ECHO OLE DB Data Provider

Version 2.0

How to Contact Us

OSIsoft, Inc.
777 Davis St., Suite 250
San Leandro, CA 94577 USA
or PO Box 727
San Leandro, CA 94577 USA
(1) 510-297-5800 (main phone)
(1) 510-357-8136 (fax)
(1) 510-297-5828 (support phone)
support@osisoft.com

OSIsoft Canada ULC
1155 University St., Ste. 612
Montreal (QC) H3B 3A7 Canada
(1) 514-493-0663
(1) 514-493-0980 (fax)

OSIsoft, Ltd.
Unit A1, 255 Rawson St.
Auburn, Sydney 2144, Australia
(61) 2-9648 1511
(61) 2 9648 1522 (fax)

European Joint Venture
OSI Software GmbH
Hauptstrasse 30
D 63674 Altenstadt, Germany
(49) 6047-989 0
(49) 6047-989 188 (fax)

OSI Software, Ltd
Level 4, 126 Khyber Pass Rd.
Grafton Auckland New Zealand 1003
or PO Box 8256
Auckland New Zealand 1003
(64) 9-522-5900
(64) 9-522-5201 (fax)

Wired City
Level 3 Septimus Roe Square
256 Adelaide Terrace
Perth WA 6000 Australia
(61) 8 9218 9780

OSI Software Asia, Pte Ltd.
152 Beach Road
#09-06 Gateway East
Singapore 189721
(65) 391-1811
(65) 295-2488 (fax)

OSIsoft, Inc.
1266 Nan Jing West Road 200040
Shanghai China

For additional information, please see our Web site:
http://www.osisoft.com

ECHO Developers Guide.doc

Contents

Chapter 1 Introduction

Product Overview 1-2
Introduction to OLE DB 1-3
  OLE DB Architecture 1-3
  OLE DB Data Provider for ECHO 1-5

Chapter 2 Installation

Installation / Uninstallation Guidelines 2-2

Chapter 3 Configuration for Data Access

Configuration Attributes 3-2
ECHO OLE DB Specific Attributes 3-3
ECHO OLE DB Login Dialog Box 3-5
Universal Data Link (UDL) 3-6
  Creating a UDL File 3-6
  Editing with the Data Link Properties Dialog Box 3-7
  Connection String Format 3-10
  Setting Maximum Processing Time for Queries 3-10

Chapter 4 Supported ANSI SQL 92 (Subset)

SELECT Statements 4-2
INSERT, UPDATE, DELETE, CREATE/DROP DATABASE Statements 4-4
CAST Operator 4-5

Chapter 5 Catalogs and Tables

Catalog Usage 5-2
CREATE DATABASE / DROP DATABASE Statements 5-3
Table Structure- ECHO Catalog 5-4
Table Structure - Historians Catalog 5-7
Supported Data Types 5-14
Supported Time Stamp Formats 5-15
Examples of Other Statement Use 5-16
  Simple Expressions 5-16
  Simple Table Queries 5-16
  NULL Values 5-17
  DISTINCT 5-17
  TOP n 5-17
  GROUP BY, HAVING 5-17
  ORDER BY 5-17
  Inner Joins 5-17
  Nested Queries 5-18
  Subqueries 5-18
  UNION 5-18
  INSERT INTO SELECT 5-18

Chapter 6 ActiveX Data Objects - ADO

Description of the ActiveX Data Object....6-2
ADO Object Model 6-4
ADO Databinding ActiveX Controls 6-5
ADO and DBTIMESTAMP 6-6
Samples Provided....6-7
  ADO Intro Sample 6-7
  ADO.NET Intro Sample....6-9
  ASP Sample 6-10
  MyECHOSQL Sample....6-11
  UDL Sample....6-11

Chapter 7 OLE DB Server Environments

Microsoft SQL Server Linked Server....7-2
  Configuration Through the Enterprise Manager 7-3
  User Account and DCOM Settings for OutProc Instantiation....7-8
  Four-Part Names 7-10
  Sample SQL Statements....7-11
  Pass-Through Queries 7-13
  Known Linked Server Issues....7-14
Oracle Generic Connectivity 7-16
  Configuration of Generic Connectivity 7-17

Chapter 8 Advanced Topics

OLE DB – ODBC Transformation....8-2
  MS SQL Server 2000 as ODBC Gateway....8-3
  Oracle 8i/9i Generic Connectivity....8-4
  Attunity Connect 8-5
  Microsoft Access 2000 / 2002 as ODBC Client....8-6
Microsoft Excel 2000 / 2002 8-7
RowsetViewer 8-9
  Connection 8-10
  SQL Statement Execution 8-11
MS SQL Reporting Services 8-12

Chapter 9 Troubleshooting

Error Messages....9-2
Log File 9-3

Appendix A Tested OLE DB Clients

Clients Tested....A-2

Glossary

Index

Chapter 1 Introduction

This chapter outlines the features of the OLE DB Data Provider for the Embedded Component Historian Object (ECHO).

Product Overview

The Embedded Component Historian Object (ECHO) by OSIsoft is a tool that Manufacturers use to add a scalable, robust, high-performance time-series data historian to their hardware devices or software products.

ECHO is an open system, and as such, it is necessary to provide industry standard interfaces that allow other vendor products to connect to it. The two most important are OPC and OLE DB.

The ECHO SDK allows developers to write code that connects to ECHO using any development environment that supports COM (Component Object Model).

ECHO OLE DB, on the other hand, provides standard public interfaces that allow users to connect any client that works with OLE DB data sources. Programming is therefore reduced to configuration tasks. The number of these clients is continually growing because OLE DB standards are widely accepted.

ECHO itself is not a relational database, and ECHO OLE DB only provides a relational view on ECHO data by implementing SQL. Both configuration and the actual archive data is represented in the form of catalogs and tables.

Introduction to OLE DB

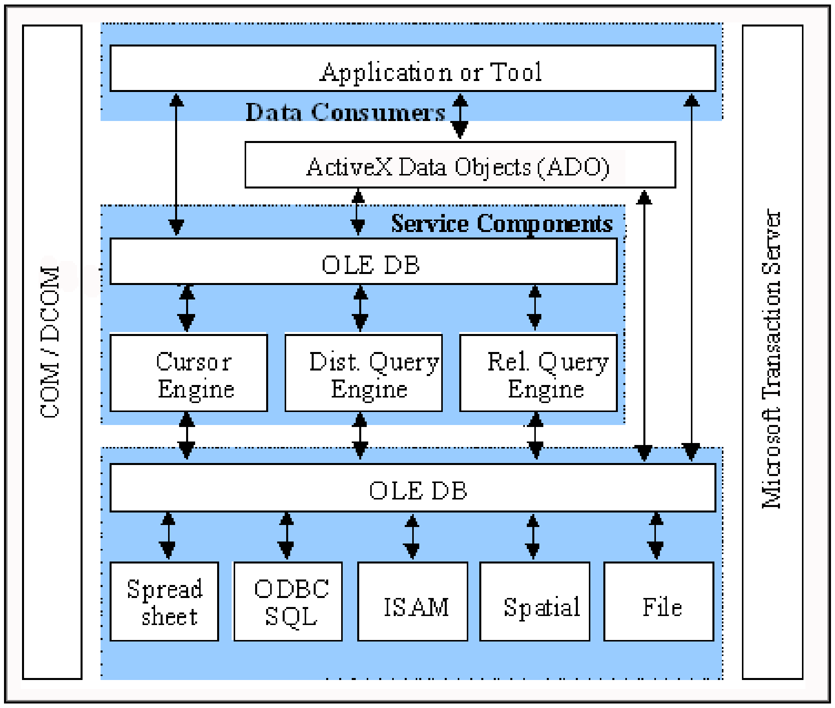
OLE DB is an open specification designed to build on the success of ODBC by providing an open standard for universal data access. Whereas ODBC was created to access relational databases, OLE DB is designed for both relational and non-relational information sources, including mainframe ISAM/VSAM and hierarchical databases, email and file system stores, text, graphical and geographical data, custom business objects, and more.

OLE DB defines a collection of COM interfaces that encapsulate various database management system services. These interfaces allow developers to create software components that implement such services. OLE DB components consist of the following.

- Data Providers, which expose data.
- Data Consumers, which use data.
- Service Providers, which extend the functionality of data providers by implementing extended interfaces not natively supported by the data stores; such as query processors, cursor engines, and synchronization service.

OLE DB Architecture

The following figure illustrates the Universal Data Access (UDA) architecture. This is a developing platform Microsoft defines for multi-tier enterprise applications that require access to diverse relational or non-relational data sources across intranets or the Internet.



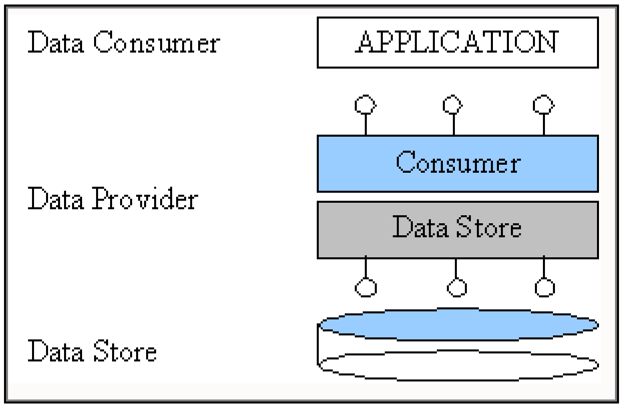
To give users universal access to external data stores, applications must be able to access the data store, read the data, and copy or modify it. To do this, applications must either recognize the format of the stored data, or else they must operate through a secondary program that can handle the data format.

For example, a financial report designer application might use a secondary ODBC driver to access data in a relational database. Such a secondary program does the following tasks.

- Accepts requests for data from the application.
- Accesses the data from the data store.
- Returns the data to the application.

Data access through secondary program requires a common language of communication, or interface, between the data requester (the application) and the secondary program. The more powerful the interface, the better the communication that is achieved.

The Microsoft OLE DB specification is a powerful interface that is supported by many commercially available applications. In addition to using these commercial applications for OLE DB, developers can also write customized applications that use the OLE DB interface for data access. When OLE DB is the interface used for data access, the application is called the OLE DB data consumer, and the secondary program is called the OLE DB data provider. The following figure shows how the OLE DB data provider and the data consumer use a common interface to communicate.



It is the function of the OLE DB data provider to convert OLE DB data requests received from the data consumer to the native format of the data store. Each data store used can have a unique native format.

Although data can be stored in a variety of formats, these formats fall into two broad categories.

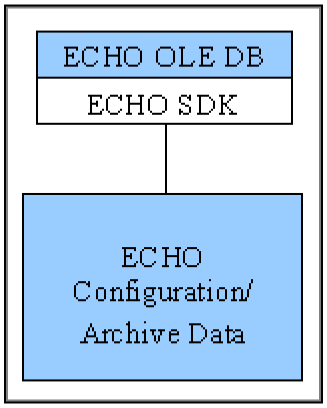
- **Relational data** – This data is stored in a database that is governed by a Relational Database Management System (RDBMS). The RDBMS presents the data as tables that consist of rows and columns. Typically, the RDBMS supports SQL to retrieve and update the data. The RDBMS also manages user access to the data. Microsoft SQL Server and Oracle are examples of RDBMS(s) that store relational data.
- **Non-Relational data** – This data is stored in a file system instead of tables of rows and columns. These systems usually do not provide a way to sort and retrieve the data in their files beyond simple text searching. Email and word processor files are examples of non-relational data stores. Some databases, such as Lotus Notes do not meet the strict definition of relational databases.

Structured Query Language (SQL) is often used by applications to read, retrieve, and update complex data in a relational data store.

NOTE: OLE DB data providers that support SQL processing are sometimes referred to as OLE DB (SQL).

OLE DB Data Provider for ECHO

ECHO itself is not a relational database and ECHO OLE DB exposes only the configuration and archive data in a relational way. See the following figure.



The ECHO OLE DB provider meets the OLE DB 2.0 specification, and in its basic form it is not dependent on any other component coming with Microsoft Universal Data Access Components (MDAC) suite.

However, it does require the ECHO SDK to be installed to connect to the ECHO archive engine. For a detailed description of the ECHO specific components and versions the ECHO OLE DB requirements, see the "Installation" chapter.

Chapter 2 Installation

This chapter describes how to install the ECHO OLE DB Data Provider.

Installation / Uninstallation Guidelines

Before You Start

It is recommended that you complete the following actions before you run the setup program.

- Before installing the OLE DB Provider, log on to your Windows system using an account with administrator privileges.
- Close any programs, particularly OSIsoft client or OSIsoft applications that are currently running.
- Uninstall any previous versions of the OLE DB Data Provider, including beta and pre-release versions.
- Verify that your operating system is one of the following: Windows XP Professional, Windows Server 2003, or Windows 2000 Professional (Service Pack 2 or later) operating systems.
- Verify that you have the following software package installed.

The ECHO SDK provides COM and DCOM access to ECHO archive engines. Therefore, the ECHO SDK Release 1.3.0 or greater must be resident on the client node.

System Requirements

The ECHO OLE DB Data Provider is designed to run on the Windows NT family of operating systems. Before installing, verify that your system meets the following requirements.

- Pentium ® (or Pentium-compatible) processor-based system (Pentium III 800 MHz processor or faster is recommended).
- At least 128 MB of RAM (256 MB recommended).
- Hard disk space: 70 MB of hard disk space needs to be available. For installation, 160 MB must be available.
- Monitor capability: Minimum of 256 colors and 800 X 600 pixel resolution.
- Windows XP Professional, Windows XP Server, or Windows 2000 Professional (Service Pack 2 or later) operating systems.

Installing the ECHO OLE DB Data Provider

Install the ECHO OLE DB Data Provider in either of two ways.

- The set-up kit – This set-up kit includes MDAC and the ECHO OLE DB. Each component is only installed or updated when necessary. The set-up kit uses the Microsoft Installer (MSI) technology and updates this as well.
- The plain Microsoft Installer (MSI) kit (by way of a download from Technical Support). This kit automatically copies and registers the ECHO OLE DB related DLL. This kit creates the necessary directories on your hard disk, and copies the files into the appropriate directories.

OverviewUninstalling the ECHO OLE DB Data Provider

To remove the OLE DB Data Provider from your system, use the standard Windows **Add/Remove Programs** utility in the Control Panel..

Installed Components

The list of installed files is provided in the release notes document ReadMe.doc.

In addition to the ECHO OLE DB Data Provider itself, these tools and examples are also installed.

- ADO Intro – This sample introduces the basic principles of ADO. For the resulting data set presentation, the Microsoft DataGrid Control is used.
- ADO.NET Intro – This simple example describes how to use ECHO OLE DB provider in “managed environment”.
- ASP (Active Server Pages) – Example that prints out results of a SELECT query in a web browser.
- MyECHOSQL - Simple C++ console application that executes a batch of SQL queries through ECHO OLE DB.
- UDL - Example of the UDL file

Chapter 3 Configuration for Data Access

This chapter describes how to configure the connection information.

Configuration Attributes

Before accessing data via an OLE DB data provider, you must provide the connection (initialization) information. This information can be saved in special text files, named Universal Data Link (UDL) or specified directly in the client application (data consumer).

The following table lists the configuration attributes.

Configuration Attributes

OLE DB Property	Description	Corresponding UDL Attribute
DBPROP_INIT_DATASOURCE (Required)	Name of the server to connect to.	"Data Source"
DBPROP_AUTH_USERID	User Name	"User ID"
DBPROP_INIT_CATALOG	Initial catalog (database)	"Initial Catalog"
DBPROP_INIT_PROMPT	Prompt mode (designates if the provider should ask for the missing information)	This property is not persisted in the UDL file.
DBPROP_INIT_PROVIDERSTRING	Provider specific attributes	"Extended Properties"
DBPROP_INIT_HWND	Window handle from the calling application (is used as a parent window of the dialog prompting for the missing information)	This property is not persisted in the UDL file.

ECHO OLE DB Specific Attributes

ECHO OLE DB Data Provider supports additional attributes that are set through the **DBPROP_INIT_PROVIDERSTRING** property (see the preceding table). These attributes are listed in the table below.

Specific Attributes

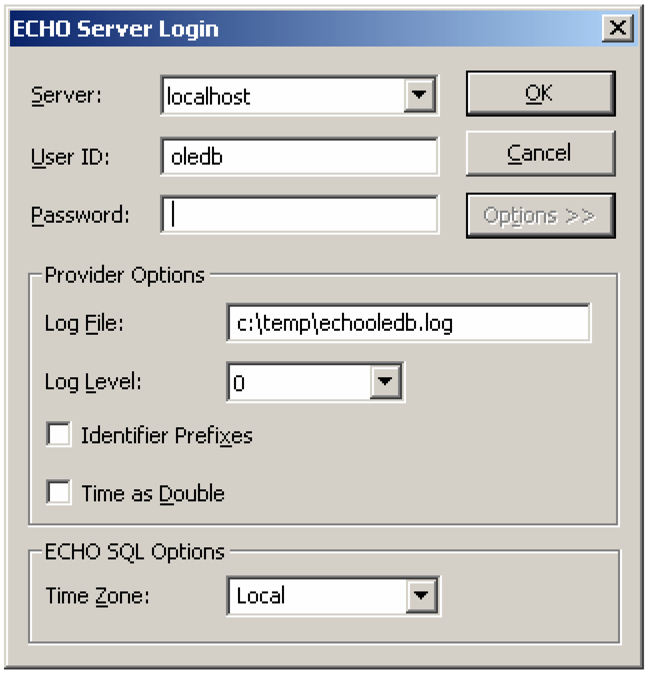
Name	Description
Always Return Rowset	Forces statements, such as INSERT, UPDATE, DELETE, CREATE DATABASE, etc., to return the number of affected rows in the form of a rowset. This allows you to use these 'non-SELECT statements' within the Linked Server environment, in combination with the OPENQUERY function. See section "Pass-Through Queries"in the "OLE DB Server Environments" chapter for more information. Example: *Always Return Rowset=True;* *Always Return Rowset=False; (default)*
Command Timeout	When used, the DBPROP_COMMANDTIMEOUT property is ignored. Value is in seconds. Example: *Command Timeout=10;*
Defer Execution	When set to true, the provider does not forward the SELECT * FROM table.. statement to ECHO engine. This statement is automatically generated by clients, such as SQL Server, DTS (Data Transformation Services), etc., before any DML statement to obtain metadata information. Example: *Defer Execution=true;* *Defer Execution=false; (default)*
Disable Server Selection	Flag indicates that the Server field in the login dialog box is disabled. This prevents server changes during the connection. Example: *Disable Server Selection=True;* *Disable Server Selection=False; (default)*
Identifier Prefixes (Not supported in Version 2.0 beta)	Flag indicates that names of columns, tables, and catalogs will be prefixed by "ch_." This handles naming conflicts when object names collide with client application keywords. Example: *Identifier Prefixes = True;* *Identifier Prefixes = False; (default)*
Log File	Full path to the log file. Example: *Log File = c:\temp\log\choledb.log; (default: no log file)*
Historians by GUID (Not supported in Version 2.0 beta)	Flag indicates that catalog names will be resolved through GUID. By default, catalog names correspond to IRP (Identity Resolution Plug-in) names (ECHO 1.3+) Example: *Historians by GUID = False; (default)* *Historians by GUID = True;*

Name	Description
Log Level	The amount of information printed in Log File increases with the Log Level as follows: 0 (default) : initialization properties, software versions 1: SQL queries, used optimization, query duration 2: OLE DB interfaces 3: OLE DB interfaces – more details Example: *Log Level=1;*
Shorten Primary Keys	This property forces the length of a string column (designed as a primary key) to be 255. The property is implemented because Microsoft Access automatically converts such a string column (length > 255) to the data type 'memo'. Example: *Shorten Primary Keys=true; (default false)*
Time as Double	When set to true the columns of the data type time are transformed to the data type of double. The value is in seconds.
Time Zone	Local: time stamps are in local timezone (default). Server: time stamps are adjusted according to the time zone of the archive engine. UTC: time stamps are in UTC Time stamps used in WHERE clauses are adjusted according to the given option as well. Example: *Time Zone=UTC*

NOTE: Each client application configures the provider in a different way. See the "Advanced Topics" chapter for more information about clients.

