XYZ Corporation

KLM Replacement
Architecture and Functional Specification

Version 1.1

Revision History

Date	Version	Description	Author
27 March 2002	1.0	First draft	Someone
4 April, 2002	1.1	Updated database diagram and functional spec sections from David and Jim	Brian Loomis

Table of Contents

KLM Replacement
Architecture and Functional Specification

1. Introduction

This document describes the high-level architectural components of the KLM replacement system, standards applicable to the development, technology selection, deployment specifications, and other system characteristic notes (security, caching, etc.). This document serves as a jumping-off point for subsystem (component) functional specifications that provide developer-level documentation.

1.1 Purpose

Conceptual, or logical, business objects are captured through requirements analysis and are refined into designed, or physical, classes. Typically a single logical business object will have a corresponding class in each of the presentation services, business services, and data services tiers at implementation. After the logical object model is a drafted, separate diagram plus interaction diagrams for each tier are designed and will be maintained in the project design folder. All objects will be grouped into libraries (DLLs) or assemblies after considering performance implications, network deployment, and other architectural aspects.

For logical design documents of the KLM replacement application, please refer to the Visio diagrams in the project documentation folder. The database schema diagram is also contained in a separate Visio diagram.

1.2 Scope

The sections below describe the related documentation on this project, the terminology used when describing system attributes, and then contains sections describing the deployment and functional specification (design) of each tier of the application (user/presentation services, business services, and data services).

1.3 References

The following documents serve as background reference for this project (stored on the SharePoint Team Services site at \\ppoqmcclure\sharepoint.htm and in \\kioscdocs\kiosc :

- Project vision document (KLM Vision.doc)
- Master schedule and project plan (KLM Replacement Main Project Plan.mpp)
- System requirements document (KLM System Requirements.doc)
- System architecture document (KLM System Arch.doc)
- System development plan (KLM System Development Plan.doc)
- System design diagram (Design Diagram.doc)
- Logical design model (KLM Logical Model.vsd)

The following terms are used throughout the document:

Applications:

WYSIWYG Editor – the user interface application within KLM replacement that allows authoring of labels (creation and editing) as well as Manual Print functionality.
Manual Print – the user interface application within KLM replacement that allows only searching and printing of labels.
Print engine – a subsystem (component) of KLM replacement that assembles parameterized labels from the repository and user inputs and manages the output to a specific printer instance.
Converter – the application within KLM replacement that migrates existing label data into the KLM

replacement database.

Labels:

Label – the simplest printable unit within the KLM replacement repository. A label may have a title, associated graphics and metadata. Basic operations on labels include creation, creation from an existing label (cloning), viewing, editing (update), saving, printing, reprinting, deleting, and archiving.
Label template – same as Label. The distinction between a template and a label instance is that the template is the basic information for any instance of a label; when the label is printed, the specific information for that label is inserted in the audit log (the template remains unchanged).
Thumbnail – a smaller version of the final printed label as it would appear on the screen before printing. For performance reasons, the thumbnails may be pre-generated and stored in the repository.
Graphics – a file or files associated with a label and required for its printing. The graphics will have the raw bitmap(s) as well as metadata such as searchable keywords and a title.
Label Metadata – descriptive information about a label. Metadata includes searchable keywords and categorization information (a hierarchical associative list), label stock size, version information and tags, access permissions, BOM part number, status (?), a flag for whether to track reprints or not, reference to owning repository (owner database for cache management), product line site and station identifiers, prompts, check-in/checkout information (user id), and product association (mapped products). Note that metadata is not a single table in SQL Server and will most likely be distributed through a variety of lookup tables.
Prompts – descriptions of information required or optionally supplied by the user when physically printing a label. Prompts may be static text, require the user to enter optional or mandatory data, or may be an automatically supplied value requiring user confirmation. For example, a serial number may be automatically generated based on sequence in a series, the part number may be supplied from a product association, a third prompt may supply a static text string.
Serial number – a type of prompt where a number is generated and formatted based on product association requirements and confirmed by the user at print time. Serial numbers may be compound fields composed of static text, incrementing values (integer, hexadecimal, or other with a specified starting seed value or block number for an assigned site/user), or user-entered text.
Label Stock – the size and dimensions of a specific paper stock used for printing labels. The stock also includes the vendor identification and type. For example, a particular label may be printed on stock of size 5.5” W x 6.8”L from Avery on form 2280.
Version – the revision of a given label including information such as tags (“PRODUCTION VERSION”, “DRAFT”, “APPROVED”) and a sequential number (version 1, 2, 3, etc.).
Audit Log – the recorded capture of system events including the printing of labels (and actual parameters), and reprints. Other events include security events.

Printers:

Printer type – the list of printer types (Zebra, generic PostScript, etc.) and relevant parameters for access (network-aware, resolution, etc.).
Zebra printer – a printer capable of understanding the ZPL format
Generic printer – a printer capable of printing PostScript or generic List and Label format that generally does not understand ZPL.
Printer instance -- A specific printer instance will have a printer type plus values for each parameter; for instance, “Printer 56 is a Zebra printer on the network at \\myZServer\ZPrinter1 with a resolution of XYZ.”
Print format type – the print specification and representation for a label or print job.
ZPL format – a format type understood by Zebra printers.
LBL format – a format type understood by generic printers.

Users and security:

User – a set of credentials to access the KLM replacement applications; may consist of Windows userid and password or an external application name. Also includes information on how to contact the user, such as phone number or email. Users are logically collected in groups, belonging to a site, and users have roles that they can be assigned to.
Role – the job functions within the KLM replacement system that a user or group can perform (maps to a group of permissions). Example roles may include label printing only (print-only user), engineer/design labels, create labels (label authoring), administrate system (administrator), external application integration (system user). Each role will enable or disable specific features of the application depending on the permissions allowed to the role; for example, label authoring may enable the menu items in the WYSIWYG editor to create, edit and save labels.
Group – a collection of users or other groups for administrative purposes. Groups can be assigned roles. Note that the most broad set of permissions may apply to a given user, so that a user in the label authoring role and in a group that only has label printing role will get authoring permissions
Permission – a basic operation on a label (such as creation, editing, saving, printing). Permissions are assigned to roles so that a role may contain 1 or more permissions and then users and groups can be assigned to the role.
Repository – a data storage mechanism (file system, database, etc.) that provides labels and associated details to appropriate users. Repositories may be local on a particular machine (e.g., a local XML file cache) or may be centralized on a SQL Server installation. Repository attributes include a repository name, unique identifier, and whether the repository can serve as a master repository to other repositories (e.g., whether users can create new or edit existing labels in this repository).
Site – same as Repository.
Subscription – a method for one repository to reference updated labels in a separate repository. Subscription attributes include the method of subscription (push or pull), which items are available through the subscription, and the method that updates are sent (SQL DTS, email, FTP, etc.). For example, the Panang repository may subscribe to the Colorado Springs repository for all labels as a push subscription (on-change) via SQL DTS.

External applications:

External application – any application that provides data to or receives data from the KLM application.
KTime – the time provider application within KIOSC.
KBusinessRules – the custom logic provider within KIOSC for serial number validation.
KSecurity – the security module within KIOSC.

2. System Characteristics

This section discusses general architecture characteristics that apply to the entire system and may be used in many parts of the functional specification. Specifically, this sections discusses the versioning of labels, the security infrastructure, data formats of labels in the database, and multiple repository/caching approaches.

2.1 Versioning of Lables

Labels may exist in multiple versions (or revisions). Each version may represent a particular status of the label (approved for production, not approved, pending approval), or be applicable to a particular set of part numbers (e.g., part number revisions 1-5 on product XYZ get version 4 of the label, whereas part number revisions 6 or higher get version 10 of the label).

When manual print users search for a label, they will see the current approved label for the given part number revision for the product they are working with. In the label WYSIWYG designer, the user can select any version of a label to view or select the checked out version to edit and save. External applications follow the same rules as manual print users.

Note that certain status changes may involve manual approval by a second person before moving to an approved for production status. This will be discussed in the WYSIWYG use cases below.

The labels table will group versions of a label by "title", or the text name for the label. The particular revision is an incrementing integer in the Revision column. A new row is added on label checkout in the WYSIWYG designer, with a new revision number. The status identifier is a GUID lookup into the label_status table. Sample schema is shown below:

Labels table

Title	Revision	ID	Status Id	Text	CheckoutUser
"My Label"	1	{5E2891FA-...4200A}	**{5E2891FA-...4225A}**	Some LBL text goes here…	NULL
"My 2nd Label"	1	{5E2891FA-...4200B}	{5E2891FA-...4225C}	Initial draft of LBL	NULL
"My 2nd Label"	2	{5E2891FA-...4200C}	{5E2891FA-...4225C}	Second draft of LBL	NULL
"My 2nd Label"	3	{5E2891FA-...4200D}	**{5E2891FA-...4225A}**	Final version for production	NULL
"My 3rd Label"	1	{5E2891FA-...4200E}	{5E2891FA-...4225C}	Initial draft of LBL	NULL
"My 3rd Label"	2	{5E2891FA-...42010}	{5E2891FA-...4225C}	Second draft of LBL	NULL
"My 3rd Label"	3	{5E2891FA-...42011}	**{5E2891FA-...4225A}**	Final version for production	NULL
"My 3rd Label"	4	{5E2891FA-...42012}	{5E2891FA-...4225C}	First rewrite for new part number	NULL
"My 3rd Label"	5	{5E2891FA-...42013}	*{5E2891FA-...4225B}*	Ready for approval	DOMAIN/JoeUse

Label_Status table

ID	Name	Description
{5E2891FA-...4225A}	Approved	This status is for labels that can go into production
{5E2891FA-...4225B}	Pending Approval	This status is for labels that must be approved by a label designer before going to Approved
{5E2891FA-...4225C}	Not Approved	This is a holding status

2.2 Application Security

Application security will be provided through specific tables in the KLM replacement database to authenticate and authorize users into groups of functional capability as well as data access to all or part of the label repository.

