ以上の <code>AseError</code> オブジェクトのコレクションを表します。                                       |
| 構文   | <code>AseErrorCollection Errors</code>                                                 |
| アクセス | 読み込み専用                                                                                 |
| 使用法  | <code>AseErrorCollection</code> クラスには、 <code>AseError</code> クラスのインスタンスが常に 1 つ以上含まれます。 |

## Message プロパティ

|      |                                                |
|------|------------------------------------------------|
| 説明   | エラーを説明するテキストを返します。                             |
| 構文   | <code>string Message</code>                    |
| アクセス | 読み込み専用                                         |
| 使用法  | このメソッドは、最初の <code>AseError</code> のメッセージを返します。 |

## AseFailoverException クラス

説明 HA クラスタに設定されているセカンダリ・サーバへの Adaptive Server のフェールオーバーが成功したときに返される例外を表します。

基本クラス `AseException`

`AseFailoverException` には、コンストラクタはありません。通常、`AseFailoverException` オブジェクトは `catch` 内で宣言されます。次に例を示します。

```
...
catch( AseFailoverException ex )
{
    MessageBox.Show( ex.Message, "Warning!" );
}
```

参照 [「AseException クラス」\(162 ページ\)](#)

## Errors プロパティ

|      |                                        |
|------|----------------------------------------|
| 説明   | データ・ソースから送信された警告のコレクションを表します。          |
| 構文   | <code>AseErrorCollection Errors</code> |
| アクセス | 読み込み専用                                 |

## Message プロパティ

|      |                                 |
|------|---------------------------------|
| 説明   | データ・ソースから送信されたエラーの完全なテキストを表します。 |
| 構文   | <code>string Message</code>     |
| アクセス | 読み込み専用                          |

## ToString メソッド

|     |                                 |
|-----|---------------------------------|
| 説明  | InfoMessage イベントの文字列表現を取得します。   |
| 構文  | <code>string ToString( )</code> |
| 戻り値 | InfoMessage イベントを表す文字列。         |

## AseInfoMessageEventArgs クラス

|       |                                        |
|-------|----------------------------------------|
| 説明    | InfoMessage イベント・ハンドラに渡されたイベント引数を表します。 |
| 基本クラス | EventArgs                              |

## Errors プロパティ

|      |                                        |
|------|----------------------------------------|
| 説明   | サーバによって返された実際のエラー・オブジェクトのコレクションを表します。  |
| 構文   | <code>AseErrorCollection Errors</code> |
| アクセス | 読み込み専用                                 |

## Message プロパティ

|      |                             |
|------|-----------------------------|
| 説明   | エラー・メッセージを表します。             |
| 構文   | <code>string Message</code> |
| アクセス | 読み込み専用                      |

## AseInfoMessageEventHandler デリゲート

|       |                                                                                           |
|-------|-------------------------------------------------------------------------------------------|
| 説明    | AseConnection の InfoMessage イベントを処理するメソッドを表します。                                           |
| 構文    | <code>void AseInfoMessageEventHandler ( object sender, AseInfoMessageEventArgs e )</code> |
| パラメータ | <b>sender</b> : イベントの発生元。<br><b>e</b> : イベント・データが格納された AseInfoMessageEventArgs オブジェクト。    |

## AseParameter クラス

|       |                                                      |
|-------|------------------------------------------------------|
| 説明    | AseCommand のパラメータと、オプションとして DataSet カラムへのマッピングを表します。 |
| 基本クラス | MarshalByRefObject                                   |
| 実装    | IDbDataParameter、IDataParameter                      |

### AseParameter コンストラクタ

|      |                                                                                                                                                                                                                                                                                           |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 構文 1 | <b>AseParameter( )</b>                                                                                                                                                                                                                                                                    |
| 構文 2 | <b>AseParameter( string <i>parameterName</i>, object <i>value</i> )</b>                                                                                                                                                                                                                   |
| 構文 3 | <b>AseParameter( string <i>parameterName</i>, AseDbType <i>dbType</i> )</b>                                                                                                                                                                                                               |
| 構文 4 | <b>AseParameter( string <i>parameterName</i>, AseDbType <i>dbType</i>, int <i>size</i> )</b>                                                                                                                                                                                              |
| 構文 5 | <b>AseParameter( string <i>parameterName</i>, AseDbType <i>dbType</i>, int <i>size</i>, string <i>sourceColumn</i> )</b>                                                                                                                                                                  |
| 構文 6 | <b>AseParameter( string <i>parameterName</i>, AseDbType <i>dbType</i>, int <i>size</i>, ParameterDirection <i>direction</i>, bool <i>isNullable</i>, byte <i>precision</i>, byte <i>scale</i>, string <i>sourceColumn</i>, DataRowVersion <i>sourceVersion</i>, object <i>value</i> )</b> |

|       |                                                                       |
|-------|-----------------------------------------------------------------------|
| パラメータ | <b>value</b> : パラメータの値であるオブジェクト。                                      |
|       | <b>size</b> : パラメータの長さです。                                             |
|       | <b>sourceColumn</b> : マッピング対象のソース・カラムの名前。                             |
|       | <b>parameterName</b> : パラメータの名前。                                      |
|       | <b>dbType</b> : AseDbType 値の 1 つ。                                     |
|       | <b>direction</b> : ParameterDirection 値の 1 つ。                         |
|       | <b>isNullable</b> : フィールドの値として null を許可する場合は “true”、許可しない場合は “false”。 |
|       | <b>precision</b> : Value の解決に適用する、小数点以上と以下両方の総桁数。                     |
|       | <b>scale</b> : Value の解決に適用する、小数点以下の桁数。                               |
|       | <b>sourceVersion</b> : DataRowVersion 値の 1 つ。                         |

### AseDbType プロパティ

|      |                            |
|------|----------------------------|
| 説明   | パラメータの AseDbType を表します。    |
| 構文   | AseDbType <b>AseDbType</b> |
| アクセス | 読み込みと書き込み                  |



|      |                                                                                                                                                                                                               |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| アクセス | 読み込みと書き込み                                                                                                                                                                                                     |
| 実装   | IDataParameter.ParameterName                                                                                                                                                                                  |
| 使用法  | <ul style="list-style-type: none"><li>Adaptive Server ADO.NET Data Provider は、<b>@name</b> パラメータで示される、位置を指定するパラメータを使用します。</li><li>デフォルトは空の文字列です。</li><li>出力パラメータ (準備のありなしに関係なく) には、ユーザ指定のデータ型が必要です。</li></ul> |

## Precision プロパティ

|      |                                                                                                                                                                                                                                                                              |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明   | Value プロパティを表すために使用する最大桁数を表します。                                                                                                                                                                                                                                              |
| 構文   | byte <b>Precision</b>                                                                                                                                                                                                                                                        |
| アクセス | 読み込みと書き込み                                                                                                                                                                                                                                                                    |
| 実装   | IDbDataParameter.Precision                                                                                                                                                                                                                                                   |
| 使用法  | <ul style="list-style-type: none"><li>このプロパティの値は、<b>Value</b> プロパティを表すために使用する最大桁数です。デフォルト値は 0 で、これは Adaptive Server ADO.NET Data Provider が <b>Value</b> プロパティの精度を設定することを意味します。</li><li><b>Precision</b> プロパティは、<b>decimal</b> と <b>numeric</b> 型の入力パラメータにのみ使用します。</li></ul> |

