

Document Manager 2.x

Architecture Overview

Created: Thursday, 6 October 2011

Last Modified: Thursday, 6 October 2011

Edited By Tim Cowell

Revision: 1

Copyright © 2011 Document Logistix Ltd. All rights reserved.

Version History

Version	Date	Author	Comments
1.0	6.10.2011	Tim Cowell	First release for external publication.

Table of Contents

DOCUMENT MANAGER 2.X	1
ARCHITECTURE OVERVIEW	1
VERSION HISTORY	1
TABLE OF CONTENTS	2
INTRODUCTION	5
FUNCTIONAL OVERVIEW	6
CORE FEATURES	6
CORE BUSINESS OBJECTS	6
<i>Database</i>	6
<i>Drawer</i>	6
<i>Folders and Subfolders</i>	6
<i>Document</i>	7
<i>Annotations</i>	7
<i>Storage Locations</i>	7
<i>User</i>	7
<i>User Group</i>	7
CORE PROCESSES / ACTIONS	8
<i>Scan and Store</i>	8
<i>Index and Search</i>	8
<i>Workflow</i>	8
SOME KEY QUESTIONS	9
CONCEPTUAL ARCHITECTURE	11
DOCUMENT MANAGER RICH CLIENT.....	11
CONFIGURATION DATABASE	11
CONTENT DATABASE.....	11
STORAGE LOCATIONS	12
API BASED SERVICES AND MODULES	12
<i>Import Service</i>	12
<i>Office Add-in</i>	12
<i>Sorting Office</i>	12
<i>Mail Archiver</i>	12
DATABASE BASED SERVICES AND MODULES	13
<i>Indexing Service</i>	13
<i>OCR Service</i>	13
TOKADMIN APPLICATION.....	13
DOCUMENT MANAGER WEB.....	13
PHYSICAL ARCHITECTURE	13
API - STATIC CLASS DIAGRAM	15
<i>TOKOPEN_ENVIRONMENT</i>	16
<i>TOKOPEN_VIEW</i>	16
<i>TOKOPEN_DOCTYPELIST</i>	16

TOKOPEN_DOCTYPE.....	16
PUBLIC ENUM TOKOPEN_DOCCLASS.....	17
TOKOPEN_ANNOTATIONLIST.....	17
TOKOPEN_ANNOTATION.....	17
TOKOPEN_POINTS.....	17
TOKOPEN_POINT.....	17
TOKOPEN_USERLIST.....	17
TOKOPEN_USER.....	17
TOKOPEN_IMAGELIST.....	17
TOKOPEN_IMAGEVIEW.....	17
TOKOPEN_GROUPLIST.....	18
TOKOPEN_GROUP.....	18
HOTKEYSCREENS.....	18
HOTKEYSCREEN.....	18
TOKOPEN_MEDIALIST.....	18
TOKOPEN_MEDIA.....	18
TOKOPEN_DOMAINLIST.....	18
TOKOPEN_DOMAIN.....	18
TOKOPEN_DRAWERLIST.....	18
TOKOPEN_DRAWER.....	19
TOKOPEN_WORKFLOWS.....	19
TOKOPEN_WORKFLOW.....	19
TOKOPEN_ACLLIST.....	19
TOKOPEN_ACL.....	19
TOKOPEN_CRITERIALIST.....	19
TOKOPEN_VIEWLIST.....	19
TOKOPEN_ERRORS.....	19
TOKOPEN_ERROR.....	19
TOKOPEN_DOCLIST.....	20
TOKOPEN_DOCUMENT.....	20
TOKOPEN_PAGELIST.....	20
TOKOPEN_PAGE.....	20
TOKOPEN_TEXTSEARCH.....	20
TOKOPEN_FIELDLIST.....	20
TOKOPEN_FIELD.....	20
TOKOPEN_PERMISSIONSCACHE.....	20
DESIGN - CONCEPTS AND PRINCIPLES.....	21
DATA STORAGE.....	21
AUDITING.....	21
<i>Archiving Audit Logs.....</i>	<i>21</i>
SECURITY AND PERMISSIONS.....	22
<i>Security.....</i>	<i>22</i>
<i>Permissions / Access Control.....</i>	<i>23</i>
TOKOPEN_PRIVILEGE.....	23
TOKOPEN_EXTENDED_PRIVILEGE.....	23
PUBLIC ENUM TOKOPEN_PRIVILEGE.....	23

END ENUM.....23

PUBLIC ENUM ENUMPRIVILEGETYPE23

 LICENSING 24

Named User License..... 24

Concurrent Use License 24

Manage Sessions..... 24

A LIST OF CURRENTLY LOGGED IN USERS CAN BE ACCESSED VIA THE TOKADMIN APPLICATION. IF REQUIRED, USER SESSIONS CAN BE ENDED FROM WITHIN THIS INTERFACE.24

Server License..... 24

 CONFIGURATION 25

TokAdmin 25

TokOpen.ini..... 25

 WORKFLOW 25

Code 26

 EXTENSIBILITY..... 26

COM API Callbacks..... 26

ScreenScrape/Hot Keys 26

APPENDIX A – EXAMPLE API USAGE XXVII

 'FOUND ACCOUNTS PAYABLE – TAKE A REFERENCE..... XXVIII

END WITH..... XXVIII

 'PRINT ALL DOCUMENTTYPE NAMES TO CONSOLE XXVIII

 'SET CRIT = NEW TOKOPEN_CRITERIA..... XXVIII

 'ADD FIELDS TO VIEW; IF YOU DON'T DO THIS NO DATA IS DISPLAYED IN THE VIEW XXVIII

 'EXECUTE SEARCH XXVIII

 IF VIEW.DOCUMENTS.COUNT = 0 THEN..... XXIX

Introduction

Document Manager (previously known as TokOpen) is a mature and successful Document management system. It has been extensively enhanced throughout the last 10 years, mainly in response to ever changing customer requirements. A high pace of development has been sustained during this time.

This document seeks to detail Document Managers functional scope, architecture and implementation.

However, the solution is vast, and to document it in entirety is beyond the scope of this exercise. A pragmatic approach has been employed. To make the task more manageable, a focus is made on those areas of the system that are most likely to be encountered during maintenance of the system and development of bespoke integrations. Essentially, this means that most attention is given to the API exposed by the system. The database schema is not covered in great detail.

The 2.x Document Manager product set is a Rich Client application (workstation installation) primarily developed in Visual Basic with a number of .Net components. It has a .COM API with object names prefixed 'TOKOPEN_' which is the original product name, and has been retained for API compatibility.

There is now also a Web based version of the product with a Web Service interface and .Net core services interface for integration purposes. The scope of this document is purely the 2.x product set.