A user is currently assigned access to applications via the User_Authorization table. Each user has an entry in this table and is assigned access to each application. This access is hierarchical in nature in that a user's access includes all the capabilities of lower access levels.

There is a need to be able to assign capabilities, or features, to users independently of their relationship to the "level" of other users. To address this need the Kiosc Security Enhancement has been proposed. Another goal is to be able to more easily assign a new user access that existing users already have, but still on an application-by-application basis.

Applications needing to have such additional features, must define their feature list. Care should be taken so that no two features are mutually exclusive as the features may be enabled one way for one Group and then enabled in the opposite fashion in another Group, and a user might get assigned to both groups. Each feature is enabled and disabled independently as they are assigned to a Group, for that application.

A Group could be assigned features from more than one application. When a given application requests a user's access, only the features pertaining to that application are provided.

Users assigned to a Group, receive access to the features assigned to the group by an application, however the users also have to be given access to the application by the User_Authorization table.

A user can be assigned to none, one, or more groups. When assigned to multiple groups an application is provided a superset of the features the user has access to by "OR-ing" the features from the multiple Groups. If a user is not

assigned to a Group and the application requesting the user's access has features defined, all the features will be disabled for the user.

Another capability of this new security is addressing the need of some applications to allow access to "areas" of their applications which some of the application's features apply, such as create, modify, or delete. Instead of adding separate features, like "Save Site1" and "Save Site2", there would be one feature "Save" and then the areas, "Site1" and 'Site2" would be independently associated with different (or in some cases the same) Groups, thus allowing the application to know that the users has the "Save" feature but only for those "areas" that are associated with the Group. This allows a new "area" to be added in the future without the program having to be modified in order to handle it. The new area is handled just like the existing areas; it becomes just one more item in an existing list (such as products).

An additional feature for the KLM application will be the association of the Label_ID and the Group.

The following table graphics show how this schema is implemented:

Access_Group_Features: Used to define the features of an application

App_Name	Feature_Text	Feature_Position
KLM	Label Maintenance	1
KLM	Print Label	2
KLM	Reprint Label	3

Access_Group_Names: Used to define an application's features for a group of users

Group_Id	App_Name	Group_Name	Feature_List	Access_Level
1	KLM	Process Engr - BDA	YYY	
2	KLM	Production Operator	NYN	
3	KLM	Production Lead	NYY	

Access_Group_Users: Used to assign Users to Groups

Kiosc_Id	Group_Id
1010	1
1020	2
1030	2
1040	2
1050	3

Access_Categories: Used as a lookup table for the application's broad categories, or scope

App_Name	Categorty_Type	Category_Value	Category_ID
KLM	Process Area	BDA	1
KLM	Process Area	CIO	2
KLM	Process Area	PSG	3
KLM	Process Area	SNAP	4
OIT	Configuration	CIO	5
OIT	Configuration	PSG	6
KDE	Configuration	BDA	7
KDE	Configuration	PSG	8

Access_Category_Assigned: Used to assign categories to Groups

Category_ID	Group_Id
1	1

2	1
1	2
2	2
3	2
4	2

2.3 Data Formats and Conversion

Labels will be stored in the database in LBL format (**Text** column in Labels table). Caching of ZPL format may also be done for performance (**ZPLText** column in Labels table). Graphics will be stored in BMP format or other LBL-supported formats and will have a column describing the format type in the graphics table.

2.4 Multiple Repositories and Caching

Caching and multiple repositories will be designed for Phase 2.

3. Use Cases

This section describes the actors and use cases for the KLM system. Actors are defined as the different users that may interact with the system from a broad functional perspective. Usually actors are defined by their role in the process that the system supports.

3.1 User Scenario Actors

Print-user – a user in this role will be able to perform use cases including: login, search, view, print, re-print, set user options, and add a printer.

Editor, designer, and engineer – a user in this role will be able to perform use cases including: all operations allowed to the print user plus

Repository administrator – a user in this role will be able to create new repositories (including importing and exporting data), manage subscriptions between repositories, manage access permissions to the repository, and generate reports. This user will also establish web services and verify application configuration (through end-to-end operations such as printing a test label to a networked printer).

External application user – a user (or application) in this role will communicate with the KLM application through web services or an integrated DLL mechanism to perform all KLM operations allowed to a print-user.

3.2 Use Cases

The use cases are performed by actors in the system and represent end-to-end scenarios that the end-user will spend the majority of their time performing. The use cases usually map fairly closely to final QA test scenarios, though particular operations within a use case may also be tested with specific test cases in unit test or integration/system test plans.

3.2.1 Manual Print

1. **Login** to KLM application. Prompt user for KIOSC username and password. Use common KIOSC login screens from common KIOSC DLL (if enhanced features are needed, let's build it into the common DLL). No generic logins allowed. Login process should return allowed feature set to the application.

2. **Search** for a label. Accept user inputs for key label metadata fields. Also take input for keyword search. Use this information to return a set of labels. Return just the latest approved/released versions (for Manual Print). Return both a list of labels and optionally thumbnail images. The KLM-UI will narrow the labels a user may access based on the group information and site the user logs in under. Further search criteria are provided to select a label based on creator, most recently assessed, application used in, keyword, station_type, label_id, label_stock and graphic used.
3. **View/preview** a label – Present a view of the label (thumbnail?). Show the presaved thumbnail (JPEG or other decided upon format) from the database. Do not use the List'N'Label OCX (would require a license for each box).
4. **Print** the label – Assure that the prompts have been completed and print the selected label using the print engine. Generate serial number;Add to audit log
5. **Reprint** a label – if metadata for requested label allows unrestricted reprinting then same as above (Print use case). By default, disallow reprints. If reprint rules are allowed at manual station, enforce them appropriately, including checking permissions for that user. If track reprints are not set to true, then print same as above. If it is set, then label must be allowed for KLM manual printing, and user must have reprint authorization.
6. **Print bulk labels** – Get the quantity from the user. Print that quantity of labels properly, advancing auto-generated fields. Track quantity printed in case job is stopped. Provide means to stop printing.
7. **Params from file** – Complete prompts from a file. Advance to next line for each label. Key off prompt names in top row. Use comma-separated file.
8. **Printer management** (networked or local). Use common interface that can be shared with other applications. Have user set information about printer location (COM1, COM2, LPT1, etc.), baud rates, and loaded stock size. Provide test functions that test communication and settings with the printer(s). This would be program callable (see external application use cases). For user, provide a test print function.

3.2.2 WYSIWYG Designer

1. **Create** a label (new one from scratch). A user must have the appropriate credentials to create a new label in KLM. The List and Label OCX is initialized with a list of variables the user may work with in the LnL Designer. The information for the variable must contain: 1) the variable name, i.e. prompt, 2) sample variable content, and 3) the type of object the variable will apply to, i.e. a text box, a barcode, a drawing, etc. The object types are LnL object references. The LnL OCX is initialized by the LLDefineVariableStart and available variables are specified with LLDefineVariable or LLDefineVariableExt methods. The LnL Designer is then invoked, the new label wizard may be invoked prior to the designer or by the user selecting New from the Designer menu.

   The user selects the target printer and the label stock from the wizard. The user selects and positions label objects on the label template, enters the object content by specifying text, a filename, or selecting a variable defined for the object type. The user assigns a name to each object type, both static and variable objects. The object name is the same as the variable name for variable types. The label is saved as a label file having a .lbl file extension. The label is saved in a default directory, specified before the designer is invoked. If the label is saved in any place other than the default directory, the label will not be assigned a label_id and rev and further db entries will not be made. The label proceeds through the check-in process after approval has been given.

2. Create a **clone** of a label (copied from the template of another label). A user requires the same credentials to copy a label as to create a label. The same variable definition must occur for the LnL OCX as Create New label if this is a new instance of the OCX. The user selects a label using the search criteria and opens an existing label in the designer and either a) saves the current label with a new filename, or b) copies some or all of the label objects from the existing label, creates a new label and pastes the objects into the new label. The label is saved with a new filename in the default directory and the label is then submitted to the approval and check-in process.

