

# Louisiana Department of Insurance

## Software Development Standards



## Table of Contents

<b>EXECUTIVE SUMMARY .....</b>	<b>4</b>
<b>LDI SOFTWARE DEVELOPMENT STANDARDS GOAL .....</b>	<b>5</b>
<b>EXCEPTIONS, CHANGES AND AMENDMENTS TO SOFTWARE DEVELOPMENT STANDARDS .....</b>	<b>6</b>
<b>IMPLEMENTATION OBJECTIVES .....</b>	<b>8</b>
<b>LDI SYSTEMS INTEGRATION .....</b>	<b>9</b>
<b>LDI ACCEPTABLE TOOLS AND TECHNOLOGIES BY CATEGORY .....</b>	<b>10</b>
<b>THE FOLLOWING TOOLS AND TECHNOLOGIES ARE ACCEPTABLE FOR THE DEVELOPMENT OF NEW LDI SOFTWARE SYSTEMS: .....</b>	<b>10</b>
<b>LDI PROJECT MANAGEMENT .....</b>	<b>12</b>
<b>LDI PROJECT AND SOFTWARE DELIVERABLES .....</b>	<b>14</b>
<b>DOCUMENTATION DELIVERABLES .....</b>	<b>15</b>
<i>User Documentation/ User Manual .....</i>	<i>16</i>
<i>Technical Documentation .....</i>	<i>17</i>
<b>LDI APPLICATION LOOK AND FEEL GUIDELINES .....</b>	<b>20</b>
<b>LDI DATABASE NORMALIZATION .....</b>	<b>21</b>
<b>NETWORK, SERVERS AND MISCELLANEOUS STANDARDS .....</b>	<b>22</b>
<b>LDI DEVELOPMENT, TEST AND PRODUCTION ENVIRONMENTS .....</b>	<b>24</b>
<b>METHODS AND PROCEDURES TO MOVE NEW SYSTEMS AND UPDATES INTO THE PRODUCTION ENVIRONMENT .....</b>	<b>26</b>
<b>SYSTEM AND APPLICATION LAUNCHING POINT .....</b>	<b>27</b>
<b>PERMISSIONS STRUCTURE, PASSWORDS, AND SUPPORTED COMPUTER SOFTWARE .....</b>	<b>28</b>
<b>NAMING CONVENTIONS AND DATA STANDARDS .....</b>	<b>29</b>
<b>APPENDICES .....</b>	<b>30</b>
<b>CODING (LANGUAGE) RECOMMENDATIONS .....</b>	<b>31</b>
<b>VISUAL C#.NET .....</b>	<b>33</b>
<b>SOURCE SAFE RECOMMENDATIONS (SOURCE CONTROL) .....</b>	<b>34</b>
<b>INTEGRATION GUIDELINES .....</b>	<b>35</b>

Document last updated on March 31, 2010



## **Executive Summary**

This document represents a basis for the overall design, implementation, development, deployment, and documentation, for which all work performed on current applications and systems as well as future systems deployed at LDI must adhere. Included within this basis are the general and specific requirements as defined by the department. These requirements cover the internal systems on which the department depends for day to day operations. The external systems and Internet applications which the department provides for public use are addressed in this document. The increasing complex external systems are becoming more important for proper department operation.

In addition, the method of integration for all applications which bonds the applications and systems together and creates a seamless department wide application is included. Finally, the structure and storage of all data within the department databases is described, and a pattern of implementation expressed.

These standards have been developed with the cost of implementation in mind, and it is believed that these standards will have a minimal cost impact to the department when implemented. There is no risk to the department in employing these suggested standards. The standards have been designed to ensure maximum future flexibility, greatest growth potential and lowest cost of maintenance. A greater risk to the department's operations exists if developers do not follow or correctly employ these minimum standards as more applications are developed.

The complexity and critical nature of the department's automated systems in relation to the operations and function of the department necessitate that these standards be implemented and followed. The department's overall IT plan for a completely integrated system is currently in progress. The plan for an integrated system extends the already completed integrated systems of LDI. The plan depends on the standards being implemented and followed while adherence to the standards is monitored.