Functional Overview

Document Manager is a scalable document management system that tracks and stores electronic documents and/or images of paper documents. It has the capability to index documents for search and retrieval. It also features a flexible workflow system that can be used to dictate how a document flows through an organisation. Additionally, Document Manager's integration framework enables streamlined business processes involving multiple systems. Documents are stored securely in the Document Manager system and access to them is controlled via a comprehensive permissions system.

Core Features

This section briefly introduces the core features of the Document Manager system. These features are categorised as either, business objects (data) or business processes (behaviour to manipulate data).

Core Business Objects

This section outlines the core business objects maintained by the system.

Database

The database stores reference data about where the documents stored in the system are located. It also stores the majority of other data for the Document Manager system, e.g. users, document annotations, permissions, index fields and values... Connection details for the Document Manager database server are persisted in the *TokOpen.ini* file located in the **Document Manager** installation folder. Document Manager can display multiple databases to users. This allows for content to be categorised at a high level, for example, by department or geo-graphical location.

Drawer

Drawers are analogous with filing cabinets; they contain any number of **Folders** and **Subfolders**, which are repositories for **Documents**. Drawers provide a convenient means of further categorising documents. Typically, Drawers are titled to represent business departments or high level processes, for example, 'Customer Services' or 'Accounts Payable'. Each Drawer may have a unique indexing structure, defining index fields for all folders and documents contained within it. To further refine the content stored, [Document Types](#) can be assigned for use within a Drawer. Drawers are permissionable entities.

Folders and Subfolders

Folders are akin to the suspension files found within a drawer in a filing cabinet. Which Index Fields are assigned to a Folder is determined at the Drawer level. All Folders within a Drawer share the same index fields. These fields provide an effective type description for Folders. So, for example, all Folders within an 'Accounts Payable' Drawer may have a 'Supplier Name' field. Therefore, each folder created within a Drawer represents a specific supplier and the documents contained within them will relate to that supplier.

Folders may have any number of **Subfolders** which in turn may themselves contain Subfolders and so forth recursively. Subfolders have the same index fields as parent folders. They provide a means

of subdividing Folder content. For example, if there were so many Folders in the 'Accounts Payable' Drawer as to be unmanageable, you may consider creating Subfolders to subdivide suppliers documents into Financial Years.

Folders and Subfolders are both permissionable entities. When created, Subfolders inherit their parent's permissions list.

Document

Documents within the Document Manager system can be any file type; Word document, TIFF, PDF, Binary etc... Documents 'inherit' index fields from the Drawer they are contained within. They are permissionable entities.

Document Manager implements the concept of **Document Types**. In Document Manager, a document type is a named class of document that has a number of parameters, some dependant on the file type (image, application doc, URL or ERM). Workflows are associated with document types. Document Types can have default access restrictions assigned to them. If Document Type access restrictions are specified they overwrite any restrictions inherited from the folder/drawer level.

Annotations

There are many annotation types available in Document Manager including, sticky notes, highlights and redactions.

Redaction is the term used to describe the action of blocking out portions of an image with a filled rectangle. This prevents underlying content from being viewed. In Document Manager, users cannot delete, move or hide redactions unless they are permissioned to do so. This is an important security feature of the system that prevents unauthorised viewing of personal or sensitive data. Redactions may also be associated with a particular document type. This allows redactions to be automatically applied to documents as they are created or viewed.

Storage Locations

Documents are not stored within the Document Manager database; they are stored on a storage device that is generally accessed via a UNC path. Also important to note is that fact that there is no 1:1 relationship between Document Manager drawers/folders/subfolders and folders on the file system. The storage locations within Document Manager are 'logical' and as such the documents contained within a single folder may physically reside in a number of actual locations. These physical locations (**Storage Locations**) are determined by Document Types but can be overridden at the Drawer level.

User

Users of the Document Manager system must authenticate before they can access its functionality. Documents, Folders and Workflows, Screen Scrape, Administration and Batch Import are all permissionable at a user level. Users may belong to one or more User Groups.

User Group

User Groups in Document Manager provide a convenient bucket into which one can place users who should share the same privileges. User Groups have access to the same range of potential privileges as Users. Privileges are binary, either granted or not, there is no 'deny' privilege. The effective

privileges for any user are the sum of the user's privileges and those of any User Groups they belong to.

Figure 1, below, shows the basic hierarchy of high level core business objects in the system.

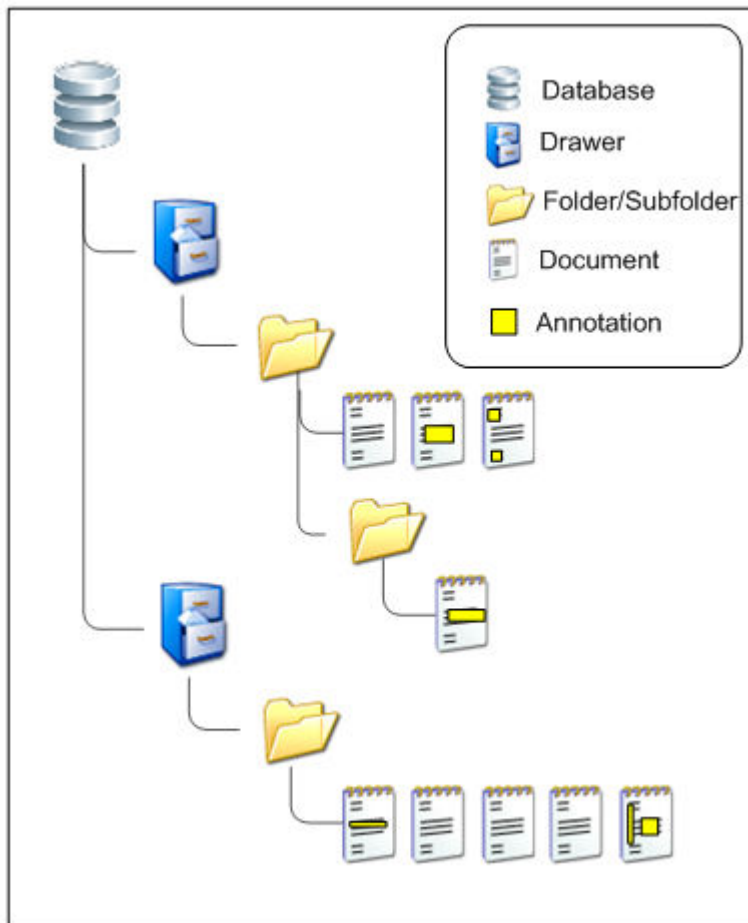


Figure 1: Document Manager Basic Concepts

Core Processes / Actions

Scan and Store

Document Manager can store electronic documents or images of paper based documents. Essentially, any format document may be stored in the system. Paper based documents are scanned into the system and stored in one of a number of standard image formats, eg, tiff or jpg.