ECHO OLE DB Login Dialog Box

According to the **DBPROP_INIT_PROMPT** OLE DB property, the ECHO OLE DB login dialog box appears during the initialization of the provider. This allows you to select the ECHO Server and enter the ClientID. Additionally, after clicking **Options**, you set the **Log File** (ECHO OLE DB specific attribute) as shown in the following figure.



ECHO Server Login

Server: localhost

User ID: oledb

Password:

Provider Options

Log File: c:\temp\echooledb.log

Log Level: 0

☐ Identifier Prefixes

☐ Time as Double

ECHO SQL Options

Time Zone: Local

Universal Data Link (UDL)

A Universal Data Link is a text file with the .udl extension containing the connection information. This version of the connection information is referred to as a connection string. This file allows you to reuse the saved information in client applications.

Creating a UDL File

To create a UDL file, follow the steps below.

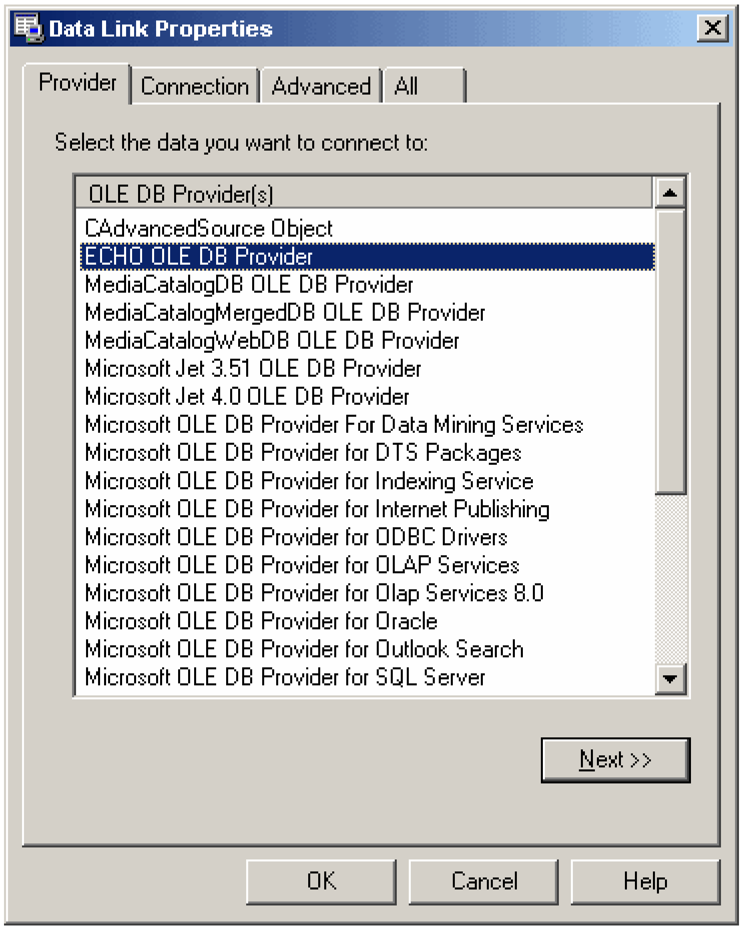
1. Open the Windows Explorer.
2. Select the folder in which the udl file is to be created.
3. Right-click in the right pane and create a new text document.

   Verify that the file extensions are visible. On the **Tools** menu, click **Folder Options**. On the View tab, deselect the **Hide file extensions for known file types** checkbox and click **OK**.
4. Rename the file and assign it the .udl extension.

NOTE: On Windows NT 4.0 operating systems, MDAC must be installed before the UDL file is created.

Editing with the Data Link Properties Dialog Box

Double-click the UDL file in the Windows Explorer, and the **Data Link Properties** dialog box appears. This tabbed dialog box, shown below, exposes all the properties that the selected OLE DB provider supports. The Provider tab allows you to select the OLE DB Provider. This selection influences the appearance of other tabs.

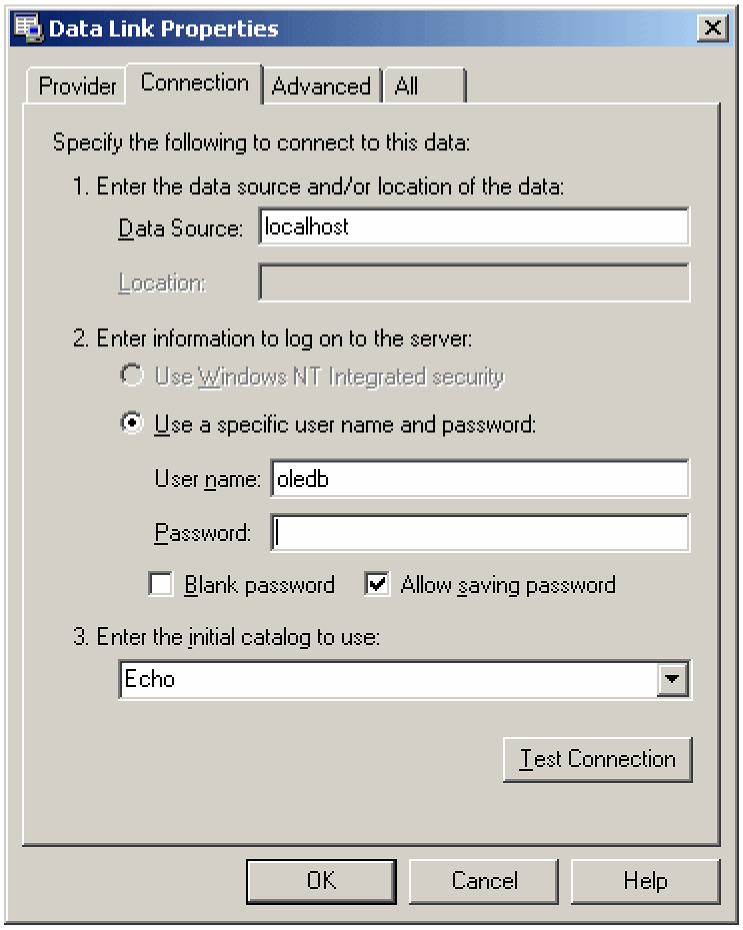


Select the data you want to connect to:

OLE DB Provider(s)

CAdvancedSource Object
ECHO OLE DB Provider
MediaCatalogDB OLE DB Provider
MediaCatalogMergedDB OLE DB Provider
MediaCatalogWebDB OLE DB Provider
Microsoft Jet 3.51 OLE DB Provider
Microsoft Jet 4.0 OLE DB Provider
Microsoft OLE DB Provider For Data Mining Services
Microsoft OLE DB Provider for DTS Packages
Microsoft OLE DB Provider for Indexing Service
Microsoft OLE DB Provider for Internet Publishing
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for OLAP Services
Microsoft OLE DB Provider for Olap Services 8.0
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for Outlook Search
Microsoft OLE DB Provider for SQL Server

In the Connection tab, enter the server name and the authentication information and select the initial catalog. The Echo catalog is the default selection.



Specify the following to connect to this data:

1. Enter the data source and/or location of the data:

Data Source: localhost

Location:

2. Enter information to log on to the server:

Use Windows NT Integrated security

[✓ Use a specific user name and password:]

User name: oledb

Password:

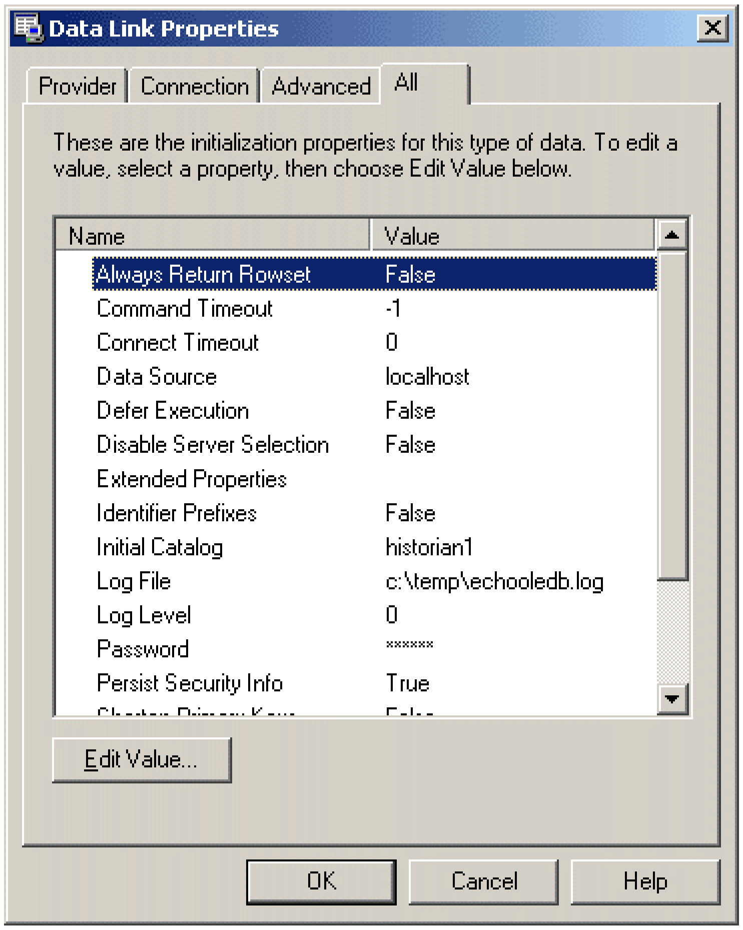
Blank password [✓ Allow saving password]

3. Enter the initial catalog to use:

Echo

The Advanced tab does not contain information supported by the ECHO OLE DB Data Provider, so this tab is not shown.

The All tab lists all the properties that can be set for the specified provider. Click **Edit Value** to change a property value.



These are the initialization properties for this type of data. To edit a value, select a property, then choose Edit Value below.

Name	Value
Always Return Rowset	False
Command Timeout	-1
Connect Timeout	0
Data Source	localhost
Defer Execution	False
Disable Server Selection	False
Extended Properties	
Identifier Prefixes	False
Initial Catalog	historian1
Log File	c:\temp\echooledb.log
Log Level	0
Password	******
Persist Security Info	True

Connection String Format

When you click **OK** to close the **Data Link Properties** dialog box, the connection string is saved in the UDL file. Its format is as follows.

```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=CHOLEDB.1; User ID=oledb;Initial Catalog=echo;Data Source=localhost; "Log
File=c:\temp\echo\echo_oledb.log;"
```

Setting Maximum Processing Time for Queries

Queries on the ECHO archive data tables can be potentially expensive in execution time; therefore, the timeout per consumer through the **DBPROP_COMMANDTIMEOUT OLE DB** property on the command object can be defined. The query is aborted after the timeout expires and a message is returned regarding this event.

NOTE: Each client sets this property in a different way. See the "Advanced Topics" chapter for more information about clients. The default value of this property is 60 seconds.

Chapter 4 Supported ANSI SQL 92 (Subset)

This chapter describes the SQL syntax that is supported by ECHO OLE DB.

SELECT Statements

Syntax

```
SELECT [ALL | DISTINCT] [TOP integer] SelectExpression [[AS] Alias][, …]
[FROM Table [[INNER] JOIN JoinedTable [[AS] JoinedTableAlias] ON JoinCondition][, …]]
[WHERE WhereCondition]
[GROUP BY GroupByColumn[, …]]
[HAVING HavingCondition]
[ORDER BY OrderByColumn [ASC | DESC][, …]]
[UNION [ALL] SelectExpression [UNION …]]
```

NOTE: Square brackets denote optional parts of the statement.

Supported Features

SELECT clause

- ALL and DISTINCT keywords.
- TOP n
- Column aliases with or without the AS keyword.
- Column aliases with an equal sign ("Alias = SelectExpression").
- Star in combination with the table ("Alias SELECT table_alias).*"

FROM clause

- Table aliases with or without the AS keyword.
- Inner joins specified by comma-separated table names.
- Inner joins specified by the INNER, JOIN and ON keywords.
- Nested queries.

GROUP BY clause

- Non-constant and non-aggregate expressions.

HAVING clause

- Condition constructed from aggregate expressions and grouping columns (grouping columns have to be specified exactly the same way as in the GROUP BY clause).

ORDER BY clause

- Non-negative integers representing the position of the sorted column in the SELECT list.
- Qualified or nonqualified column names (need not appear in the SELECT list); in case of an aggregate query, the column must be specified in the GROUP BY clause.

UNION clause

- ALL modifier.

Expressions

- Standard SQL numeric functions: abs, acos, asin, cos, exp, log, log10, sin, tan.
- Standard SQL time and date functions: day, hour, minute, month, second, year.
- Standard SQL string functions: concat (can have 2 or more arguments), lcase, left, mid, length, right, ucase.
- Aggregate functions: AVG, COUNT(arguments: * - all rows are counted; column name - rows, in which value of the column is NULL, are not counted), MAX, MIN and SUM. (all functions support the DISTINCT keyword).
- Arithmetic operators: +, -, *, /, unary + and unary -.
- String operators: +,
- Comparison operators: <, =, <=, >, <>, >=, LIKE (supports both ANSI SQL wildcard characters ("%" and "_"), IN, NOT IN, IS NULL, IS NOT NULL and BETWEEN *exp.* AND *exp*.
- LIKE operator can contain the ESCAPE 'EscapeCharacter' argument.
- Subqueries (IN, NOT IN and EXISTS operators, ANY and ALL modifiers).
- Boolean operators: AND, NOT, OR.
- Numeric constants: integer numbers, float numbers with or without exponent.
- Character constants in single quotes.
- Symbolic constants: true, false.
- Numeric or character constants represented by a SELECT statement (enclosed in parentheses).
- Identifiers can be quoted by double quotes, but this is not required.

INSERT, UPDATE, DELETE, CREATE/DROP DATABASE Statements

Syntax

```
INSERT [INTO] Table (Column[, …]) {VALUES(Value[, …]) | SelectExpression}
```

NOTE: Curly brackets denote alternative syntax.

```
UPDATE Table SET Value = Expression[, …] [WHERE WhereCondition]

DELETE [FROM] Table [WHERE WhereCondition]

CREATE DATABASE DatabaseName [ON  (PATH = 'Path' [, SIZE = Size ]) ]

DROP DATABASE DatabaseName
```

CAST Operator

Syntax

```
CAST ( expression AS data_type )
   Expression   any valid expression
   data_type ECHO engine supported data types.
```

NOTE: For list of supported data types see section "Supported Data Types" in the "Catalogs and Tables" chapter.

Chapter 5
Catalogs and Tables

This chapter describes the catalogs (databases) that ECHO OLE DB uses.

Catalog Usage

ECHO OLE DB tables are divided into several catalogs (databases). The default catalog, called ECHO, consists of tables containing the configuration data. The other catalogs represent individual historians (OLE DB Catalog = ECHO Historian) - catalog names correspond to the ECHO Historian names. Alternatively, based on the ECHO OLE DB proprietary attribute "Historians by GUID" (see the "ECHO OLE DB Specific Attributes" section in the "Configuration for Data Access" chapter), the corresponding globally unique identifier (GUID) can be used to recognize catalogs.

NOTE: Enclose the GUID with double quotation marks as shown below:

```
SELECT * FROM "B8DD90D5-E775-4882-85CB-
4647840A3DB1".data
```

All available catalogs and tables are listed in the following table.

Catalogs and Tables

Catalog Name	Table Name	Description	Comment
ECHO (tables containing ECHO configuration data).	Historians	Each historian represents a separate catalog. See the table below.	Can be updated, but insertions and deletions cannot be done.
	Metadata	Information related to ECHO node object.	Can be updated, but insertions and deletions cannot be done.
	Versions	Versions of ECHO database engine, ECHO SDK, ECHO OLE DB	Read only.
ECHO Historian Alternatively the actual historian's GUID can be used. Enclose GUIDs in double quotation marks.	Data	Data of all DataStreams. Value and ValueEx are Variants	Can be updated.
	Data2	Data of all DataStreams. Contrary to the Data table variants are broken to discrete data types.	Can be updated. Data2 table is provided for clients that cannot work with variants.
	DataFiles	Represents the ECHO SDK DataFiles collection	Rows (new DataFiles) can be inserted, but updates and deletions cannot be done.
	DataStreams	Represents the ECHO SDK DataStreams collection	Can be updated.

CREATE DATABASE / DROP DATABASE Statements

The CREATE DATABASE statement creates a new ECHO historian. As a result, the new ECHO OLE DB catalog appears and the newly created historian is listed in the Historians table.

Sample SQL Statements

```
CREATE DATABASE historian1 ON (PATH='c:\echo\historians',SIZE=16000)
```

The PATH parameter is optional and the default path is determined by the ECHO installation.

NOTE: When specified, the PATH parameter must point to an existing directory that does not contain any previously created historians.

