



ECHO

ECHO Embedded Component Historian Object

Technical Data Sheet

The ECHO Embedded Component Historian Object by OSISOft enables a developer to easily add data archiving to hardware devices and software products with minimal compromise of performance and no large resource requirements. The resulting data historian is scalable, robust, high-performance and audit-trackable in real time. ECHO design assures the developer of ease of integration, scalability, high capacity, connectivity and functionality.

ECHO Capabilities

ECHO is a powerful development tool that provides the fastest possible way to store and retrieve time-series data. It supports data archives of more than 1,000,000 variables per second, scales to match your application using as little as 2MB disk and 2MB RAM (plus cache). An ECHO archive stores up to 1 million data streams while occupying minimal storage space. All historian management and data functions are integrated through its SDK, and it leverages the native functionality of Windows environment, including:

- Audit logging & Performance counters
- Internationalization & localization (both time and language)
- Native 64bit 100ns resolution
- Fully threaded, allowing multiple historian instances to run simultaneously on a node or machine

Each ECHO data stream will encapsulate all time-series data with no data loss and no performance degradation over time, including Time, Value, Extended Value (see detailed list). ECHO imposes virtually no limitations on storage or retrieval of data and supports all variant data types (see detailed list). To enable forecast simulation data, it supports time into past and future.

ECHO OLE DB supports SQL queries, data management and link server architecture so you can create tags and data streams using this function, and supports ADO: SQL controls enable building data streams and historians; uses ANSI SQL 92 subset.

ECHO Exception Reporting supports optional client-side exception reporting through the ECHO SDK. Exception reporting is the process of sending events to the archive engine only when there has been a significant change in the monitored value. The purpose of exception reporting is to reduce the number of events stored in the archive without compromising the information. Because it is implemented in the SDK, traffic between the ECHO clients and the archive engine is also reduced.

ECHO Event notifications: clients may sign up to be notified when significant events occur in ECHO such as configuration changes and time-series data being added, modified or deleted within the ECHO historians. Clients may also sign up to receive completion status regarding asynchronous operations the client made to ECHO.

Choice of data archiving strategies include (a) high speed circular buffer historian which performs automatic space reclamation and (b) high speed archive which employs rolling file strategy similar to OSISOft's flagship PI System

Create self-healing historians with ECHO: a new feature checks archive on power up and can self-heal any data corruptions.

OPC Data Acquisition servers support other OPC-compatible DA or HDA clients.

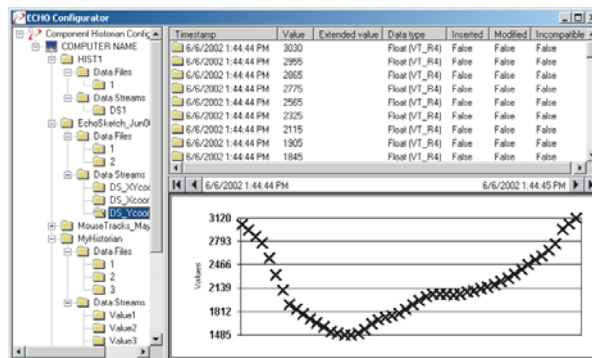
The ECHO package consists of two basic components:

- ECHO *Software Developer Toolkit (SDK)* provides the tools for creating applications and interfaces that interact with the archive engine.
- ECHO *Archive Engine* that manages the data historians and the reading/writing of process data. A set of time stamped data values, along with their extended data descriptions (if any), are defined as a data stream.

ECHO Configurator

The ECHO Configurator is a utility used to create historians for storing process data. This Configurator, shown in the figure below, is

provided as a sample program. It is a fully functional client and you can use it to create historians and add data points (data streams) to ECHO historians. You can manually write values for individual data streams and you can manually select data streams from which you want to read archived values.

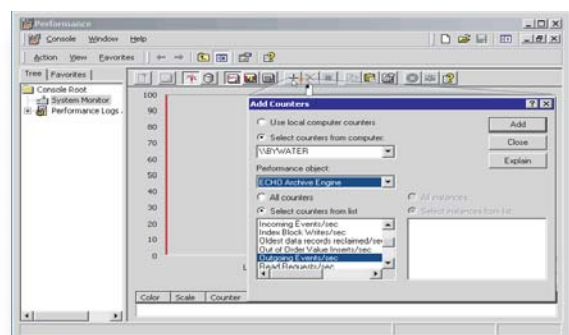


The Configurator is provided as an ActiveX control (in source code form) and can be embedded in an application or used unchanged with the provided test container, ConfigCtlTest.exe. This sample application allows the user to configure an entire historian network.