## Scale プロパティ

|      |                                                                              |
|------|------------------------------------------------------------------------------|
| 説明   | Value の解決に適用する小数点以下の桁数を表します。                                                 |
| 構文   | byte <b>Scale</b>                                                            |
| アクセス | 読み込みと書き込み                                                                    |
| 実装   | IDbDataParameter.Scale                                                       |
| 使用法  | デフォルトは 0 です。Scale プロパティは、 <b>decimal</b> と <b>numeric</b> 型の入力パラメータにのみ使用します。 |

## Size プロパティ

|      |                             |
|------|-----------------------------|
| 説明   | カラム内のデータのバイト単位での最大サイズを表します。 |
| 構文   | int <b>Size</b>             |
| アクセス | 読み込みと書き込み                   |
| 実装   | IDbDataParameter.Size       |

- 使用法
- このプロパティの値は、カラム内のデータのバイト単位での最大サイズです。デフォルト値は、パラメータ値から推論されます。
  - **Size** プロパティは、binary と string 型で使います。
  - 可変長のデータ型では、**Size** プロパティはサーバに送信するデータの最大量を示します。たとえば、**Size** プロパティを使用して、サーバに送信するデータ量を、最初の 100 バイトまでの文字列値に制限できます。
  - 明示的に設定しない場合は、指定したパラメータ値の実際のサイズから、このサイズが推論されます。固定長のデータ型では、**Size** の値は無視されます。この情報は参照目的で取得できます。この場合、パラメータ値をサーバに送信するときに Adaptive Server ADO.NET Data Provider が使用する最大バイト数が返されます。

## SourceColumn プロパティ

- 説明 DataSet にマッピングされており、値の読み込みや書き込みに使用されるソース・カラムの名前を表します。
- 構文 `string SourceColumn`
- アクセス 読み込みと書き込み
- 実装 IDbDataParameter.SourceColumn
- 使用法 SourceColumn に空の文字列以外が設定されている場合、パラメータの値は SourceColumn 名で指定されるカラムから取得されます。Direction が Input に設定されている場合、値は DataSet から取得されます。Direction が Output に設定されている場合、値はデータ・ソースから取得されます。Direction が InputOutput の場合は、両方の組み合わせです。

## SourceVersion プロパティ

- 説明 Value を読み込むときに使用する DataRowVersion を表します。
- 構文 `DataRowVersion SourceVersion`
- アクセス 読み込みと書き込み
- 実装 IDbDataParameter.SourceVersion
- 使用法 パラメータ値が Current と Original のどちらに設定されているかを判別するために、Update 中に UpdateCommand によって使用されます。これによって、プライマリ・キーが更新できるようになります。InsertCommand と DeleteCommand では、このプロパティは無視されます。このプロパティは、Item プロパティ、または DataRow オブジェクトの GetChildRows メソッドによって使用される DataRow のバージョンに設定されます。



## ToString メソッド

|      |                              |
|------|------------------------------|
| 説明   | ParameterName が含まれる文字列を表します。 |
| 構文   | string ToString()            |
| アクセス | 読み込みと書き込み                    |

## Value プロパティ

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明   | パラメータの値を表します。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 構文   | object Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| アクセス | 読み込みと書き込み                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 実装   | IDataParameter.Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 使用法  | <ul style="list-style-type: none"> <li>入力パラメータでは、値は、サーバに送信される <b>AseCommand</b> にバインドされます。出力パラメータと戻り値パラメータでは、<b>AseCommand</b> の完了時と <b>AseDataReader</b> をクローズした後に、値が設定されます。</li> <li>null パラメータ値をサーバに送信するときは、null ではなく DBNull を指定します。システムでの null 値は、値を持たない空のオブジェクトです。</li> <li>アプリケーションでデータベースの型を指定する場合は、Adaptive Server ADO.NET Data Provider がデータをサーバに送信するときに、バインドされた値がその型に変換されます。Adaptive Server ADO.NET Data Provider が <b>IConvertible</b> インタフェースをサポートする場合は、どのような型の値でも変換されます。指定した型が値と互換性を持たないときは、変換エラーが発生する場合があります。</li> <li>Value の設定によって、DbType と AseDbType の両方のプロパティを推論できます。</li> <li>Value プロパティは Update によって上書きされます。</li> </ul> |

## AseParameterCollection クラス

|       |                                                                                                          |
|-------|----------------------------------------------------------------------------------------------------------|
| 説明    | AseCommand のすべてのパラメータと、オプションとして DataSet カラムへのマッピングを表します。                                                 |
| 基本クラス | オブジェクト                                                                                                   |
| 実装    | ICollection、IEnumerable、IDataParameterCollection                                                         |
| 使用法   | AseParameterCollection には、コンストラクタはありません。<br>AseParameterCollection は AseCommand.Parameters プロパティから取得します。 |
| 参照    | <a href="#">「Parameters プロパティ」 (118 ページ)</a>                                                             |

## Add メソッド

|       |                                                                                                                                                                                                                                                                |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | AseParameter を AseCommand に追加します。                                                                                                                                                                                                                              |
| 構文 1  | <code>public AseParameter Add( AseParameter P )</code>                                                                                                                                                                                                         |
| 構文 2  | <code>public AseParameter Add( object P)</code>                                                                                                                                                                                                                |
| 構文 3  | <code>public AseParameter Add( string name, AseDbType dataType )</code>                                                                                                                                                                                        |
| 構文 4  | <code>public AseParameter Add( string name, object value )</code>                                                                                                                                                                                              |
| 構文 5  | <code>public AseParameter Add( string name, AseDbType dataType, AseParameter size)</code>                                                                                                                                                                      |
| 構文 6  | <code>public AseParameter Add( string name, AseDbType dataType, int size, string sourceColumn)</code>                                                                                                                                                          |
| 構文 7  | <code>public AseParameter Add( string parameterName, AseDbType dbType, int size, ParameterDirection direction, Boolean isNullable, Byte precision, Byte scale, string sourceColumn, DataRowVersion sourceVersion, object value)</code>                         |
| パラメータ | <b>value</b> : 構文 1 と 2 では、 <i>value</i> はコレクションに追加する AseParameter オブジェクトです。構文 3 では、 <i>value</i> は接続に追加するパラメータの値です。<br><b>parameterName</b> : パラメータの名前。<br><b>aseDbType</b> : AseDbType 値の 1 つ。<br><b>size</b> : カラムの長さ。<br><b>sourceColumn</b> : ソース・カラムの名前。 |
| 戻り値   | Add は、AseCommand によって使用されるパラメータ・リストにパラメータを挿入します。戻り値は、リストに追加された新しいパラメータです。                                                                                                                                                                                      |

## Clear メソッド

|    |                                |
|----|--------------------------------|
| 説明 | コレクションからすべての項目を削除します。          |
| 構文 | <code>void Clear( )</code>     |
| 実装 | <code>ICollection.Clear</code> |