## **LDI Software Development Standards Goal**

The goal of this document is to establish common standards including system integration for which all present and future automated systems will obey.

These standards will provide the department maximum flexibility, increase the flow of information between different systems within the department and create a foundation from which all systems will start.

These standards cannot be circumvented in any state of an applications or systems lifecycle. Only by all systems embracing these standards will the overall goals of the department be achieved.

## **Exceptions, Changes and Amendments to Software Development Standards**

All exceptions, changes and amendments to the Software Development Standards must be formally approved. There will be no deviations from the published software standards without approval. All subsequent publications or addendums to this document adhere to the same policy of no changes without approval. Any questions concerning what issues need approval should be directed to the IT Director for clarification.

Below is an overview of the approval process:

1. Identify and document all desired changes or deviations from the published Software Development Standard.
2. Document the basis and specific issues requiring the change or deviation from the published standard. Give an explanation in support of the deviation from the published standards.
3. Provide the IT Project Leader with all documentation for revising the standards.
4. Based on provided documentation, the current overall IT plan and the current and future needs of the department, the Project Leader will, after consulting with the IT Director, make a determination on the validity of the request. If the request is valid, and is of such nature that it does not require a committee recommendation, the Project Leader will issue a standards update on an IT Project Form to the published software development standard.
5. If a decision is reached which determines that a more thorough review is warranted, the IT Director will convene an IT standards committee to review the change.
6. The IT committee will review the proposal and make a determination on the viability of the proposed modification to the Standards and either approve, modify, or disapprove it, and submit their decision to the IT Director who will make a final decision, which should probably be done within five working days.

7. The Deputy Commissioners of Management and Finance will be advised of all changes made to the IT Software Development Standards and should be consulted for approval on any major change to the IT Software Development Standards which may affect the budget or entities outside the IT division.

New standards will only take effect when an approved standards update on an IT Project Form is issued by the Project Leader and the Software Development Standards has been updated and posted.

The Software Standards documentation is currently maintained on LDI File share under its own subfolder. The IT Director, Project Leader and Web Team must be immediately notified if this location is changed for any reason.

## **Implementation Objectives**

The overall objective during implementation of these standards is to ensure and verify proper integration and consistency across the automated system or systems being developed or maintained. Correct integration within the department databases and current automated systems is crucial for the operation of the department. The process and scheme of implementation must be consistent with other databases and systems as to minimize the future development and maintenance costs to the department. Finally, only by consistent implementation can the department be secure in relying on the systems for dependable operation, and be secure in the accuracy of the information being stored and processed.

## **LDI Systems Integration**

A primary goal of the departmental software standards is the creation of a set of common interfaces within the department's databases which will allow the flow of information between databases, systems, and applications without error, maximize speed and decrease the need for special interfaces.

The introduction of all new systems **should** integrate into Department's database paradigm.

New systems working in concert will ensure that special data transformation programs and routines do not have to be developed, deployed and relied upon for daily department operations unless external sources or regulation requires otherwise.

## **LDI Acceptable Tools and Technologies by Category**

**The following tools and technologies are acceptable for the development of NEW LDI Software Systems:**

### **Operating Systems**

- Microsoft Windows Server Server 2008 / 2008 R2
- Microsoft Windows XP SP2 (Desktop) / 7 (Desktop)

### **Database Engines**

- SQL Server 2005/2008

### **Programming Languages, Tools and Technologies**

- NET Framework 3.5/4.0
- Visual Studio 2008/2010
- C#.NET
- ASP.NET
- XML
- Active Reports 3
- Telerik ASP.net radControls Suite
- Viewlet Builder
- EC Software Help Pro
- TNT Screen Capture
- AJAX – Visual Studio 2008/2010 (.NET framework 3.5/4 Built-in)
- SQL Reporting Services

### **Database Design Tools**

- Microsoft Visio 2007

### **Administrative Tools and Technologies**

- Microsoft Source Safe 2005
- Microsoft Project 2003 / 2007
- Red Gate SQL Data Compare
- XML Spy
- One Note 2007
- Sharepoint 2007
- Office 2007