3. **Edit** a label. A user must have appropriate credentials to edit a label. The label is selected by the search criteria. If the user selects Edit from the KLM-UI, The KLM-UI will provide a display of labels currently checked out by the current user and provide the capability to undo the checkout or submit the label for check-in.

   The label is and designated a status_id of edit (numeric equivalent) when the edit button or menu item is selected (only if the label is not already under edit). The labels table is updated with an entry specifying a new rev, which is editing, status_id and modified_time_stamp. The user is notified by display or email of the label_id and rev they must/should enter in the file description when they use 'save as' in the LnL designer. The LnL Designer is invoked with menu items enabled to save or save as, with the default directory specified. Saving the file in any other location will essentially render void and useless any changes made to the label.

4. **Check out** a label. A user must have appropriate credentials to check out a label. The user selects the label he/she wishes to checkout. An entry is made in the Labels table with a new Label_Id, indicating the label status is being edited and by whom. If the label is already checked out, indicated by an entry with Status_Id indicating under edit, the user is notified on the display. One person can only check out a label at a time. The LnL designer is invoked with the specified label. The user may save the label in the default directory until ready for approval / check-in. Saving the label in the default directory will cause an entry in the database to be created, parsing the .lbl file and entering meta-information relevant to the label.

5. **Check in** a label. The .lbl file is parsed for all variables referenced in the file. The variable name must be present in the Prompts table as Prompt_Text. If there are any missing variables, the label is rejected for check-in. The label description is parsed for the label_id and label_rev and checked against the db for being under edit by the submitter (current user) or assigned a new label id and rev and Meta information is entered. The label name is verified unique. Meta information must be entered in the KLM-UI because the label printer definition file is a combination of text and binary and is reserved for modification by LnL. The .lbl and .lbp files are saved as blobs in the database or saved to a specified directory for storage/archival.

6. **Versioning** labels – edit, approve; make the label the one for production! Audit log

3.2.3 External Application

External applications, for the purpose of this document, are defined as KIOSC system applications which are not intrinsically parts of the KLM system, but which may call KLM in order to print labels as part of a production process. Non-production users should print labels through KLM's manual interface. Examples of external applications would be AOP, KCI and KDE. (Note that this is not the complete list of allowed external applications.)

9. **Identify** (AppName, UserId, ComputerName)
   The calling application executes the Identify method and passes the required parameters, before making any other calls to the KLM interface. KLM will process the Identify method by: Ensuring that all required parameters are present
   - Checking that a database entry exists for the passed ComputerName – if found, KLM will cache the printer information for use during the PrintLabels call

- Caching the passed information so that it is available for future audit logging purposes.
- If any parameters are missing in the Identify call, or if the passed ComputerName does not have a defined printer setup, an error will be raised by KLM with the appropriate descriptive text. Note: It would be possible, at this point, for KLM to check a list of authorized applications, perhaps from a database table, to determine if the application was, in fact, permitted to call KLM, but this functionality is not planned at this time.

10. **PrintLabels** (SiteCode, StationType, Product, Optional OptionPN, Optional OptionRev, Optional <Other Parameters>) The external application calls the PrintLabels function and passes the appropriate parameters, which are specific to the process step being performed. Note that the calling routine may pass more parameters than are actually required to print the labels needed at the process step.
    - KLM uses the available parameters to compose a SQL statement, which it executes against the local labels database. It then evaluates the query results. If no matching labels are found, the function return value is set and the routine exits.
    - If one or more labels are found, KLM will check the required parameters for each of the labels against the list of parameters passed by the calling program. If any required parameters are missing, the function return value is set and the routine exits. If any serial number generation is required KLM will execute the appropriate stored procedure to acquire the data from the database. Note that if multiple labels specify the exact same "type" of generated serial number format, all of the labels in that call will receive the same serial number.
    - KLM passes the label file, along with the required parameters and printer information, to the labels print driver. The print driver will determine which labels are printed to which printer based on either label stock size, port number or printer name (TBD).
    - The Ids of all labels printed, as well as their parameter values will be cached for possible label reprints. (This information will be cleared on each new call to the PrintLabels function.)
    - Returns: Success if at least one label was found and printed or Failure if no labels were found for the specified process step, or if the passed parameters were insufficient to print the required label(s). This routine will also have to provide a mechanism for returning to the calling program all serial numbers (and associated serial number tracking_ids) that were auto-generated during the course of printing the label(s).

11. **Reprint Label** (AuthorizingUserId) If no previous call was made to the PrintLabels function, or if the passed AuthorizingUserId is missing or invalid, the routine will simply set a return status and exit.
    - If a *single* label was printed as a result of the previous PrintLabels call, that label is immediately reprinted. If multiple labels were printed, KLM will display a form on the calling machine showing thumbnail views of *each* of the labels that were just printed and will allow the user to select which label (or labels) are to be reprinted. The selected labels will then be reprinted.
    - KLM will make a log entry for every reprinted label and will associate the passed AuthorizingUserId and any label serial number information within the reprint log.
    - Returns: Success if at least one label was found and printed, or Failure if no labels were printed due to inadequate credentials or because no labels were available for reprint.
12. **Printer Setup** (ComputerName, AuthorizedMaintenanceId) As a result of the PrinterSetup call, KLM will display a form on the calling machine that allows the user to specify:
    - Printer Name(s)
    - Printer Type(s)
    - Port(s)
    - Setup String(s) [such as: "9600,8,n,1"]
    - Network address (for networked printers)
    - Label Stock Size(s)

    Any existing information for the specified ComputerName (found in the KLM database) will be pre-entered on the form. Once all new or changed information has been provided and has passed

validation, it will be registered into the KLM database to be used for future label printing functions.

13. **TestPrinter** (ComputerName, PrinterName/PortNo) The remote program calls the TestPrinter routine and passes all of the required parameters. KLM checks its database to determine the label stock size loaded into the specified printer. KLM then selects the appropriately sized "test" label and sends it to the print driver. Returns: Success if a label was printed, or Failure if no labels were printed due to unknown computer name, bad printer name or port number or failure to find a test label of the appropriate size.

3.2.4 Administrative Use Cases

14. **TBD – Phase 2)** Security management will be defined in Phase 2. Possibly including: add/modify/delete users, groups, roles, permissions; audit log; integrated with KSecurity? Manage client-caching strategies, adding/removing repositories/sites, reporting on label usage (reprints, etc.), view/filter audit log, archiving unused labels and log entries, import/export raw label tables to external apps.
15. **Delete** a label. Audit log.
16. **Add searchable keywords** for a label. Auditable.
17. **Remove** searchable keywords for a label. Auditable.
18. **Add category association** for a label. Labels may appear at more than one place in the hierarchy. Select the label, then select when in the tree view the label should be added. Need to be able to view the whole tree, plus the set of associations for a given label. Auditable.
19. **Remove category association** for a label. Select the label; get a list of the categories it is in. Select one category and remove. Auditable.
20. **Change category hierarchy**. Be able to add/delete/rename tree view category titles. Auditable.
21. Change the **default stock size** used for a label. Auditable
22. **Add stock type**. Auditable
23. **Remove** stock type. Auditable
24. **Update** stock type… Auditable
25. **Add available stock** size. Auditable
26. **Remove** available stock size. Auditable
27. **Modify/update** available stock size. Auditable.
28. **Add stock vendor.** Auditable.
29. **Remove stock vendor**. Auditable.
30. Change access permissions for a label (who can use the label). Auditable
31. Change the ability to do reprints or not on a label. Auditable.
32. Change the associated part number, BOM part number, etc. What about the status flag? Or is this pat of versioning? Change the product site and line station for a label. Auditable
33. Change associated product/mapped product for a label. Auditable.

34. Define prompts for a label and verify that they are formatted correctly when printed. What are all the prompt types? Auditable.

35. Manage available serial numbers and generation process for a label.

4. Deployment Constraints

This section describes the target platform of the KLM application and deployment considerations. The first subsection describes the current hardware and software platform that are the minimum supported target platform for KLM. The second subsection describes the platform supported for various .NET technologies. The final subsection describes the draft deployment process for the KLM application.

4.1 Target Platform Description

The KLM rich client should run on a PC with at least 16MB RAM, 133MHz, etc. with Windows Xp/NT4 Pro/etc…

The KLM web client should run on any machine with Internet Explorer 5.01 or higher…

The KLM web service… converter, repository, etc….

4.2 .Net Platform Description

The .NET platform is supported over a wide variety of Microsoft platforms. Some platform configurations provide full support whereas others provide only partial (or degraded) support.

