

WHITE PAPER

The Alchemy® Software Developers Kit (SDK)



INFORMATION MANAGEMENT RESEARCH

Information Management Research

6025 South Quebec Street, Suite 260
Englewood, CO 80111
303-689-0022

www.imrgold.com

Summer 2003 Version 7

Abstract

This paper describes the Alchemy Software Developers Kit (SDK) consisting of COM (Component Object Model) object interfaces and ActiveX controls. The Alchemy SDK extends the use of the popular Alchemy product line into mission critical business applications by enabling the control of Alchemy database objects through industry-standard interfaces. This brings affordable document storage and retrieval solutions to more business applications and solves IT problems of compliance, security and information access to unstructured documents.

Overview

The Alchemy SDK provides a mechanism for developers to incorporate Alchemy functionality into a custom application using the Component Object Model (COM) specification. The Alchemy SDK consists of two parts.

1. Alchemy Objects provide the main functionality of Alchemy while allowing programmatic control and flexibility over how Alchemy is implemented in a custom application.
2. The Alchemy SDK provides a set of three ActiveX controls that may be added to an application by dropping a representation of the control from a development environment toolbox onto your project's COM compliant container—typically a designer form of some type.

Use these controls to provide visual display of a specific aspect of Alchemy's functionality.

Use Models for the Alchemy SDK

Because a developer has access to every Alchemy dialog, the easy to navigate Alchemy Tree View and user-friendly Viewer panes allow for a wide variety of uses. For example, the SDK allows for automatically opening of the search dialog in the Alchemy interface, or opening the same search dialog within another application. A programmer can create a customized interface to simplify scanning, indexing, viewing and searching, where the standard Alchemy interface may be considered too complicated for the end user who only performs limited Alchemy functions.

A programmer using the Alchemy SDK can also block features from the end user. For example, users who are assigned the task of indexing documents can be provided through explicit functionality, while other functions are disabled such as Build Database, Delete Database, Clear Database and the entire Design Menu.

Any of these use models can be "standalone," where Alchemy is not interacting with any other software, or can be embedded into another application that opens Alchemy dialogs or customized Alchemy dialogs.

One example of effective SDK use is for a business that needs to track customer or user interaction with the Alchemy archive system. Every time a user views a record or performs a search query, the details can be written to an audit log. This is useful for

regulatory compliance, security, billing purposes or to justify system growth based on usage.

Integration with existing business applications

The SDK enables Alchemy to serve as a document repository, storing image or document files with pointers for a relational database application to locate documents. Unlike other storage systems that can only provide a path to the file, Alchemy can store any data field or fields as a pointer to retrieve the document. The developer simply passes the query string into the Alchemy system and receives a results list or the exact document called for.

Integration with Lotus Notes and other workflow software

The SDK enables Alchemy to serve as an archive repository for Notes or other workflow environments. Documents can be scanned and stored in Alchemy, and retrieved back into the workflow process; annotations may be added in the Alchemy viewer and saved back into the database.

Customize the Alchemy user interface

A custom user interface (UI) developed using the SDK can replace the standard Alchemy UI. This may be desirable in applications where the Alchemy UI is “overkill”, or is considered too complex for the end user, or where the Alchemy UI exposes more functionality than is desired.

Simple Search

This use model has been defined as a simple document retrieval interface with one or two query fields. This can easily be developed with the SDK.

SDK Structure

Windows Operating Systems

The Alchemy SDK is used to create applications that run on a Microsoft Windows computer. Programs will operate on the same platforms as the standard Alchemy Windows product.

- Microsoft Windows 2000
- Microsoft Windows NT Workstation & Server 4.0
- Microsoft Windows 2000 Professional & Server
- Microsoft Windows XP Professional
- Microsoft Windows 2003

Standard automation interfaces

The Alchemy SDK uses industry standard automation interfaces.