Index and Search

Typically, as part of the document import process, a number of index fields are given values. For example, 'invoice number' or 'supplier name'. These fields can be searched in any combination to aid in retrieving documents. Additionally, an add-on module enables full text search capability.

Workflow

Workflow allows you to automate business processes involving documents. In Document Manager, workflows are associated with particular Drawers, Users and Document Types. Each action in the workflow may change the state of the document's index fields and cause email notifications to be raised or other actions to be performed (for example, move the document to another folder or

execute an external application). Conditions can be specified that control whether or not a workflow action may be performed. The combination of conditions and actions define the structure of the workflow; this is how state transitions are specified. Associating workflow actions with specific users, drawers and document types allows tight control over to whom, which documents and when a workflow is applicable.

Some Key Questions

There are a number of generic questions one may ask of any document management solution. This section poses these questions and supplies Document Manager's answer to each.

Where will documents be stored?

Documents can be stored on any network storage device accessible via a UNC path. (In actual fact, use of UNC paths is not enforced – rather it is a best practise suggestion. It is possible to specify a local path using a drive letter. Of course, this limits use of the solution to a single machine thus is generally only used in demo systems.) Users of the Document Manager system should normally have read/write access to these document storage paths.

Where will people need to go to access documents?

Documents may be accessed via the Document Manager thick client or the Document Manager web client. Each of these interfaces employs the Drawer/Folder/Sub Folder/Document structure outlined above.

What methods are used to organise documents to assist in later retrieval?

Documents in Document Manager are stored in logical Drawers and Folders. These serve to categorise documents. For example, all invoices could be stored in a drawer named 'Invoices'. Within that drawer, invoices could be subdivided by company name (with each folder representing a single company).

How will documents be indexed?

All Documents and Folders have some number of index fields associated with them. These fields are specified at the Drawer level of the repository hierarchy. Thus all folders within a drawer share the same index fields. The same is true of all documents with a drawer. Field values can be manually inputted or be automatically extracted from content via OCR. This index information is stored in the Document Manager database for use in search and retrieval. In addition, full text indexing is available for most document types.

How will documents be found?

Browse: Documents may be located by browsing the Drawer/Folder structure.

Search: Documents may be searched for by any index field value. Additional search criteria include, document type, drawer, permission and text (if full text indexing is enabled).

How are documents kept secure?

Outside of Document Manager, Documents are kept secure via the native features of the storage device employed to store them. For example, Window's folder security should be used to ensure that documents can only be accessed by the appropriate Document Manager users. Additionally, documents maintained by the Document Manager system may be stored encrypted. This added security prevents documents being read or modified if they are accessed by unauthorised personnel. Within Document Manager, access to documents is controlled via ACLs (Access Control Lists) that apply at the Drawer, Folder, Document and Annotation level. Users authenticate with the Document Manager system at system start up.

How can documents be recovered in case of destruction from fires, floods or natural disasters?

There are no specific features within Document Manager that cater for disaster recovery. However, the Document Manager documentation recommends best practise in this regard (ensure backups are made regularly and stored in a fire proof safe).

How can documents be preserved for future use?

There are no specific features that support document retention or archiving within Document Manager. These subjects are covered in the documentation and best practise guidelines are provided.

How are documents made available to the people that need them?

Documents can be retrieved by browsing or searching. Workflow actions can result in email notifications being sent to individuals required to progress the process. Saved searches can be configured to run periodically or on system logon. These are effective means of ensuring new documents are brought to the attention of the required individuals within an organisation.

If documents need to pass from one person to another, what are the rules for how their work should flow?

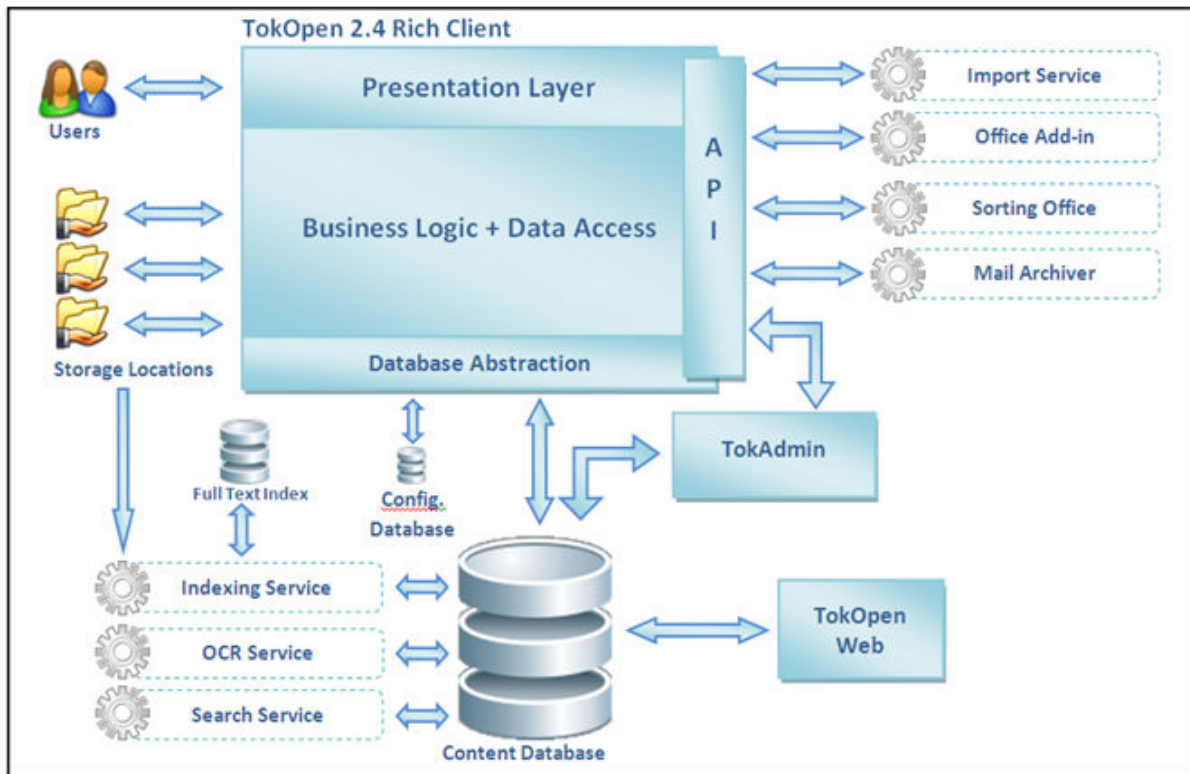
Document Manager's Workflow Actions can inform users that they should instigate the next action in a Workflow. Each action in the workflow possesses a list of permissible users that may execute it. Additionally, each action can be made reliant on a certain state (combination of index field values) before it may be executed. In this way, a complex workflow process can be produced, that ensures Documents flow through business processes in an efficient and consistent manner.