The **basic .NET Framework SDK** (including ADO .NET, Windows Forms, and XML web service clients) is supported on the following platforms (Internet Explorer 5.01+ and Windows Installer 2.0 are also required):

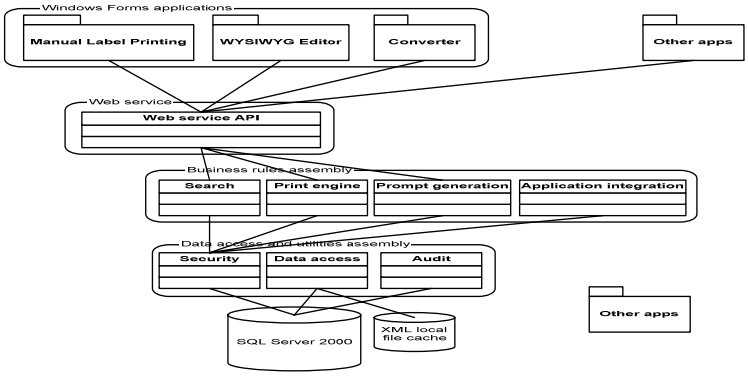
- Windows 2000 (Pro, Server, Advanced Server)
- Windows NT 4 Workstation or Server with Service Pack 6a+
- Windows Millennium Edition
- Windows 98, Windows 98SE, and
- Windows XP (Home or Pro).

To **host** ASP .NET or XML web services, the server must be capable of IIS 5.0+ and running Windows 2000 (Pro, Server or Advanced Server, with Service Pack 2) or Windows XP Pro with MDAC 2.7+.

To **access SQL Server** using the SQL Server .NET Data Provider, the client must be capable of MDAC 2.6+.

General **hardware requirements** vary by client (Windows Forms and Windows Services) or server. Client machines should be Pentium 90MHz+, 32MB+ RAM (required), and 90MB+ (recommended). Server machines should be Pentium 133MHz+, 128MB+ (required), and 256MB+ (recommended).

4.3 Deployment Process



The web client application requires IIS 5.0 o higher and is deployed with the MSI package. The rich client requires the CLR and connectivity to a SQL Server repository…

The following deployment scenarios are envisioned (Visios): XYZ parent site (Colorado Springs, Penang, Dundalk), connected online vendor, disconnected vendor, and single workstation mode.

The CLR can be installed separately as the Windows Component Update (from the Visual Studio installation media or the merge module, or SMS push…)

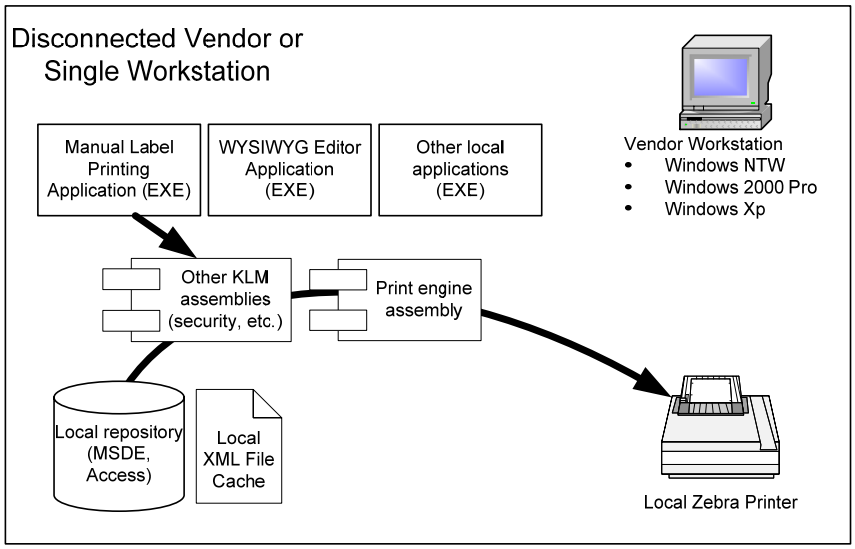


Figure 1. Disconnected vendor scenario.

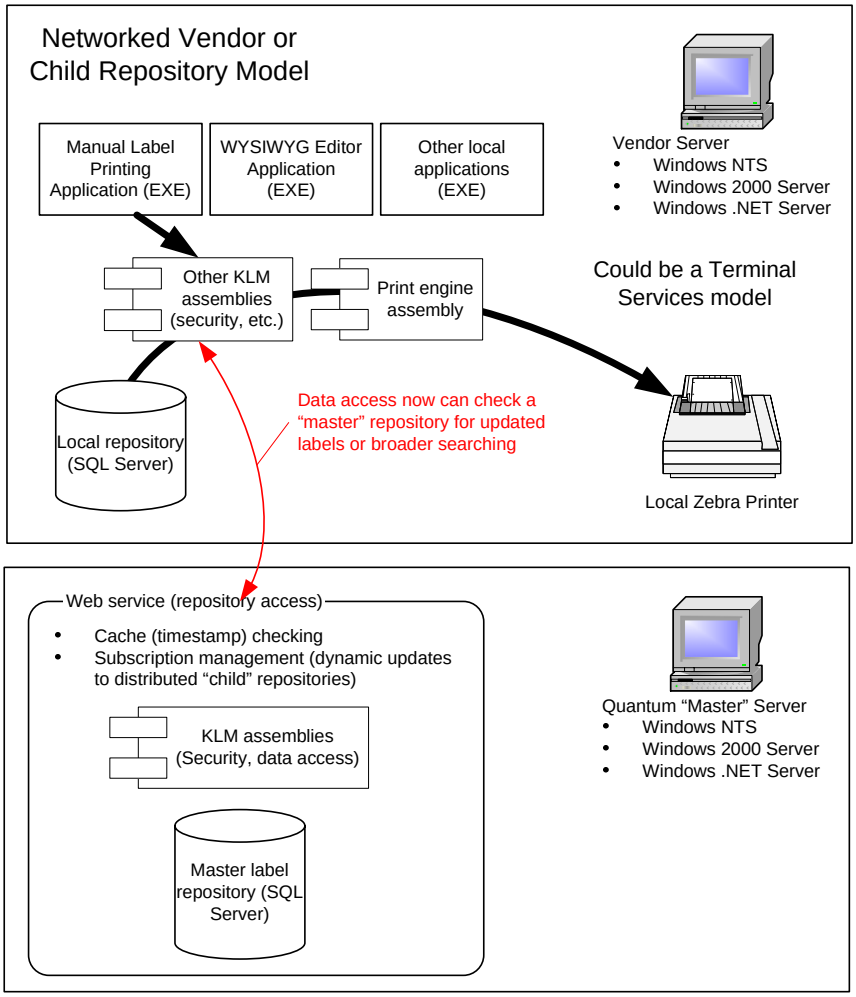


Figure 2. Network-aware vendor.

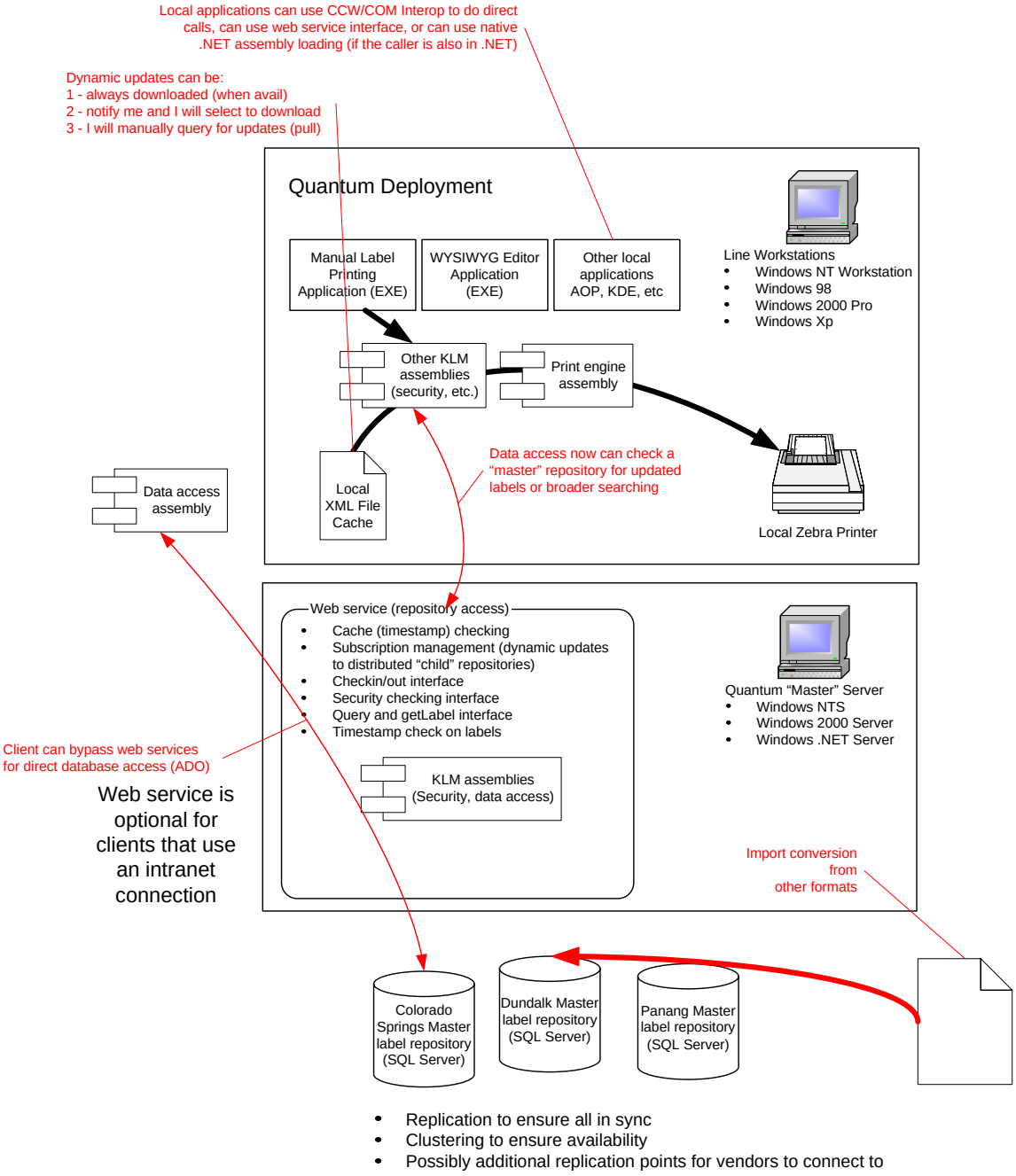


Figure 3. XYZ Master Repository Deployment.