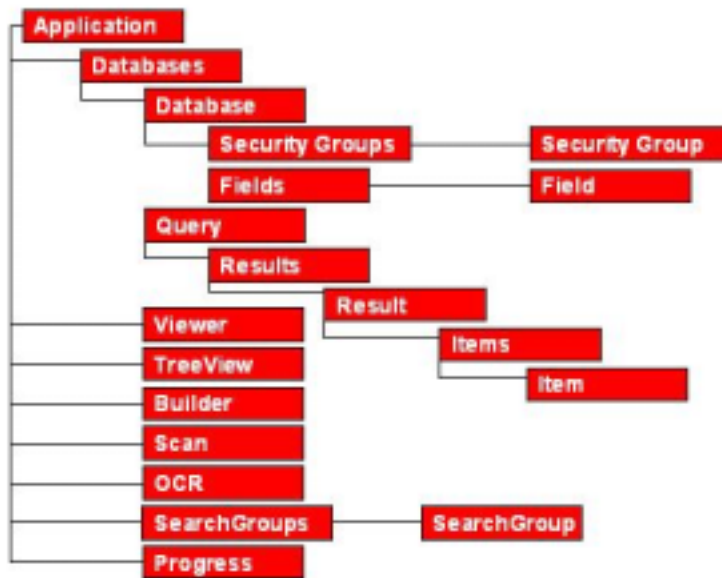
- COM (Component Object Model) object library
- ActiveX controls
- ODBC V2.0

Suitable Programming or scripting environments

- Alchemy version 7.0 or higher installed for both the development environment and for the custom application.
- A COM-compliant development environment, and should provide an ActiveX container in order to site the Alchemy ActiveX controls. Examples of this type of environment include Microsoft Visual Basic, Microsoft Visual C++, Sybase PowerBuilder, and Inprise C Builder. Users of the Alchemy SDK should be proficient in one of these languages and should have a good working knowledge of the Alchemy product.

Object Library overview

The objects can be understood by reviewing the following hierarchical overview of the functionality available to the programmer from the underlying database file structure. Programmers using C++ to write complex applications use these objects.



Application Object

The **Application** object is the top-level object in the Alchemy object model and contains and controls all other objects in the hierarchy of Alchemy objects. You cannot create additional **Application** objects, and the **Application** object is a property of other objects. Use the **Application** object to create and copy databases, execute batch commands, instantiate **Viewer**, **TreeView**, **Query** and **Builder** objects.

Databases Collection

The **Databases** collection contains all the open **Database** objects of the **Application** object. Add, remove or return a count of database objects from the **Databases** collection using standard collection methods: Add, Remove and Count.

Database Object

The **Database** object represents an Alchemy database. Use the **Database** object, its properties and methods to open, close, build or destroy an Alchemy database. May also be used to manipulate attributes and objects of an open database

Builder Object

The **Builder** object provides a way to add items to a new or existing Alchemy database and build or rebuild the database. Use the **Builder** object's methods to execute a build process, or apply an Alchemy DataGrabber definition file to an Alchemy database.

Groups Collection

The **Groups** collection contains all stored **Group** objects of a **Database** object. Add, remove or return a count of **Group** objects from the **Groups** collection using standard collection methods: Add, Remove and Count.

Group Object

The **Group** object represents a group of user accounts that have common access permissions for a given Alchemy Database. Use the **Group** object and its properties to invoke or to change the security password to an Alchemy database.

Fields Collection

The **Fields** collection contains all stored **Field** objects in a **Database** object. The **Fields** collection of a **Database** object represents the **Field** objects in a record. Add, remove or return a count of **Field** objects from the **Fields** collection using standard collection methods: Add, Remove and Count.

Field Object

The **Field** object represents a column of data with a common data type within a record. Use the **Field** object and its properties to retrieve the name, data type and indexing method information for a field in a given database.

Viewer Object

The **Viewer** object is used to display any information contained in an Alchemy database. Use the **Viewer** object and its methods to display any file either external to or contained within an Alchemy database. Use the **Viewer** object and its properties to control display of the toolbar, viewer pane, profile tab, and profile pane.

TreeView Object

The **TreeView** object displays a hierarchical list of items contained within an Alchemy Database. Each item can have a list of sub-items associated with it. Use the **TreeView** object to display all the elements of an Alchemy database, or a list of databases currently open in Alchemy.

Item Object

The **Item** object represents any entity contained in an Alchemy Database. An Item represents anything contained in a result object such as a folder, document or image. A parent-child relationship exists among the different items identified by the Folder and Has Children properties. Use the **Item** object's methods to

navigate the various folders in a **Result** object, create and delete Items, as well as launch applications associated with a given Item.

Items Collection

The **Items** collection contains all stored **Item** objects in a **Result** object. The **Items** collection of a **Result** object represents the **Item** objects from a qualifying Query. Add, remove or return a count of Item objects from the **Items** collection using standard collection methods: Add, Remove and Count.