How are documents created?

All Documents within the Document Manager system must be associated with a Document Type. Documents can be created within a selected folder. When a Document is created the user must supply values for all required Index Fields. The required index fields presented depend on the Drawer within which the new Document exists. It is also possible to provide custom access permissions for a new document which will override those of the Folder/Drawer.

Conceptual Architecture

The following diagram provides a high level view of the Document Manager system:



Document Manager Rich Client

The core of the system, the Document Manager rich/thick client is logically architected as a 2-tier application. This consists of a Presentation layer coupled to a combined Business Logic/Data Access layer (BL/DAL). SQL command strings are persisted in code within BL/DAL. All SQL strings conform to the ANSI SQL standard. On top of the BL/DAL there is a veneer of database abstraction which executes SQL commands and ensures they are formatted correctly for whichever database engine is hosting the Document Manager database.

The majority of business objects and behaviours of the BL/DAL are exposed via an OLE server; this means that Document Manager may be automated from any programming language that understands COM interfaces, e.g., VB, C++, C#.

Configuration Database

The Document Manager configuration database stores details of all available Document Manager content databases. This database is also interrogated by the Document Manager system to determine the default database to connect to on start up. The Document Manager client can only be connected to a single content database at any one time.

Content Database

The Document Manager content database contains information pertaining to:

- Drawers
- Folders
- Workflow
- Document Types
- Users
- User Groups
- Privileges
- Domain Fields
- Common Fields
- Document Manager General Parameters
- Document Manager Web Settings
- Document Metadata
- Storage Locations

Note that the actual documents themselves are not stored within the database; refer to the section on [Storage Locations](#) for more information on how and where documents are persisted.

The Document Manager system may reference multiple content databases; however, only one open connection can be maintained at any time. Databases may exist in any of the supported RDMS: Microsoft Access, Microsoft SQL Server (versions 7 and later), MySQL (versions 4.0 and later) or Sybase SQL Anywhere (versions 5 and later).

Storage Locations

Documents are stored on a networked device that is accessed via a UNC path. Storage locations within Document Manager are 'logical' and, as such, the documents contained within a single folder may physically reside in a number of different actual locations. Logic to deal with availability of removable media is implemented in the system. When opening a document, a check is made to ensure that its storage location is available, if not, the user is informed of which media the document is stored on and where it should be loaded.

API Based Services and Modules

A number of services and modules extend the functionality of the core rich client. The following services do so via the Document Manager.exe OLE Server based API.

Import Service

The Document Manager Import Service (Windows service) enables automatic import of documents from one or more file locations against a pre-defined schedule.

Office Add-in

The Office Add-in enables documents to be opened from and saved to a Document Manager system.

Sorting Office

Provides rules based batch capture and indexing of documents into the Document Manager system. Sorting office enables automatic metadata identification and extraction, side-by-side manual entry of index fields, database lookups and barcode reading functionality. As a single offering, this functionality offers a powerful and efficient document assimilation capability to users of Document Manager. As such, Sorting Office features in most Document Manager installations.

Mail Archiver

This Windows service enables archiving of company email. Integration with e-Marc (an email archiving solution) enables copies of emails to be sent to Document Manager as they are sent or received. Emails are then stored securely in Document Manager and indexed for search and retrieval.

Database Based Services and Modules

A number of services interact directly with the Document Manager database:

Indexing Service

This service creates full text indexes of document content for search and retrieval. Indexing functionality is provided by the popular [dtSearch](#) search engine.

OCR Service

This service extracts text from image formats for indexing by the Indexing Service. The 3rd party Accusoft OCR Engine is used to provide optical character recognition functionality for the OCR service.

TokAdmin Application

The TokAdmin application provides a user interface to manage the Document Manager system. In general TokAdmin interacts directly with Document Manager databases, bypassing any BL/DAL. However, it does use the TokOpen.exe OLE Server API when managing document types and templates.

Document Manager Web

This is an ASP based web based client to the Document Manager system. It allows you to publish documents to the World Wide Web or an intranet. Users can search, retrieve, view and execute workflow on documents they are permissioned for without the need for the Document Manager rich client. Users are also able to edit document metadata (indexes) and upload documents.

Document Manager Web is intended to be superseded by the new .NET based thin client.

Physical Architecture

This section details the physical architecture of the Document Manager system. As the API is likely to provide the main focus for ongoing maintenance and bespoke development, the discussion focuses on the Document Manager Rich Client and its OLE API. It is considered that the main core of the Document Manager 2.4 product will remain relatively stable for the foreseeable future so I do not investigate the inner workings of the system to a great level of detail. It is expected that the 2.4 code will be deprecated in favour of the .NET incarnation 2-3 years from the time of writing (August 2008).

The Document Manager 2.4 solution is written primarily in VB 6 with some small portions in C#. The code base is mature, having been developed over the last 10 years. The magnitude of the code base is as follows:

- **Lines:** 237,690
- **Classes:** 214
- **Forms:** 170
- **Modules:** 71

As stated previously, the aim of this document is not to exhaustively detail all aspects of the architecture; the focus shall be on those areas that provide the majority of functionality and the public interface to the system.

In general, the code follows the 2-tier structure outlined in the [Conceptual Architecture](#) section above.

The next section details the objects exposed by the Document Manager OLE Server. Where deemed appropriate.

It is somewhat incomplete, but it remains accurate enough to be of use. Some sample code is provided within the API documentation.

An overview of the class structure of Document Manager's public API follows.

TOKOPEN_ENVIRONMENT

This is the top level object in the Document Manager object hierarchy. It contains references to the following collections:

- Users
- Groups
- Views
- Media
- document types
- Drawers
- Domains
- Images
- Screen scrape keys
- Searches

Collectively, these lists define a Document Manager environment.

The TOKOPEN_ENVIRONMENT class cannot be directly instantiated and should only be referenced via an instance of the TOKOPEN_VIEW class.

TOKOPEN_VIEW

The view class is the usual access point to the Document Manager environment. In addition to granting access to the Document Manager environment, each view may maintain criteria and results for a document search. A View can be made visible so that it appears in the Document Manager user interface or left invisible for API access only. The view object also exposes behaviour to create new documents, storage locations and document types.