### **Documentation Technologies**

- Adobe Photoshop
- Adobe Acrobat Pro
- Microsoft Word 2003/2007
- EC Software Help Pro
- TNT Screen Capture 2

**❖ For all maintenance performed, the original development environment and/or application initially used can be utilized.**

## **LDI Project Management**

The department has chosen Microsoft Project as its primary automation tool to assist in managing all IT projects. In addition to the reports which can be generated by Microsoft Project, the department has mandated that the following additional documents be produced.

- Requirements Document
- Design / Definition / Specifications Document
- Project Plan and Work Breakdown Structure
- Scope Document
- Weekly Status Reports
- Issue Descriptions
- Change Requests / Issue Description
- Sign Off Sheets
- Test Plan
- Test Plan Results

Requirements document, Scope document, Design / Definition documents, overall project plan and work breakdown are all due before programming on a project begins. These documents should at a minimum determine the functionality, operational capability, and features of the system, define the whole organizational structure of the system, explain any critical dates in the timeline of the project, illustrate any possible problems, and define the critical path for project completion.

A Gantt chart is appropriate method for displaying timelines and the critical path for a project.

Completed weekly status reports including timesheets, change requests/issue description sheets, sign off sheets, and test plan results are to be given to both IT and the respective division's IT coordinator for project tracking.

At minimum, the Scope Document must include, but is not limited to the following sections:

- Objectives of the project
- Scope of the project
  - In Scope items that will be developed
  - Out of Scope items discovered in interviews of staff that are beyond the original statement of work and should not be considered as items in the project
- Deliverables that will be produced
- Assumptions of the contractor in development of the scope document
- Risks associated with the project

## **LDI Project and Software Deliverables**

All software and maintenance projects at the department, either development projects or maintenance projects have a 1) software and 2) documentation component. These two components should be delivered in a form consistent with department standards in accordance with the LDI Acceptable Tools and Technologies by Category section of this standard.

All objects written on the SQL server should be written in Transact SQL or CLR languages.

## **Documentation Deliverables**

LDI requires that all documentation be a consistent organized collection of documents that describe the global structure, purpose, operation, maintenance and data requirements for a program. All documentation for LDI systems will include:

- User Documentation / User Manual
- Program Source Code and Technical Documentation

Program and Technical documentation should have at least these sections:

- **System Design Overview**
- **Operational Environment**
- **Object Reference**
- **Database Models**
- **Entity Relationship Diagrams**
- **Database Normalization**
- **Stored Procedure Reference**
- **Table Reference / Data Dictionary**
- **Integration with LDI's Integrated Database**
- **Security Reference**
- **ASP/ C# Source Code**
- **Stored Procedure Source Code**
- **SQL Script Printout**
- **Report Descriptions**
- **Deployment Instructions**

All documentation should be submitted in electronic format, Adobe PDF format preferred, in addition to paper.

Documentation deliverables are due when the software deliverable is completed or according to the contract between LDI and the contractor.

## **User Documentation/User Manual**

User Documentation/User manual is to be written from a client perspective. The purpose of the document is to empower the client to be self-sufficient.

### Application

- ❖ How To – Includes any of the following that apply to the project. All documentation is to be written in “User Manual” format. Include screen capture images for easy understanding.
  - Security Maintenance
  - Application Maintenance
  - Application Usage
  - Expirations – Any expirations which will affect the system (i.e. rollover of data, temporary permissions, and site and/or application certificates)

## **Technical Documentation**

All technical documentation should be written from a design and support perspective. The reader of the documents should be assumed to fully understand the technology and grasp the problem for which the system provides a solution. The only instance when technology should be explained is when the technology is being utilized in a non-standard method, or in a technique that has not been exploited previously by the department.