## Contains メソッド

|       |                                                          |
|-------|----------------------------------------------------------|
| 説明    | AseParameter がコレクション内にあるかどうかを示す値を返します。                   |
| 構文 1  | <code>bool Contains( object value )</code>               |
| 構文 2  | <code>bool Contains( string value )</code>               |
| パラメータ | <b>value</b> : 対象となる AseParameter オブジェクトの値。構文 2 の場合は名前。  |
| 戻り値   | AseParameter オブジェクトがコレクション内にある場合は“true”、それ以外の場合は“false”。 |

- 実装
- 構文 1 では `ICollection.Contains` を実装。
  - 構文 2 では `IDataParameterCollection.Contains` を実装。

## CopyTo メソッド

説明 `AseParameter` オブジェクトを `AseParameterCollection` から、指定した配列にコピーします。

構文 `void CopyTo( array array int index )`

パラメータ **array** : `AseParameter` オブジェクトのコピー先となる配列。

**index** : 配列の開始インデックス。

実装 `ICollection.CopyTo`

## Count プロパティ

説明 コレクション内にある `AseParameter` オブジェクトの数を表します。

構文 `int Count`

アクセス 読み込み専用

実装 `ICollection.Count`

## IndexOf メソッド

説明 コレクション内での `AseParameter` の位置を取得します。

構文 1 `int IndexOf( object value )`

構文 2 `int IndexOf( string parameterName )`

パラメータ **value** : 位置を調べる `AseParameter` オブジェクト。

**parameterName** : 位置を調べる `AseParameter` オブジェクトの名前。

戻り値 コレクション内での `AseParameter` の 0 から始まる位置。

- 実装
- 構文 1 では `ICollection.IndexOf` を実装。
  - 構文 2 では `IDataParameterCollection.IndexOf` を実装。

## Insert メソッド

説明 コレクション内の指定したインデックスに `AseParameter` を挿入します。

構文 `void Insert( int index object value )`

|       |                                                                                                 |
|-------|-------------------------------------------------------------------------------------------------|
| パラメータ | <b>index</b> : パラメータの挿入先となる、コレクション内の 0 から始まるインデックス。<br><b>value</b> : コレクションに追加する AseParameter。 |
| 実装    | IList.Insert                                                                                    |

## Item プロパティ

|        |                                                                                  |
|--------|----------------------------------------------------------------------------------|
| 説明     | 指定したインデックスまたは名前を持つ AseParameter を表します。                                           |
| 構文 1   | AseParameter <b>this</b> [ int <i>index</i> ]                                    |
| 構文 2   | AseParameter <b>this</b> [ string <i>parameterName</i> ]                         |
| パラメータ  | <b>index</b> : 取得するパラメータの 0 から始まるインデックス。<br><b>parameterName</b> : 取得するパラメータの名前。 |
| プロパティ値 | AseParameter                                                                     |
| アクセス   | 読み込みと書き込み                                                                        |
| 使用法    | C# では、このプロパティは AseParameterCollection クラスのインデクサになります。                            |

## Remove メソッド

|       |                                                  |
|-------|--------------------------------------------------|
| 説明    | メソッドに渡された AseParameter をコレクションから削除します。           |
| 構文    | void <b>Remove</b> ( object <i>value</i> )       |
| パラメータ | <b>value</b> : コレクションから削除する AseParameter オブジェクト。 |
| 実装    | IList.Remove                                     |

## RemoveAt メソッド

|       |                                                                                                                                    |
|-------|------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | パラメータのインデックスまたは名前に基づいて、コレクションからパラメータを削除します。                                                                                        |
| 構文 1  | void <b>RemoveAt</b> ( int <i>index</i> )                                                                                          |
| 構文 2  | void <b>RemoveAt</b> ( string <i>parameterName</i> )                                                                               |
| パラメータ | <b>index</b> : 削除するパラメータの 0 から始まるインデックス。<br><b>parameterName</b> : 削除する AseParameter オブジェクトの名前。                                    |
| 実装    | <ul style="list-style-type: none"><li>構文 1 では IList.RemoveAt を実装。</li><li>構文 2 では IDataParameterCollection.RemoveAt を実装。</li></ul> |

## AseRowsCopiedEventArgs クラス

説明 AseRowsCopiedEvent の引数データを表します。

基本クラス EventArgs

### AseRowsCopiedEventArgs コンストラクタ

説明 AseRowsCopiedEventArgs クラスの新しいインスタンスを初期化します。

構文 `void AseRowsCopiedEventArgs( int num )`

パラメータ **num** コピーするローの数。

### Abort プロパティ

説明 バルク・コピー操作をアボートするかどうかを指定します。

構文 `bool Abort`

アクセス 読み込みと書き込み

### RowCopied プロパティ

説明 コピーされたローの数。

構文 `int RowCopied`

アクセス 読み込みと書き込み

## AseRowsCopiedEventHandler デリゲート

説明 AseRowsCopied イベントを処理するメソッドを表します。

構文 `void AseRowsCopiedEventHandler( Object sender, AseRowsCopiedEventArgs e )`

パラメータ **e** イベント引数。

**sender** イベントをトリガしたオブジェクト。

## AseRowUpdatedEventArgs クラス

説明 RowUpdated イベントのデータを提供します。

基本クラス RowUpdatedEventArgs

### AseRowUpdatedEventArgs コンストラクタ

説明 AseRowUpdatedEventArgs クラスの新しいインスタンスを初期化します。

構文 **AseRowUpdatedEventArgs**( DataRow *dataRow*, IDbCommand *command*, StatementType *statementType*, DataTableMapping *tableMapping* )

パラメータ **dataRow** : Update によって送信された DataRow。

**command** : Update の呼び出し時に実行される IDbCommand。

**statementType** : 実行するクエリの種類を指定する StatementType 値の1つ。

**tableMapping** : Update によって送信された DataTableMapping を表します。

### Command プロパティ

説明 Update を呼び出したときに実行される AseCommand を表します。

構文 AseCommand **Command**

アクセス 読み込み専用

### Errors プロパティ

説明 コマンドの実行時に、Adaptive Server によって生成されたすべてのエラーを表します。RowUpdatedEventArgs から継承されます。

構文 Exception **Errors**

プロパティ値 コマンドの実行時に、Adaptive Server によって生成されたエラーを表します。

アクセス 読み込みと書き込み

### RecordsAffected プロパティ

説明 SQL 文の実行によって変更、挿入、削除されたローの数を表します。RowUpdatedEventArgs から継承されます。

構文 int **RecordsAffected**

|        |                                                                            |
|--------|----------------------------------------------------------------------------|
| プロパティ値 | 変更、挿入、または削除されたローの数。影響を受けたローがない場合または文が失敗した場合は 0、 <b>Select</b> 文の場合は -1 です。 |
| アクセス   | 読み込み専用                                                                     |

## Row プロパティ

|      |                                                                            |
|------|----------------------------------------------------------------------------|
| 説明   | <b>Update</b> によって送信された <b>DataRow</b> を表します。RowUpdatedEventArgs から継承されます。 |
| 構文   | <b>DataRow Row</b>                                                         |
| アクセス | 読み込み専用                                                                     |

## StatementType プロパティ

|      |                                                                                            |
|------|--------------------------------------------------------------------------------------------|
| 説明   | 実行された SQL 文の種類を表します。RowUpdatedEventArgs から継承されます。                                          |
| 構文   | <b>StatementType StatementType</b>                                                         |
| アクセス | 読み込み専用                                                                                     |
| 使用法  | StatementType は、 <b>Select</b> 、 <b>Insert</b> 、 <b>Update</b> 、または <b>Delete</b> のいずれかです。 |