The TOKOPEN_VIEW object utilises MultiUse instancing. To ensure that the view object can be properly disposed; you should call the Kill_Public_Instance method to remove your reference when you are finished working with the object. When you create a TOKOPEN_VIEW object, two references are created: one from the code that created the object, and one from the shell TokOpen.exe. When you destroy your reference to a view object, Document Manager still has its own reference, so the object remains in memory. This is why you must inform Document Manager to release its reference using the kill_public_instance method. If you create a TokOpen_View via the API and want to leave the view visible to the user in Document Manager **do not** use the kill_public_instance.

Finally, remember to add columns to your view. If you specify no columns, the view will not display any documents. This gives the mistaken impression that the view's search has returned no results.

TOKOPEN_DOCTYPELIST

The TOKOPEN_DOCTYPELIST is exposed via the [TOKOPEN_ENVIRONMENT](#) parameter of a [TOKOPEN_VIEW](#) instance. It is a collection of all document types available to the system (expressed as TOKOPEN_DOCTYPE objects).

TOKOPEN_DOCTYPE

This class represents an instance of a Document Manager document type. It defines the following parameters: name, document class, extension, encryption settings, storage locations and redactions.

The TOKOPEN_DOCCLASS enumeration is used to specify what kind of file is associated with a Document Type. The document class determines which parameters are associated with a document.

For example, Application documents have a file type and an optional document template associated with them, whereas Image documents have redaction and image format settings.

Note that Sub Folders are Documents with a document class of 'Container'. The TOKOPEN_DOCCLASS enumeration follows:

```
Public Enum TOKOPEN_DOCCLASS
    Application = 0    'ie Word document
    image       = 1    'ie TIFF File
    Container   = 2    'ie A Folder
    NotSupported = 3    'ie Deleted
    Exturl      = 4    'external URL
    ERMDocument = 5    '.CLD document
End Enum
```

TOKOPEN_ANNOTATIONLIST

A collection of TOKOPEN_ANNOTATION objects. Each collection member describes a single TOKOPEN_ANNOTATION applied to a document.

TOKOPEN_ANNOTATION

Instances of this class describe the position, appearance, and permissions for a single annotation for a specified document/document page.

TOKOPEN_POINTS

A collection of TOKOPEN_POINT objects; used to specify the location and size of a TOKOPEN_ANNOTATION.

TOKOPEN_POINT

A single XY co-ordinate used to specify a point on screen.

TOKOPEN_USERLIST

Accessible via an instance of the TOKOPEN_ENVIRONMENT class; instances of this class maintain a read only snapshot of all users of the Document Manager system. This snapshot can be updated by calling the 'refresh' method which will query the Document Manager content database and update the collection. The TOKOPEN_USERLIST class also contains the usual accessor methods.

TOKOPEN_USER

This class defines simple properties for a Document Manager user (Id, Short Name, Long Name, Email 1 and Email 2). The Id property is used throughout the API to reference users.

TOKOPEN_IMAGELIST

Accessible via an instance of the TOKOPEN_ENVIRONMENT class; the TOKOPEN_IMAGELIST specifies a collection of TOKOPEN_IMAGEVIEW objects.

TOKOPEN_IMAGEVIEW

This class represents a single page (image) from an image based document. It maintains a reference to the TOKOPEN_DOCUMENT object it forms part of and an 'frmimage' form which handles the actual image rendering.

TOKOPEN_GROUPLIST

A collection of TOKOPEN_GROUP objects. This collection is accessed via the TOKOPEN_ENVIRONMENT object.

TOKOPEN_GROUP

Objects of this class represent an instance of a Document Manager user group. This class contains the expected parameters: Name, Description, Members, and Permissions. In addition, this class also exposes a handy Boolean parameter 'am_a_member' which specifies whether the currently logged on user is a member of the group or not. This value is specified as the TOKOPEN_GROUPLIST collection is initialised or refreshed. User Groups are persisted in the **Groups** table of Document Manager database.

HOTKEYSCREENS

A collection of HOTKEYSCREEN objects. This collection is accessed via the TOKOPEN_ENVIRONMENT object reference of a TOKOPEN_VIEW instance.

HOTKEYSCREEN

Hot keys provide access to screen scrape functionality. The class specifies parameters required to locate the window and fields to be scraped.

Screen scraping is the method by which Document Manager executes a search based on data displayed in some other application. For example, from within a human resources (HR) application, you may wish to quickly search for all Document Manager documents related to a particular employee. With the HR application open, the user would press the pre-defined Hotkey; Document Manager could intercept the key press and execute a search based on an employee payroll number scraped from the HR application.

TOKOPEN_MEDIALIST

Storage Locations for documents are defined by the TOKOPEN_MEDIALIST collection; it is a collection of TOKOPEN_MEDIA objects.

TOKOPEN_MEDIA

This class defines the properties required to access a Storage Location.

TOKOPEN_DOMAINLIST

A collection of TOKOPEN_DOMAIN objects.

TOKOPEN_DOMAIN

In TokOpen, a domain is a globally available enumeration e.g., a 'Department' domain could contain values: 'Human Resources', 'Finance', 'Sales' and 'Development'. Note that the TOKOPEN_DOMAIN class does not actually contain its values, rather it provides access to Id, Name, Description and Status parameters. Use the property `GetDomainValues(DomainName As String) As Collection` method of the TOKOPEN_DOMAINLIST class to obtain a Domain's values.

TOKOPEN_DRAWERLIST

A collection of TOKOPEN_DRAWER objects. This collection is accessed via an instance of the TOKOPEN_ENVIRONMENT class.

TOKOPEN_DRAWER

Instances of this class define the properties of a single Drawer in the Document Manager system. These properties include: index fields, permissions and associated workflows. Note that the TOKOPEN_DRAWER class does not provide direct access to the folders contained within it. Documents and Folders are accessed via a TOKOPEN_VIEW according to specified TOKOPEN_CRITERIA.

TOKOPEN_WORKFLOWS

A collection of TOKOPEN_WORLFLOW objects. This collection is accessed via a TOKOPEN_DRAWER object reference. Also contains helper functions that list valid workflows for a document or create a context menu for it.

TOKOPEN_WORKFLOW

An instance of this class represents a single workflow associated with one or more TOKOPEN_DOCTYPES. The `Apply_to_Document` method executes the workflow action against a supplied TOKOPEN_DOCUMENT.

See [Workflows](#) section for more detail.