To automate reading and writing time-series data, sample code has been provided for both data storage and retrieval. The user can also examine the source code for examples of how to implement this functionality.

Monitoring ECHO Performance

You can monitor performance parameters for ECHO, such as the number of write operations, the number of write operations per second, incoming events per second, etc. You can add these to the Windows Performance Monitor as shown in this display:



Analyzer Stress Tool

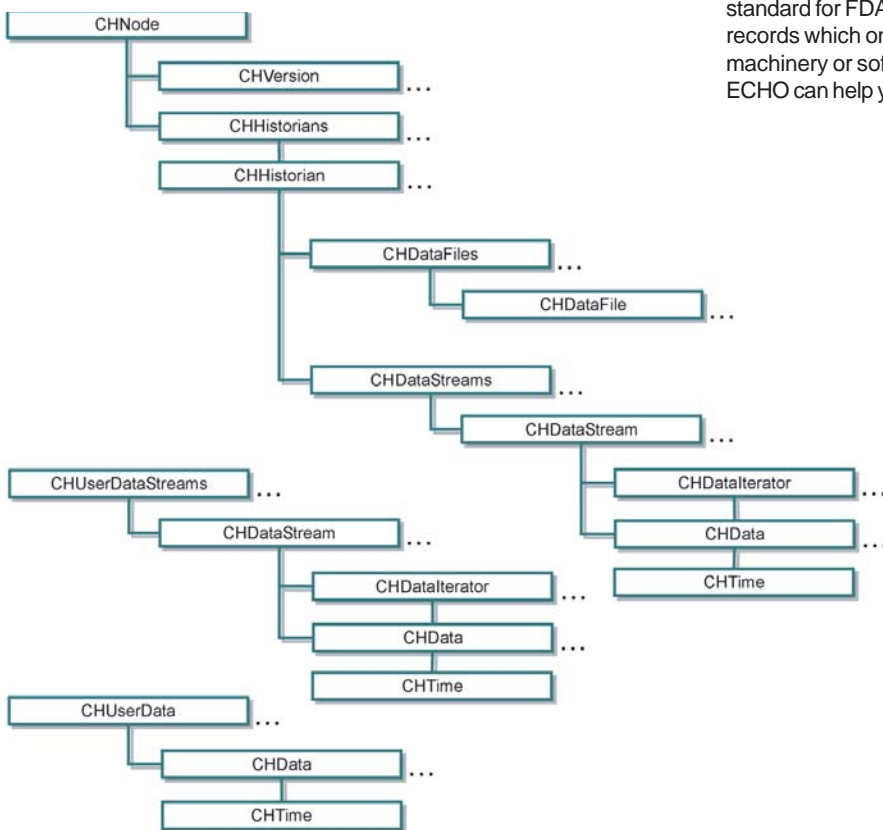
The ECHO stress tool enables the developer to test performance of their system and to quantify its speed of storage and data retrieval by loading the application with the amount, type and speed of information intended to be archived.

Full Size Estimator

Estimator provides information on the size of an archive based on the data type, resolution. Output can be plotted from CSV file.

ECHO Example Code

Simple and generic programming examples are included in various languages to show the use and capabilities of the archive system and to serve as templates for development of customized applications.



ECHO Architecture and Functionality

ECHO Integration

In addition to being for integration into custom applications ECHO has been developed to integrate easily into other front and back office applications, including relational databases and analysis tools. Communication connectors provide data from the archive to a wide range of standard interfaces such as OPC, OLEDB, and .NET.

Archive Engine

The archive engine is a Windows service that manages the historians and their associated data files. A data historian consists of one or more sets of time-stamped data values, which are defined as data streams. For example, you can define a single data historian to manage all time-series data for all the sets of data (data streams) for a given production line, and then define other historians to handle other production lines.

The archive engine starts automatically when the computer powers up. One archive engine can be located on a node or machine, and it can support multiple applications with which it exchanges data, and multiple data historians. The archive engine is scaleable, which means it can manage data streams that range from very few in number to up to one million.