## Status プロパティ

|        |                                                                                                   |
|--------|---------------------------------------------------------------------------------------------------|
| 説明     | Command プロパティの UpdateStatus を表します。RowUpdatedEventArgs から継承されます。                                   |
| 構文     | <b>UpdateStatus Status</b>                                                                        |
| プロパティ値 | 次の UpdateStatus 値のいずれか。Continue ( デフォルト )、ErrorsOccurred、SkipAllRemainingRows、または SkipCurrentRow。 |
| アクセス   | 読み込みと書き込み                                                                                         |

## TableMapping プロパティ

|      |                                                                                     |
|------|-------------------------------------------------------------------------------------|
| 説明   | <b>Update</b> によって送信された <b>DataTableMapping</b> を表します。RowUpdatedEventArgs から継承されます。 |
| 構文   | <b>DataTableMapping TableMapping</b>                                                |
| アクセス | 読み込み専用                                                                              |

## AseRowUpdatingEventArgs クラス

説明 RowUpdating イベントのデータを提供します。

基本クラス RowUpdatingEventArgs

### AseRowUpdatingEventArgs コンストラクタ

説明 AseRowUpdatingEventArgs クラスの新しいインスタンスを初期化します。

構文 **AseRowUpdatingEventArgs**( DataRow *row*, IDbCommand *command*, StatementType *statementType*, DataTableMapping *tableMapping* )

パラメータ

- row** : 更新する DataRow。
- command** : 更新時に実行する IDbCommand。
- statementType** : 実行するクエリの種類を指定する StatementType 値の 1 つ。
- tableMapping** : Update によって送信された DataTableMapping を表します。

### Command プロパティ

説明 Update の実行時に実行する AseCommand を表します。

構文 AseCommand **Command**

アクセス 読み込みと書き込み

### Errors プロパティ

説明 コマンドの実行時に、Adaptive Server によって生成されたすべてのエラーを表します。RowUpdatingEventArgs から継承されます。

構文 Exception **Errors**

プロパティ値 コマンドの実行時に、Adaptive Server によって生成されたエラーを表します。

アクセス 読み込みと書き込み

### Row プロパティ

説明 Update によって送信された DataRow を表します。RowUpdatingEventArgs から継承されます。

構文 DataRow **Row**

アクセス 読み込み専用



## StatementType プロパティ

|      |                                                          |
|------|----------------------------------------------------------|
| 説明   | 実行された SQL 文の種類を表します。RowUpdatingEventArgs から継承されます。       |
| 構文   | StatementType <b>StatementType</b>                       |
| アクセス | 読み込み専用                                                   |
| 使用法  | StatementType は、Select、Insert、Update、または Delete のいずれかです。 |

## Status プロパティ

|        |                                                                                                 |
|--------|-------------------------------------------------------------------------------------------------|
| 説明     | Command プロパティの UpdateStatus を表します。RowUpdatingEventArgs から継承されます。                                |
| 構文     | UpdateStatus <b>Status</b>                                                                      |
| プロパティ値 | 次の UpdateStatus 値のいずれか。Continue (デフォルト)、ErrorsOccurred、SkipAllRemainingRows、または SkipCurrentRow。 |
| アクセス   | 読み込みと書き込み                                                                                       |

## TableMapping プロパティ

|      |                                                                        |
|------|------------------------------------------------------------------------|
| 説明   | Update によって送信された DataTableMapping を表します。RowUpdatingEventArgs から継承されます。 |
| 構文   | DataTableMapping <b>TableMapping</b>                                   |
| アクセス | 読み込み専用                                                                 |

## AseRowUpdatedEventHandler デリゲート

|       |                                                                                   |
|-------|-----------------------------------------------------------------------------------|
| 説明    | AseDataAdapter の RowUpdated イベントを処理するメソッドを表します。                                   |
| 構文    | void <b>AseRowUpdatedEventHandler</b> ( object sender, AseRowUpdatedEventArgs e ) |
| パラメータ | <b>sender</b> イベントの発生元。<br><b>e</b> イベント・データが格納された AseRowUpdatedEventArgs。        |

## AseRowUpdatingEventHandler デリゲート

|       |                                                                                                    |
|-------|----------------------------------------------------------------------------------------------------|
| 説明    | AseDataAdapter の RowUpdating イベントを処理するメソッドを表します。                                                   |
| 構文    | void <b>AseRowUpdatingEventHandler</b> ( object <i>sender</i> , AseRowUpdatingEventArgs <i>e</i> ) |
| パラメータ | <b>sender</b> イベントの発生元。<br><b>e</b> イベント・データが格納された AseRowUpdatingEventArgs。                        |

## AseTransaction クラス

|       |                                                                                                                                                                                                                                 |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | SQL トランザクションを表します。                                                                                                                                                                                                              |
| 基本クラス | オブジェクト                                                                                                                                                                                                                          |
| 実装    | IDbTransaction                                                                                                                                                                                                                  |
| 使用法   | <ul style="list-style-type: none"><li>• AseTransaction には、コンストラクタはありません。AseTransaction オブジェクトを取得するには、AseConnection.BeginTransaction() メソッドを使用します。</li><li>• トランザクションにコマンドを関連付けるには、AseCommand.Transaction プロパティを使用します。</li></ul> |
| 参照    | <a href="#">「BeginTransaction メソッド」 (131 ページ)</a> 、 <a href="#">「トランザクション処理」 (75 ページ)</a> 、 <a href="#">「AseCommand オブジェクトを使用したローの挿入、更新、削除」 (42 ページ)</a>                                                                        |

## Commit メソッド

|    |                          |
|----|--------------------------|
| 説明 | データベース・トランザクションをコミットします。 |
| 構文 | void <b>Commit</b> ( )   |
| 実装 | IDbTransaction.Commit    |

## Connection プロパティ

|      |                                                                                                             |
|------|-------------------------------------------------------------------------------------------------------------|
| 説明   | トランザクションに関連付けられている AseConnection オブジェクトを表します。トランザクションが無効になっている場合は null 参照 (Visual Basic の場合は “Nothing”) です。 |
| 構文   | AseConnection <b>Connection</b>                                                                             |
| アクセス | 読み込み専用                                                                                                      |

**使用法** 1つのアプリケーションが複数のデータベース接続を保持し、それぞれの接続で0または1つのトランザクションが実行される場合があります。このプロパティを使用すると、`BeginTransaction` によって作成された特定のトランザクションに関連付けられている接続オブジェクトを識別できます。

## IsolationLevel プロパティ

**説明** このトランザクションの独立性レベルを指定します。

**構文** `IsolationLevel IsolationLevel`

**アクセス** 読み込み専用

**プロパティ値** このトランザクションの独立性レベル。 `ReadCommitted` (デフォルト)、`ReadUncommitted`、`RepeatableRead`、または `Serializable` を指定できます。

**実装**  `IDbTransaction.IsolationLevel`

## Rollback メソッド

**説明** データベース・トランザクションをロールバックします。

**構文** `void Rollback()`

**実装**  `IDbTransaction.Rollback`

**使用法** `Rollback` は同期的です。最初に `BeginTransaction()` を呼び出してから、返された `AseTransaction` に対して `Rollback` を呼び出します。

## TraceEnterEventHandler デリゲート

**説明** `TraceEnter` イベントを処理するメソッドを表します。

**構文** `void TraceEnterEventHandler( AseConnection connection, Object source, string method, Object[] parameters)`

**パラメータ**

- connection** イベントが発生した接続。
- source** イベントをトリガしたオブジェクト。
- method** 入力されたメソッド。
- parameters** 入力されたメソッドのパラメータ。