The SIZE parameter specifies the size in KBytes of the first data file for the created historian. This (first) data file is always created (as a consequence of the CREATE DATABASE command).

The SIZE parameter is optional (default size is 32Mb).

The DROP DATABASE statement drops the existing historian and a corresponding row is removed from the Historians table, as well as from the list of ECHO OLE DB catalogs.

The pertaining data files are deleted.

```
DROP DATABASE historian1
```

or depending on the "Historians by GUID" attribute:

```
DROP DATABASE "BCCB9371-3CF3-4F5C-BBE9-CC2D0D62CEEE"
```

NOTE: The "Historians by GUID" attribute is not supported in Version 2.0 beta.

Table Structure- ECHO Catalog

NOTE: Case is preserved in table names and column names, but the implemented SQL is case insensitive.

The ECHO catalog contains the following three tables.

- Historians
- Metadata
- Versions

Historians Table Structure

Column	Data Type	Part of Primary Key	Can be Set to Null	Comments [1]
CreationTimestamp	Timestamp	no	no	Read only.
CriticalPercentFull	Int32	no	no	
CriticalReclamationPercent	Int32	no	no	
Descriptor	WString	no	no	
EnableReclamation	Bool	no	no	
ExtendedData	Variant	no	yes	
FillupStrategy	Int32	no	no	
HighPercentFull	Int32	no	no	
HighReclamationPercent	Int32	no	no	
Identity	WString	yes	no	Read only.
LastServerFlushTime	Timestamp	no	no	Read only
LogAggressiveReclamationEvents	Bool	no	no	
LogCHDatastreamDataCoercionEvents	Bool	no	no	
LogCHDatastreamFilterEvents	Bool	no	no	
LogCHDatastreamInsertInOrderEvents	Bool	no	no	
LogCHDatastreamInsertOutOfOrderEvents	Bool	no	no	
LogCHDatastreamManualFlushEvents	Bool	no	no	
LogCHDatastreamPropertyModifyEvents	Bool	no	no	
LogCHDatastreamRemoveEvents	Bool	no	no	
LogCHDatastreamReplaceEvents	Bool	no	no	
LogCreateCHDataFileEvents	Bool	no	no	
LogCreateCHDatastreamEvents	Bool	no	no	
LogDeleteCHDatastreamEvents	Bool	no	no	
LogErrorMessages	Bool	no	no	
LogGrowCHDataFileEvents	Bool	no	no	
LogInfoMessages	Bool	no	no	
LogManualFlushEvents	Bool	no	no	
LogPropertyModifyEvents	Bool	no	no	
LogWarningMessages	Bool	no	no	
MaxEntrySize	Int32	no	no	

Column	Data Type	Part of Primary Key	Can be Set to Null	Comments[1]
Name	WString	yes	yes	
ServerFlushRate	Int32	no	no	
ThrowAwayLimitPercentFull	Int32	no	no	
ThrowAwayScanRate	Int32	no	no	
ThrowAwayScanRatePercent	Int32	no	no	
1 See ECHO SDK CHHistorian object property description for more details.				

Sample SQL Statements

The following statement renames the historian.

```
UPDATE Historians SET Name = 'historian1' WHERE Descriptor = 'DB1'
```

NOTE: Renaming the Name column in the Historians table affects the catalog names.

It is not possible to insert or delete rows in this table. See the "CREATE DATABASE / DROP DATABASE Statements"section for more information on this topic.

Metadata Table Structure

Column	Data Type	Part of Primary Key	Can be Set to Null	Comments[1]
Descriptor	WString	no	no	
Identity	WString	yes	no	Read only.
IDResolutionPluginIdleTimeout	Int32	no	yes	
IDResolutionPluginIsRegistered	Bool	no	no	Read only.
IDResolutionPluginProgID	WString	no	yes	Read only.
LogCHHistorianManualFlushEvents	Bool	no	no	
LogCHHistorianPropertyModifyEvents	Bool	no	no	
LogClientConnections	Bool	no	no	
LogHistorianCreateEvents	Bool	no	no	
LogHistorianDeleteEvents	Bool	no	no	
LogPerfCounterResetEvents	Bool	no	no	
LogPropertyModifyEvents	Bool	no	no	
LoggingEnabled	Bool	no	no	
Name	WString	yes	no	Read only.
ServerResponseTimeout	Int32	no	no	
ServerTimeZone	WString	no	no	Read only.
1 See the ECHO SDK **CHNode** object property description for more details.				

Sample SQL Statements

The following statement renames the descriptor.

```
UPDATE Metadata SET Descriptor = 'This node runs ECHO version 1.3'
```

The Metadata table always contains one row, holding information related to the ECHO node.

Versions Table Structure

Column	Data Type	Part of Primary Key	Can be Set to Null	Comments
Build	WString	no	no	Item build number.
Descriptor	WString	no	no	
HasTimeout	Bool	no	no	
Item	WString	yes	no	Item name.
Location	WString	no	no	Full path to the installation directory.
Timeout	Timestamp	no	yes	
Version	WString	no	no	Version in the string form.

The Version table is read only. It contains three rows holding information about versions of ECHO OLE DB, ECHO SDK and the ECHO archive engine.

Table Structure - Historians Catalog

ECHO OLE DB considers each ECHO historian as a catalog. Therefore, there are as many catalogs as ECHO historians. Each catalog lists the following tables:

- Data
- Data2
- DataFiles
- DataStreams

Data Table Structure

Column	**Data Type**	**Part of Primary Key**	**Can be Set to Null**	**Comments** [1]
DataIncompatible	Bool	no	no	Read only.
DataStreamIdentity	WString	yes	no	Foreign Key to DataStreams table.
DataStreamName	WString	no	yes	
Index	Int32	yes	no	Index of the same time stamp succession.
IsTransient	Bool	no	no	
Timestamp	Timestamp	yes	no	
Value	Variant	no	yes	
ValueEx	Variant	no	yes	
WasInsertedOutOfOrder	Bool	no	no	Read only.
WasReplaced	Bool	no	no	Read only.
1 See ECHO SDK **CHData** object property description for more details.				

Sample SQL Statements

The following statement selects all data for one DataStream using a subquery.

```
SELECT * FROM historian1.Data WHERE DataStreamIdentity IN (SELECT Identity FROM
DataStreams WHERE Name = 'DS_I4')
```

The following statements insert one value into the ECHO archive. The data stream GUID (**DataStreamIdentity**) is taken from the DataStreams table (INSERT into SELECT construction).

```
INSERT INTO historian1.Data (DataStreamIdentity,Timestamp,Value) SELECT Identity,
date('27-Mar-2003 16:19:21.1234'),123 FROM historian1.DataStreams WHERE Name = 'DS_I4'

INSERT INTO historian1.data (DataStreamName, Timestamp, Value) VALUES ('DS_I4','16-Mar-
2004',1)
```

The following statements update a value in ECHO archive.

```
UPDATE historian1.Data SET Value = 321 WHERE DataStreamIdentity IN (SELECT Identity FROM
historian1.DataStreams WHERE Name = 'DS_I4') AND Timestamp = '27-Mar-2003 16:19:21.1234'

UPDATE data SET Value=1 WHERE DataStreamName = 'DS_I4' AND Timestamp='2004-03-16'
```

The following statements delete from the ECHO archive.

```
DELETE FROM historian1.Data WHERE DataStreamIdentity IN
(SELECT Identity FROM historian1.DataStreams WHERE Name = 'DS_I4') Timestamp =
'27-Mar-2003 16:19:21.1234'

DELETE FROM data WHERE DataStreamName = 'DS_I4' AND Timestamp='2004-03-16'
```

Data2 Table Structure

Column	Data Type	Part of Primary Key	Can be Set to Null	Comments[1]
DataIncompatible	Bool	no	no	Read only.
DataStreamIdentity	WString	yes	no	Foreign Key to DataStreams table.
DataStreamName	WString	no	yes	
Index	Int32	yes	no	Index of the same time stamp succession.
IsTransient	Bool	no	no	
Timestamp	Timestamp	yes	no	Read only.
Value_BOOL	Bool	no	yes	Variant type of Value is VT_BOOL.
Value_BSTR	WString	no	yes	Variant type of Value is VT_BSTR.
Value_CY	Int64	no	yes	Variant type of Value is VT_CY
Value_DATE	Timestamp	no	yes	Variant type of Value is VT_DATE.
Value_I1	Int8	no	yes	Variant type of Value is VT_I1.
Value_I2	Int16	no	yes	Variant type of Value is VT_I2.
Value_I4	Int32	no	yes	Variant type of Value is VT_I4.
Value_R4	Float32	no	yes	Variant type of Value is VT_R4.
Value_R8	Float64	no	yes	Variant type of Value is VT_R8.
Value_UI1	UInt8	no	yes	Variant type of Value is VT_UI1.
Value_UI2	UInt16	no	yes	Variant type of Value is VT_UI2.
Value_UI4	UInt32	no	yes	Variant type of Value is VT_UI4.
ValueEx_Bool	Bool	no	yes	Variant type of ValueEx is VT_BOOL.
ValueEx_BSTR	WString	no	yes	Variant type of ValueEx is VT_BSTR.
ValueEx_CY	Int64	no	yes	Variant type of ValueEx is VT_CY.
ValueEx_DATE	Timestamp	no	yes	Variant type of ValueEx is VT_DATE.
ValueEx_I1	Int8	no	yes	Variant type of ValueEx is VT_I1.
ValueEx_I2	Int16	no	yes	Variant type of ValueEx is VT_I2.
ValueEx_I4	Int32	no	yes	Variant type of ValueEx is VT_I4.
ValueEx_R4	Float32	no	yes	Variant type of ValueEx is VT_R4.

Column	**Data Type**	**Part of Primary Key**	**Can be Set to Null**	**Comments** [1]
ValueEx_R8	Float64	no	yes	Variant type of ValueEx is VT_R8.
ValueEx_UI1	UInt8	no	yes	Variant type of ValueEx is VT_UI1.
ValueEx_UI2	UInt16	no	yes	Variant type of ValueEx is VT_UI2.
ValueEx_UI4	UInt32	no	yes	Variant type of ValueEx is VT_UI4.
WasInsertedOutOfOrder	Bool	no	no	Read only
WasReplaced	Bool	no	no	Read only
1 See ECHO SDK **CHData** object property description for more details.				

The Data2 table contains all columns that can be found in the Data table. The variant columns (Value and ValueEx) are broken into descrete data types. In this version of ECHO OLE DB provider, only the following variant data types are supported.

- VT_BOOL
- VT_BSTR
- VT_CY
- VT_DATE
- VT_I1
- VT_I2
- VT_I4
- VT_R4
- VT_R8
- VT_UI1
- VT_UI2
- VT_UI4

Other data types supported by ECHO are transformed to the nearest equivalent of the types listed above. For example, VT_UI4 is converted to VT_I4.

The Data2 table is implemented in order to support some OLE DB clients that cannot handle variants. For example, it is not possible to update the variant columns for clients communicating with ECHO through MS SQL Server Linked Server. In this case, the Data2 table provides a solution.

DataFiles Table Structure

Column	Data Type	Part of Primary Key	Can be Set to Null	Comments [1]
ArchiveNumber	Int32	yes	no	Read only.
BlockSize	Int32	no	no	Read only.
CreationTimestamp	Timestamp	no	no	Read only.
FileSize	Iint32	no	no	
FreeBlockCount	Int32	no	no	Read only.
HistorianIdentity	WString	yes	no	Read only.
Pathname	WString	no	no	Read only.
PercentBlocksUsed	Float64	no	no	Read only.
Status	Int32	no	no	Read only.
TotalBlockCount	Int32	no	no	Read only.
UsedBlockCount	Int32	no	no	Read only.
1 See ECHO SDK **CHDataFile** object property description for more details.				

Sample SQL Statements

The following statement creates a new ECHO data file.

```
INSERT INTO historian1.DataFiles (Pathname) VALUES ('c:\echo\historians\')
```

NOTE: The value specified for **Pathname** must point to an existing directory.

DataStreams Table Structure

Column	Data Type	Part of Primary Key	Can be Set to Null	Comments[1]
CreationTimestamp	Timestamp	no	no	Read only.
DataType	WString	no	no	See the "Supported Data Types" section in this chapter for allowed keywords.
Descriptor	WString	no	no	
DiscardDuplicateTimestamps	Bool	no	no	
DiscardDuplicateValues	Bool	no	no	
EarliestOnlineDate	Timestamp	no	yes	Read only.
ExtendedData	Variant	no	yes	
FutureEditLimit	Int32	no	no	
HasData	Bool	no	no	Read only.
Identity	WString	yes	no	Read only.
LatestOnlineDate	Timestamp	no	yes	Read only.
LogDataCoercionEvents	Bool	no	no	
LogFilterEvents	Bool	no	no	
LogInsertInOrderEvents	Bool	no	no	
LogInsertOutOfOrderEvents	Bool	no	no	
LogManualFlushEvents	Bool	no	no	
LogPropertyModifyEvents	Bool	no	no	
LogRemoveEvents	Bool	no	no	
LogReplaceEvents	Bool	no	no	
Name	WString	no	yes	
PastEditLimit	Int32	no	no	
SizeClientWriteCache	Int32	no	no	
ThrowAwayLimit	Int32	no	no	
TimePrecision	Int32	no	no	
1 See ECHO SDK **CHDataStream** object property description for more details.				

Sample SQL Statements

The following statement creates a new ECHO data stream.

```
INSERT historian1.DataStreams (Name,DataType) VALUES ('DS_I4','VT_I4')
```

Use these keywords to specify data stream data type when inserting into the DataStreams table.

- VT_ARRAY
- VT_BOOL
- VT_BSTR
- VT_CY
- VT_DATE
- VT_EMPTY
- VT_I1
- VT_I2
- VT_I4
- VT_INT
- VT_R4
- VT_R8
- VT_UI1
- VT_UI2
- VT_UI4
- VT_UINT

The following statement renames a data stream.

```
UPDATE historian1.DataStreams SET Name = 'DS_I4_OLD' WHERE name = 'DS_I4'
```

The following statement deletes a data stream.

```
DELETE FROM historian1.DataStreams WHERE Name = 'DS_I4_OLD'
```

Supported Data Types

ECHO OLE DB internally maps supported data types to the VT_* (variant) used in the ECHO-SDK as listed in the table below.

Data Type Mappings

This type maps to ...	… this keyword
Bool	VT_BOOL
Float32	VT_R4
Float64	VT_R8
Int8	VT_I1
Int16	VT_I2
Int32	VT_I4
Int64	VT_CY
Timestamp	VT_DATE
UInt8	VT_UI1
UInt16	VT_UI2
UInt64	VT_UI4
WString	VT_BSTR

You can use all the data types listed in the table above in the CAST operator.

Examples

```
SELECT CAST(Value AS WString) FROM historian1.Data WHERE Timestamp > '01-Jan-2004'
SELECT CAST(Value AS VT_BSTR) FROM historian1.Data WHERE Timestamp > '01-Jan-2004'
..
```

Supported Time Stamp Formats

VT_UI4 – (UInt32)

The following absolute formats are supported:

```
"yyyy-mm-dd hh:mm:ss.fr"
"dd-mmm-yyyy hh:mm:ss.fr"
```

Formats supported in Windows Regional Options.

The provider implements a 'shortcut' to current time '*' star. Relative times can be expressed as follows: '*-30s, '*-1m', '*-12h', '*-10d'.

You can use the built-in *Date()* function to explicitly force the conversion to the Timestamp data type.

Examples

```
SELECT Date('*')
SELECT Date('*')-Time('30s')
SELECT * FROM historian1.Data WHERE Timestamp > '*-30d'
SELECT Timestamp-Time('1m') FROM historian1.Data WHERE Timestamp > '2004-01-01'
```

Examples of Other Statement Use

Simple Expressions

```
SELECT 1

SELECT 1 WHERE 1 = 1 ORDER BY 1

SELECT -1+1*1-1/(1+1)

SELECT abs(-1) abs_minus_one, acos(1) acos_one, asin(1) asin_one, cos(3.14)
cos_ch, exp(1) exp_one, log(1) log_one, log10(1) log10_one, sin(3.14) sin_ch,
tan(3.14) tan_ch

SELECT concat('OLE DB Provider', ' ', 'for', ' ', 'ECHO'), 'OLE DB Provider' + ' ' +
'for' + ' ' + 'ECHO'

SELECT lcase('OLE DB Provider'), left('OLE DB Provider', 5), length('OLE DB
Provider'), mid('OLE DB Provider', 5, 1), right('OLE DB Provider', 5), ucase('OLE
DB Provider')

SELECT lcase(left('OLE DB Provider', length('OLE DB Provider') - 5))

SELECT string(1), string(1.0)

SELECT AVG(1) avg_one, COUNT(*) _count, MAX(1) max_one, MIN(1)
min_one, SUM(1) sum_one HAVING COUNT(*) = 1

SELECT 1, 1., 1.1, 1e1, 1.e1, 1.1e1, 1e-1, 1.e-1, 1.1e-1

SELECT acos(2), asin(2), log(-1), log10(-1)
```

Simple Table Queries

```
SELECT * FROM Versions

SELECT 1 FROM historian1.DataStreams

SELECT * FROM historian1.Data WHERE
Timestamp = '2003-01-27 15:43:01.1234567'

SELECT * FROM "B8DD90D5-E775-4882-85CB-4647840A3DB1".DataStreams

SELECT  COUNT(*) FROM historian1..Data WHERE DataStreamName =
'DS_I4'

SELECT  MIN(Timestamp) FROM historian1..Data WHERE DataStreamName =
'DS_I4' AND Value > 90

SELECT * FROM historian1..Data WHERE DataStreamName = 'DS_I4' AND
Timestamp > '*-1d'

SELECT * FROM historian1..Data WHERE DataStreamName = 'DS_I4' AND
Timestamp BETWEEN  '*-1d' AND '*-23h'
```

NULL Values

```
SELECT * FROM historian1.DataStreams WHERE
EarliestOnlineDate IS NOT NULL
```

```
SELECT * FROM Historians WHERE Name IS NOT NULL
```

DISTINCT

```
SELECT DISTINCT * FROM historian1.Data WHERE
DataStreamName = 'DS_I4'
```

```
SELECT COUNT(DISTINCT Value) FROM historian1.Data
```

TOP n

```
SELECT TOP 1 * FROM historian1.Data
```

```
SELECT TOP 10 Timestamp,Value FROM historian1.Data  WHERE
DataStreamName =  'DS_I4'  ORDER BY DataStreamName, Timestamp DESC
```

GROUP BY, HAVING

```
SELECT DataStreamName, AVG(Value_R4) average, MAX(Value_R4)
maximum, COUNT(Value_R4) [count] FROM  historian1.Data2 WHERE
DataStreamName =  'DS_I4' AND Timestamp > '01-Jan-2004' GROUP BY
DataStreamName
```

```
SELECT DataStreamName, MAX(Value_R4) FROM  historian1.Data2 WHERE
DataStreamName =  'DS_I4' AND Timestamp > '01-Jan-2004' GROUP BY
DataStreamName HAVING COUNT(Value_R4) > 1
```

ORDER BY

```
SELECT * FROM historian1.Data WHERE DataStreamName =  'DS_I4' AND
Timestamp > '01-Jan-2004' ORDER BY Value DESC, ValueEx
```

Inner Joins

```
SELECT ds.Name, dat.Timestamp, dat.Value FROM
historian1.Data dat, historian1.DataStreams ds
WHERE ds.Identity = dat.DataStreamIdentity AND ds.Name = 'DS_I4' AND
Timestamp > '01-Jan-2004'
```

```
SELECT ds.Name, ds.Descriptor, dat.* FROM historian1.DataStreams ds
JOIN  historian1.Data dat ON ds.Identity = dat.DataStreamIdentity  WHERE
Timestamp > '01-Jan-2004'
```

Nested Queries

```
SELECT c1 FROM (SELECT 1 c1) t1
```

Subqueries

```
SELECT * FROM historian1.Data WHERE DataStreamIdentity IN (SELECT
Identity FROM historian1.DataStreams WHERE  Name = 'DS_I4') AND
Timestamp > '01-Jan-2004'
```

UNION

```
SELECT 1 UNION SELECT 1
```

```
SELECT 1 UNION ALL SELECT 1
```

```
SELECT * FROM historian1.DataStreams UNION ALL
SELECT * FROM historian2.DataStreams ORDER BY earliestonlinedate
```

INSERT INTO SELECT

```
INSERT INTO historian1.Data (DataStreamName,Timestamp,Value)
SELECT 'DS2_I4', Timestamp,Value
FROM historian1.Data
WHERE
DataStreamName = 'DS_I4'
```

NOTE: The INSERT INTO … SELECT … construction allows you to copy data from one data stream to another.