Standard technical documentation to be produced for all projects:

- **System Design Overview**
- **Operational Environment**
- **Object Reference**
- **Database Models**
- **Entity Relationship Diagrams**
- **Database Normalization**
- **Stored Procedure Reference**
- **Table Reference / Data Dictionary**
- **Integration with LDI's Integrated Database**
- **Security Reference**
- **ASP/ C# Source Code**
- **Stored Procedure Source Code**
- **SQL Script Printout**
- **Report Descriptions**
- **Deployment Instructions**

Below is a synopsis for each document:

- ❖ **System Design Overview**
  - Overview: Provide an overview of the system developed.
  - Program Specifications: Specifications developed in the planning phase of the project.
  - Functions: Specify the system/subsystem functions.
  - System/Subsystem Logic: Describe the logic flow of the entire system/subsystem in the form of a flowchart/diagram.
  
- ❖ **Operational Environment**

- Operations: Describe the operating characteristics of the user and computer centers or sites where the software will be operational.
- Equipment: Identify the equipment and software required for the operation of the software to be developed.
- Support Software: Describes any if needed.
- Interfaces: Describe and define all interfaces to the system. These include interfaces with other department databases and application systems, external databases and systems to the department and specific user interface requirements.
- ❖ **Object Reference**
  - Overview: Identifies and describes all program and data objects developed.
  - Sub Programs: Any separate subprograms required for system functionality are described.
- ❖ **Database Models**
  - Overview: High level description of database design and goals and database models developed for system.
- ❖ **Entity Relationship Diagrams**
  - Overview: All relationship diagrams for system.
- ❖ **Database Normalization**
  - Overview: Description of normalization implementation.
- ❖ **Stored Procedure Reference**
  - Stored Procedures: All stored procedures developed for system which are located on the departmental database are defined and described.
  - Special Dependencies: Any special or extraordinary stored procedure dependencies are defined and described.
  - Views: All database views are described and defined
- ❖ **Table Reference / Data Dictionary**
  - Table Schemas
  - Schemas and Table Definitions: All table layouts are defined with types, length and size where appropriate.
  - Database Diagrams: Diagrams with identifying primary and foreign keys with all indexes.
  - Data Dictionary: Document describing all fields by field name with description of information to be stored in data field.

- ❖ **Integration with LDI's Integrated Database**
  - Integration Description: Describes the implementation of the system with the departmental integrated database. This includes logical descriptions of program functionality.
- ❖ **Security Reference**
  - Model: Define the security model utilized for the system. All algorithms for security verification and encryption are described.
  - Database: All table, stored procedure, and view permissions by active directory users and groups.
  - User: All specific update, view, and create permissions by active directory users and groups.
- ❖ **ASP/ C# Source Code**
  - Source Code: All source code of the system divided by module.
- ❖ **Stored Procedure Source Code**
  - Stored Procedure Source: All transact SQL source code for all procedures, divided by procedure.
- ❖ **SQL Script Printout**
  - Scripts: All transact SQL source code for all scripts.
- ❖ **Report Descriptions**
  - Overview: Description of each report. Included are reporting requirements, all input parameters and sample output.
- ❖ **Deployment Instructions**
  - Overview: Detailed technician level instructions for installing the developed system.
  - Dependencies: Description of all required support software by operating system, operating system version, support software, support software version, and software location on server and client computers.

## **LDI Application Look and Feel Guidelines**

All applications and systems developed for the department should conform as closely as possible to the Windows design metaphor. In addition to conforming to the Windows design metaphor, applications need to conform to the current color scheme used by the department. The department has chosen Microsoft products as the department's standard for base installations on all department workstations. Designing programs and systems to closely conform to the Microsoft standard will minimize the training costs and reduce the time required for users to become familiar with a new application or system.

The Microsoft guidelines can be found in the Microsoft publication, *Official Guidelines for User Interface Developers and Designers*. This manual can be found on the Microsoft website at the following address:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/ch00b.asp>

All deviations or extensions from this standard must be approved by the IT division. Additional approval may be required from the IT coordinator for the division or divisions for whom the application or system is being developed.

## **LDI Database Normalization**

Correct data expression is of utmost importance to the department. A *third normal form* is desired for the relational database developed for the LDI. A relational database that satisfies the 1NF, 2NF, and 3NF is sufficiently well-designed to support all mission critical-business applications.