5. Presentation Services Overview

This section describes the user interfaces provided as part of the KLM project. The user interfaces are described by subsection for the WYSIWYG editor (rich-client, also including print-only features), the web-enabled print-only application, the conversion utility, and installation package. Each subsection has initial screen mockups, descriptive text, application configuration file format notes, data validation notes, and pseudo-code for interactions with business services (described in the following section).

WYSIWYG (rich client) editor

The rich client editor (main interface) will be a Windows Forms EXE providing the user interface elements for Manual Print and WYSIWYG Editor use cases.

Project namespace: **XYZ.KLM.Application**

Project Type: ❑ **DLL** ✓ Windows Forms EXE ❑ Web service ❑ WebForms app

The class uses the following other components:

- XYZ.KLM.DataAccess
- XYZ.KLM.Search
- XYZ.KLM.Security
- XYZ.KLM.Editor

The Windows Form project supports a startup method, menu item and toolbar click methods (event handlers), a close method, and has a specified configuration file format.

The initial screen will look like Figure 1.

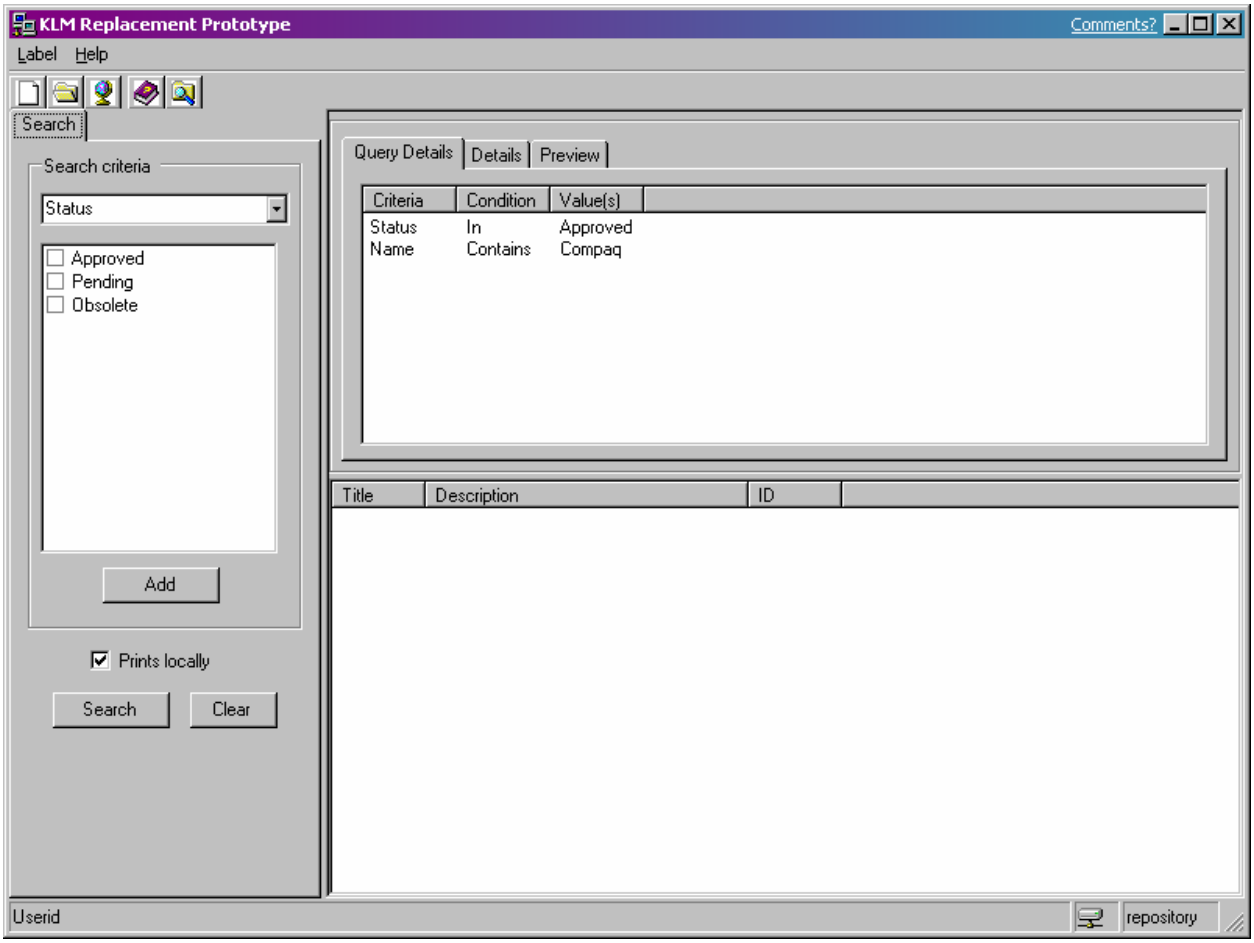


Figure 1. Initial KLM screen.

The main screen is divided into display areas for title bar, menu bar, tool bar, search panel, search results panel, selected item display panel, and status area. The application title bar has the name of the application and maximize/minimize buttons. The menu bar has entries for the events defined below. The toolbar has icons for activating menu items as specified below.

The search panel is displayed on the left side of the main form (shown above) and allows the user to select a criteria and value(s), allows the user to add the criteria to the search, and allows the user to initiate the search (populating the search results panel).

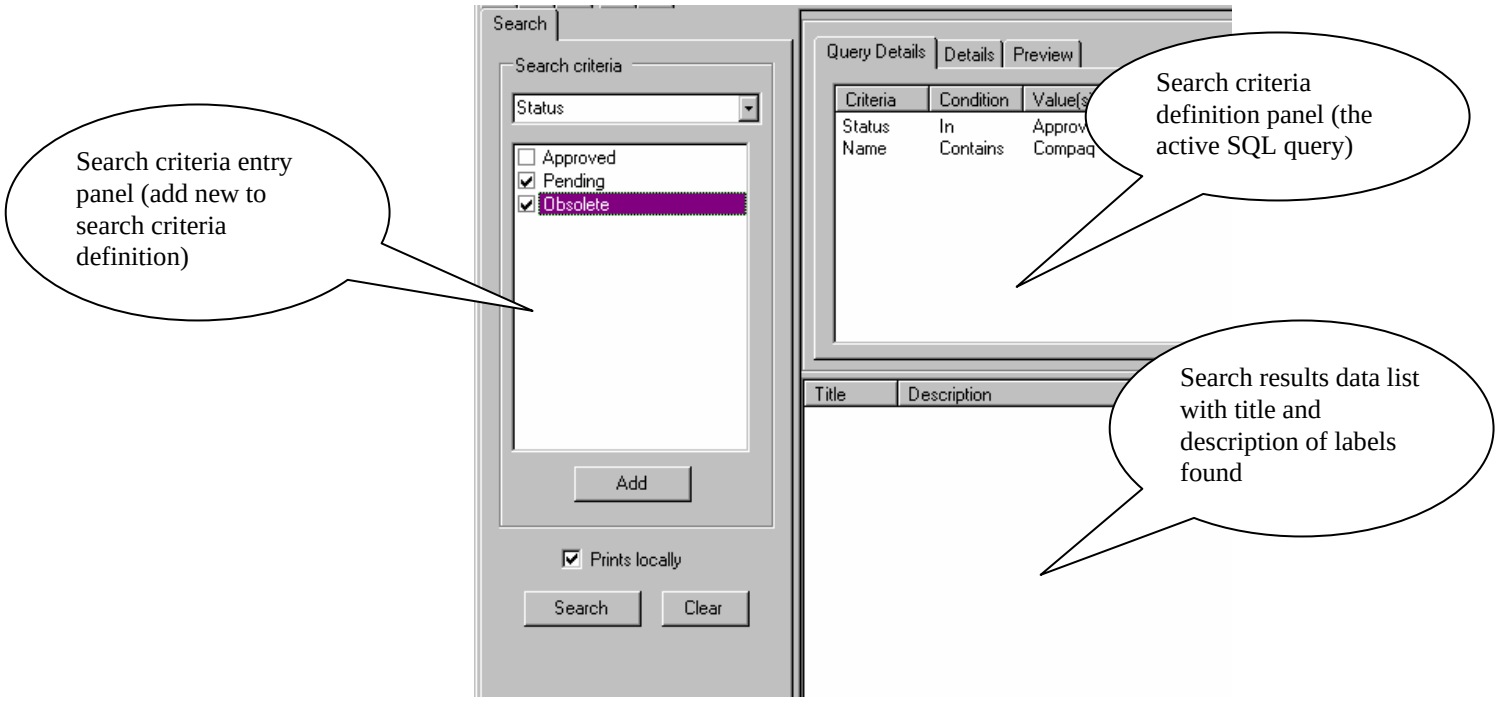


Figure 2. Search tab and query definition (details) tab showing active search criteria.