Chapter 6
ActiveX Data Objects - ADO

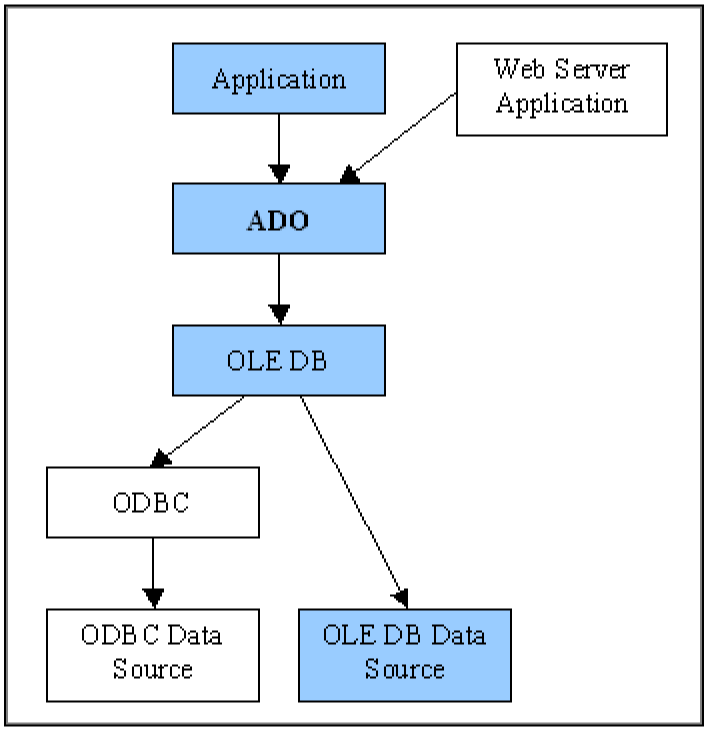
This chapter describes the use of the ActiveX Data Objects with the ECHO OLE DB Data Provider.

Description of the ActiveX Data Object

The ADO (ActiveX Data Objects) is a set of high-level Automation interfaces on top of OLE DB. It simplifies the access to OLE DB data by eliminating the low-level operations, such as managing memory resources or component aggregation. ADO provides the following advantages.

- Ease of use – The data-access task analogous to "Hello World" requires only three lines of code.
- Programming language neutrality – The ADO can be used with languages such as Visual Basic, Java, C++, VBScript, or JScript.
- Provider neutrality – The ADO can access data from any OLE DB source. In addition, ADO adapts seamlessly to support less functional OLE DB providers.
- No loss of OLE DB functionality – The ADO allows C++ developers access to the underlying OLE DB interfaces.

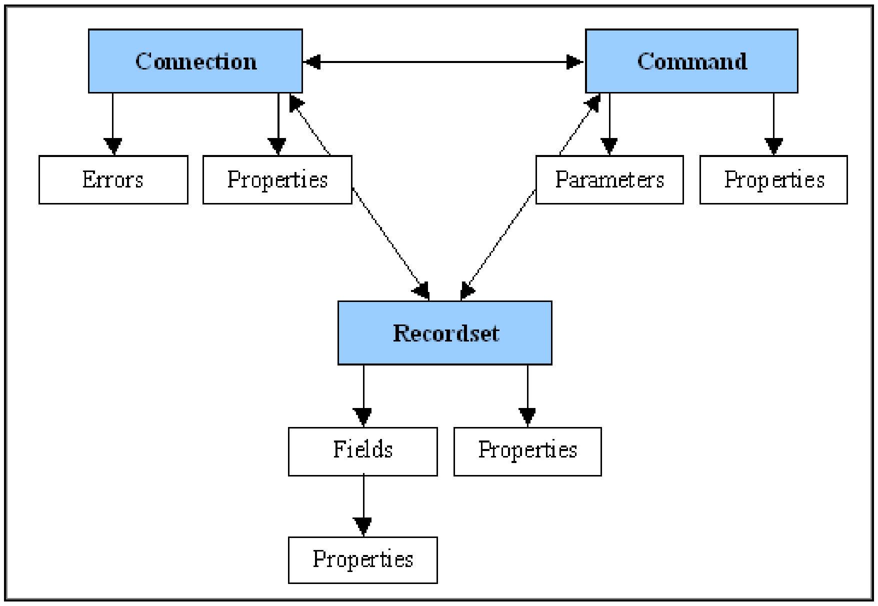
The following figure shows the typical configuration of applications that used ADO.



NOTE: All ADO samples provided here use the shaded boxes "Application - ADO - OLE DB (ECHO OLE DB) - OLE DB Data Source (ECHO Historian)"

ADO Object Model

ADO is made up of seven objects as shown in the following figure.



The Connection, Command, and Recordset objects are top-level objects that you create and destroy independently of each of the other objects. Although you create the Parameter object independently of a Command object, it must be associated with a command before it can be used. The Field, Error, and Property objects exist only within the context of their parent objects, and cannot be created separately.

The ADO Connection object represents a connection to the source of data. It defines properties of the connection, assigns the scope of local transactions, and provides a central location for retrieving errors.

The ADO Command object represents the data-definition or data-manipulation statement to be executed. In the case of a relational provider, this is an SQL statement. The Command object allows you to specify parameters and customize the behavior of the statement to be executed. A collection of Parameter objects exposes the parameters.

The ADO Recordset object is the actual interface to the data, whether it is the result of a query or was generated in some other fashion. The Recordset object provides control over the locking mechanism, the type of cursor, the number of rows to access at a time, etc.

The Recordset object exposes a collection of Field objects that contain the metadata about the columns in the recordset, such as name, type, length, and precision, as well as the actual data values themselves. The Recordset object can be used to navigate through records and change data (assuming the underlying provider is updatable).

ADO Databinding ActiveX Controls

ADO Databinding ActiveX Controls encapsulate the ADO functionality and allow the displaying and changing of the underlying data.

There are two types of Databinding ActiveX controls.

- Data-Source Controls – A data-source control encapsulates a database query and the retrieved rowset. The Microsoft ADO Data Control provides a user interface consisting of a series of buttons to iterate through the data.
- Data-Bound Controls – A data-bound control presents the data. Data-bound controls connect to data-source controls to receive data and present the data through a variety of user interfaces. An example of the data-bound control is the Microsoft DataGrid Control.

NOTE: More information about ADO is available on the Microsoft's Universal Data Access web site at http://www.microsoft.com/data/ado. Although you can store data in a variety of formats, these formats fall into two broad categories.

ADO and DBTIMESTAMP

ECHO stores time stamps with up to 100-nanosecond precision. OLE DB uses the DBTIMESTAMP structure, which allows up to 1-nanosecond time precision. Because ADO maps the DBTIMESTAMP to a DATE data type that does NOT implement the subsecond part; seconds are rounded. Most of the data-bound controls therefore show time stamps with only seconds, even if the underlying DBTIMESTAMP contains the subsecond part. As of publication, OSIsoft is not aware of any grid that displays a precision greater than seconds.

Samples Provided

All samples are written in Visual Basic (VB). For the result presentation, samples use Microsoft Databinding ActiveX Controls (Microsoft ADO Data Control and Microsoft DataGrid Control).

- ADO_Intro – This sample introduces the basic principles of ADO. For the resulting data set presentation, the Microsoft DataGrid Control is used.

NOTE: To be opened in design mode, samples require the licensed version of the Microsoft DataGrid Control. This control is installed (with the full license) with Microsoft Visual Basic or Microsoft Office Developer Tools.

- ADO.NET Intro – This simple example describes how to use ECHO OLE DB provider in a "managed environment."
- ASP - Sample that shows a simple ADO query called from an ASP page (IIS Web Server required).

ADO Intro Sample

This sample demonstrates the basic principles of ADO. The sample displays the results of this query in the Microsoft DataGrid Control.

```
SELECT CreationTimestamp,Name,Descriptor FROM historians
```

To create the sample, follow the steps below.

1. Run Visual Basic and create a new "Standard EXE" project.
2. Select the **Project - References** menu item and check the **Microsoft ActiveX Data Objects 2.7 Library** in the **References** dialog box. This loads the ADO type library into the Visual Basic environment.
3. Select the **Project - Components** menu item and check the **Microsoft DataGrid Control 6.0** in the **Components** dialog box. This loads the DataGrid Control type library into the Visual Basic environment.
4. Click the **DataGrid Control** icon in the toolbox and place it on the main form. Change its name to "DataGrid."
5. Add the **Form_Load** event handler.
6. Enter the following code.

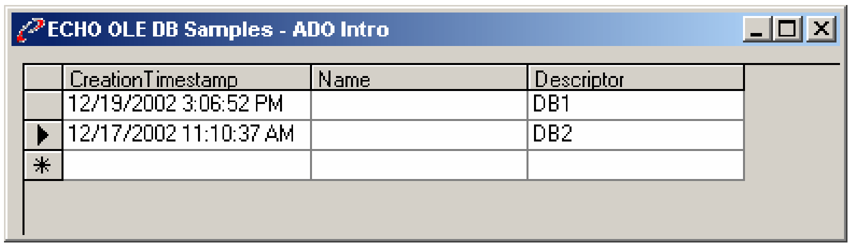
```
Dim adoCn As New ADODB.Connection
With adoCn
 Provider = "CHOLEDB"
 Properties("Prompt") = adPromptComplete
   Open
End With
Dim adoRs As New ADODB.Recordset
adoRs.Open "SELECT CreationTimestamp,Name,Descriptor
from historians ", adoCn, adOpenStatic,
adLockOptimistic, adCmdText

Set DataGrid.DataSource = adoRs
```

```
DataGrid.AllowUpdate = True
DataGrid.AllowAddNew = False
DataGrid.AllowDelete = False
```

7. Save the project and its form as "ADO Intro.frm" and "ADO Intro.vbp."
8. Run the sample.

The following figure illustrates how the running sample appears.



ECHO OLE DB Samples - ADO Intro

	CreationTimestamp	Name	Descriptor
	12/19/2002 3:06:52 PM		DB1
▶	12/17/2002 11:10:37 AM		DB2
*			

The code that you enter does the following:

1. Create a new ADO Connection object.
   `Dim adoCn As New ADODB.Connection`
2. Open a connection to the ECHO Server through the ECHO OLE DB Data Provider.
   `adoCn.Open "Provider=CHOLEDB;"`
3. Create a new ADO Recordset object.
   `Dim adoRs As New ADODB.Recordset`
4. Execute the query and save the data in the created Recordset object. Arguments of the adoRs.Open method are set to the most common values.
   ```
   adoRs.Open "SELECT CreationTimestamp,Name,Descriptor
   FROM orians ", adoCn, adOpenStatic,
   adLockOptimistic, adCmdText)
   ```
5. Assign the Recordset object to the DataGrid Control.
   `Set DataGrid.DataSource = adoRs`
6. Sets the updatability flags of the DataGrid Control. The DataGrid Control allows point attribute changes.

The connection string in ADO (first argument of the ADO Connection "Open" method) is compatible with the UDL file. You can even specify the connection string to use attributes from the UDL file, for example, "FILE NAME= Test.udl;"

Note that the ADO Command object is not used. This approach is possible when the command specific properties or methods are not needed.

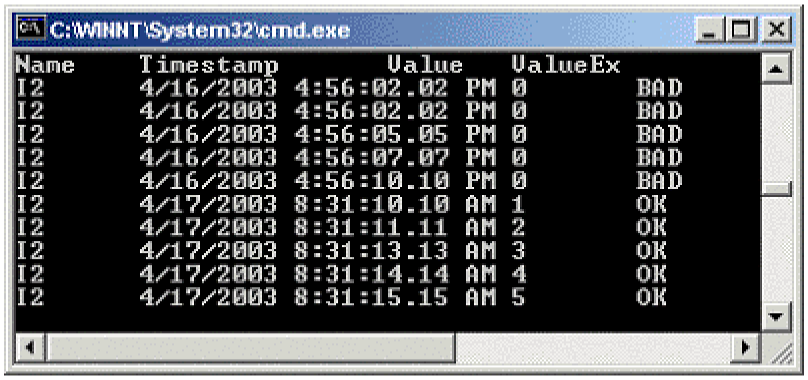
ADO.NET Intro Sample

This sample demonstrates how to use ECHO OLE DB provider in the a "managed environment." To achieve this, the managed provider for OLE DB providers from the .NET Framework must be used. The managed provider actually represents a way from the .NET application to "OLE DB powered datasources".

ADO.NET Intro sample is created using MS Visual Basic.NET and the sample code does the following.

1. Import the OLE DB layer. The OLE DB .NET Data Provider classes are located in the **System.Data.OleDb** namespace.
```
Imports System.Data.OleDb
```
2. Open a connection to the ECHO Server through the ECHO OLE DB Data (via the managed provider for OLE DB providers).
```
Dim cn As New OleDbConnection()
cn.ConnectionString = "Provider = CHOLEDB; Data
Source = localhost; Log File=c:\temp\echo_net.log;"
cn.Open()
```
3. Create the ADO.NET command object and assign it the connection.
```
Dim cmd As New OleDbCommand()
cmd.CommandText =
"SELECT ds.Name,dat.Timestamp,dat.Value,dat.ValueEx
FROM historian1.Data dat,historian1.DataStreams ds
WHERE dat.DataStreamIdentity = ds.Identity AND
ds.name = 'Your_DS Name' "
```
4. Execute the query. The result set is returned in the form of a data reader object.
```
Dim dataReader As OleDbDataReader =
cmd.ExecuteReader()
```
5. Loop the data reader object extracting the individual records from the result set and print them onto the console window.

The following figure shows the result.



C:\WINNT\System32\cmd.exe

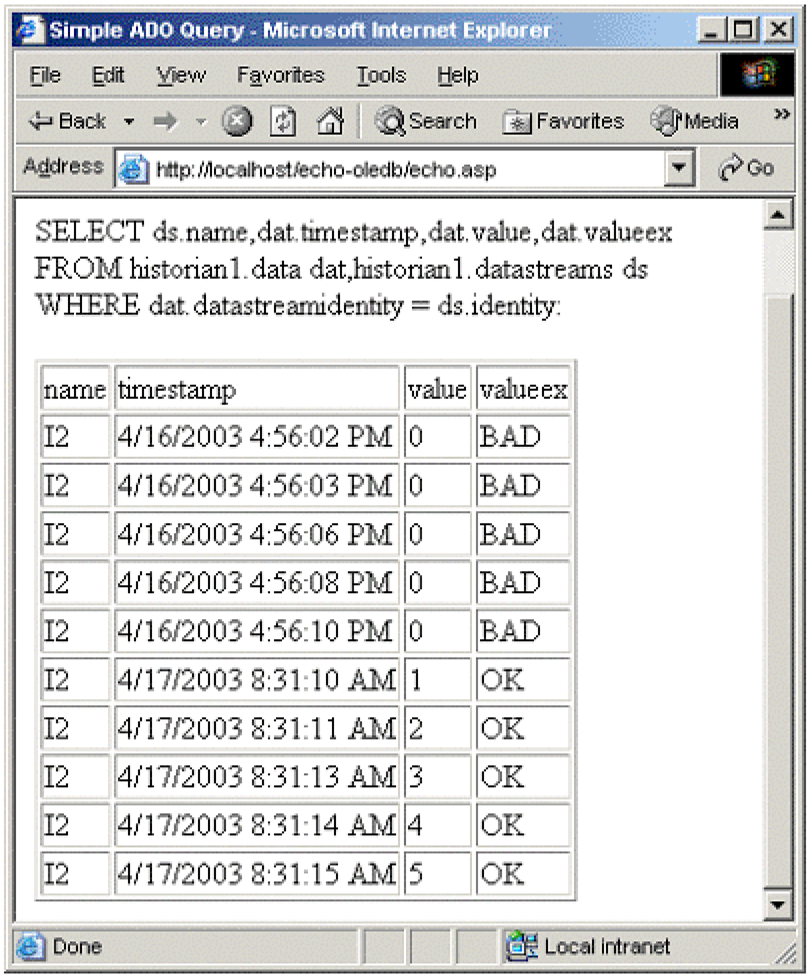
```
Name      Timestamp            Value    ValueEx
I2        4/16/2003 4:56:02.02 PM 0         BAD
I2        4/16/2003 4:56:02.02 PM 0         BAD
I2        4/16/2003 4:56:05.05 PM 0         BAD
I2        4/16/2003 4:56:07.07 PM 0         BAD
I2        4/16/2003 4:56:10.10 PM 0         BAD
I2        4/17/2003 8:31:10.10 AM 1         OK
I2        4/17/2003 8:31:11.11 AM 2         OK
I2        4/17/2003 8:31:13.13 AM 3         OK
I2        4/17/2003 8:31:14.14 AM 4         OK
I2        4/17/2003 8:31:15.15 AM 5         OK
```

ASP Sample

This sample shows how to instantiate the ECHO OLE DB provider within an IIS (Internet Information Server) using VBScript language. Data from ECHO is stored within an ADO recordset (described in the ADO Intro example). Rows from the ADO recordset are finally displayed in a table within the browser.