## TraceExitEventHandler デリゲート

|       |                                                                                                                                                    |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 説明    | TraceExit イベントを処理するメソッドを表します。                                                                                                                      |
| 構文    | <code>void TraceExitEventHandler( AseConnection <i>connection</i>, Object <i>source</i>, string <i>method</i>, Object[] <i>returnValue</i>)</code> |
| パラメータ | <b>connection</b> イベントが発生した接続。<br><b>source</b> イベントをトリガしたオブジェクト。<br><b>method</b> 終了したメソッド。<br><b>returnValue</b> 終了したメソッドの戻り値。                   |

# 索引

## A

- AcceptChangesDuringFill プロパティ
  - ASE ADO.NET Data Provider API 138
- Add メソッド
  - ASE ADO.NET Data Provider API 170
- ADO.NET プロバイダ
  - ASE ADO.NET Data Provider API 103
  - POOLING オプション 33
  - 接続プール 33
- API リファレンス
  - ASE ADO.NET Data Provider API 103
- ASE .NET Data Provider API
  - GetUInt16 メソッド 153
- ASE ADO.NET Data Provider
  - C# プロジェクトでの DLL 参照の追加 29
  - Simple コード・サンプルの使用 11
  - Table Viewer コード・サンプルの使用 16
  - Visual Basic .NET プロジェクトでの DLL 参照の追加 29
  - エラー処理 78
  - コード・サンプルの使用 11
  - サンプル・プロジェクトの実行 9
  - 時刻値の取得 70
  - システムの稼働条件 2
  - ストアード・プロシージャの実行 72
  - 説明 1
  - データのアクセス 36
  - データの更新 36
  - データの削除 36
  - データの挿入 36
  - データベースへの接続 31
  - トランザクション処理 75
  - 配備 2
- ASE ADO.NET Data Provider API
  - AcceptChangesDuringFill プロパティ 138
  - Add メソッド 170
  - API リファレンス 103
  - AseCommand クラス 113
  - AseCommand コンストラクタ 114
  - AseCommandBuilder クラス 119
  - AseCommandBuilder コンストラクタ 120
  - AseConnection クラス 125
  - AseConnection コンストラクタ 126
  - AseDataAdapter クラス 137
  - AseDataAdapter コンストラクタ 137
  - AseDataReader クラス 144
  - AseDbType プロパティ 165
  - AseDbType 列挙型 157
  - AseError クラス 159
  - AseErrorCollection クラス 161
  - AseException クラス 162
  - AseInfoMessageEventArgs クラス 163
  - AseInfoMessageEventHandler デリゲート 164
  - AseParameter クラス 165
  - AseParameter コンストラクタ 165
  - AseParameterCollection クラス 169
  - AsePermission クラス 112
  - AsePermission コンストラクタ 112
  - AsePermissionAttribute クラス 112
  - AsePermissionAttribute コンストラクタ 113
  - AseRowUpdatedEventArgs クラス 174
  - AseRowUpdatedEventArgs コンストラクタ 174
  - AseRowUpdatedEventHandler デリゲート 177
  - AseRowUpdatingEventArgs クラス 176
  - AseRowUpdatingEventArgs メソッド 176
  - AseRowUpdatingEventHandler デリゲート 178
  - AseTransaction クラス 178
  - BeginTransaction メソッド 131
  - Cancel メソッド 114
  - ChangeDatabase メソッド 131
  - Clear メソッド 170
  - Close メソッド 131, 145
  - Command プロパティ 174
  - CommandText プロパティ 114
  - CommandTimeout プロパティ 114
  - CommandType プロパティ 115
  - Commit メソッド 178
  - Connection プロパティ 115, 178
  - ConnectionString プロパティ 132
  - ConnectionTimeout プロパティ 133

## 索引

- Contains メソッド 170
- ContinueUpdateOnError プロパティ 138
- CopyTo メソッド 161, 171
- Count プロパティ 162, 171
- CreateCommand メソッド 133
- CreateParameter メソッド 115
- CreatePermission メソッド 113
- DataAdapter プロパティ 120
- Database プロパティ 133
- DbType プロパティ 166
- DeleteCommand プロパティ 120, 138
- Depth プロパティ 145
- DeriveParameters メソッド 120
- Direction プロパティ 166
- Dispose メソッド 121, 145
- ErrorNumber プロパティ 159
- Errors プロパティ 162, 163, 174, 176
- ExecuteNonQuery メソッド 116
- ExecuteReader メソッド 116
- ExecuteScalar メソッド 117
- ExecuteXMLReader メソッド 117
- FieldCount プロパティ 145
- Fill メソッド 139
- FillError イベント 139
- FillSchema メソッド 140
- GetBoolean メソッド 146
- GetByte メソッド 146
- GetBytes メソッド 146
- GetChar メソッド 147
- GetChars メソッド 147
- GetDataTypeName メソッド 148
- GetDateTime メソッド 148
- GetDecimal メソッド 149
- GetDeleteCommand メソッド 121
- GetDouble メソッド 149
- GetFieldType メソッド 149
- GetFillParameters メソッド 140
- GetFloat メソッド 150
- GetInsertCommand メソッド 121
- GetInt16 メソッド 150
- GetInt32 メソッド 150
- GetName メソッド 151
- GetOrdinal メソッド 151
- GetSchemaTable メソッド 152
- GetString メソッド 153
- GetUInt32 メソッド 153
- GetUInt64 メソッド 154
- GetUpdateCommand メソッド 122
- GetValue メソッド 154
- GetValues メソッド 154
- InfoMessage イベント 134
- Insert メソッド 171
- InsertCommand プロパティ 122, 141
- IsClosed プロパティ 155
- IsDBNull メソッド 155
- IsNullable プロパティ 166
- IsolationLevel プロパティ 179
- Item プロパティ 155, 162, 172
- Message プロパティ 159, 163
- MissingMappingAction プロパティ 141
- MissingSchemaAction プロパティ 141
- NextResult メソッド 156
- Open メソッド 134
- ParameterName プロパティ 166
- Parameters プロパティ 118
- Precision プロパティ 167
- Prepare メソッド 118
- QuotePrefix プロパティ 123
- QuoteSuffix プロパティ 124
- Read メソッド 156
- RecordsAffected プロパティ 156, 174
- RefreshSchema メソッド 124
- Remove メソッド 172
- RemoveAt メソッド 172
- Rollback メソッド 179
- Row プロパティ 175, 176
- RowUpdated イベント 142
- RowUpdating イベント 142
- Scale プロパティ 167
- SelectCommand プロパティ 125, 143
- Size プロパティ 167
- SourceColumn プロパティ 168
- SourceVersion プロパティ 168
- SqlState プロパティ 159
- State プロパティ 134
- StateChange イベント 135
- StatementType プロパティ 175, 177
- Status プロパティ 175, 177
- TableMapping プロパティ 175, 177
- TableMappings プロパティ 143
- ToString メソッド 159, 164, 169
- Transaction プロパティ 119