The search results panel displays a list view of the results of the SQL query, including the name and TBD metadata on the returned label set. When the user selects one of the list view items, the selected item display panel is populated with the details of that label. The search button causes the search function to ExecuteQuery (in the search component); the clear button causes the ClearSearchCriteria method to execute in the search component. Selecting "Prints Locally" adds search criteria for selecting only labels that can be printed on this computer (labels that match the printer types attached to this machine and also match stock sizes loaded in those specific printers).

The selected item display panel is a tabbed display with panels for selected item preview and details (e.g., metadata).

Startup - login to the application and load main form

Inputs:

- Optional pointer to application configuration file (String, command line argument)

Processing:

1. Get Site_code, KIOSCRef connect string, and KLM connect String and retain in memory
   - If the path to the configuration file was specified on the command line, then read from there. (If a read error is encountered, write to the application log, show a message box saying an error occurred, and exit the application.)
   - Else check the current application path for the configuration file; if not there, log the error, present a message box, and end the application.
2. Instantiate the security class (see below) and call validate method to get user credentials and authorization level.
   - If "no access" returned, then show a message box with "You're not authorized to use this application" and exit the application. Retain the User Id and Authorization level in memory for later use (object?).
   - Otherwise show the splash screen while loading main form
     - Show splash screen (below)

- Load main form
- Hide splash screen on return from load main form
- Show main form

3. Load main form (from above)
   - Set the visible menu items visible based on authorization level (see table below)
     - **General User**: View, Print, Bulk Print, Reprint
     - **Maintenance User**: All of General User menu items plus Printer management
     - **Reprint Authorize**: All of the maintenance user menu items
     - **Label Creator**: All of reprint authorize menu items plus New (Label), Copy, and Edit
     - **Label Approver**: All of label creator menu items plus Check-in
     - **Developer**: All of the label approver menu items plus **TBD** possible app debugging features
   - Enable menu items (if visible) for initial form load
     - Search, PrinterManagement, New (Label), Exit, Help
   - Instantiate search class and get search criteria to load in combo box
   - **TBD** use ktimeservice unless we go with server side time stamps

Outputs:

- Main application form displayed

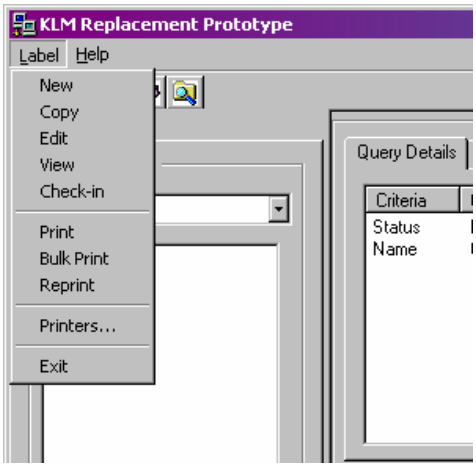


Figure 2. Label menu items.

Menu Item: Label→New

This menu item is always available authorized user credentials. This calls the LNLDesigner component (instantiated at application startup) to CreateNewLabel.

Menu Item: Label→Copy

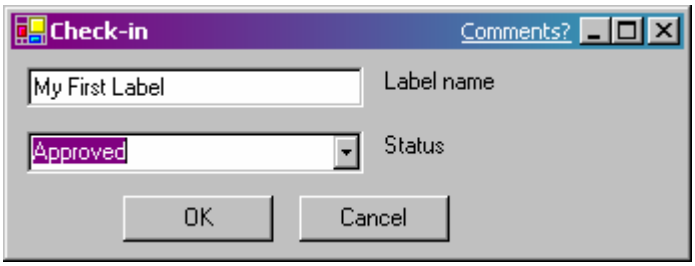
This menu item is only enabled when a label is selected from search results. This calls the LNLDesigner component to CopyExistingLabel.

Menu Item: Label→Edit

This menu item is only enabled when a label is selected from search results. This calls the LabelManager component to CheckOutLabel and then (if successful) calls the LNLDesigner to EditExistingLabel.

Menu Item: Label→Check-in

This menu item is only enabled when a versioned label is selected in the search results and the user has "Approver" authorization. Updates the status on the labels to reflect "Approved" or "Released". This is the only metadata item allowed changed with this command. See screen below.



SearchCriteriaAdded Event

This event is raised when the user selects to add new search criteria. This event causes a call to GetSearchCriteria( ) in the Search component and rebinds the resulting dataset to the query definition panel.

Configuration file structure

The application configuration file looks like:

```
<application>
  <configuration>
    <add key="siteCode" value="32">
    <add key="KIOSCConnectString" value="server=foo;database=k2dev;uid=sa;passwd=">
    <add key="KLMConnectString" value=" server=foo;database=k2dev;uid=sa;passwd=">
    <add key="LNLObjectTypes" value="barcode;text;">
  </configuration>
</application>
```

Installation package

The installation package is a Microsoft Installer package built for the rich client (WYSIWYG editor), web application interface (print-only), the basic web service for integration, and the repository deployment. The user selects which type of install/uninstall to perform…

6. Business Services Overview

This section describes the business logic components (assemblies) and the web services interface for the KLM application. The business components will be designed in separate functional specifications but the initial interfaces and interactions will be specified in this document.

LNLDesigner component

The LNLDesigner component will be a DLL providing methods to instantiate, control, and respond to events for the List and Label OCX.

Project namespace: **XYZ.KLM.Editor**

Project Type: ✓ **DLL** ❑ Windows Forms EXE ❑ Web service ❑ WebForms app

The class uses the following other components:

- XYZ.KLM.DataAccess

The class supports the following methods:

Constructor

Processing:

4. Instantiate the LNLDesigner, map event handler for Close event.
5. Set component status to “No label”

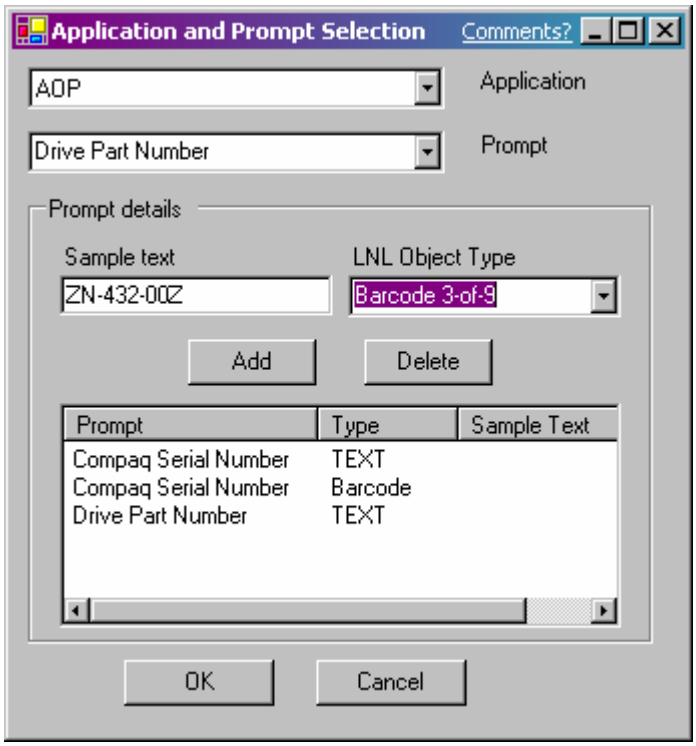
CreateNewLabel

Inputs:

- None (menu click)

Processing:

6. Get the application name from the user using a dialog box (shown below)



7. Once the application name is selected, call the Data Access component and get the list of prompts, fill in the checked list box. Read the application configuration file for List and Label types (for drop down list). If the user changes the application name selection then clear the LNL variable name buffer (LLDefineVariableStart).
8. Validate that each checked prompt has an LNL object type. Once the user selects OK, use the List and Label variable buffer DLL to pre-load the variable names for the OCX.
9. Set the component state to "New Label".
10. Invoke the New Label Wizard (in LNL).
11. Invoke the designer (Show).