Strict definitions of the fourth and fifth normal forms should be used sparingly since the cost/benefit relationship typically does not warrant their utilization and can actually result in performance degradation.

## **Network, Servers and Miscellaneous Standards**

- All new servers set up for development which may require input from the Internet must be secured via a web security certificate. It is the responsibility of the developer requiring the new server to consult with LDI Network and Maintenance contractors as well as LDI technicians to determine whether the new server requires a security certificate.
- Applications shall not be developed and tested on a production server.
- All new servers set up to house web applications externally accessible must be on the DMZ. There will be no exceptions. It is the responsibility of the developers who are setting up the server along with the Network contractors and IT technicians to see that this standard is enforced.
- Backing up new servers.
  - a. All new servers, regardless of whether or not they are used for development, testing or production must be added to the SANS for backup. There will be no exceptions.
  - b. All code on development/testing and production servers shall be placed on a drive other than the C: drive. In other words, SOURCE CODE OR SITE DIRECTORIES SHALL NOT be placed on the C: drive.
- Delivery of documentation and source code from a developer to the department will be completed using the Department of Insurance Sign Off sheet. The time and location for a smooth turnover shall be agreed on by all parties and signed off.
- Session time-outs need to be evaluated by all the developers on-site in order to set the correct time-out. Not all session time-outs are the same.

- All programmers and technicians working at the LDI must share and discuss their problems and solutions as well as their progress regarding the applications they are developing. In order to affect this exchange of ideas, monthly meetings will be held which must include the developers and project managers from every developing entity at the LDI.
- Logins, badges, and security. All contractor accounts will be disabled by IT immediately upon completion of a contract when a contractor's services are no longer needed. The badges will be collected by the IT technical staff in charge of Active Directory.
  - a. IT Technical staff in charge of Active Directory will disable the active directory accounts based on a list submitted by the IT Project Leader.
  - b. The IT staff or Contract / DBA will disable all database and/or server accounts from a list submitted by the IT Project Leader.

## **LDI Development, Test and Production Environments**

In order to create the most robust environment for systems development and production, the areas of development and production have been logically and physically separated into the following regions:

- Development
- Testing
- Production

Only in the **system development environment** are databases and program code objects created, and modified and actual system or program coding occurs. The development environment is characterized by the unique ability of the developer to make dynamic changes to their system development area without prior authorization or coordination.

During and after development, all system code developed or updated is controlled and cataloged through Microsoft Visual Source Safe. The location and management of the source safe server will be controlled by the LDI IT department.

The **application testing environment** is an exact replica of the production system which the system is designed to function. This environment allows the developed code to be tested against final production schemas before being moved into production. The primary difference between the development environment and the testing environment is the ability for the developer to make on the fly changes to the underlying database. The testing databases are not to be changed without prior coordination. The testing environment is to be used for all system and program testing by the developers, LDI IT staff and LDI users.

A SQL job is scheduled and routinely executed to keep the data in the test environment consistent with the production environment.

The **production environment** is logically and physically separated on different servers. LDI internal application databases are also physically

separated from external and Internet and WEB application databases. This allows for maximum flexibility and security with departmental data. The production environment is designed for maximum uptime and the fastest possible response time. As a result, no on the fly changes are allowed within the production environment. All changes to either the databases or system programs have to schedule with LDI IT staff, support personnel, and the LDI division responsible for the automated system.

Unhandled exceptions that cannot be corrected immediately should be trapped and the captured information sent to a log file for evaluation. During the testing and debugging phase of development, errors do not have to be trapped on the development and test servers but these errors are unacceptable in the production environment so the developer should use their own procedures to correct these errors in the development environment. A procedure of adding a “continue” or a “cancel” routine and/or button on the interface is not acceptable.

## **Methods and Procedures to Move New Systems and Updates into the Production Environment**

Once a newly developed system or existing system update has been thoroughly tested by the developer, and department staff, and the department staff has approved the system update; then the system can be moved to the production environment for operations.