Audit Tracking

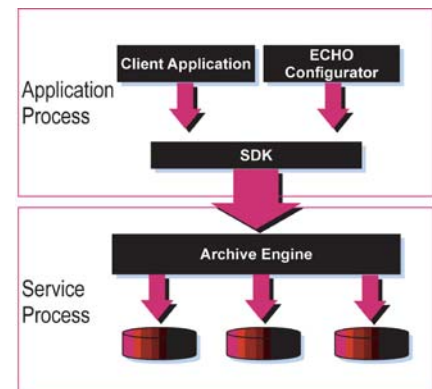
An ECHO data historian helps meet strict FDA regulations for electronic documentation and record keeping. The 21 CFR Part 11 standard for FDA-regulated industries now requires permanent records which only authorized personnel can change. If your machinery or software system records large amounts of data, ECHO can help you comply with Part 11.

ECHO SDK

The ECHO Software Development Kit (SDK) provides both automation and COM access to the ECHO archive engine written in any COM-supported languages such as Visual Basic, VBScript, as well as custom applications written in C, C++, C# or Java.

The SDK presents a hierarchical model of objects and collections that represent underlying ECHO features and concepts. A detailed description of objects is given below under the heading "Object Types."

The architectural relationship of the archive engine to the SDK and client applications is shown in the following figure.



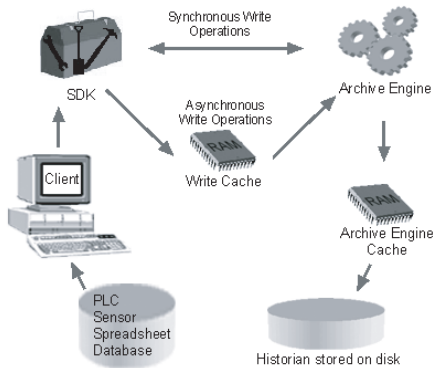
Objects

Code samples included in ECHO can interact with the ECHO SDK to create an historian, data streams, and add/retrieve data. ECHO enables the user to create objects that support the entire variety of functions necessary for data storage and retrieval.

The (partial) object model for the SDK is shown in the figure above left.

Writing Data

The following figure shows how the SDK sends data values for a data stream to a historian.

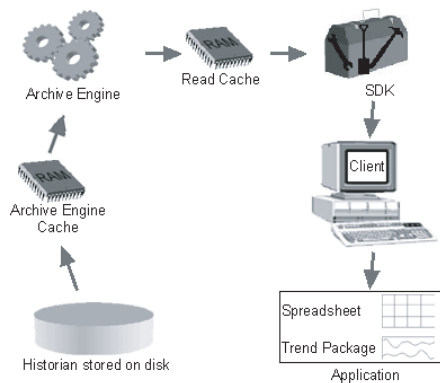


For fastest performance, write operations send data to the Write Cache. The size of the cache is user defined. When the cache is full, a user-configured timer times out, or a client application requests that the cache be flushed, the data is archived in the Historian. Note that a request by a client to do a synchronous write operation also causes the Write Cache to be flushed.

Two types of write operations are supported: a synchronous write (greatest performance, limited error reporting) and asynchronous write (does not use the Write Cache, it is the preferred method when specific notification that an operation was successful is required).

Reading Data

The following figure shows how the SDK reads data values for a data stream from a historian and sends it to the client.



To retrieve a data value from a historian, a client must specify a time stamp and a data stream Name or GUID. The SDK caches data that is read from the archive engine to reduce the number of read requests and responses.

Types of Data Supported

- Boolean (true / false)
- Decimal variables stored as 96-bit (12-byte) unsigned integers scaled by variable power of 10
- Double-precision floating point
- Empty data value
- Signed four-byte integer
- Signed integer (size is that of default integer)
- Signed one-byte integer
- Signed two-byte integer
- Signed four-byte Integer
- Single-precision floating point
- String
- Unsigned four-byte integer
- Unsigned integer (size is that of the default integer)
- Unsigned one-byte integer
- Unsigned two-byte integer
- Unsigned four-byte Integer
- Variant arrays
- Canonical Type Arrays
- Visual Basic currency variable (64-bit integer)
- Visual Basic date variable
- 32-bit Windows error code
- Blobs (Binary large objects)

Deliverable Software Components

The ECHO product provides all software components needed to install and use the historian services. These components include the following:

- **CHSvc and support binaries** - the archive engine, a Windows service that handles the management of historians.
- **License File** - ECHO license DLL provides the hooks to manage a wide range of licensable features including the number of historians, datastreams, client connections, demo timeout, maximum data storage and many more. Designed to work with any licensing model
- **CHSDK** - the COM / OLE Automation interfaces needed to write custom applications. The ECHO Software Developer Toolkit (SDK) is a set of COM interfaces that provide access to the historian system. OLE Automation is supported for Visual Basic, including late binding for scripting languages. The main SDK data objects are CHNode, CHHistorians, CHHistorian, CHDataFiles, CHDataFile, CHDataStreams, CHDataStream, and CHData. Several other minor objects are provided. The SDK also supports client-side caching and several types of asynchronous events.

- **ECHO Configurator** - a sample tool used for configuring historians, creating data streams, and archiving data. It is provided as an ActiveX control (in source code form) as an example of the SDK capabilities.
- **Analyzer** - tool for analyzing your system by configuring large amounts of data quickly and then gathering and logging performance numbers for specified time intervals. Provided in source code form
- **Windows PerfMon support** - an interface to PerfMon.exe for logging performance counters. Extended to include monitoring of archive performance

ECHO Development System Requirements

ECHO is designed to run on any 32-bit Windows PC. Before installing ECHO, verify that your development system meets the following requirements. Your system may need additional tools to support CE.NET development.

- Pentium processor-based system (Pentium II 300 MHz processor or faster is recommended)
- At least 64 MB of RAM (128 MB recommended)
- CD-ROM Drive
- Windows Vista, XP, XP Embedded, Server 2003, Server 2008, 2000, CE.NET 4.2
- Client development requires COM- or .NET-enabled programming tools such as Visual Studio 6.0 or Visual Studio .NET (CE Platform builder may also be required to include ECHO into your CE image)
- Hard disk space: 10 MB of hard disk space needs to be available, depending on selected options and platforms (20 MB recommended)
- Monitor capability: VGA with 256 colors and 640 x 480 pixel resolution (256 colors and 800 x 600 pixel resolution recommended)
- Mouse or other pointing device compatible with Windows operating systems

ECHO Target System Requirements

ECHO is designed to run on any 32-bit or 64-bit Windows system. Specific performance requirements will be based partly on your application. Before installing ECHO, verify that your target system meets the following requirements.

- Windows Vista, XP, XP Embedded, Server 2003, Server 2008, 2000, CE.NET 4.2
- Storage space: 1MB storage plus space for local archive

ECHO Support and Maintenance Program

- Developer Hotline connects you directly to the development team for complete and timely answers
- FAQ and issue status and defect reporting
- On-site application modeling assistance
- Service agreements are available based on major software revision
- Application consulting services available
- Technical training available

ECHO-Based Applications

- Embedded, small footprint or mobile
- High-speed and precision data
- Remote monitoring or machine-to-machine (M2M)
- Remote data collection supporting PI

Integration with the PI System

- PI Connector – allows data to be used directly by a PI server
- PI Interface – Fast aggregation of data across disparate or slow networks
- OPC server – Data access or historical data access
- OLEDB – Relational data provider

Royalty-based ECHO License

- Supports long-term OEM agreements
- Runtime redistributables with embedded installation modules
- Flexible server-based licensing controls integrate with any licensing scheme
- Allows for field changes from demo to unlimited or anywhere in between
- Source escrow option

Summary: ECHO Functionality

Performance

Data Archive	1,000,000+ values per second
Data Retrieval	1.5 Million values per second
Data Storage	Unlimited (media only)
Historians per node	1 to 100
Data Streams	Up to 1,000,000
Data Access Time	10 microseconds (avg.)
Validation	Supports 21 CFR Part 11

Technical Support

Phone support	Direct support by OSISOFT engineers
Email	24x7 assistance
Web Site	FAQ's, issues and upgrades
Documentation	Programmers and users reference manuals
Platforms:	Windows Vista, XP, XP Embedded, Server 2003, Server 2008, 2000, CE.NET 4.2

Early support for Microsoft standards

NOTE: Performance benchmarks are based on a dual processor running at 1.8GHz. Actual performance is CPU bound.

For additional information and a free demo package visit us at www.echohistorian.com