Outputs:

- None

OnListAndLabelDesigner Close (mapped event)

Inputs:

- LBL file name
- Component state

Processing:

12. When the user clicks "Exit" on the List and Label designer, the LBL file will be written to c:\temp or KLM application temp directory… The designer window closes
13. The application will check the application temp directory for the LBL file, and if not found allow user to browse to save location
14. The application will prompt the user for label metadata (if the component state is "Edit Label" then get current metadata from database). Need screen snapshot of Metadata screen.
15. The application parses the LBL file for variables and prompts
16. Generate a JPEG image of the LBL (if one doesn't already exist) using the LNL DLLs.
17. LBL, JPEG, prompts, and metadata are saved to the database. If the component state is

"New Label", then insert the LBL text in a NEW row in the Labels table (revision = 1). If the component state is "New Label" then add a row to Label_Checkouts for the current user. If the component state is "Edit Label" then insert the LBL text/graphic in a NEW row in the labels table (revision = previous revision + 1) with status checked out (to current user).

18. Delete the LBL and JPEG files on disk.

Outputs:

- None

CopyExistingLabel

Inputs:

- Selected label id

Processing:

19. Get the application name from the user using a dialog box (shown below). The default value is the application of the original label.

See picture above.

20. Once the application name is selected, call the Data Access component and get the list of prompts for the application area, fill in the checked list box. Read the application configuration file for List and Label types (for drop down list). If the user changes the application name selection then clear the LNL variable name buffer (LLDefineVariableStart).

21. Validate that each checked prompt has an LNL object type. Once the user selects OK, use the List and Label variable buffer DLL to pre-load the variable names for the OCX from the selected prompts.

22. Set the component state to "New Label"

23. Invoke the designer (Show).

Outputs:

- None

EditExistingLabel

Inputs:

- Selected label id (assumed checked out)

Processing:

24. Get the existing LBL file from database to application temporary folder, plus prompts, and graphics.

25. Get the application name from the user using a dialog box (shown above). The default value is the application of the original label. Populate the prompts list from the existing prompts in the database.

26. Once the application name is selected, call the Data Access component and get the list of prompts for the application area, fill in the checked list box. Read the application configuration file for List and Label types (for drop down list). If the user changes the application name selection then clear the LNL variable name buffer (LLDefineVariableStart) and re-clear the prompt list.

27. Validate that each checked prompt has an LNL object type. Once the user selects OK, use the List and Label variable buffer DLL to pre-load the variable names for the OCX from the selected prompts.

28. Set the component state to "Edit Label"

29. Invoke the designer (Show).

Outputs:

- None

Application integration component

The application integration component will be a DLL providing the CallableInterface class with methods for remote applications to perform key KLM functions. External applications are trusted and do not need to be authenticated.

Project namespace: **XYZ.KLM.CallableInterface**

Project Type: ✓ **DLL** ❑ Windows Forms EXE ❑ Web service ❑ WebForms app

The class uses the following other components:

- XYZ.KLM.DataAccess

The class supports the following methods:

Identify

Inputs:

- AppName (String) – the application name of the caller
- User Id (Integer) – the user identifier of the caller (trusted)
- Computer Name (String) – the physical computer location of the caller
- Site Code (String) – the site code of the caller

Operations:

1 Validate input method parameters (for non-null strings or negative/zero userid)
  1.1 Throw "Missing parameter error" if not valid
2 Connect to the KLM database
  2.1 Call Data Access method `Connect()` with the site code parameter (to determine which connect string is used)
3 Get printer parameters for given computer name
  3.1 Call Data Access method `GetPrinterDetailsForComputer()`
  3.2 If no printer details available, throw "Printer not defined error"
4 Add all items to cache for future calls
  4.1 Cache is a private member variable, m_Cache, of type Hashtable. Key-value pairs are inserted for the input parameters and the DataTable returned with printer details.

Outputs:

- Return Code (Boolean) – success or failure of getting the printer details
- (Cache of Inputs)
- (Cache or Printer Parameters)

Exceptions:

- Missing required method parameter
- Could not connect to database (passed up from DataAccess component)
- Printers not defined for specified computer name
- Could not create cache (memory violation)

Database Tables Needed:

Printers (table name)	**Data type**
ComputerName	Nvarchar (50)
PrinterName	Nvarchar (50)
PrinterType	UniqueIdentifier – FK to PrinterTypes table
StockId	UniqueIdentifier – FK to AvailableStock table
NetworkId	Nvarchar (200)

PrintLabels

Inputs:

- StationType (Int32, short) – the station type of the caller
- Product (string) – the product of the caller
- Option_PN (string) – the option part number
- Option_Rev (string) – the revision of the option part number
- <other parameters TBD>

Operations:

1. Validate method parameters (non-null strings, etc.)
2. Query database to determine labels to be printed at this station/process step
    a. If no labels found matching search criteria, throw "no label defined at this station error"
3. Match passed parameters to required parameters

Outputs:

- Return Code (Boolean) – success or failure of ?

Exceptions:

- Missing required method parameter
- Could not connect to database (passed up from DataAccess component)
- No label defined at this station
- Missing required label parameter

LabelManager component

The label manager component will be a DLL providing general label management functions.

Project namespace: **XYZ.KLM.Editor**

Project Type: ✓ **DLL** ❑ Windows Forms EXE ❑ Web service ❑ WebForms app

The class uses the following other components:

- XYZ.KLM.DataAccess

The class supports the following methods:

CheckOutLabel

Inputs:

- Label id
- User id

Processing:

30. Check if an entry exists in database checkout table
    - If the checkout user matches the requested user id for checkout
        - Return status "Already checked out"
    - If a checkout user exists but does NOT match the requested user id
        - Return status "Checked out to another user"
    - Else
        - Insert row in checkout table for requested user id (must be transacted)
        - Return status "Checkout success"

Outputs:

- Checkout status enumeration (success, failed/another user, failed/already checked out to you)

Security component

The security component will be a DLL providing the Security class with methods to authenticate and authorize users in the KLM system.

Project namespace: **XYZ.KLM.Security**

Project Type: ✓ **DLL** ❑ Windows Forms EXE ❑ Web service ❑ WebForms app

The class uses the following other components:

- XYZ.KLM.DataAccess

The class supports the following methods:

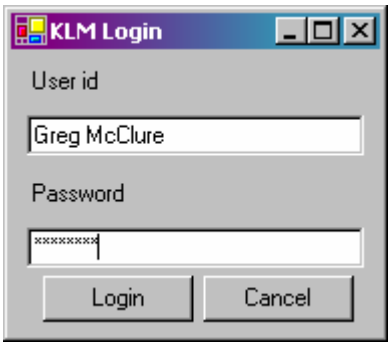
Validate - Authenticate and Get Authorization level

Inputs:

- Connect String to site specific KIOSCRef database

Processing:

31. Prompts user for user id and password (see screen below)
    - Mask password word



32. Performs authentication and Obtains authorization level
    - Encrypt password (import current KIOSC encryption scheme)
    - Query database with user id and encrypted password for authorization level
        - If no records provide up to 3 retries
        - Else (after 3rd retry), show a message box: "Come back when you remember your password."
33. Map to enumerated type and return output

Outputs:

- Enumerated value of authorization level: Developer, Label Approver, Label Creator, Reprint Authorize, Maintenance User, General User, No Access. Note these maybe expanded to accommodate separate process engineering groups or other non-hierarchical relationships.

Search component

The search component will be a DLL providing the Search class with methods to find labels and prompts for specified metadata (keywords, id's, titles, etc.).

Project namespace: **XYZ.KLM.Search**

Project Type: ✓ **DLL** ❑ Windows Forms EXE ❑ Web service ❑ WebForms app

The class uses the following other components:

- XYZ.KLM.DataAccess

The class supports the following methods:

Constructor

Inputs:

- Connect string to KLM database (String)

Processing:

34. Set an internal variable to store the connect string
35. Call ClearSearchCriteria( )

GetSearchCriteria

Inputs:

- Authorization level (some queries don't get a choice on label status)

Processing:

36. Create dataset from scratch with name column (maybe others **TBD**)
37. Insert rows in dataset for: Station Type(s), Site Code, Stock Size, Category, **Customer**, Graphics, **Bar Code Type**, Product, Part No, Part No Rev, Label Name, Label Rev
38. Call AddSearchCriteria( ) for certain search criteria that are applied by authorization level
    - "Status" = "Approved" for regular Print User

Outputs:

- Data set of search criteria with a column called "Name" in a table called "Criteria"

GetChoices – for populating a checked list box on the display

Inputs:

- Criteria name (string)

Processing:

39. Create an empty dataset from scratch with value column and checked column (bit)
40. Select the values from the appropriate table in the database based on the criteria name:
    - Station Type, Site Code, Stock Size, Category, Customer, Bar Code Type, Product, Part Number, Part Number – Revision are looked up in database
    - For label revision, label name, graphics name – these are "LIKE" criteria and return an empty dataset

Outputs:

- Data set of criteria values and whether they are checked by default (if the criteria is a select-from-list type)

GetSearchQuery - for populating Query Definition panel

Inputs:

- None

Processing:

41. Return a reference to the active search criteria dataset

Outputs:

- Data set of search criteria selected in the active search collection (columns for criteria name, type of query condition (LIKE, IN), and values)

AddSearchCriteria

Inputs:

- Criteria name (string)
- IsLikeQuery (Boolean) – specifies whether the value should be converted to a LIKE query in the SQL string (if False, then the equals operator is applied)
- Value (string)

Processing:

42. Add a row into the active search criteria dataset for the inputs specified
43. Raise a SearchCriteriaAdded event to the user interface element (to repaint with new query definition panel contents)

Outputs:

- Data set of search criteria with a column called “Name” in a table called “Criteria”

ExecuteSearch – for populating Query Results list

Inputs:

- None

Processing:

44. Build a SQL query from the active search criteria dataset

Outputs:

- Data set of search results with columns for label name and TBD other label attributes

ClearSearchCriteria

Inputs:

- None

Processing:

45. Create dataset from scratch for active search criteria with columns for criteria name, type of query condition (LIKE, IN), and values

Outputs:

- None

7. Data Services Overview

This section describes the data access component of the KLM replacement project and the KLM repository schema. The data access component will be designed in a separate functional specification but the initial interface and interactions will be specified in this document. The repository discussion will provide general guidelines and schema layout and a separate diagram will show the detailed schema including all tables, stored procedures, etc.