Due to the complex nature of the department's automated systems, exact methods and procedures for shifting systems or programs into production cannot be standardized.

LDI staff, working in conjunction with the entity responsible for completing the project, will determine the best method for transitioning a new system or updated system into the LDI production environment.

## **System and Application Launching Point**

All developed systems and applications are launched from the department's Intranet site. The Intranet is the only departmentally approved location for a system to be launched. Within the Intranet, shortcuts and aliases to the actual program or initiation program are allowed. The program or application itself is not to be stored on the Intranet server. All links from the Intranet are to be relative in nature, and absolute locations are never to be used.

Program and system dependencies for the client workstations, which are dynamic in nature, and can be updated on the fly, are to be stored within the program or system directories. The dependencies are then updated upon launching the program or system by the user when the user initiates the program.

For Web-based application, the timeout period should be set for 20 minutes. If approved by the IT Project Manager, the timeout period can be extended to 45 minutes but written justification must be submitted with the request.

## **Permissions Structure, Passwords, and Supported Computer Software**

All system, application and database permissions are handled by the Windows Server active directory, and roles defined within the database server.

All users within the department are divided into functional areas or groups. Windows Server has the groups defined within its active directory. Users are defined by which group they belong to. The group itself determines which permissions or abilities the group has. All database roles are a subset of the group as defined within the active directory. Together the group's permissions and database roles describe all actions a user can perform on the departmental network. The Department's Active Directory will control the internally developed application and web-based systems must have passwords that must meet the requirements of LDI IT SEC-POL-003 as adopted by LDI. To meet the current requirements for password complexity, the password is required to be at least eight (8) characters in length, case sensitive, and contains at least three (3) of the following categories:

- English upper case
- English lower case
- Base 10 digits
- Non-alphanumeric characters (% , & , ! , etc.)

No blank passwords are permitted.

No other method or technique of managing users or user's permissions is allowed.

As per IT SEC-POL-010, no non-IT supported software is to be installed on computers. If you wish to test new software, you are responsible to report the software to IT for determination if the licensing criterion for the software is compliance with State mandates and LDI's policy.

## **Naming Conventions and Data Standards**

All tables, stored procedures, database views, code modules, programs, and reports should comply with the department's naming conventions.

The department's current naming convention for tables, stored procedures, database views, and reports has the application name followed by an underscore preceding the name of the object. This standard is consistent across all systems.

All code or reference tables have an underscore and the characters cde following the table name.

Currently, the department is developing and extending the departmental name convention guide lines.

Database fields cannot be an empty string or spaces. Nulls must be used in the fields. Date fields must contain either a date or a null.

## **Appendices**

## **Coding (Language) Recommendations**

### *Overview*

A comprehensive coding standard encompasses all aspects of code construction and not just the language used. While developers should prudently implement a standard, it should be adhered to whenever practical. Completed source code should reflect a harmonized style, as if a single developer wrote the code in one session. At the inception of a software project, establish a coding standard to ensure that all developers on the project are working in concert. When the software project incorporates existing source code, or when performing maintenance on an existing software system, the coding standard should state how to deal with the existing code base.

The readability of source code has a direct impact on how well a developer comprehends a software system. Code maintainability refers to how easily that software system can be changed to add new features, modify existing features, fix bugs, or improve performance. Although readability and maintainability are the result of many factors, one particular facet of software development upon which all developers have an influence is coding technique. The easiest method to ensure a team of developers will yield quality code is to establish a coding standard, which is then enforced at routine code reviews.

Using solid coding techniques and good programming practices to create high-quality code plays an important role in software quality and performance. In addition, if you consistently apply a well-defined coding standard, apply proper coding techniques, and subsequently hold routine code reviews, a software project is more likely to yield a software system that is easy to comprehend and maintain.

Although the primary purpose for conducting code reviews throughout the development life cycle is to identify defects in the code, the reviews can also enforce coding standards in a uniform manner. Adherence to a coding standard is only feasible when followed throughout the software project from inception to completion. It is not practical, nor is it prudent, to impose a coding standard after the fact.