- Update メソッド 143
- UpdateCommand プロパティ 125, 144
- UpdatedRowSource プロパティ 119
- Value プロパティ 169
- ASE ADO.NET Data Provider サンプル・アプリケーションの使用 11
- ASE ADO.NET Data Provider の概要 1
- ASE ADO.NET Data Provider を使用したアプリケーション開発 29, 81
- AseCommand クラス
  - ASE ADO.NET Data Provider API 113
  - Visual Studio .NET プロジェクトでの使用 14
  - 使用 36
  - 説明 36
  - データの検索 37
  - データの更新 42
  - データの削除 42
  - データの挿入 42
- AseCommand コンストラクタ
  - ASE ADO.NET Data Provider API 114
- AseCommandBuilder クラス
  - ASE ADO.NET Data Provider API 119
- AseCommandBuilder コンストラクタ
  - ASE ADO.NET Data Provider API 120
- AseConnection 関数
  - Visual Studio .NET プロジェクトでの使用 19
- AseConnection クラス
  - ASE ADO.NET Data Provider API 125
  - Visual Studio .NET プロジェクトでの使用 14
  - データベースへの接続 31
- AseConnection コンストラクタ
  - ASE ADO.NET Data Provider API 126
- AseDataAdapter
  - プライマリ・キー値の取得 62
- AseDataAdapter クラス
  - ASE ADO.NET Data Provider API 137
  - Visual Studio .NET プロジェクトでの使用 20
  - 結果セットのスキーマ情報の取得 61
  - 使用 49
  - 説明 36
  - データの検索 49
  - データの更新 50
  - データの削除 50
  - データの挿入 50
- AseDataAdapter コンストラクタ
  - ASE ADO.NET Data Provider API 137
- AseDataReader クラス
  - ASE ADO.NET Data Provider API 144
  - Visual Studio .NET プロジェクトでの使用 15
  - 使用 37
- AseDbType プロパティ
  - ASE ADO.NET Data Provider API 165
- AseDbType 列挙型
  - ASE ADO.NET Data Provider API 157
  - データ型 157
- AseError クラス
  - ASE ADO.NET Data Provider API 159
- AseErrorCollection クラス
  - ASE ADO.NET Data Provider API 161
- AseException クラス
  - ASE ADO.NET Data Provider API 162
- AseInfoMessageEventArgs クラス
  - ASE ADO.NET Data Provider API 163
- AseInfoMessageEventHandler デリゲート
  - ASE ADO.NET Data Provider API 164
- AseParameter クラス
  - ASE ADO.NET Data Provider API 165
- AseParameter コンストラクタ
  - ASE ADO.NET Data Provider API 165
- AseParameterCollection クラス
  - ASE ADO.NET Data Provider API 169
- AsePermission クラス
  - ASE ADO.NET Data Provider API 112
- AsePermission コンストラクタ
  - ASE ADO.NET Data Provider API 112
- AsePermissionAttribute クラス
  - ASE ADO.NET Data Provider API 112
- AsePermissionAttribute コンストラクタ
  - ASE ADO.NET Data Provider API 113
- AseRowUpdatedEventArgs クラス
  - ASE ADO.NET Data Provider API 174
- AseRowUpdatedEventArgs コンストラクタ
  - ASE ADO.NET Data Provider API 174
- AseRowUpdatedEventHandler デリゲート
  - ASE ADO.NET Data Provider API 177
- AseRowUpdatingEventArgs クラス
  - ASE ADO.NET Data Provider API 176
- AseRowUpdatingEventArgs メソッド
  - ASE ADO.NET Data Provider API 176
- AseRowUpdatingEventHandler デリゲート
  - ASE ADO.NET Data Provider API 178
- AseTransaction クラス
  - ASE ADO.NET Data Provider API 178
  - 使用 75

## 索引

- ### B
- BeginTransaction メソッド
    - ASE ADO.NET Data Provider API 131
- ### C
- Cancel メソッド
    - ASE ADO.NET Data Provider API 114
  - ChangeDatabase メソッド
    - ASE ADO.NET Data Provider API 131
  - Clear メソッド
    - ASE ADO.NET Data Provider API 170
  - Close メソッド
    - ASE ADO.NET Data Provider API 131, 145
  - ColumnSize 152
  - Command プロパティ
    - ASE ADO.NET Data Provider API 174, 176
  - CommandText プロパティ
    - ASE ADO.NET Data Provider API 114
  - CommandTimeout プロパティ
    - ASE ADO.NET Data Provider API 114
  - CommandType プロパティ
    - ASE ADO.NET Data Provider API 115
  - Commit メソッド
    - ASE ADO.NET Data Provider API 178
  - Connection プロパティ
    - ASE ADO.NET Data Provider API 115, 178
  - ConnectionString プロパティ
    - ASE ADO.NET Data Provider API 132
  - ConnectionTimeout プロパティ
    - ASE ADO.NET Data Provider API 133
  - Contains メソッド
    - ASE ADO.NET Data Provider API 170
  - ContinueUpdateOnError プロパティ
    - ASE ADO.NET Data Provider API 138
  - CopyTo メソッド
    - ASE ADO.NET Data Provider API 161, 171
  - Count プロパティ
    - ASE ADO.NET Data Provider API 162, 171
  - CreateCommand メソッド
    - ASE ADO.NET Data Provider API 133
  - CreateParameter メソッド
    - ASE ADO.NET Data Provider API 115
  - CreatePermission メソッド
    - ASE ADO.NET Data Provider API 113
- ### D
- DataAdapter
    - 結果セットのスキーマ情報の取得 61
    - 使用 49
    - 説明 36
    - データの検索 49
    - データの更新 50
    - データの削除 50
    - データの挿入 50
    - プライマリ・キー値の取得 62
  - DataAdapter プロパティ
    - ASE ADO.NET Data Provider API 120
  - Database プロパティ
    - ASE ADO.NET Data Provider API 133
  - DbType プロパティ
    - ASE ADO.NET Data Provider API 166
  - DeleteCommand プロパティ
    - ASE ADO.NET Data Provider API 120, 138
  - Depth プロパティ
    - ASE ADO.NET Data Provider API 145
  - DeriveParameters メソッド
    - ASE ADO.NET Data Provider API 120
  - Direction プロパティ
    - ASE ADO.NET Data Provider API 166
  - Dispose メソッド
    - ASE ADO.NET Data Provider API 121, 145
  - DSURL 85
- ### E
- EncryptPassword 88
  - ErrorNumber プロパティ
    - ASE ADO.NET Data Provider API 159
  - Errors プロパティ
    - ASE ADO.NET Data Provider API 162, 163, 174, 176
  - ExecuteNonQuery メソッド
    - ASE ADO.NET Data Provider API 116
  - ExecuteReader メソッド
    - ASE ADO.NET Data Provider API 116
    - 使用 38
  - ExecuteScalar メソッド
    - ASE ADO.NET Data Provider API 117
    - 使用 39
  - ExecuteXMLReader メソッド
    - ASE ADO.NET Data Provider API 117