TOKOPEN_ACLLIST

A collection of TOKOPEN_ACL objects. This collection is used by the TOKOPEN_DRAWER and TOKOPEN_DOCUMENT classes to defines the access control for instances of these classes.

TOKOPEN_ACL

This class provides a container for a set of privileges. The `set_my_rights` method is used to OR together two sets of privileges.

TOKOPEN_CRITERIALIST

The TOKOPEN_CRITERIALIST class is a collection of TOKOPEN_CRITERIA which define an index field, its data type, valid values and current value. It is typically used for two main purposes, to determine the indexing structure for a drawer, and for creating a list of criteria to perform a search.

TOKOPEN_VIEWLIST

A collection of TOKOPEN_VIEW objects. This collection is accessed via the TOKOPEN_ENVIRONMENT object accessed from a TOKOPEN_VIEW instance. Note that new TOKOPEN_VIEW's add themselves to the TOKOPEN_VIEWLIST when created. Most windows in the Document Manager system (explorer, search results, etc) present the content of a view.

TOKOPEN_ERRORS

A collection of TOKOPEN_ERROR objects. Any errors that occur during an operation relating to a TOKOPEN_VIEW are stored in a global instance of this class. This allows suitable error messages to be displayed to the user should an exception occur during execution.

TOKOPEN_ERROR

This class provides a simple datastructure that describes a single error message.

TOKOPEN_DOCLIST

A collection of TOKOPEN_DOCUMENT objects with extensive behaviour to manage each document in the collection. For example, you can use the TOKOPEN_DOCLIST to check documents in/out, delete them, mail or print them etc...

TOKOPEN_DOCUMENT

This class represents a single document within the Document Manager system. If the TOKOPEN_DOCUMENT's class is Image(1) then it will populate a TOKOPEN_PAGELIST collection with a scanned image for each page of the document. The class exposes similar document management functionality as the TOKOPEN_DOCLIST class. Internally, these functions are implemented by creating a temporary TOKOPEN_DOCLIST, adding the TOKOPEN_DOCUMENT to it and calling the relevant function, eg:

```
Set doc = New TOKOPEN_DOCUMENT
Set doclist = New TOKOPEN_DOCLIST
doclist.Add , doc
Action_Print = doclist.Action_Print(showprintdialog)
```

Reminder: Folders and Sub folders are Documents of type 'Container'.

TOKOPEN_PAGELIST

This class exposes a collection of document pages. The TOKOPEN_PAGELIST is accessed via an instance of the TOKOPEN_DOCUMENT class. However, if the TOKOPEN_DOCUMENT does not yet specify opened a document, its page list will be empty.

Note that this collection is only populated for Documents of type 'image'.

TOKOPEN_PAGE

The class specifies the file containing a single page image. It also specifies display properties and a key to decrypt the page file if encrypted.

TOKOPEN_TEXTSEARCH

This class specifies and executes a full text search. Search results are returned via 3 collections, filenames (names of files matched), hitcounts and words (the words from the search string matched in each file).

TOKOPEN_FIELDLIST

The TOKOPEN_FIELDLIST class is a collection of TOKOPEN_FIELD objects. It is generally used to specify which columns are displayed in a view.

The TokOpen_Fields enumeration represents to all Document Manager system fields

TOKOPEN_FIELD

This class represents a single field within the Document Manager system. Fields may be specific to a Drawer or common to many. Common Fields, shared across multiple drawers, enable cross-drawer searching. Each drawer may have up to 10 document fields and 20 folder fields.

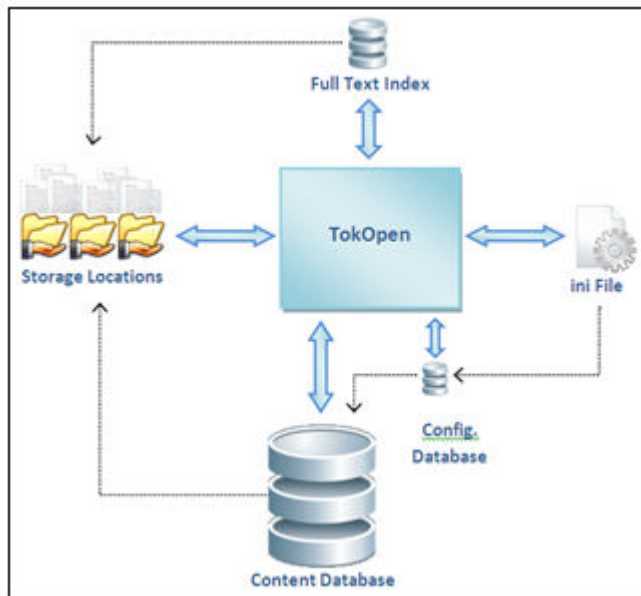
TOKOPEN_PERMISSIONSCACHE

This class is used to hold details of permissions for an object. This improves performance by preventing the system from having to re-read the permission list each time it has to test user rights.

Design - Concepts and Principles

Data Storage

The following diagram depicts where data is persisted within the Document Manager system.



Document Manager Data Storage

The Document Manager system stores the majority of its data in a main content database. Of course the notable exception is that no documents are stored within the database. Rather a reference to each document's network location is persisted. The configuration database provides connection details for all available content databases. TokOpen's ini file specifies the connection details for the configuration database. It also stores parameters specific to an instance of the thick client. Finally, if the feature is enabled, a full text index is persisted separately from other Document Manager content.

Auditing

The Document Manager and TokAdmin programs write detailed audit logs of user activity. For example, if a user edits a document or folder, or the administrator creates a new user, information about these events is persisted. These audit records are stored in the content database. Events are categorised into one of four types, *folder*, *document*, *annotation* or *user*. Document Manager maintains a separate database table for each of these audit event types. These tables share a largely similar schema which describes who did what, when and from where. For example, the audit can express events of such as 'Joe Soap deleted /drawer1/folder1/document1 on 12.12.2005:14:30. Joe was logged into the system from workstation COMP001'.

Archiving Audit Logs

It is possible to archive audit logs. Archived logs are stored as documents within the Document Manager system. The **Audit** Document Type is used when creating audit archive files. *Audit* is a built-

in read-only Document Type. Documents of this type cannot be deleted or moved once created. Archiving audit logs allows you to reduce the size of the 'live' audit in the database and hence improve its performance. It is recommended that the Audit Document Type is configured to encrypt documents. This prevents unauthorised reading and modification of audit logs.