### *Language*

Choosing a programming language depends on your language experience and the scope of the application you are building. While small applications are often created using only one language, it is not uncommon to develop large applications using multiple languages. For example, if you are extending an application with existing XML Web services, you might use a scripting language with little or no programming effort. For client-server applications, you would probably choose the single language you are most comfortable with for the entire application. For new enterprise applications, where a large team of developers create components and services for deployment across multiple remote sites, the best choice might be to use several languages depending on developer skills and long-term maintenance expectations.

Because of the wide range of choices for developing a new system under, it is recommended that the project leader be responsible for the ultimate platform decisions of a project.

The .NET Platform programming languages — including Visual C#, Managed Extensions for C++, and many other programming languages from various vendors — use .NET Framework services and features through a common set of unified classes. The .NET unified classes provide a consistent method of accessing the platform's functionality (Microsoft Windows).

The following sections identify the preferred current standard language of the LDI and will help the project leader choose the right programming language for the project. If the developer determines that Visual C#.net **cannot** be used in the development of the project, they must document the reason and present the documentation to the Project Leader for approval as per the Exception, Changes, and Amendments to Software Development Standards section of this document.

## Visual C#.NET

Visual C# (pronounced C sharp) is designed to be a fast and easy way to create .NET applications, including Web services and ASP.NET Web applications. Applications written in Visual C# are built on the services of the common language runtime and take full advantage of the .NET Framework.

C# is a simple, elegant, type-safe, object-oriented language recently developed by Microsoft for building a wide range of applications. Anyone familiar with C and similar languages will find few problems in adapting to C#. C# is designed to bring rapid development to the C++ programmer without sacrificing the power and control that are a hallmark of C and C++. Because of this heritage, C# has a high degree of fidelity with C and C++, and developers familiar with these languages can quickly become productive in C#. C# provides intrinsic code trust mechanisms for a high level of security, garbage collection, and type safety. C# supports single inheritance and creates Microsoft intermediate language (MSIL) as input to native code compilers.

C# is fully integrated with the .NET Framework and the common language runtime, which together provide language interoperability, garbage collection, enhanced security, and improved versioning support. C# simplifies and modernizes some of the more complex aspects of C and C++, notably namespaces, classes, enumerations, overloading, and structured exception handling. C# also eliminates C and C++ features such as macros, multiple inheritance, and virtual base classes. For current C++ developers, C# provides a powerful, high-productivity language alternative.

Visual C# provides solutions for the following:

- Windows Application.
- Class Library.
- Windows Control Library.
- ASP.NET Web Application.
- ASP.NET Web Service.
- Web Control Library.
- Console Application.
- Windows Service.

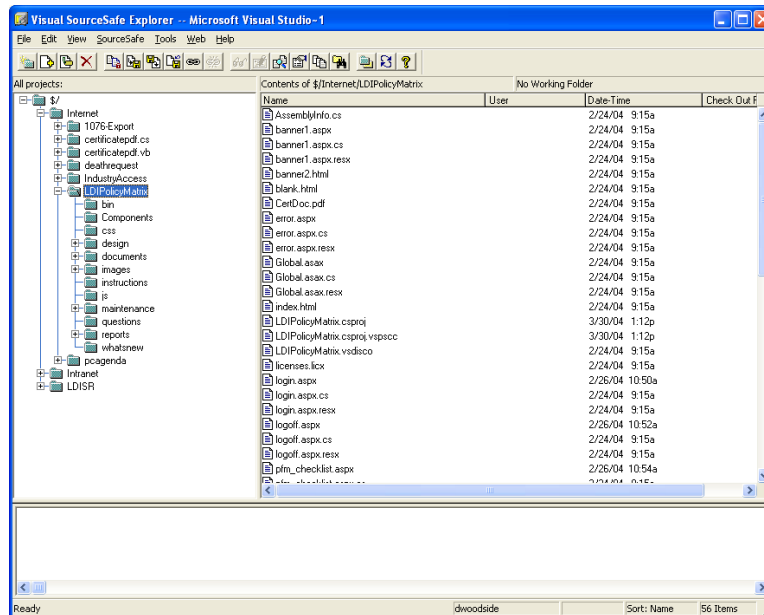
### *Naming Guidelines*

It is recommended that the department adopt a unified naming guideline for all development. Microsoft has a fairly complete document which should serve as a start. It can be viewed at the following location:

## Source Safe Recommendations (Source Control)

From a simplistic perspective, a Software Configuration Management (SCM) product, like Visual SourceSafe (VSS), is a central library, or database, of documents that make up a project. Visual SourceSafe can store any stream of bits, such as project plans, specification documents, database objects, source code, and any other artifacts of your project. As a best practice, all project artifacts, not just source code, should be contained in a Visual SourceSafe database for easy access, sharing among team members, and most importantly, for version control. This database should be centrally located and administered by an assigned LDI staff member.

It is recommended to use Visual SourceSafe since it keeps with the department standard of utilizing a Microsoft platform.



## Integration Guidelines

### *Overview*

The Louisiana Department of Insurance is presently going through a major rewrite of its systems. Most of the new systems being developed are being custom developed onsite. Each of these project groups is assigned a business unit to work within but little communication about system design and business logic crosses from group to group. To facilitate this communication an internal process needs to be put in place to assist in bringing these groups together. It is recommended that Project Leader perform the following task:

- Set IT strategic direction for the department (with approval from the IT director and selected department leaders)
- Identify department leaders
- Review ongoing status of projects (Measuring a Project's Success)
- Set development guidelines
- Resolve issues which affect multiple projects

### IT Strategic Plan

Simply put, strategic planning determines where an organization is going over the next year or more, how it's going to get there and how it'll know if it got there or not. *The focus of a strategic plan is usually on the entire organization.*

Quite often, an organization's strategic planners already know much of what will go into a strategic plan. However, development of the strategic plan greatly helps to clarify the organization's plans and ensure that key leaders are all "on the same script". Far more important than the strategic plan document, is the strategic planning process itself.

The vision put forth in the plan should project into the future an IT environment that will deliver needed services to LDI staff and outside parties. The vision for IT is to support and align projects with the Business Plan and the needs of the industry. The specifics of LDI's future IT environment are directly linked to the agency's strategic business plan that provides the programmatic framework for all LDI work. By practicing strict strategic alignment of IT resources with programmatic priorities, LDI can achieve a match between industry and internal needs and technical sophistication.

### Identifying Department Leaders

To provide needed support for IT initiatives, key leaders in the LDI need to be sought out and a working relationship built. The first step should be in identifying all key system at the department and creating owners of these systems in the client community. This ownership inherently builds a relationship between IT and the clients using the systems.

These key department leaders know firsthand the business and the issues the systems need to resolve.

### Review Ongoing Status of Projects

To accurately measure the success of a project, there must be a set of standards to measure against. While all projects undertaken are unique and have varying measurements, there are some which are common to all projects.

Common measurements:

- Adherence to timeline
- Adherence to budget
- Percent project complete (based on task from the function specification but not budget or timeline)
- Level of client satisfaction
  - Client response to demonstrations of product during each iteration
- Other measurements defined by the project leader and the primary client
  - Adherence to statement of work
  - Report from test results

A standard report should be developed which takes these measurements into account and reported on a regular interval.

### Setting Development Guidelines

The IT Division should set guidelines for not only defining the platforms the applications either execute on or the technology within which they are developed but should also set the standards by which the applications are developed. This includes but is not necessarily limited to programming frameworks, languages, naming conventions and documentation.

### Resolve Issues which Affect Multiple Projects

By being involved with all the projects at the LDI, the IT Division will from time to time come upon issues which affect multiple projects or the entire organization. These issues should be resolved so that the organization receives a unified result. As an example, over time there have been issues which have come up at the department (i.e. a consistent numbering schema for entities) where multiple solutions have been implemented by various projects. In the end, a unified solution will need to be implemented and the existing systems converted.