**F**

|                               |     |
|-------------------------------|-----|
| FieldCount プロパティ              |     |
| ASE ADO.NET Data Provider API | 145 |
| Fill メソッド                     |     |
| ASE ADO.NET Data Provider API | 139 |
| FillError イベント                |     |
| ASE ADO.NET Data Provider API | 139 |
| FillSchema メソッド               |     |
| ASE ADO.NET Data Provider API | 140 |
| 使用                            | 61  |

**G**

|                               |     |
|-------------------------------|-----|
| GAC                           |     |
| 配備と設定                         | 7   |
| 配備と設定に使用しない                   | 8   |
| GetBoolean メソッド               |     |
| ASE ADO.NET Data Provider API | 146 |
| GetByte メソッド                  |     |
| ASE ADO.NET Data Provider API | 146 |
| GetBytes メソッド                 |     |
| ASE ADO.NET Data Provider API | 146 |
| 使用                            | 68  |
| GetChar メソッド                  |     |
| ASE ADO.NET Data Provider API | 147 |
| GetChars メソッド                 |     |
| ASE ADO.NET Data Provider API | 147 |
| 使用                            | 68  |
| GetDataTypeName メソッド          |     |
| ASE ADO.NET Data Provider API | 148 |
| GetDateTime メソッド              |     |
| ASE ADO.NET Data Provider API | 148 |
| GetDecimal メソッド               |     |
| ASE ADO.NET Data Provider API | 149 |
| GetDeleteCommand メソッド         |     |
| ASE ADO.NET Data Provider API | 121 |
| GetDouble メソッド                |     |
| ASE ADO.NET Data Provider API | 149 |
| GetFieldType メソッド             |     |
| ASE ADO.NET Data Provider API | 149 |
| GetFillParameters メソッド        |     |
| ASE ADO.NET Data Provider API | 140 |
| GetFloat メソッド                 |     |
| ASE ADO.NET Data Provider API | 150 |
| GetInsertCommand メソッド         |     |
| ASE ADO.NET Data Provider API | 121 |
| GetInt16 メソッド                 |     |
| ASE ADO.NET Data Provider API | 150 |

|                               |     |
|-------------------------------|-----|
| GetInt32 メソッド                 |     |
| ASE ADO.NET Data Provider API | 150 |
| GetName メソッド                  |     |
| ASE ADO.NET Data Provider API | 151 |
| GetOrdinal メソッド               |     |
| ASE ADO.NET Data Provider API | 151 |
| GetSchemaTable メソッド           |     |
| ASE ADO.NET Data Provider API | 152 |
| 使用                            | 47  |
| GetString メソッド                |     |
| ASE ADO.NET Data Provider API | 153 |
| GetTimeSpan メソッド              |     |
| 使用                            | 70  |
| GetUInt16 メソッド                |     |
| ASE ADO.NET Data Provider API | 153 |
| GetUInt32 メソッド                |     |
| ASE ADO.NET Data Provider API | 153 |
| GetUInt64 メソッド                |     |
| ASE ADO.NET Data Provider API | 154 |
| GetUpdateCommand メソッド         |     |
| ASE ADO.NET Data Provider API | 122 |
| GetValue メソッド                 |     |
| ASE ADO.NET Data Provider API | 154 |
| GetValues メソッド                |     |
| ASE ADO.NET Data Provider API | 154 |

**H**

|    |    |
|----|----|
| HA | 93 |
|----|----|

**I**

|                               |          |
|-------------------------------|----------|
| IndexOf メソッド                  |          |
| ASE ADO.NET Data Provider API | 171      |
| InfoMessage イベント              |          |
| ASE ADO.NET Data Provider API | 134      |
| Insert メソッド                   |          |
| ASE ADO.NET Data Provider API | 171      |
| InsertCommand プロパティ           |          |
| ASE ADO.NET Data Provider API | 122, 141 |
| IsClosed プロパティ                |          |
| ASE ADO.NET Data Provider API | 155      |
| IsDBNull メソッド                 |          |
| ASE ADO.NET Data Provider API | 155      |
| IsNullable プロパティ              |          |
| ASE ADO.NET Data Provider API | 166      |

## 索引

IsolationLevel プロパティ  
ASE ADO.NET Data Provider API 179  
Item プロパティ  
ASE ADO.NET Data Provider API 155, 162, 172

## K

Kerberos 95  
稼動条件 96  
プロセスの概要 95  
kinit ユーティリティ 97

## L

LDAP 85

## M

Message プロパティ  
ASE ADO.NET Data Provider API 159, 163  
MissingMappingAction プロパティ  
ASE ADO.NET Data Provider API 141  
MissingSchemaAction プロパティ  
ASE ADO.NET Data Provider API 141

## N

NextResult メソッド  
ASE ADO.NET Data Provider API 156

## O

Open メソッド  
ASE ADO.NET Data Provider API 134

## P

ParameterName プロパティ  
ASE ADO.NET Data Provider API 166  
Parameters プロパティ  
ASE ADO.NET Data Provider API 118  
POOLING オプション  
ADO.NET プロバイダ 33  
Precision プロパティ  
ASE ADO.NET Data Provider API 167  
Prepare メソッド  
ASE ADO.NET Data Provider API 118

## Q

QuotePrefix プロパティ  
ASE ADO.NET Data Provider API 123  
QuoteSuffix プロパティ  
ASE ADO.NET Data Provider API 124

## R

Read メソッド  
ASE ADO.NET Data Provider API 156  
RecordsAffected プロパティ  
ASE ADO.NET Data Provider API 156, 174  
RefreshSchema メソッド  
ASE ADO.NET Data Provider API 124  
Remove メソッド  
ASE ADO.NET Data Provider API 172  
RemoveAt メソッド  
ASE ADO.NET Data Provider API 172  
Rollback メソッド  
ASE ADO.NET Data Provider API 179  
Row プロパティ  
ASE ADO.NET Data Provider API 175, 176  
RowUpdated イベント  
ASE ADO.NET Data Provider API 142  
RowUpdating イベント  
ASE ADO.NET Data Provider API 142

**S**

- Scale プロパティ
  - ASE ADO.NET Data Provider API 167
- SelectCommand プロパティ
  - ASE ADO.NET Data Provider API 125, 143
- Size プロパティ
  - ASE ADO.NET Data Provider API 167
- SourceColumn プロパティ
  - ASE ADO.NET Data Provider API 168
- SourceVersion プロパティ
  - ASE ADO.NET Data Provider API 168
- SqlState プロパティ
  - ASE ADO.NET Data Provider API 159
- SSL (Secure Sockets Layer) 90
- State プロパティ
  - ASE ADO.NET Data Provider 34
  - ASE ADO.NET Data Provider API 134
- StateChange イベント
  - ASE ADO.NET Data Provider API 135
- StatementType プロパティ
  - ASE ADO.NET Data Provider API 175, 177
- Status プロパティ
  - ASE ADO.NET Data Provider API 175, 177
- Sybase.Data.AsaClient.DLL
  - Visual Studio .NET プロジェクトでの参照の追加 29

**T**

- TableMapping プロパティ
  - ASE ADO.NET Data Provider API 175, 177
- TableMappings プロパティ
  - ASE ADO.NET Data Provider API 143
- Time 構造体
  - ASE ADO.NET Data Provider の時刻値 70
- TimeSpan
  - ASE ADO.NET Data Provider 70
- ToString メソッド
  - ASE ADO.NET Data Provider API 159, 164, 169
- Transaction プロパティ
  - ASE ADO.NET Data Provider API 119