Security and Permissions

Security

User permissions percolate through 5 layers of privilege. These layers correspond to the [business object hierarchy](#) presented earlier in this document:

- Database (essentially application level access)
- Drawer
- Folder and Sub Folder
- Document
- Annotation

It is helpful to think of each of these layers as containers. The Database is a container for Drawers, which contains a number of Folders which contain Sub Folders and so on. For a permission to be effective, it must be granted at all higher layers. For example, to grant a user 'Modify' rights to a Folder, you must also grant the same rights to the Drawer containing the folder and the top level Database.

Users will only see Drawers, Folders, Documents and Document Content they have permissions to. For example, if a user does not have the rights to view a Drawer, it will not appear in the Document Manager client. Within a Drawer the user only sees those Folders they have been granted access to. Access to Documents and Annotations is achieved in the same manner.

Document Security and Storage Locations

From within the Document Manager client, access to documents is governed by the security/access control model outlined above. However, this document security is bypassed if they are accessed directly through the file system. The capabilities of the file system should be used to control access to these document locations. Best practise suggests that ACLs to these locations should mirror those specified within TokOpen. Additional protection can be achieved by configuring Document Manager to encrypt Documents. This ensures that even if direct access is made to documents, they cannot be read or modified.

Encryption

As mentioned above, Document Types can be configured to be stored on disk encrypted. The following encryption algorithms are made available by default: DES, RC2 or Rijndael. Alternatively, you can supply your own encryption as a COM component that implements certain methods and properties.

Encryption is used to protect sensitive data throughout the system. Database connection strings, user names and passwords are persisted encrypted. The Blowfish algorithm is utilised to encrypt data of this type.

Permissions / Access Control

Document Manager expresses all available privileges using four enumerations:

```
TOKOPEN_PRIVILEGE
TOKOPEN_EXTENDED_PRIVILEGE
TOKOPEN_ANNOTATION_PRIVILEGE
TOKOPEN_EXT_ANNOTATION PRIVILEG
```

For example:

```
Public Enum TOKOPEN_PRIVILEGE
    DOCUMENT_CREATE      =    1
    FOLDER_CREATE        =    2
    BATCH_ADMIN          =    4
    DOCUMENT_OPEN        =    8
    OVERLAY_CREATE       =   16
    DOCUMENT_MODIFY      =   32
    FOLDER_MODIFY        =   64
    ....
End Enum
```

Each enumeration represents a logical set of related privileges. Note that values within these enumerations are represented by series where each number is a factor 2 higher than the previous (eg. 1, 2, 4, 8, 16, 32, 64...).

ORing all granted permissions and expressing the answer as a binary number creates a ‘bitmap’ representing a user’s rights. With reference to the TOKOPEN_PRIVILEGES enumeration above, if a user were granted DOCUMENT_CREATE and FOLDER_CREATE privilege, they would have a user rights value of 3. Or, expressed in binary: 11. To determine if a user has a specified privilege, one merely needs to AND the required privilege with the user’s granted privileges; a result of 1 means that the user has been granted that privilege. For example, continuing the above example, to test that the user has the DOCUMENT_CREATE privilege we AND the granted privileges (11) with the requested privilege (01): $11 \text{ AND } 01 = 1$. The requested permission is granted.

Effective Rser rights are determined by an OR of the user and group granted privileges.

As mentioned above, the TOKOPEN_ENVIRONMENT class holds enumerations representing all available privileges.

User rights are maintained by the system as a four element array of type `long`. Each element in this array represents the privileges granted from one of the 4 privilege enumerations above. The PRIVILEGETYPE enumeration ensures consistent access into the array:

```
Public Enum enumPrivilegeType
    NORMAL_PRIVILEGE      =    1
    EXTENDED_PRIVILEGE    =    2
    ANNOTATION_PRIVILEGE  =    3
    EXT_ANNOTATION_PRIVILEGE =    4
End Enum
```

The `Have_Rights_To` method in the [TOKOPEN_PERMISSIONCACHE](#) class provides functionality to test user permissions against specified business objects. Note that the ‘Refresh’ method is never actually called.

Licensing

Each Document Manager database requires a separate server license.

There are two type of client license in TokOpen, **Named User License** and **Concurrent Use License**. A Document Manager database may implement one or other of these models; mixed authentication is not possible.

Named User License

Licenses are assigned to specific login names. This user type may only maintain one open session. With this model, there is a 1:1 relationship between users and licenses; it is typically used when there are fewer users of the system and/or clients want to guarantee users access to the system.

Concurrent Use License

Licenses are not assigned to any particular user. This type of license is suitable if there are many users, but they use Document Manager infrequently. Employing this license model can be cheaper than a **Named User** implementation.

Manage Sessions

A list of currently logged in users can be accessed via the TokAdmin application. If required, user sessions can be ended from within this interface.

Server License

Server licenses are persisted as an encrypted string saved to file with the 'lic' extension. License files are imported into the Document Manager database at install time or via the TokAdmin application post install.

When a server license is installed, the license file is decrypted and the individual license options are written to the `toklicense` table in the content database. An encrypted form of the license is also stored in this table. License details can be viewed in TokAdmin:

The screenshot shows a 'License Information' dialog box with the following fields and options:

- Licensed To: Nick Savva
- License Type: NETWORK
- Serial No.: 11230
- Full Users: 25, View Users: 25, Workflow Users: 25
- Named (radio), Concurrent (radio selected)
- Sorting Office Users: 5
- Expiry Date: 08 July 2009
- Expiry Message: (empty)
- Version: 2.5, BETA features Enabled (checkbox)
- Optional Features:
 - Workflow (checked), Version Control (checked), Desktop Scanning (checked), LDAP (unchecked)
 - Auditing (checked), Enhanced Admin (checked), Encryption (checked), ObjServer (unchecked)
 - Checkout (checked), Screen Scrape (checked), Advanced Annotations (checked), CD Viewer (checked)
 - Text Search (checked), Web Sites (checked), Web Data Export (checked), TokAlerter (checked)
- Tokopen Lite Restrictions:
 - Users: (empty), Max Pages/Objects: (empty)
 - Drawers: (empty), Workflow Actions: (empty), Saved Searches: (empty)
- Reseller Information:
 - Company Name: Infonic plc
 - Telephone: 01908 366 388
 - Email: support@infonic.com
 - Website: http://www.infonic.com

Document Manager 'View License' Dialog

Configuration

TokAdmin

The bulk of system configuration is conducted via the TokAdmin application. This application provides a user interface to manage the Document Manager system. In general TokAdmin interacts directly with Document Manager databases, bypassing any BL/DAL. However, it does use the TokOpen.exe OLE Server API when managing document types and templates.

TokOpen.ini

Connection details for the Document Manager database server are persisted in the **TokOpen.ini** file located in the **Document Manager** folder of the installation. This file also includes parameters specifying:

- Licensing (Licensee and license type)
- Last user logged in details (user name and window size)
- Appearance (Theme appearance,)
- OCR and ERM settings

Workflow

Workflow allows you to automate business processes involving documents.