To create the sample, follow the steps below.

1. Run the Internet Services Manager and create the new Virtual Directory and name it ECHO-OLEDB.
2. Copy the echo.asp file into a real directory, to which the Virtual Directory (specified in step 1) points.
3. Run the Internet Explorer and point to the echo.asp file using the http protocol.
   `http://localhost/echo-oledb/echo.asp`


Simple ADO Query - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/echo-oledb/echo.asp
Go
Done
Local intranet

SELECT ds.name,dat.timestamp,dat.value,dat.valueex FROM historian1.data dat,historian1.datastreams ds WHERE dat.datastreamidentity = ds.identity;

name	timestamp	value	valueex
I2	4/16/2003 4:56:02 PM	0	BAD
I2	4/16/2003 4:56:03 PM	0	BAD
I2	4/16/2003 4:56:06 PM	0	BAD
I2	4/16/2003 4:56:08 PM	0	BAD
I2	4/16/2003 4:56:10 PM	0	BAD
I2	4/17/2003 8:31:10 AM	1	OK
I2	4/17/2003 8:31:11 AM	2	OK
I2	4/17/2003 8:31:13 AM	3	OK
I2	4/17/2003 8:31:14 AM	4	OK
I2	4/17/2003 8:31:15 AM	5	OK

MyECHOSQL Sample

This sample, written in C++, executes a batch of SQL queries. Queries can be written in a text file, which is then pointed in start-up parameters of the MyECHOSQL.exe. The script_sample.sql creates a new historian, adds several data streams, writes some data, and then cleans everything up. Results are stored in script_output.txt.

NOTE: Create the historian in an existing directory that is empty, adjusted according to your environment.

```
CREATE DATABASE hist1 ON
(PATH='c:\echo\historians',SIZE=16000).
```

UDL Sample

This sample is an example of a UDL (Universal Data Link) file that can be used for establishing a connection to the ECHO Archive Engine.

Chapter 7 OLE DB Server Environments

This chapter describes server access to OLE DB data sources. The linked server configuration allows SQL Server to execute commands against OLE DB data sources on different servers.

Microsoft SQL Server Linked Server

Linked Server is a term for a virtual RDBMS server that is linked into Microsoft SQL Server. The linked server configuration allows the Microsoft SQL Server to execute commands against OLE DB data sources on different servers.

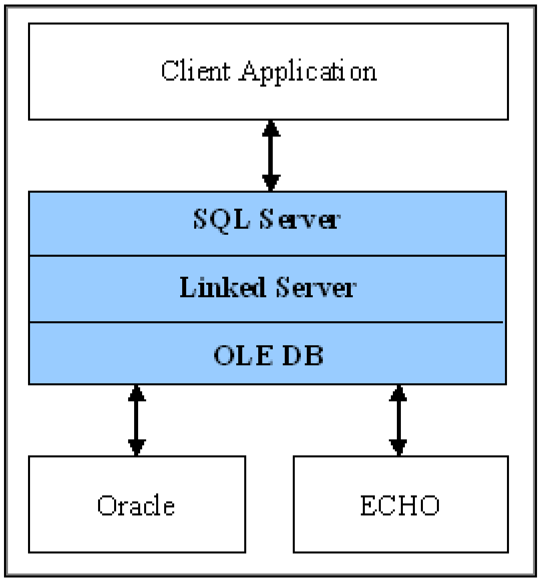
Linked servers offer these advantages.

- Remote server access.
- Ability to issue distributed queries, updates, commands, and transactions on heterogeneous data sources across the enterprise.
- Ability to address diverse data sources similarly.

The Linked Servers functionality is available in Microsoft SQL Server 7.0 and greater.

NOTE: ECHO OLE DB requires Microsoft SQL Server 2000 to operate as a Linked Server.

The following figure illustrates the linked server architecture of the Microsoft SQL Server.



Distributed queries are queries, that access data stored in SQL Server (homogeneous data) plus data stored in a data store other than SQL Server (heterogeneous data from SQL Server point of view). Distributed queries operate as if all data is stored in SQL Server. Each distributed query can reference multiple linked servers and can perform either update or read operations against each individual linked server.

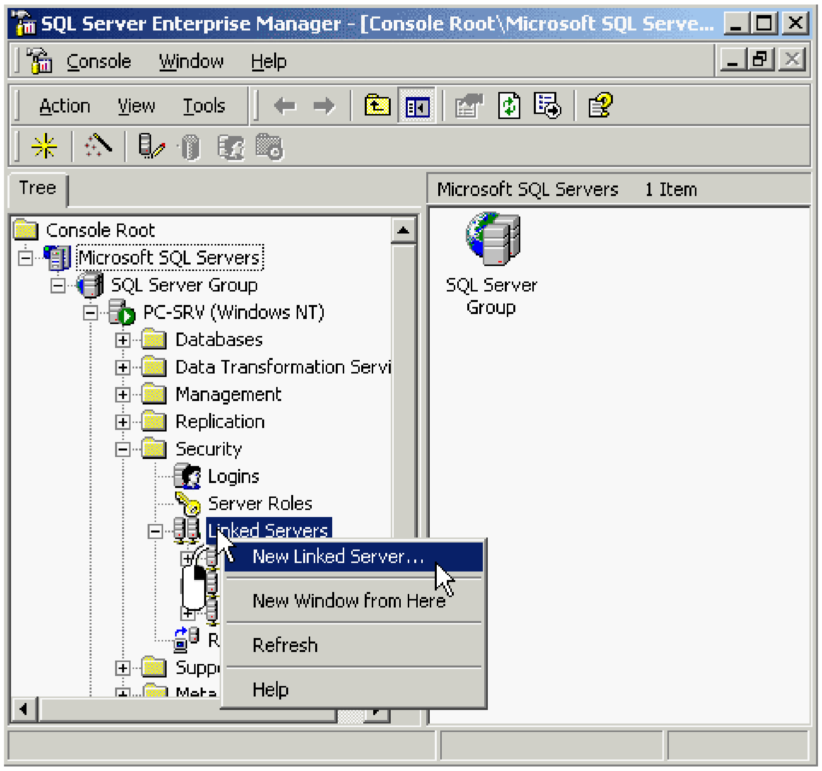
Configuration Through the Enterprise Manager

To define a linked server, specify an OLE DB provider and its data source. The OLE DB provider DLL must be present on the same server as SQL Server. Do this either through the SQL Server Enterprise Manager or through stored procedures. For more information on configuration by stored procedures, refer to the SQL Server documentation.

To configure a new linked server with the Enterprise Manager, follow the steps below.

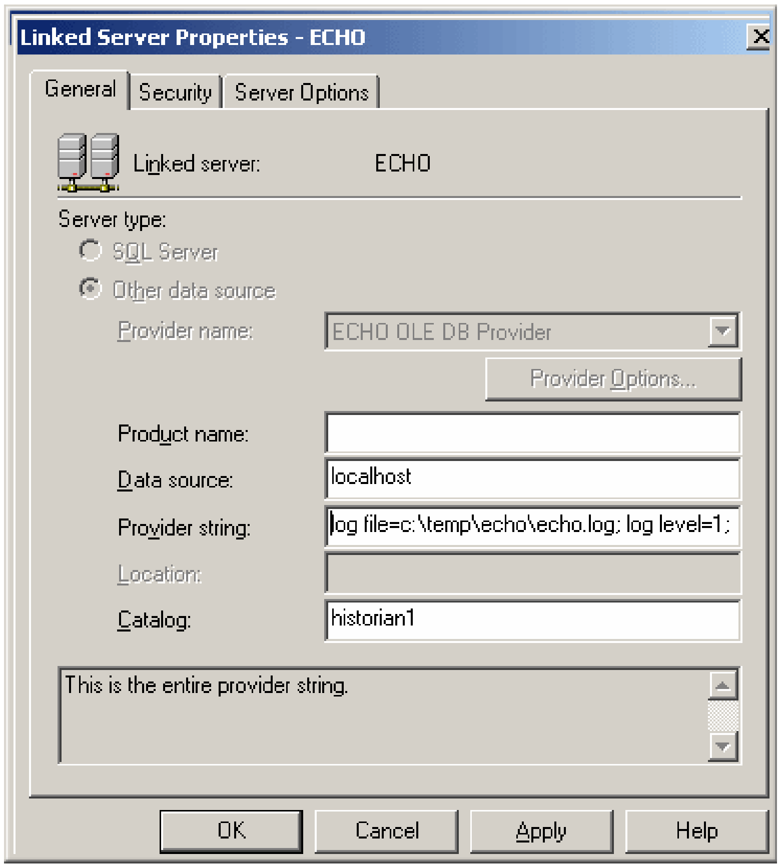
Open the Enterprise Manager

1. Run the Enterprise Manager.
2. On the tree under the Console Root, locate **Linked Servers under Microsoft SQL Servers - SQL Server Group –** [Server Name] **- Security**.
3. Right-click **Linked Servers** and select **New Linked Server**, as shown in the figure below.



4. In the **Linked Server Properties** dialog box that appears, select the General tab.

The fields of the General tab are shown in the following figure.



Linked server: ECHO

Server type:
SQL Server
Other data source

Provider name: ECHO OLE DB Provider

Product name:

Data source: localhost

Provider string: log file=c:\temp\echo\echo.log; log level=1;

Location:

Catalog: historian1

This is the entire provider string.

5. On the General tab, enter the information into the fields, described in the following table.

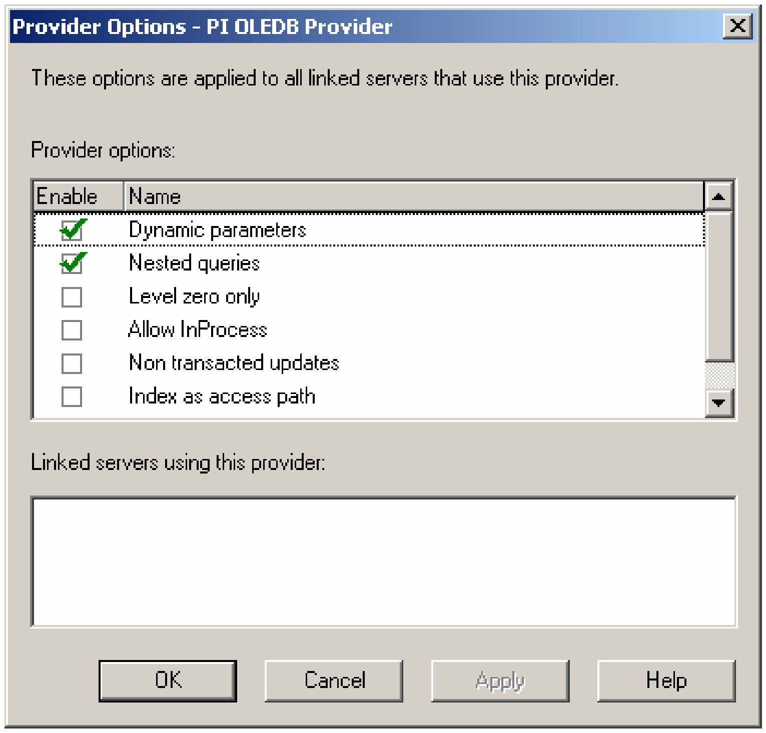
Linked Server Properties – ECHO Dialog Box – General Tab

Field	**Description**	**Typical Value**	**Remarks**
Linked Server	Name of the linked server being created.	ECHO.	Required.
Provider Name	Name of the OLE DB data provider.	ECHO OLE DB Provider.	Required.
Product Name	-	-	Leave this field blank.
Data Source	Name of the ECHO server.	Enter the appropriate server name for your application, localhost for example.	Required.
Provider String	Corresponds to the UDL Extended Properties attribute. For more, see the "ECHO OLE DB Specific Attributes" section in the "Configuration for Data Access" chapter.	-	Optional
Catalog	The default catalog.	Echo	Optional

Specify Provider Options

Continue the configuration by specifying the provider options.

1. Click **Provider Options**. The **Provider Options – PI OLEDB Provider** dialog box appears.


Provider Options - PI OLEDB Provider
These options are applied to all linked servers that use this provider.
Provider options:
Enable
Name
Linked servers using this provider:
OK
Cancel
Apply
Help

Enable	Name
✓	Dynamic parameters
✓	Nested queries
☐	Level zero only
☐	Allow InProcess
☐	Non transacted updates
☐	Index as access path

2. Enter the information into the fields, described in the following table.
3. Then click **OK** to close the dialog box and return to the General tab.

Provider Options – PI OLEDB Provider Dialog Box

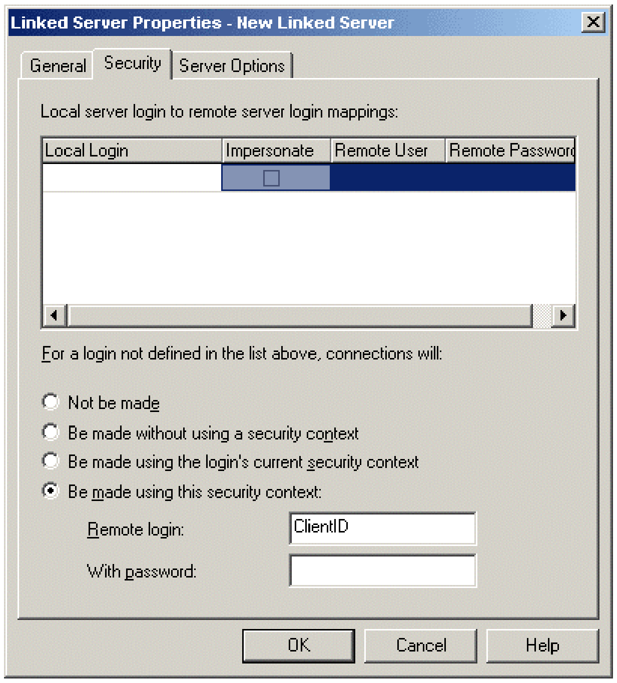
Field	**Description**	**Supported by ECHO OLE DB**
Dynamic Parameters	Indicates that the provider allows the "?" parameter marker syntax for parameterized queries.	Yes
Nested Queries	If nonzero, indicates that the provider allows nested SELECT statements in the FROM clause.	Yes
Level Zero Only	Base level OLE DB interfaces are invoked against the provider.	Do not select this option; multiple base level interfaces are supported.

Field	Description	Supported by ECHO OLE DB
Allow InProcess	SQL Server allows the provider to be instantiated as an in-process server. When this option is not set, the default behavior is to instantiate the provider outside the SQL Server process. Instantiating the provider outside the SQL Server process protects the SQL Server from errors in the provider and potentially from crashing.	ECHO OLE DB supports both in-process and out-of-process configuration. In-process configuration runs faster because the provider is instantiated within the SQL Server process itself. Out-of-process configuration is safer because the dllhost.exe surrogate instantiantes it and any possible provider problem does not affect the SQL Server core process.
Non-Transacted Updates	If nonzero, SQL Server allows updates, even if the ITransactionLocal interface is not available.	Do not select this option; the ITransactionLocal interface is supported
Index as Access Path	If nonzero, SQL Server attempts to use indexes of the provider to fetch data.	No
Disallow ad hoc Access	If a nonzero value is set, SQL Server does not allow ad hoc access through the OPENDATASOURCE and OPENROWSET functions against the OLE DB provider.	Can be selected.

Specify Security

Continue the configuration by specifying the security options.

1. Click the Security tab. The fields of this tab are shown in the following figure.


Linked Server Properties - New Linked Server

General | Security | Server Options

Local server login to remote server login mappings:

Local Login	Impersonate	Remote User	Remote Passwor
	☐		

For a login not defined in the list above, connections will:

- Not be made
- Be made without using a security context
- Be made using the login's current security context
- [✓ Be made using this security context:]

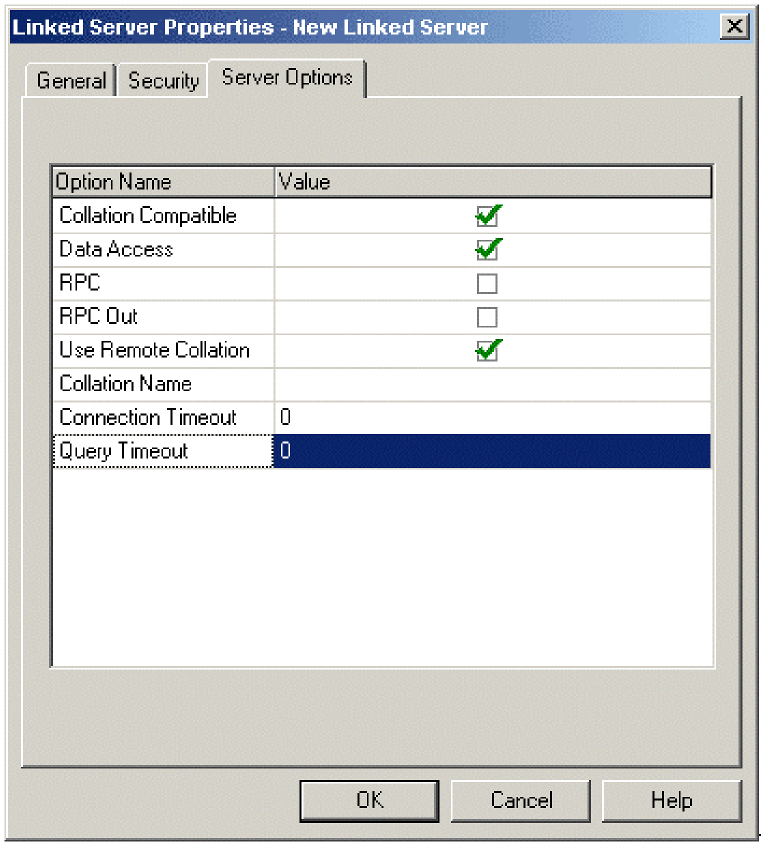
Remote login: ClientID

With password:

2. Enter the information appropriate for your application.

   You can specify authentication information separately for each SQL Server user or for all users. The figure above shows the second approach.