**U**

- Update メソッド
  - ASE ADO.NET Data Provider API 143
- UpdateCommand プロパティ
  - ASE ADO.NET Data Provider API 125, 144
- UpdatedRowSource プロパティ
  - ASE ADO.NET Data Provider API 119

**V**

- Value プロパティ
  - ASE ADO.NET Data Provider API 169

**あ**

- アクセシビリティ xv

**い**

- イベント
  - FillError イベント 139
  - InfoMessage イベント 134
  - RowUpdated イベント 142
  - RowUpdating イベント 142
  - StateChange イベント 135

**え**

- エラー処理
  - ASE ADO.NET Data Provider 78

**お**

- 応答時間
  - AseDataAdapter 137
  - AseDataReader 144
- オブジェクト
  - ASE ADO.NET Data Provider API 103

## 索引

### か

- 稼働条件
  - Kerberos 96
- 関連マニュアル xi

### く

- グローバル・アセンブリ・キャッシュ 2

### こ

- 高可用性 93
- コンストラクタ
  - AseCommand 114
  - AseCommandBuilder メソッド 120
  - AseConnection コンストラクタ 126
  - AseDataAdapter メソッド 137
  - AseParameter 165
  - AsePermission コンストラクタ 112
  - AsePermissionAttribute コンストラクタ 113
  - AseRowUpdatedEventArgs コンストラクタ 174

### さ

- サポートされている Microsoft ADO.NET の機能 99
- サンプル
  - ASE ADO.NET Data Provider 11

### し

- 時刻
  - ASE ADO.NET Data Provider を使用した取得 70
- 時刻値の取得 70
- システムの稼働条件
  - ASE ADO.NET Data Provider 2

### す

- ストアド・プロシージャ
  - ASE ADO.NET Data Provider 72

### せ

- 接続
  - ASE ADO.NET Data Provider を使用したデータベースへの接続 31
- 接続ステータス
  - ASE ADO.NET Data Provider 34
- 接続プール
  - ADO.NET プロバイダ 33

### ち

- チュートリアル
  - ASE ADO.NET Data Provider の Simple コード・サンプルの使用 11
  - ASE ADO.NET Data Provider の Table Viewer コード・サンプルの使用 16

### て

- ディレクトリ・サービス 85
- データ
  - ASE ADO.NET Data Provider によるアクセス 36
  - ASE ADO.NET Data Provider による操作 36
- データ型
  - AseDbType 列挙型 157
- データに対するアクセスと操作
  - ASE ADO.NET Data Provider の使用 36
- デリゲート
  - AseInfoMessageEventHandler デリゲート 164
  - AseRowUpdatedEventHandler デリゲート 177
  - AseRowUpdatingEventHandler デリゲート 178

### と

- 独立性レベル
  - AseTransaction オブジェクトの設定 76
- トランザクション処理
  - ASE ADO.NET Data Provider の使用 75

### に

- 認証 95

## ね

ネットワーク認証 95

## は

## 配備

ASE ADO.NET Data Provider 2

ASE ADO.NET Data Provider アプリケーション 2

パスワードの暗号化 88

発行者ポリシー・ファイル 7

パラメータ

CreateParameter メソッド 115

## ひ

必要なファイル 2

## ふ

フェールオーバー 93

プライマリ・キー

値の取得 62

プロセスの概要

Kerberos 95

プロパティ

AcceptChangesDuringFill プロパティ 138

AseDbType プロパティ 165

Command プロパティ 174

CommandText プロパティ 114

CommandTimeout プロパティ 114

CommandType プロパティ 115

Connection プロパティ 115, 178

ConnectionString プロパティ 132

ConnectionTimeout プロパティ 133

ContinueUpdateOnError プロパティ 138

Count プロパティ 162, 171

DataAdapter プロパティ 120

Database プロパティ 133

DbType プロパティ 166

DeleteCommand プロパティ 120, 138

Depth プロパティ 145

Direction プロパティ 166

ErrorNumber プロパティ 159

Errors プロパティ 162, 163, 174, 176

FieldCount プロパティ 145

InsertCommand プロパティ 122, 141

IsClosed プロパティ 155

IsNullable プロパティ 166

IsolationLevel プロパティ 179

Item プロパティ 155

Message プロパティ 159, 163

MissingMappingAction プロパティ 141

MissingSchemaAction プロパティ 141

ParameterName プロパティ 166

Parameters プロパティ 118

Precision プロパティ 167

QuotePrefix プロパティ 123

QuoteSuffix プロパティ 124

RecordsAffected プロパティ 156, 174

Row プロパティ 175, 176

Scale プロパティ 167

SelectCommand プロパティ 125, 143

Size プロパティ 167

SourceColumn プロパティ 168

SourceVersion プロパティ 168

SqlState プロパティ 159

State プロパティ 134

StatementType プロパティ 175, 177

Status プロパティ 175, 177

TableMapping プロパティ 175, 177

TableMappings プロパティ 143

Transaction プロパティ 119

UpdateCommand プロパティ 125, 144

UpdatedRowSource プロパティ 119

Value プロパティ 169

## め

## メソッド

- Add メソッド 170
- AseRowUpdatingEventArgs メソッド 176
- BeginTransaction メソッド 131
- Cancel メソッド 114
- ChangeDatabase メソッド 131
- Clear メソッド 170
- Close メソッド 131, 145
- Commit メソッド 178
- Contains メソッド 170
- CopyTo メソッド 161, 171
- CreateCommand メソッド 133
- CreateParameter メソッド 115
- CreatePermission メソッド 113
- DeriveParameters メソッド 120
- Dispose メソッド 121, 145
- ExecuteNonQuery メソッド 116
- ExecuteReader メソッド 116
- ExecuteScalar メソッド 117
- ExecuteXMLReader メソッド 117
- Fill メソッド 139
- FillSchema メソッド 140
- GetBoolean メソッド 146
- GetByte メソッド 146
- GetBytes メソッド 146
- GetChar メソッド 147
- GetChars メソッド 147
- GetDataTypeName メソッド 148
- GetDateTime メソッド 148
- GetDecimal メソッド 149
- GetDeleteCommand メソッド 121
- GetDouble メソッド 149
- GetFieldType メソッド 149
- GetFillParameters メソッド 140
- GetFloat メソッド 150
- GetInsertCommand メソッド 121
- GetInt16 メソッド 150
- GetInt32 メソッド 150
- GetName メソッド 151
- GetOrdinal メソッド 151
- GetSchemaTable メソッド 152
- GetString メソッド 153
- GetUInt16 メソッド 153
- GetUInt32 メソッド 153
- GetUInt64 メソッド 154
- GetUpdateCommand メソッド 122
- GetValue メソッド 154
- GetValues メソッド 154
- Insert メソッド 171
- IsNull メソッド 155
- Item プロパティ 162, 172
- NextResult メソッド 156
- Open メソッド 134
- Prepare メソッド 118
- Read メソッド 156
- RefreshSchema メソッド 124
- Remove メソッド 172
- RemoveAt メソッド 172
- Rollback メソッド 179
- ToString メソッド 159, 164, 169
- Update メソッド 143