In TokOpen, Workflows are assigned to particular Drawers and associated with certain Document Types and Users. Each Workflow within a Drawer is termed a 'Process'. Processes are made up from one or more actions. Actions are the state transitions of a workflow. They are comprised of a number of parameters:

- General settings:
 - Name
 - Description
 - Text to show users in pop-up menu
 - Password to execute (optional)
 - Drawer to associate with
 - Process to associate with
 - Enabled/Disabled
- Permissions
 - Users permitted to execute action
 - User Groups permitted to execute action
- Document Types to which action is applicable
- Conditions that must be satisfied before action is accessible via UI
 - Index field criteria
- Updates to be applied index fields when action is executed
- Email notifications to send when action is executed
- Other actions to perform when action is executed

Code

Workflow Conditions specify the order in which Workflow Actions are displayed in the Document Manager UI. These conditions are expressed as `TOKOPEN_CRITERIA` in code.

The `TOKOPEN_WORKFLOWS.MakeWorkflowMenu` subroutine builds a context menu for a specific document depending on its Document Type and Workflow Conditions met. This context menu will display all available workflow actions for a Document.

`TOKOPEN_WORKFLOWS.List_Valid_Workflows` returns a collection of Workflows valid for some supplied Document. The function ensures the Document's Document Type and Index Fields meet the criteria for each Workflow.

The `TOKOPEN_WORKFLOW` class represents a Workflow Action. It contains all the data and behaviour to represent Actions and apply them to valid Documents. Use the `Apply_to_Document` function to execute a Workflow Action on some supplied document.

Extensibility

One of the driving requirements for Document Manager and a key factor in the application's success has been the aim of making it easy to extend to and integrate with other systems.

COM API Callbacks

Document Manager can execute custom code in response to the following events:

- Document Creation
- Document Action (Modify, print, etc)
- Screen Scrape

It is also possible to insert an additional menu item to the context menu for Documents and Folders.

This custom code is supplied to Document Manager via specified COM components that implement certain subroutines and functions which Document Manager calls as appropriate.

See Appendix 5 of the TokAdmin User Manual for additional detail.

ScreenScrape/Hot Keys

Screen scraping is the method by which Document Manager executes a search based on data displayed in some other application. See [HOTKEYSCREENS](#) and [HOTKEYSCREEN](#) for more detail.

Appendix A – Example API Usage

The following code example requires the sample database distributed with Document Manager 2.4. The code creates a new TOKOPEN_VIEW instance to gain access to the TOKOPEN_ENVIRONMENT. It iterates the environment's Views, Drawers and Document Type collections and performs a search and sets the new View instance to true. Remember to add a reference to TokOpen.exe before running this code sample.

```
'Use early binding when developing your application.
'Late binding is recommended as this will ensure your app works with all
'versions of TokOpen

'TOKOPEN_VIEW is your access point into the API. Always create with first
Dim view As Object
'Dim view As TOKOPEN_VIEW

'TOKOPEN_CRITERIA used to specify search criteria for the TOKOPEN_VIEW
Dim crit As Object
'Dim crit As TOKOPEN_CRITERIA

'Drawer Identifier
Dim drwr As Long
'Drawer Identifier
Dim dt As Long

Dim loggedon As Boolean
Dim fld As Long
Dim vw As Long

Dim apdrawerid As Long
Dim invnofldid As Long

'Create you TOKOPEN_VIEW
Set view = CreateObject("TOKOPEN.TOKOPEN_VIEW")
'Set view = New TOKOPEN_VIEW

'Authenticate with the Document Manager system.
loggedon = False
If view.Loggedin = False Then
    view.login "", ""
    If view.Loggedin Then loggedon = True
End If
If view.Loggedin = False Then
    MsgBox "you must login"
    'must kill public instance so that Document Manager knows to dispose your
view
    view.Kill_Public_Instance
    Set view = Nothing
    Exit Sub
End If
```

```
'Iterate all Views open in Document Manager and print names to console
For vw = 1 To view.Environment.Views.Count
    Debug.Print view.Environment.Views(vw).Viewname
Next vw

'Iterate all Drawers in Document Manager environment and print names to console
For drwr = 1 To view.Environment.Drawers.Count
    Debug.Print view.Environment.Drawers(drwr).drawername

    'We're interested in the accounts payable drawer...
    If LCase(view.Environment.Drawers(drwr).drawername) = "accounts payable"
Then
    'Found accounts payable - take a reference
        apdrawerid = view.Environment.Drawers(drwr).drawerid
    End If
    'Print Drawer structure (index fields) to console
    With view.Environment.Drawers(drwr)
        For fld = 1 To .structure.Count
            'Get invoice no. field id
            If view.Environment.Drawers(drwr).drawerid = apdrawerid Then
                If LCase(.structure(fld).Fieldtitle) = "invoice no" Then
                    invnofldid = .structure(fld).fieldname
                End If
            End If
            Debug.Print vbTab & .structure(fld).Fieldtitle & vbTab &
.structure(fld).fieldname & vbTab & .structure(fld).dbFieldName
        Next fld
    End With
Next drwr

'Print all DocumentType names to console
For dt = 1 To view.Environment.DocTypes.Count
    Debug.Print view.Environment.DocTypes(dt).ShorthandName
Next dt

'search for an invoice no
If apdrawerid <> 0 And invnofldid <> 0 Then
    'we found the expected Drawer and Index Field
    'set Drawer ID and viewtype to 'locate_document' (search results)
    view.drawerid = apdrawerid
    view.ViewType = locate_document

    'Set crit = New TOKOPEN_CRITERIA
    'Specify search criteria by creating a new TOKOPEN_CRITERIA insance
    Set crit = CreateObject("TOKOPEN.TOKOPEN_CRITERIA")
    crit.fieldname = invnofldid
    crit.Condition = EqualTo
    crit.FieldValue = "0001078"
    'add new criteria to the views TOKOPEN_CRITERIALIST
    view.criteria.Add crit

    'Add fields to view; if you don't do this no data is displayed in the view
    view.Fields.Add doc_doctype
    view.Fields.Add documentdescription

    'Execute search
    view.DoSearch
```

```
    If view.Documents.Count = 0 Then
        MsgBox "Sorry no results"
    Else 'we a result - display the view!
        view.Viewname = "Invoice 0001078"
        view.Visible = True
        Exit Sub
    End If
End If

'exit code.
If loggedon Then view.logout
view.Kill_Public_Instance
Set view = Nothing
```