3. Click the Server Options tab. Enter the information appropriate for your application.



Linked Server Properties - New Linked Server

Option Name	Value
Collation Compatible	✓
Data Access	✓
RPC	☐
RPC Out	☐
Use Remote Collation	✓
Collation Name	
Connection Timeout	0
Query Timeout	0

Of the options in this tab, the **Collation Compatible** and **Query Timeout** are the most important ones for ECHO OLE DB.

Select the **Collation Compatible** option. This causes SQL Server to assume all columns and character sets are compatible with the local server character set and collation, and enables SQL Server to send comparisons on character columns to the OLE DB provider. Otherwise, SQL Server always evaluates comparisons on character columns locally.

The **Query Timeout** option corresponds to the **DBPROP_COMMANDTIMEOUT OLE DB** property. For details, see the "Maximum Processing Time for Queries" section in the "Configuration for Data Access" chapter.

You can set the other options to their default values.

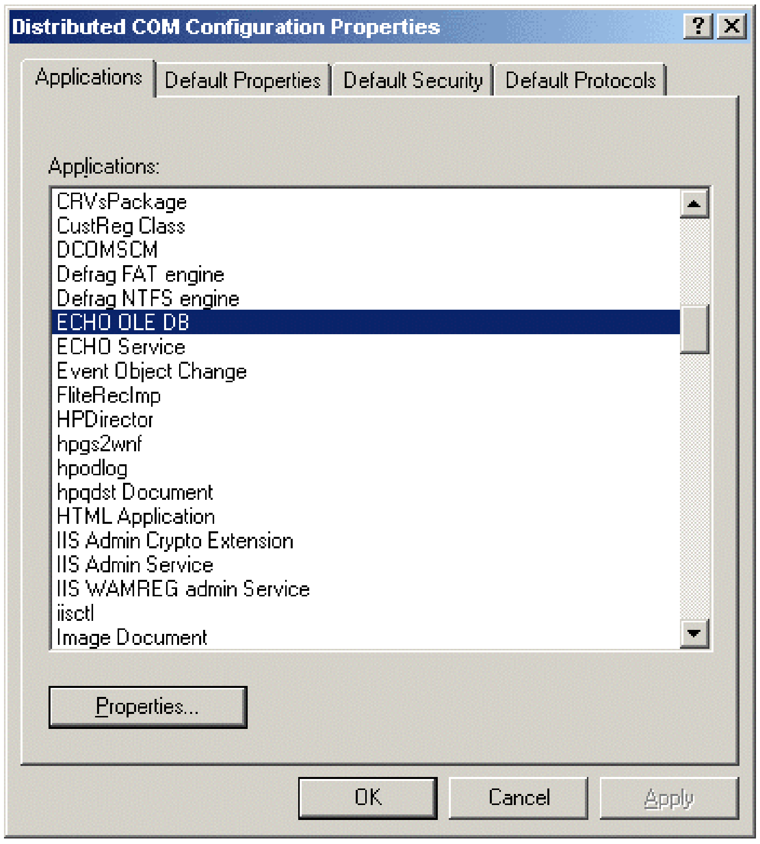
4. Click **OK** to close the **Linked Server Properties** dialog box.

User Account and DCOM Settings for OutProc Instantiation

The process of running a DLL-based COM object outside the address space of the main application is called remoting. Remoting requires that another executable be a surrogate process in place of the SQL Server executable. The default executable used by the DCOM Service Control Manager is named Dllhost.exe. The ECHO OLE DB provider is instantiated within the address space of the dllhost process when running as an OutProc. This generates issues regarding security and user account permissions. When MS SQL Server is connected using the 'SQL Server authentication' mode, the Dllhost.exe starts in the same user account as the SQL Server executable (default – LocalSystem). When connected using the 'Windows authentication' the actual Windows user must have the appropriate access and launch permissions. In the case of 'SQL Server Authentication' the permissions for the 'System' user must be addressed.

To change access and launch permissions, follow the steps below.

1. Click **Start** on the Windows **Taskbar**, then click **Run**.
2. Type `dcomcnfg` in the **Run** dialog box and click **OK**. The **Distributed COM Configuration Properties** dialog box appears as shown below.



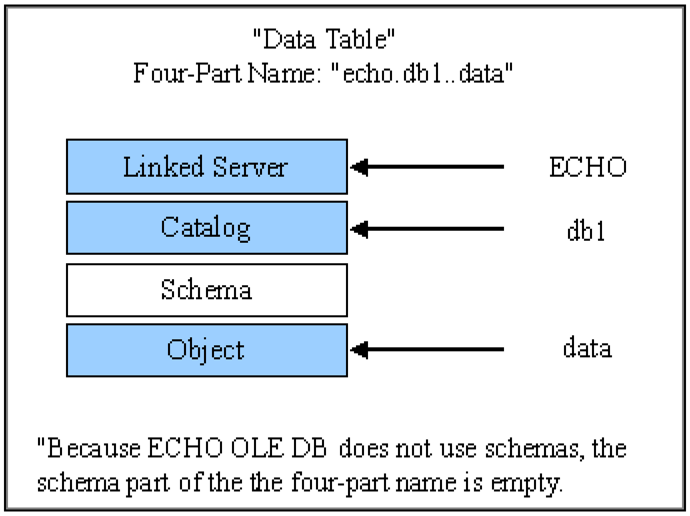
CRVsPackage
CustReg Class
DCOMSCM
Defrag FAT engine
Defrag NTFS engine
ECHO OLE DB
ECHO Service
Event Object Change
FliteRecImp
HPDirector
hpgs2wnf
hpodlog
hpqdst Document
HTML Application
IIS Admin Crypto Extension
IIS Admin Service
IIS WAMREG admin Service
iisctl
Image Document

3. On the Applications tab select ECHO OLE DB and click **Properties**. The **Properties** dialog box appears.

4. On the Security tab, select **Use Custom Access Permissions**. Click **Edit** and assign access to all users who are allowed access to the archive.
5. Select **Use Custom Launch Permissions**. Click **Edit** and assign launch permissions to all users who are allowed to access the archive.
6. Close the dialog boxes.

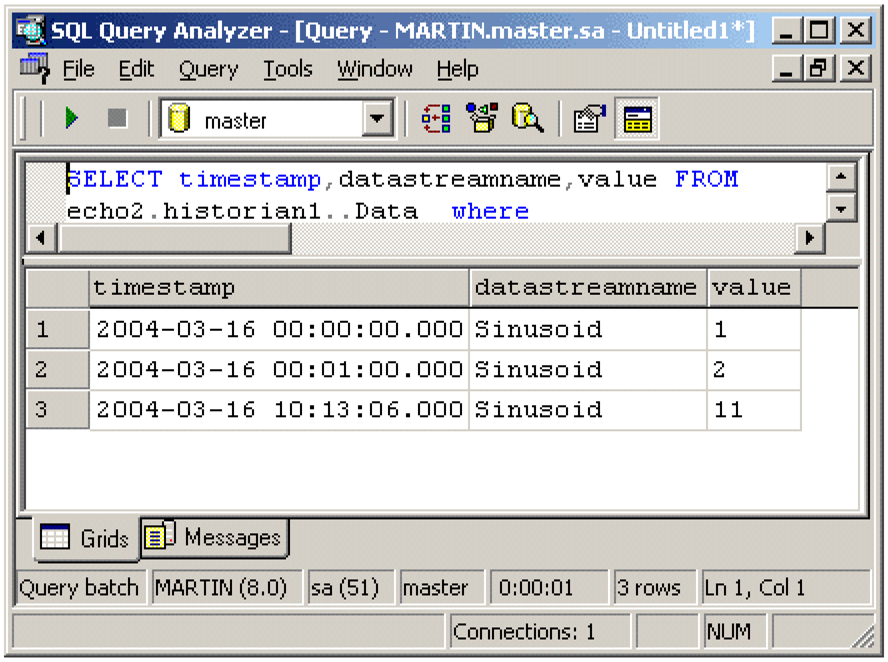
Four-Part Names

SQL Server uses four-part names for an unambiguous identification of the linked server objects (tables, views, etc.). The four-part name consists of the linked server name (name specified in the **Linked Server Properties** dialog box), catalog, schema, and object. See the figure below.


"Data Table"
Four-Part Name: "echo.db1..data"
Linked Server
Catalog
Schema
Object
ECHO
db1
data
"Because ECHO OLE DB does not use schemas, the schema part of the the four-part name is empty.

Sample SQL Statements

SQL statements accessing data through the linked server are executed the same way as "ordinary" SQL Server statements. The SQL Server Query Analyzer was used for internal testing. See the figure below.


SQL Query Analyzer - [Query - MARTIN.master.sa - Untitled1*]
File Edit Query Tools Window Help
master
Grids Messages
Query batch | MARTIN (8.0) | sa (51) | master | 0:00:01 | 3 rows | Ln 1, Col 1
Connections: 1 | NUM

```
SELECT timestamp,datastreamname,value FROM
echo2.historian1..Data  where
```

	timestamp	datastreamname	value
1	2004-03-16 00:00:00.000	Sinusoid	1
2	2004-03-16 00:01:00.000	Sinusoid	2
3	2004-03-16 10:13:06.000	Sinusoid	11

Sample statements are listed below.

Queries

```
SELECT * FROM ECHO.echo..Versions
SELECT * FROM ECHO.echo..Metadata
SELECT * FROM ECHO.echo..Historians
SELECT * FROM ECHO.historian1..DataFiles
SELECT * FROM ECHO.historian1..DataStreams WHERE Name LIKE 'DS%'
SELECT * FROM ECHO.historian1..Data WHERE Timestamp > '01-Dec-2002'
SELECT * FROM ECHO.historian1..Data2 WHERE Timestamp > '01-Dec-2002'
```

NOTE: In Query Analyzer, you can cancel (interrupt) any SELECT query during execution by pressing *Alt+Break*. Breaking any DML statement is NOT supported. This means that the by default issued '*SELECT * FROM table*' cannot be stopped. For more, see the "Known Linked Server Issues" section in this chapter.

DML statements

```
INSERT INTO ECHO.historian1..Data2 (DataStreamIdentity,[Timestamp],Value_I4)
SELECT [Identity],'27-Mar-2003 16:56:01',1 FROM ECHO.historian1..DataStreams WHERE
[Name] = 'DS_I4'
```

NOTE: Prior to executing any DML statement, the SQL Server fetches all table data. Because of this, tables that have many rows (for example, "data" and "data2") can exhibit slower response times.

To work around the limitation noted above, set the *Always Return Rowse;* property ("ECHO OLE DB Specific Attributes" section in the "Configuration for Data Access" chapter.) to *true* and execute a DML statement using the SQL Server OPENQUERY function. See Pass-Through Queries in the next section for more details.

```
SELECT * FROM OPENQUERY (ECHO,'INSERT INTO historian1..Data
       (DataStreamName,[Timestamp],Value) VALUES (''DS_I4'',''16-Mar-2004'',1)')
```

Pass-Through Queries

Sometimes it is useful to forward a query to the OLE DB data source "as is," bypassing the SQL Server preprocessing. To achieve this, the SQL Server provides three functions: **OPENDATASOURCE**, **OPENROWSET** and **OPENQUERY**. These functions can be used in the FROM clause of queries, which are then referred to as "pass-through" queries.

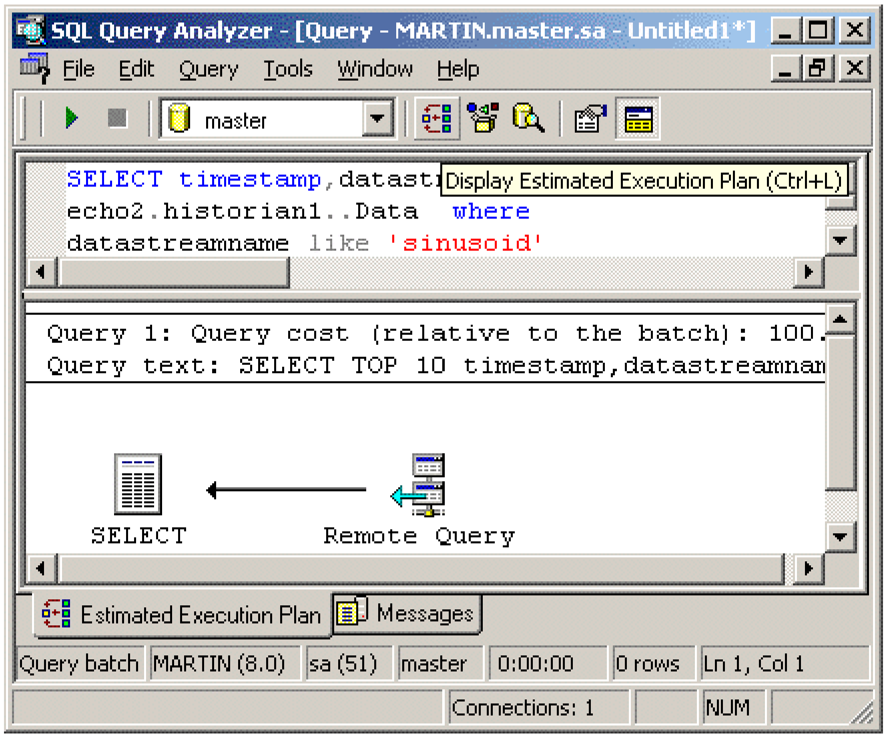
Syntax for these functions can be found in the SQL Server Books Online documentation.

Example

```
SELECT * FROM OPENQUERY(ECHO,'SELECT * FROM historian1.Data')
```

Known Linked Server Issues

The 'Estimated Execution Plan' feature in Query Analyzer can help you verify which part of the distributed query is sent to the provider, and what is evaluated on SQL Server side. Two of the Linked Server settings proved to be significant: ‘Collation Compatible' and 'Dynamic Parameters'. The figure below illustrates how to generate an Execution Plan.


SQL Query Analyzer - [Query - MARTIN.master.sa - Untitled1*]
File Edit Query Tools Window Help
master
Display Estimated Execution Plan (Ctrl+L)
SELECT
Remote Query
Estimated Execution Plan
Messages
Query batch
MARTIN (8.0)
sa (51)
master
0:00:00
0 rows
Ln 1, Col 1
Connections: 1
NUM

```
SELECT timestamp,datast
echo2.historian1..Data  where
datastreamname like 'sinusoid'
```

```
Query 1: Query cost (relative to the batch): 100.
Query text: SELECT TOP 10 timestamp,datastreamnan
```

The 'DML Rowset' proprietary setting (see the "ECHO OLE DB Specific Attributes" section in the "Configuration for Data Access" chapter) allows you to work with DML statements in the Linked Server environment, and also with larger tables like Data, Data2. By design, SQL Server always issues the 'SELECT * FROM table' statement whenever a DML statement is used. DML statements issued against a bigger 'data holding' table, for example, are likely to time out then. Setting the 'Always Return Rowset' option to True avoids this behavior by returning a spurious rowset from DMLs.

In combination with the SQL Server, the OPENROWSET or the OPENQUERY functions permit the following construction.

```
SELECT * FROM
       OPENQUERY(ECHO,'INSERT INTO data (DatastreamIdentity, Timestamp, Value) VALUES
       (''EB05C85F-6947-46C2-B6BE-C49820BC26FF'', ''19-Dec-2002 13:00:00'', 12345)')
```

The LIKE expression containing the ESCAPE argument is not forwarded to the provider.

```
SELECT * FROM echo.historian1..Data WHERE Value LIKE '\-%OSI\' ESCAPE '-'
```

To avoid this, use the pass-through query.

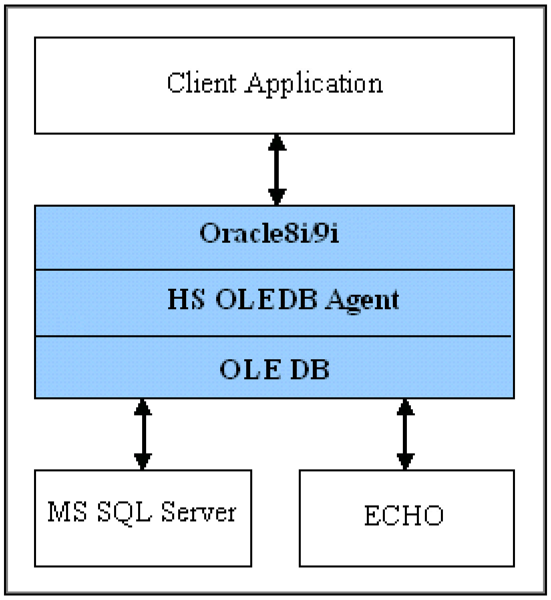
```
SELECT * FROM
       OPENQUERY(ECHO,
       'SELECT * FROM historian1.Data WHERE Value LIKE ''\-%OSI\'' ESCAPE ''-''')
```

Variant data type columns like Data.Value, Data.ValueEx,… do not evaluate properly when used in combination with parameters. SQL Server 2000 transforms the Variant data type into nVarchar and provides an inappropriate length for a parameterized query. The result is an empty item. To avoid this, use the Data2 table.

```
INSERT echo.historian1..Data2
(DataStreamIdentity, Timestamp, Value_I4)
VALUES ('EB05C85F-6947-46C2-B6BE-C49820BC26FF', '19-Dec-2002 13:00:00', 123)
```

Oracle Generic Connectivity

The Oracle 8.1.6i/9i RDBMSs is similar in concept to the MS Linked Server for accessing heterogeneous data sources. It is called Generic Connectivity. To access the non-Oracle data stores using Generic Connectivity, the Oracle 'Heterogeneous Services'- (HS) agents work with ODBC drivers or OLE DB providers. The driver/provider that you use must be installed on the same platform as the HS agent. The non-Oracle data stores (MS SQL Server,ECHO…) can reside on the same machine as the Oracle database server or on a different machine. See the figure below.



Configuration of Generic Connectivity

A client connects to the Oracle database server through Oracle Net. The Heterogeneous Services component of the Oracle database server connects through Oracle Net to the Heterogeneous Services OLE DB agent, and the agent communicates with the OLE DB provider.

To configure Generic Connectivity using OLE DB on a Windows NT operating system, follow the steps below.