7.1 Data Access Component

The data access component will be a DLL providing the DataAccess or class with methods for all KLM components to use the KLM repository (database). The caller is assumed to have been authenticated with the security component, if required.

Project namespace: **XYZ.KLM.DataAccess**

Project Type: ✓ **DLL** ❑ Windows Forms EXE ❑ Web service ❑ WebForms app

The class uses the following other components:

- XYZ.KLM???

The class supports the following methods:

Connect( )

Inputs:
- Site Code (String)

Operations:
5 Validate that a site code was passed in
  5.1 Throw "Missing parameter error" if not valid
6 Determine the connection string from the application configuration file
  6.1 If not found, select the default connection string
7 Connect to the database using the connection string

Outputs:
- Return code (success/failure)

Exceptions:
- Missing required method parameter
- Could not connect to database

GetPrinterDetailsForComputer( )

Inputs:
- ComputerName (String)

Operations:
8 Validate that a computer name was passed in
  8.1 Throw "Missing parameter error" if not valid
9 Connect to the KLM database, if not done already
  9.1 Call `Connect ()` with the site code parameter
10 Get printer details for the given computer name
  10.1 Using a T-SQL SELECT statement, get a DataTable of all printer columns in the Printers table matching the ComputerName

Outputs:
- DataTable

Exceptions:
- Missing required method parameter
- Could not connect to database

GetAllApplications( )

GetPrompts ForApplication (appName)

AddNewLabelToRepository( ) – saves a new row with Revision = 1

UpdateLabelInRepository( ) – saves a new row with Revision = Revsion + 1

AddNewLabelGraphicForLabel( )

DeleteAllLabelGraphicsForLabel(labelId)

AddNewLabelMetadataToRepository( )

CreateNewLabelFromExistingLabel( )

UpdateExistingLabelMetadataToRepository( )

QueryForLabels(stationId)

QueryForLabels(SQLStatement) – the main query from search panel, T-SQL assembled in search component

GetLabelCheckouts(labelId)

CheckoutLabel(labelId) – inserts a row in the checkout table

GetUserAuthorizationLevel(userId, password) – also does the encryption in this function

WriteApplicationEventLog( )

GetGraphicsForLabel(labelId)

GetLabelKeywordsForLabel( )

AddLabelKeywordsForLabel( )

AddPromptMappingForLabel( )

DeletePromptMappingForLabel( )

UpdateLabelStatus( )

ReadApplicationConfigurationSetting( ) – from database table (not app config file)

7.2 File Access Component

The file access component will be a DLL providing methods for all KLM components to read and write files on the local drive, including the configuration file. The caller is assumed to have been authenticated with the security component, if required.

Project namespace: **XYZ.KLM.DataAccess**

Project Type: ✓ **DLL** ❑ Windows Forms EXE ❑ Web service ❑ WebForms app

The class supports the following methods:

GetFileContentsAsBinary( ) - as Byte[], for thumbnails and JPGs

GetFileContentsAsText( ) - as string, for LBL content parsing

FileExists(strPath) returns boolean

DeleteFile(strPath)

GetApplicationTempFolder( )

GetDefaultConfigurationFileName( ) – returns the path to the default configuration file

GetConfigurationFileValue( ) - takes a string key value and looks up the string value for that key (used for initial site code, and connect strings to KIOSC ref and KLM databases)

GetConfigurationFileName( ) – returns the actual path to the configuration file being used (could be passed in by command line).

7.3 Log File Component

The log file component will be a DLL providing methods for all KLM components to write to the application log file. The caller is assumed to have been authenticated with the security component, if required.

Project namespace: **XYZ.KLM.DataAccess**

Project Type: ✓ **DLL** ❑ Windows Forms EXE ❑ Web service ❑ WebForms app

The class supports the following methods:

Constructor

Inputs:

- File path and file name (String, optional)

Operations:

1. Set an internal variable to hold the file stream name (defaults to “applog.xml”)

WriteLine( )

Inputs:

- Message (String)

Operations:

11 Validate that a site code was passed in
  11.1 Throw “Missing parameter error” if not valid

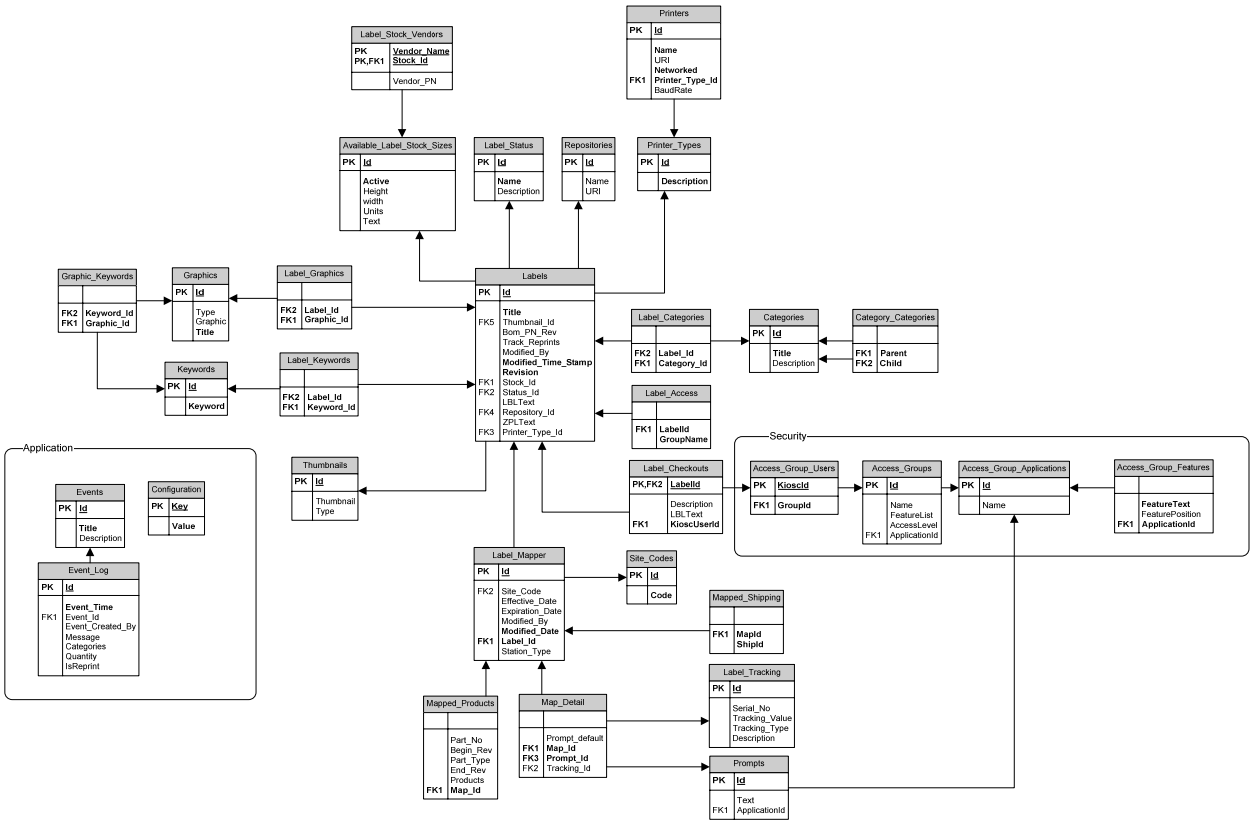
Outputs:

- Return code (success/failure)

7.4 Data Repository

The KLM repository is implemented in two flavors: a local XML file storage mechanism, and a SQL Server schema.

The SQL Server schema has the following tables (the official VISIO diagram is stored in SourceSafe under $:\database\diagrams):



Insert text to describe tables here – general usage, sample data, column types, keys and indexes

Multiple repositories may exist within an installed KLM system. Consistency between databases is maintained through the caching mechanism described above. This is implemented by the following SQL scheduled jobs…

All database tables will have audit fields that are automatically filled in by defaults. These columns include a create user id, create timestamp, last-modified user id, and last-modified timestamp. Additionally, the database contains audit tables for logging print events (and parameters), and security events…