Query Object

The **Query** object is used to define a specific query against an Alchemy database. Use the **Query** object's methods to create and execute Folder, Full Text and Profile queries and return results. Use Properties to Enable/Disable Ranked Results and Fuzzy Searching Functionality.

Search Group Object

The **Search Group** object is used to create search groups, which consist of multiple databases that can be queried through one query operation.

Result Object

The **Result** object represents the qualifying records returned from running a query against an Alchemy database. Use the **Result** objects' Properties to return a collection of qualifying **Item** Objects, a count of qualifying Items, as well as database and path information.

Results Collection

The **Results** object contains a collection of **Result** sets returned from executing a query. The **Results** collection of a **Query** object represents the qualifying records returned from the execution of a query. Add, remove or return a count of Result objects from the **Results** collection using standard collection methods: Add, Remove and Count.

Scan Object

The **Scan** object provides the ability to scan text and image documents into an Alchemy database. Use the **Scan** object to scan whole files, page subsets of a file, or a single page of a file. The object can only be used in conjunction with the Alchemy SCAN Module.

OCR Object

The **OCR** object provides the ability to perform optical character recognition on scanned documents. The **OCR** object can only be used in conjunction with the Alchemy SCAN Module.

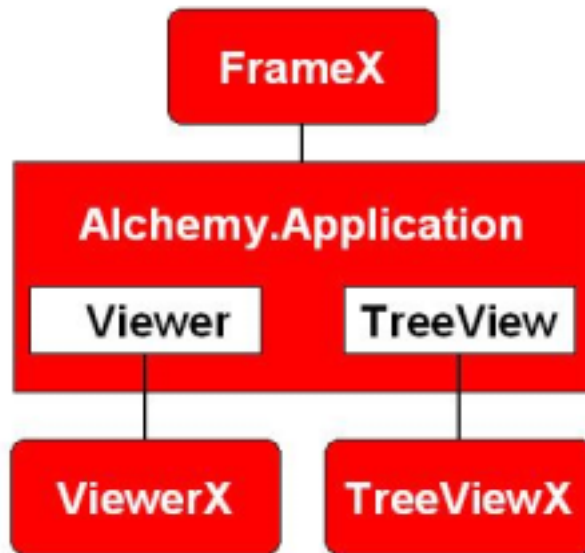
Progress Object

The **Progress** Object provides a visual update of tasks such as a database build, using the standard progress bar.

ActiveX control overview

Programmers and business analysts who are using Visual Basic to build customized applications access the ActiveX Controls.

ActiveX Controls



ActiveX Controls Available:

FrameX Control

The **FrameX** Control object is an ActiveX control representing the complete Alchemy application. Its methods and properties are the same as those of the **Application** object, and allow you to control the layout of all panes within Alchemy. Place an instance of the **FrameX** control on a Visual Basic form or other OLE-compatible container and it will manage the run-time creation and destruction of the object. Change and update standard properties at design-time, or modify **Application** properties at run-time.

ViewerX Control

The **ViewerX** Control is an ActiveX control representing the Viewer pane in Alchemy. Its methods and properties are the same as those of the **Viewer** object and allow you to control the layout and the interaction of the Viewer pane. Place an instance of the **ViewerX** control on a Visual Basic form or other OLE-compatible container and it will manage the run-time creation and destruction of the object. Change and update standard properties at design-time, or modify **Viewer** properties at run-time.

TreeViewX Control

The **TreeViewX** Control is an ActiveX control representing the TreeView pane in Alchemy. Its methods and properties are the same as those of the **Viewer** object and allow you to control the layout and the interaction of the TreeView pane. Place an instance of the **TreeViewX** control on a Visual Basic form or other OLE-compatible container and it will manage the run-time creation and destruction of the object. Change and update

standard properties at design-time, or modify **TreeView** properties at runtime.

IMR Support

Technical support for the Alchemy Software Developers Kit is available through an Incident Pack service. This service allows a developer to submit questions and code samples through email to a special support group within IMR. Details about this support program are available from your IMR reseller or from IMR at support@imrgold.com

Packaging

The Alchemy Software Developers Kit ships with a Programmers Reference Manual as well as code sample to help jump-start your special project. You must already have a license for Alchemy Gold, Premium or Pro as well as any other Alchemy add-on products that may be included in your final solution such as Alchemy SCAN or CAD. IMR offers a hands-on training program for the Software Developers Kit, which is a requirement for the purchase of the product. Current class schedules are available at <http://www.imrgold.com/en/training/>