1. Create and configure a Microsoft Data Link (.udl) file to connect to the target datastore. Test the udl to verify connectivity to the target datastore. This udl will be referenced in the initcholedb.ora file as described later on.
2. Verify that the following entries are in the tnsnames.ora and listener.ora files.

TNSNAMES.ORA

```
CHOLEDB =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS =
(PROTOCOL = tcp)(host=localhost)(port=1521)))
(CONNECT_DATA =
(SID = CHOLEDB)) <== sid needs to match listener and initHS_SID.ora
(HS=OK))  <== HS clause is placed in the description
```

LISTENER.ORA

```
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(SID_NAME = CHOLEDB) <== matches sid in   tnsnames.ora
(ORACLE_HOME = C:\oracle\ora90)
(PROGRAM = hsolesql) <== hsolesql is the agent executable
)
)
(SID_DESC =
(GLOBAL_DBNAME = ECHO)
(ORACLE_HOME = C:\oracle\ora90)
(SID_NAME = ECHO)
)
```

3. Load the new configuration by entering the following commands in a Command window.

```
lsnrctl stop
lsnrctl start
```

4. Create the initialization file.

   Oracle supplies sample initialization files named initagent.ora, where the agent might be hsodbc or hsoledb to indicate which agent the sample file can be used for as in the following.
   inithsodbc.ora
   inithsoledb.ora

   The sample files are stored in the $ORACLE_HOME\hs\admin directory.

5. To create an initialization file, copy the appropriate sample file and rename the file to initHS_SID.ora. In this example, the sid noted in the listener and tnsnames is choledb so the example file is called initcholedb.ora.

   The following entries are in the initcholedb.ora located in $ORACLE_HOME/hs/admin.

```
HS_FDS_CONNECT_INFO ="UDLFILE=c:\\Oracle\CHOLEDB\\
choledb.udl"
HS_FDS_TRACE_LEVEL = ON
HS_FDS_TRACE_FILE_NAME = choledb.trc
```

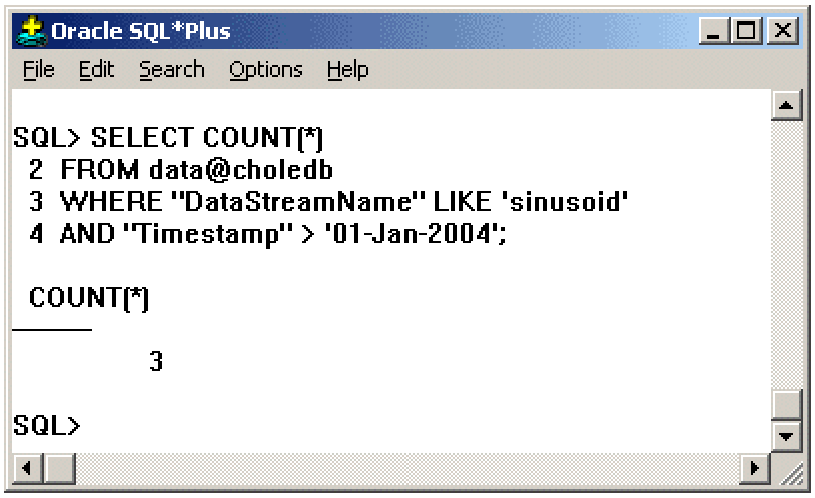
6. Create a database link to access the target database. Use appropriate quotes as noted below. Replace echouser and echopassword with a valid userid and password on the target datastore (ECHO), and create a database link to access the target database as follows.

```
CREATE PUBLIC DATABASE link CHOLEDB
CONNECT TO "echouser" IDENTIFIED BY "echopassword" USING 'CHOLEDB';
```

   Note that echopassword can be any non-empty string.
   More details about the Generic Connectivity concept are outlined in the document "4i_generic_connectivity.pdf," located on the Oracle Technology Network. Search for "video31".

7. To test, run a simple query of a known table on the target datastore in Oracle's SQL+. Enclose column names in double quotation marks (case sensitivity issue) when you use the ODBC driver to connect to Oracle. In addition, it is necessary to set the AUTOCOMMIT mode OFF. Enter the following command and refer to the figure below.

```
SELECT COUNT(*) FROM data@choledb WHERE "DataStreamName" = 'sinusoid' AND
"Timestamp" > '01-Jan-2004';
```



```
SQL> SELECT COUNT(*)
  2  FROM data@choledb
  3  WHERE "DataStreamName" LIKE 'sinusoid'
  4  AND "Timestamp" > '01-Jan-2004';

  COUNT(*)
----------
         3

SQL>
```

Related Documents

Oracle9i Heterogeneous Connectivity Administrator's Guide
June 2001
Part No. A88789-01 (a88789.pdf)

Chapter 8 Advanced Topics

This chapter describes several methods for viewing and processing data with several applications and utilities and the ECHO OLE DB Data Provider.

OLE DB – ODBC Transformation

To access OLE DB data through ODBC clients, use an OLE DB – ODBC gateway. In the simplest case, this is a generic ODBC driver for OLE DB data sources.

Tested OLE DB – ODBC scenarios

These scenarios have been tested, but other products are available.

- MS SQL Server 2000 as ODBC Gateway
- Oracle 8i/9i Generic connectivity
- Attunity Connect
- Microsoft Access 2000 / 2002 as ODBC Client

MS SQL Server 2000 as ODBC Gateway

Microsoft SQL Server can also serve as an OLE DB – ODBC gateway. To use it this way, configure a linked server pointing to an ECHO Server through ECHO OLE DB (see the "Microsoft SQL Server Linked Server" section in the "OLE DB Server Environments" chapter). Create distributed views "wrapping" its data (described below). Afterwards, access the created views from any ODBC client through the Microsoft SQL Server ODBC Driver.

NOTE: Create the Views in an MS SQL Server Database. This can be a newly created Database, for example, called ECHOViews, or any existing one.
To allow updates through ODBC clients, the Microsoft Distributed Transaction Coordinator (MS DTC) must be running.

Distributed Views

Clients, connected to SQL Server through ODBC (Microsoft SQL Server ODBC driver) or OLE DB (Microsoft OLE DB provider for SQL Server), cannot directly access the linked server tables. To avoid this problem, define SQL Server views. It is also useful to limit the accessible data range at the same time. This makes an unintentional "SELECT * FROM table" issued by the ODBC client or OLE DB consumer less disturbing.

NOTE: Although it appears that Access 2002 allows direct access to linked servers, it just automates the view creation in the background.

For the view definition, use both direct (using four-part names) and "pass-through" queries. Direct queries are faster.

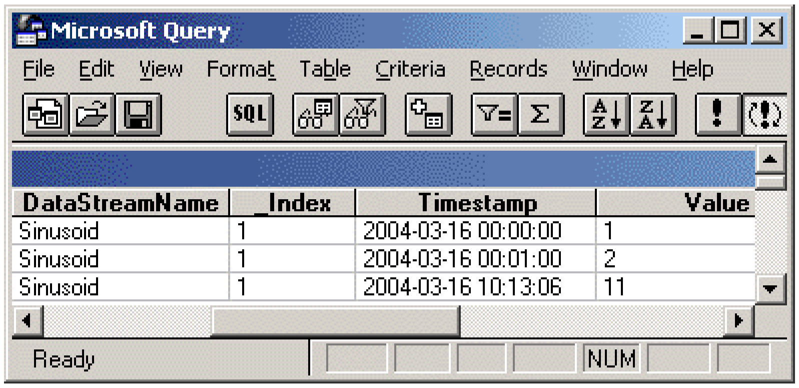
Examples

```
CREATE VIEW dbo.echo_data
AS
SELECT *
FROM ECHO.historian1..Data

CREATE VIEW dbo.echo_data2
AS
SELECT *
FROM OPENQUERY(ECHO, 'SELECT * FROM historian1.Data2')
```

Oracle 8i/9i Generic Connectivity

To use Oracle as an OLE DB – ODBC gateway, configure the Generic Connectivity as described in the "Oracle Generic Connectivity" section in the "OLE DB Server Environments" chapter. Use the ODBC client, for example MS Query, to connect to Oracle, and either specify the SQL query directly, or access the distributed views within Oracle.



DataStreamName	_Index	Timestamp	Value
Sinusoid	1	2004-03-16 00:00:00	1
Sinusoid	1	2004-03-16 00:01:00	2
Sinusoid	1	2004-03-16 10:13:06	11

NOTE: It is not possible to directly edit ECHO tables via MS Query connected to Oracle.

Attunity Connect

Attunity Connect (prior to version 3.0 called ISG Navigator) allows you to configure an ODBC data source (DSN) that accesses OLE DB data. Each such DSN can contain one or more databases (OLE DB catalogs). So configure it to represent all the catalogs in ECHO OLE DB.

A trial version of Attunity Connect can be downloaded from www.Attunity.com.

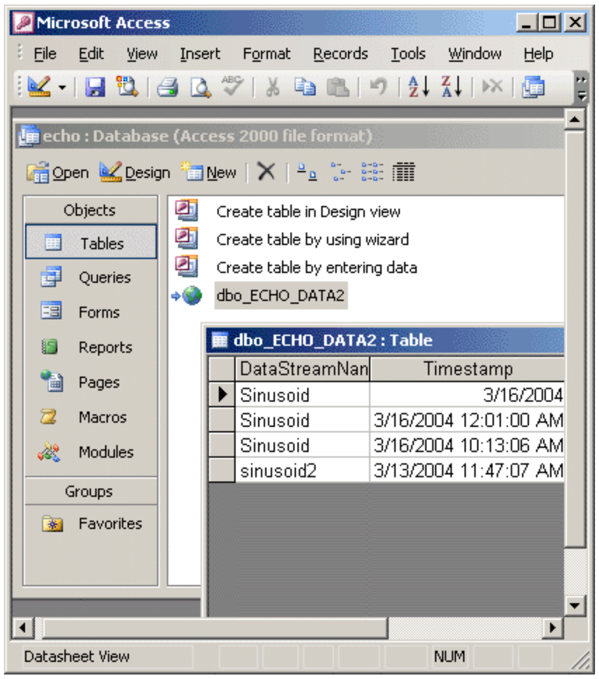
NOTE: Each Attunity DSN contains one database named "sys" by default. This database can be ignored.

Microsoft Access 2000 / 2002 as ODBC Client

Access is a very flexible RDBMS, distributed as a part of the Microsoft Office suite. It allows you to link ODBC tables and views into its database engine to work with its native data. To link an ODBC table or view, follow the steps below.

1. Run Access and create a blank Access database.
2. On the **File** menu, click **Get External Data**. Then click **Link Tables**. The **Link** dialog box appears.
3. In the **Files of Type** field, select ODBC Databases. The **Select Data Source** dialog box appears.
4. Select the appropriate ODBC data source. This can be a DSN for MS SQL Server 2000, a DSN for Oracle or a DSN for Attunity Connect.
5. In the **Link Tables** dialog box, select the tables to link.
6. In the **Select Unique Record Identifier** dialog box, select the table primary key columns.

   For editing data, Access requires a primary key to be specified.
   It is recommended that you avoid editing large tables through MS Access and MS SQL Server. This is because SQL Server always generates the "SELECT * FROM table" query before any DML statement. For details, see the "Known Linked Server Issues" section in the "OLE DB Server Environments" chapter.
7. Open the table. Refer to the figure below.


Microsoft Access
File Edit View Insert Format Records Tools Window Help
echo : Database (Access 2000 file format)
Open Design New
Objects
Tables
Queries
Forms
Reports
Pages
Macros
Modules
Groups
Favorites
Create table in Design view
Create table by using wizard
Create table by entering data
dbo_ECHO_DATA2
dbo_ECHO_DATA2 : Table
Datasheet View
NUM

DataStreamNan	Timestamp
Sinusoid	3/16/2004
Sinusoid	3/16/2004 12:01:00 AM
Sinusoid	3/16/2004 10:13:06 AM
sinusoid2	3/13/2004 11:47:07 AM

NOTE: MS Access 2002/2003 can also directly connect to an OLE DB datasource but only to the Microsoft OLE DB Provider for SQL Server.

Microsoft Excel 2000 / 2002

Microsoft Office 2000 introduced a new technology, Office Web Components, that allows interactivity to be added to Excel charts, spreadsheets, and pivot tables saved in HTML format. The Office Web Components have been further updated for Microsoft Office XP.

To import data from ECHO OLE DB tables, follow the steps below.

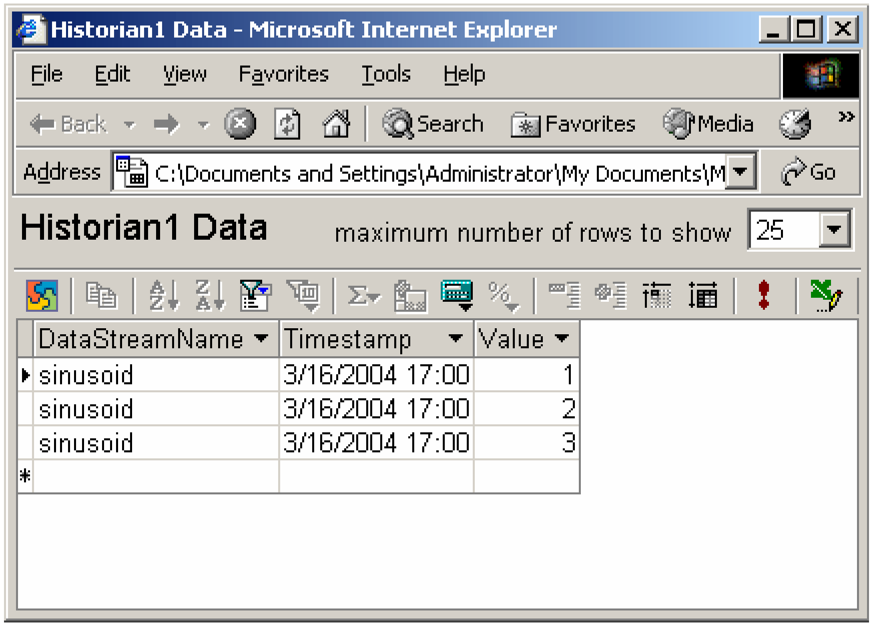
1. Run Excel.
2. On the **Data** menu, click **Import External Data**.
3. In the **Select Data Source** dialog box that appears, click **New Source Data Connection wizard**.
4. Select **Other/Advanced**, and choose ECHO OLE DB Provider.
5. Fill In the UDL connection dialog box.
6. Select the table to which you want to connect, and click **OK**.

The connection information is stored within an HTML ODC (Office Data Connection) file. Open the ODC file to view the resultset in the Internet Explorer in a fully updateable spreadsheet component.

Do not select from tables that have a large number of records, the data table, for example, without a where condition. The ODC file can be modified to execute a specific query as follows.

```
<odc:CommandType>SQL</odc:CommandType>
<odc:CommandText>SELECT * FROM historian1.Data WHERE Timestamp >='01-Jan-2004'
</odc:CommandText>
```

When you open an HTML document containing Office Web Components in Internet Explorer, it is possible to interact with the information on the page. For example sort, filter, and enter values for formula calculations. Refer to the following figure.


Historian1 Data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Documents and Settings\Administrator\My Documents\M
Go

Historian1 Data maximum number of rows to show 25

DataStreamName	Timestamp	Value
sinusoid	3/16/2004 17:00	1
sinusoid	3/16/2004 17:00	2
sinusoid	3/16/2004 17:00	3

It is possible to export data to Excel from the Internet Explorer. You can also import data from within Excel; on the **Data** menu, click **Get External Data** and select the ODC file.

Right click the imported data range to specify a background refresh frequency, so that the data is refreshed periodically .

RowsetViewer

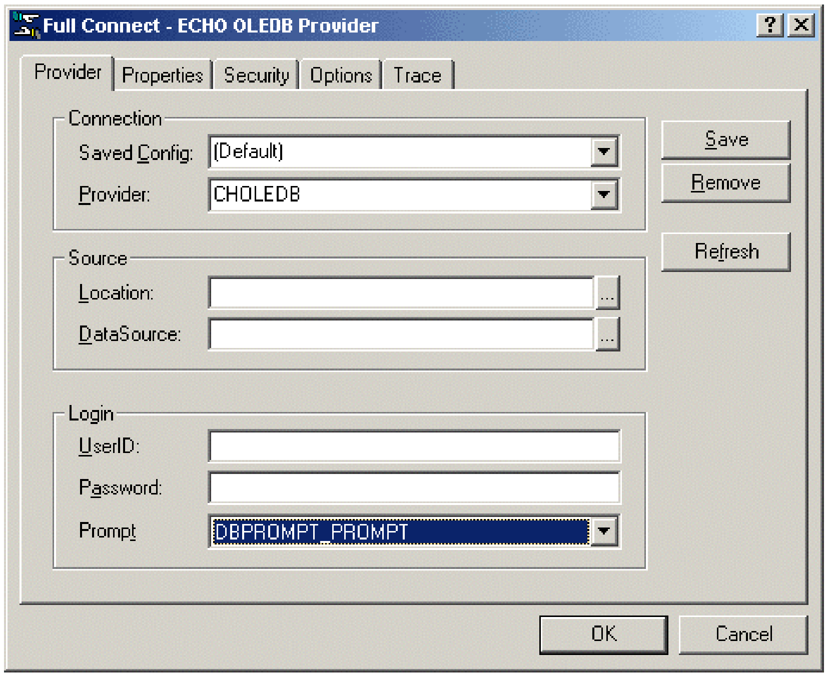
RowsetViewer is one of the basic OLE DB testing components provided to develop and test OLE DB providers. It ships with the Microsoft Data Access SDK and the Microsoft Platform SDK.

RowsetViewer communicates with OLE DB providers directly through OLE DB interfaces. This helps to diagnose the problems directly in the provider without interference from OLE DB consumers or ODBC client problems.

NOTE: If it is unclear whether a problem is caused by a consumer (there are many scenarios a consumer can follow while accessing an OLE DB provider) or the provider itself, try your queries first in RowsetViewer.

Connection

To establish a connection through ECHO OLE DB, on the **File** menu, click **Full Connect**. Then fill in the **Full Connect** dialog box. See the figure below.



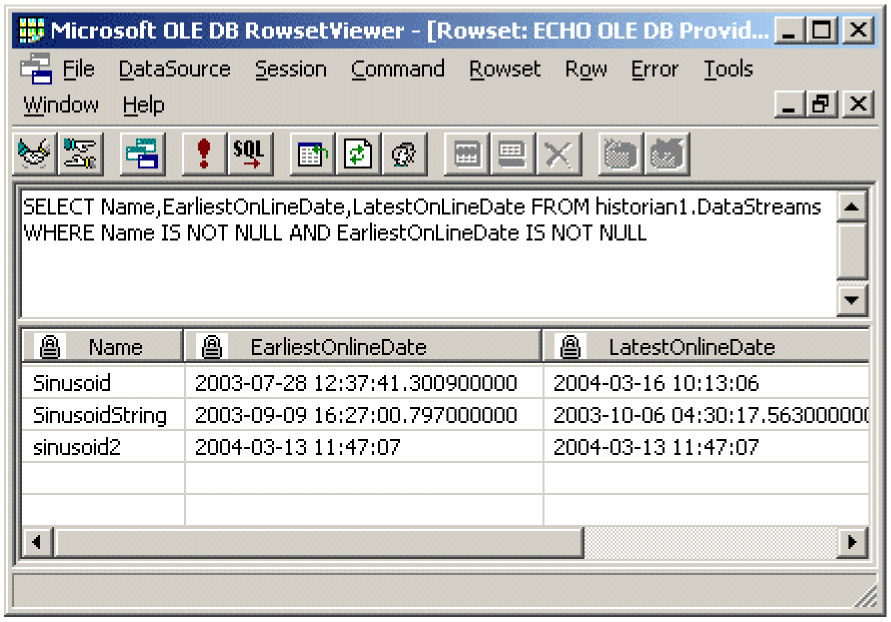
Connection
Saved Config: (Default)
Provider: CHOLEDB

Source
Location:
DataSource:

Login
UserID:
Password:
Prompt DBPROMPT_PROMPT

SQL Statement Execution

To execute an SQL statement, type it into the input window and click **SQL**. In the **ICommand::Execute** dialog box, click **OK** to confirm the statement. Refer to the figure below.


Microsoft OLE DB RowsetViewer - [Rowset: ECHO OLE DB Provid...
File DataSource Session Command Rowset Row Error Tools
Window Help
SQL

```
SELECT Name,EarliestOnLineDate,LatestOnLineDate FROM historian1.DataStreams
WHERE Name IS NOT NULL AND EarliestOnLineDate IS NOT NULL
```

Name	EarliestOnlineDate	LatestOnlineDate
Sinusoid	2003-07-28 12:37:41.300900000	2004-03-16 10:13:06
SinusoidString	2003-09-09 16:27:00.797000000	2003-10-06 04:30:17.56300000
sinusoid2	2004-03-13 11:47:07	2004-03-13 11:47:07

Note that a query timeout can be specified. Click **Properties** and set the **DBPROP_COMMANDTIMEOUT** property appropriately.

The RowsetViewer shows the subsecond part on the DBTYPE_DBTIMESTAMP data type column to a 100 ns precision.

MS SQL Reporting Services

Microsoft SQL Server 2000 Reporting Services is a server-based reporting platform that can be used to create reports with data from relational data sources, including those that are accessible through OLE DB interface.

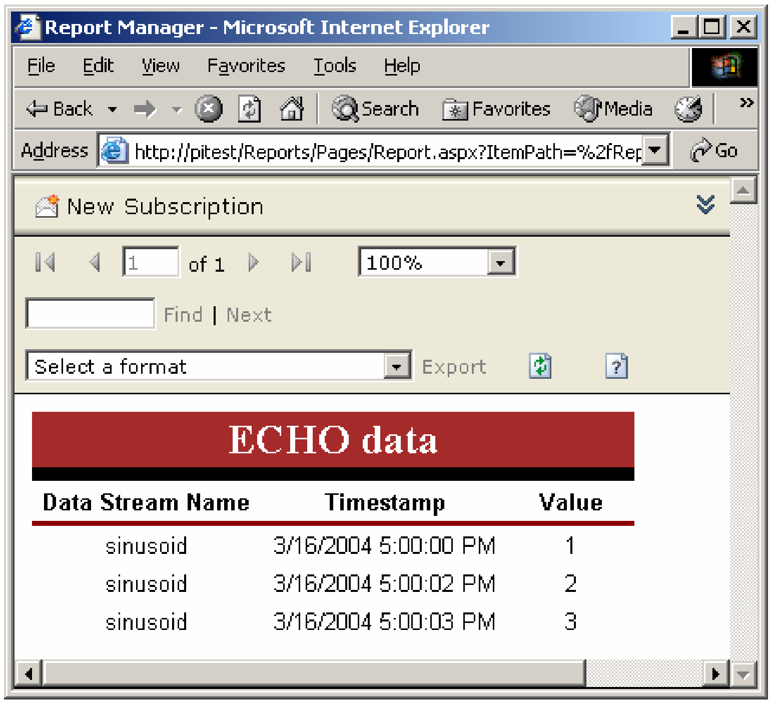
NOTE: Reporting Services is part of SQL Server 2000 license. For more information see: http://www.microsoft.com/sql/reporting

The following list of features highlights Reporting Services strengths:

- Connections to many different data sources (MS SQL Server, Oracle, IBM DB2, OLE DB, ODBC,etc) and combine disparate data into comprehensive report.
- Drag-and-drop development environment with no programming is needed.
- Rich formatting capabilities; a report can be transformed into a variety of formats like XML, Excel, PDF, graphic, Comma Separated File, etc.
- reports can be customized and delivered by way of subscription. For example reports are sent to an e-mail recipient in case a value exceeded its limit.
- Only a web browser is required for a client.

Example

Simple report generated and deployed to IIS (Internet Information Server) in MS Visual Studio .NET without any programming.


Report Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://pitest/Reports/Pages/Report.aspx?ItemPath=%2fRep
Go
New Subscription
1 of 1
100%
Find | Next
Select a format
Export

ECHO data

Data Stream Name	Timestamp	Value
sinusoid	3/16/2004 5:00:00 PM	1
sinusoid	3/16/2004 5:00:02 PM	2
sinusoid	3/16/2004 5:00:03 PM	3

Chapter 9 Troubleshooting

This chapter describes the most important troubleshooting tools and techniques available for the ECHO OLE DB Data Provider.

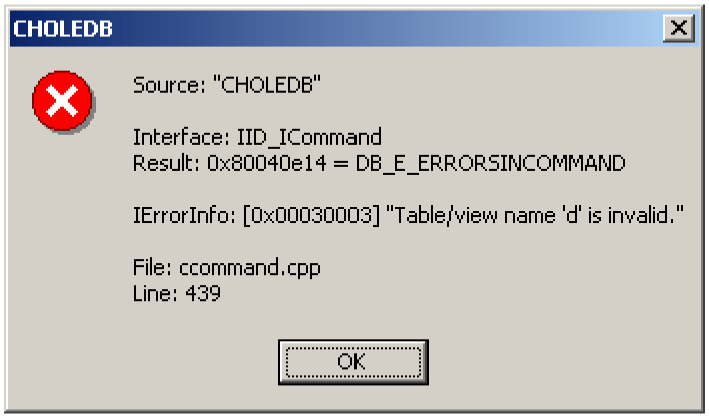
Error Messages

Error reporting in ECHO OLE DB complies with standard Automation and OLE DB error handling. Each interface method reports a return code indicating whether the method succeeded or failed. In addition, ECHO OLE DB exposes OLE DB error objects that contain extended error information, such as the detailed description of the error. Error objects can be created by any interface on any ECHO OLE DB object.

ECHO OLE DB divides error messages according to the source of a problem:

- Initialization errors – Connection problems, DLL loading errors, etc.
- General errors – Unsupported COM interfaces, out of memory, etc.
- SQL errors – Parsing errors.
- ECHO SDK errors.

The following figure shows an ECHO OLE DB error message indicated by the Rowset Viewer tool.



```
Source: "CHOLEDB"

Interface: IID_ICommand
Result: 0x80040e14 = DB_E_ERRORSINCOMMAND

IErrorInfo: [0x00030003] "Table/view name 'd' is invalid."

File: ccommand.cpp
Line: 439
```

NOTE: Error messages and warnings are printed into the specified *Log file* independent of the *Log Level* specified.

Log File

Some client applications (Microsoft SQL Server, for example) optimize execution of SQL statements. Consequently, statements actually executed by the OLE DB Data Provider do not necessarily match the original ones. Therefore, ECHO OLE DB allows you to create a log file that records executed statements and error messages.

Configure the log file through the **DBPROP_INIT_PROVIDERSTRING** OLE DB property (Extended Properties UDL property). For details, see the "Configuration Attributes" section in the "Configuration for Data Access" chapter.

The log file is assigned a version (a number is added at the end of the file name), and a new version is created when the provider is instantiated. Therefore, many log files can accumulate in the specified directory. It is recommended that you remove these files when they are no longer needed.

To increase the amount of logged information, change the setting of the *Log Level* property. The supported levels are 0-3. The higher the number, the more information that is printed.

Appendix A
Tested OLE DB Clients

This appendix lists the clients that have been tested with the OLE DB Data Provider.

Clients Tested

Client	Version
ADO	2.7
ADO.NET Microsoft .NET Framework 1.0 Microsoft Visual Studio .NET	 1.0.3705 Hotfix (325790)
ComponentOne True DBGrid	7.0.0.272
Microsoft Excel 2002 (through Office Data Connection)	10.2614.2625
Microsoft Data Grid	6.0.89.88
Microsoft SQL Server 2000 (through Linked Server) ODBC Driver for MS SQL Server	2000.80.534.0 2000.81.9031.14
Microsoft Visual Studio .NET (Server Explorer)	
Oracle 9i (through Generic Connectivity) ODBC Driver for Oracle	9.0.1.0 9.00.15.00
Rowset Viewer (part of MS platform SDK)	02.00.2905.0

Glossary

Glossary of Terms

ActiveX Data Objects (ADO)

ActiveX Data Objects (ADO) is a set of high-level Automation interfaces on top of OLE DB. It significantly simplifies the access to the OLE DB data by eliminating low-level operations like managing memory resources or component aggregation.

Catalog-

OLE DB uses three-part names for naming of tables. "Catalog" (some RDBMS use "Database" as a synonym) is the first part ("catalog.schema.table") and is intended to logically group tables. See also Relational Database Management System.

Connection String

For ADO and UDL, the "connection string" is a string containing the information necessary to establish a connection via OLE DB. See also Universal Data Link, ActiveXDataObjects.

Data Definition Language (DDL)

A language used for defining attributes of a database, e.g. table structures or views. In SQL, there's a set of statements serving for this purpose, sometimes referred to as SQL DDL (e.g. CREATE TABLE and DROP TABLE statements). See also Structured Query Language.

Data Manipulation Language (DML)

A language used to insert data in, update, and query a database. In SQL, there's a set of statements serving for this purpose, sometimes referred to as SQL DML (INSERT, UPDATE, DELETE., and SELECT statements). See also Structured Query Language.

Data Source Name (DSN)

A Data Source Name (DSN) is a name assigned to an ODBC data source. Applications can use DSNs to request a connection to a data source via ODBC. See also ODBC Data Source.

Database

See definition of Catalog.

Distributed Query

For Microsoft SQL Server, distributed query is a query that accesses data stored in SQL Server (homogeneous data) plus data stored in a data store other than SQL Server (heterogeneous data).

Generic Connectivity

For Oracle 8i,9i, similar concept as "Linked Server" within Microsoft SQL Server.

Linked Server

For Microsoft SQL Server, "Linked Server" is a term for a virtual RDBMS server that can be defined within the SQL Server to access OLE DB data sources. The linked server configuration allows SQL Server to execute commands against OLE DB data sources on different servers.

Microsoft Data Access Components (MDAC)

A toolkit that includes key technologies to enable Universal Data Access (UDA). MDAC consists of the latest versions of ADO, OLE DB, and ODBC. For more information about these technologies, see "Data Access Services" in the Platform SDK or www.microsoft.com/data/. See also ActiveX Data Objects, OLE DB, Open Database Connectivity.

Microsoft Distributed Transaction Coordinator (MS DTC)

The Microsoft Distributed Transaction Coordinator (MS DTC) is a transaction manager that allows client applications to include several different sources of data in one transaction. MS DTC coordinates committing the distributed transaction across all the servers enlisted in the transaction.

Microsoft Management Console (MMC)MMC is a common console framework for system management applications. The primary goal of MMC is to simplify administration and integrate administrative tools. MMC hosts administrative tools (called MMC snap-ins); however, the console itself provides no management functionality.

Microsoft SQL Server

A high-performance relational database management system for Microsoft Windows NT Server-based systems. Designed to meet the requirements of enterprise client/server computing and the Internet, SQL Server is tightly integrated with the Microsoft BackOffice family of servers. . For more information, see "Data Access Services" in the Platform SDK or www.microsoft.com/sql/. See also Relational Database Management System.

ODBC Data Source

A data source that can be accessed using an ODBC driver. Also, a stored definition that contains all of the connection information an ODBC application requires to connect to the data source. See also Data Source Name.

OLAP

OnLine Analytical Processing is a technology that enables client applications access data in Data Warehouses.

OLE DB

OLE DB is a set of COM interfaces that provide applications with uniform access to data stored in diverse information sources and that also provide the ability to implement additional database services. These interfaces support the amount of Database Management System (DBMS) functionality appropriate to the data store, enabling it to share its data. See also Open Database Connectivity.

OLE DB Data Consumer

Any software that calls and uses the OLE DB application programming interface (API). See also OLE DB.

OLE DB Data Provider

A software component that exposes OLE DB interfaces. Each OLE DB provider exposes data from a particular type of data source (for example SQL Server databases, Access databases, or Excel spreadsheets). See also OLE DB.

OLE DB Service Provider

A software component, which extends the functionality of OLE DB data providers by implementing extended interfaces, not natively supported by the data stores; such as query processors, cursor engines, and synchronization service. See also OLE DB Data Provider.

Open Database Connectivity (ODBC)

A data access application programming interface (API) that supports access to any data source for which an ODBC driver is available. ODBC is aligned with the American National Standards Institute (ANSI) and International Organization for Standardization (ISO) standards for a database Call Level Interface (CLI).

Relational Database Management System (RDBMS)

The RDBMS presents the data as tables that consist of rows and columns. Typically, the RDBMS supports SQL to retrieve and update the data. The RDBMS also manages user access to the data. Microsoft SQL Server and Oracle are examples of an RDBMS that stores relational data.

Structured Query Language (SQL)

A language used to insert, retrieve, modify, and delete data in a relational database. SQL is the language supported by most relational databases, and is the subject of standards published by the International Standards Organization (ISO) and the American National Standards Institute (ANSI). SQL Server 2000 uses a version of the SQL language called Transact-SQL.

Universal Data Access (UDA)

Universal Data Access (UDA) is a new Microsoft architecture that provides high-performance access to a variety of data formats (both relational and non-relational) on multiple platforms across the enterprise.

Universal Data Link (UDL)

Universal Data Link is a text file with the ".udl" extension containing the information necessary to establish a connection via OLE DB. This version of the connection information is referred to as a connection string). See also Connection String.

Index

A

ActiveX Data Object. *See* ADO
ADO
  Databinding ActiveX Controls, 6-5
  DBTIMESTAMP, 6-6
  Defined, 6-2
  Example Tools, 6-7
  Object Model, 6-4
ADO.NET Intro Example Tool, 6-7
ADO_Intro Example Tool, 6-7
Architecture
  OLE DB, 1-3
  Universal Data Access, 1-3
ASP Example Tool, 6-7
Attributes
  Configuration, 3-2
  Specific, 3-3,3-5
Attunity Connect
  ODBC Client, 8-5

C

Catalog
  Defined, 5-2
  ECHO, 5-4
  Historians, 5-7
Clients
  ODBC, 8-2
  RowsetViewer, 8-9
Configuration
  Attributes, 3-2
Configuring the Connection, 3-2,3-6
Connection
  Attributes, 3-2
  String, 3-6,3-10
CREATE DATABASE Statement, SQL, 4-4,5-3

D

Data Consumer, 1-3,1-4
Data Provider, 1-3,1-4
Data Table, 5-7
Data2 Table, 5-7
Database
  Non-Relational, 1-5
  Relational, 1-5
DataFiles Table, 5-7
DataStreams Table, 5-7
DELETE Statement, SQL, 4-4
DROP DATABASE Statement, SQL, 4-4,5-3

E

ECHO OLE DB
  Described, 1-2
Error Messages, 9-2
Example Tools, 2-3,6-7
Expressions, SQL, 4-3

F

Four-Part Names. See Linked Server Four-Part Names

G

Generic Connectivity
  Architecture, 7-16
  Configuring, 7-17
  ODBC Client, 8-4

H

Hardware Requirements, 2-2
Historians Table, 5-4

I

INSERT Statement, SQL, 4-4
Installing ECHO OLE DB, 2-2

L

Linked Server
  Architecture, 7-2
  Configuring, 7-3
  Distributed Queries, 7-2

Four-Part Names, 7-10
Known Issues, 7-14
ODBC Client, 8-3
Pass-Through Queries, 7-13
User Account/DCOM Settings, 7-8
Log File, Specific Attribute, 3-5
Log File, Troubleshooting and, 9-3

M

Metadata Table, 5-4
Model
ADO, 6-4
MS Access
ODBC Client, 8-6
MS SQL Server Linked Server. *See* Linked Server

O

ODBC Clients, 8-2
OLE DB
Architecture, 1-3
Catalogs, 5-2
Data Consumer, 1-3,1-4
Data Provider, 1-3,1-4
Described, 1-3
Example Tools, 2-3,6-7
Installing, 2-2
Requirements, 2-2
Service Provider, 1-3
Uninstalling, 2-2

P

Pass-Through Queries. See Linked Server Pass-Through Queries

R

Relational Database, 1-5
Requirements, Software, 2-2
Requirements, System, 2-2
RowsetViewer, 8-9

S

SELECT Statements, SQL, 4-2
Service Provider, 1-3
Specific Attributes, 3-3
Log File, 3-5
SQL
CREATE DATABASE Statement, 4-4,5-3
DELETE Statement, 4-4
DROP DATABASE Statement, 4-4,5-3
Expressions, 4-3
INSERT Statement, 4-4
SELECT Statements, 4-2
Statement Examples, 4-2,5-16,7-11
UPDATE Statement, 4-4
System Requirements, 2-2

T

Table
Data, 5-7
Data2, 5-7
DataFiles, 5-7
DataStreams, 5-7
Historians, 5-4
Metadata, 5-4
Versions, 5-4
Troubleshooting, 9-2

U

Uninstalling OLE DB, 2-2
Universal Data Access
Architecture, 1-3
Universal Data Link (UDL), 3-6,3-10
UPDATE Statement, SQL, 4-4

V

Versions Table, 5-4