

Sanitized Selections from:
xxxxxx and xxxxxxxxxxxxxxxx
Project (xxxxx) - Phase I
California Department of XXXX XXX
Software Design Description



Version: 1.04

Revision Date: Mar. 28, 2006

<Contracting Company Information>

1 System Architecture

xxxxx I is a Web-enabled application. On the client-side the end user performs the tasks for which they receive authorization on any browser on any computer connected to the network. On the server side, the application runs on the designated server as determined by ITSS.

System Hardware Architecture

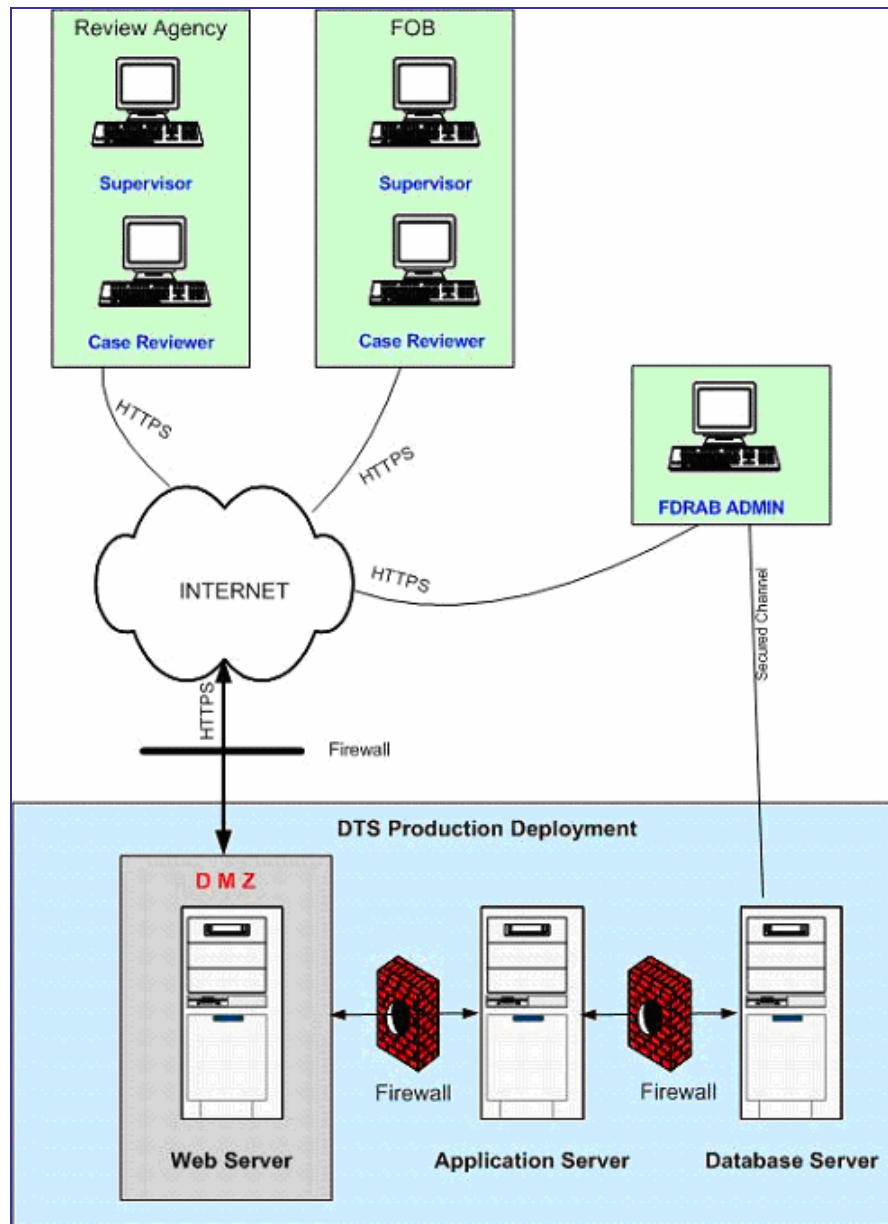


Figure 1: System Hardware Architecture

Web Server

OS	Microsoft Windows 2003 Advanced Server
Web Server	IIS 6
.Net Framework	.Net 2.0

Application Server

The application server layer will integrate and provide the presentation layer with access to the SQL database, transactions and business objects using standard well-defined web services based on a service-oriented architecture. The function of the application server is to act as host (or container) for the user's business logic while facilitating access and performance of the business application.

OS	Microsoft Windows 2003 Advanced Server
Web Server	IIS 6
.Net Framework	.Net 2.0

Data Server

OS	Microsoft Windows 2003 Advanced Server
Database	SQL 2000 Server

Requirements

This architecture satisfies the following technical and business requirements:

Req. #	Requirement
TR-003	System must operate within the existing telecommunications/computer environment supported by the CDSS and the Department of Technology Services (DTS). Connectivity for the systems will be over the public internet.
TR-006	System must execute on a Windows 2000 (or later) desktop environment, running a Microsoft Internet Explorer (MSIE) browser version 5.5 or later.
TR-007	Software development environment management and administration tools and applications must be consistent with a Windows 2000 and Windows 2003 server environment.
TR-017	All user interfaces, including all data entry, workflow, reporting, and administrative screens, must be completely web-based and work with Internet Explorer version 5.5 and later.
TR-018	System must utilize a centralized server based application that operates as a web garden environment. A web garden is defined as a server that hosts multiple web applications, as opposed to a web farm that is defined as multiple servers hosting a single web application.
TR-019	System must utilize Microsoft SQL Server 2000 as its central DBMS.
TR-020	System must support the following common database connectivity protocols: ODBC, ADO.NET, and OLE DB.

Req. #	Requirement
TR-025	System must utilize a centralized database and avoid data duplication by enforcing relational database normalization standards, specifically 3 rd normal form (e.g. each attribute of the relation is atomic, each non key attribute in the relation must be functionally dependent upon the primary key, and all attributes that are not dependent upon the primary key must be eliminated).
TR-070	System must implement standard Windows 2000 (and later) print functions, including print preview, and support for both laser and dot matrix printers for all system generated printed output.

System Software Architecture

The follow diagram provides an overview of the XXXXX I system architecture. This architecture has three layers: a Web server (or presentation) layer, an application server (or business) layer, and a database server (or data) layer:

- The Web server layer includes the (1) user interface and the (2) user interface process components.
- The application server layer includes the (4) Web services, the (3) business components, and the (5) data access logic components.
- The database server layer includes the database.

Each layer resides on a separate physical machine.

Security, operation management and communication components are present in both web server layer and application server layer, as indicated in the left-hand column.

- The security component (7) includes web service enhancement 3.0 library.
- The operation management component (6) includes exception management libraries.
- The communication component includes the business entities (8)

Each component is described in the sections that follow

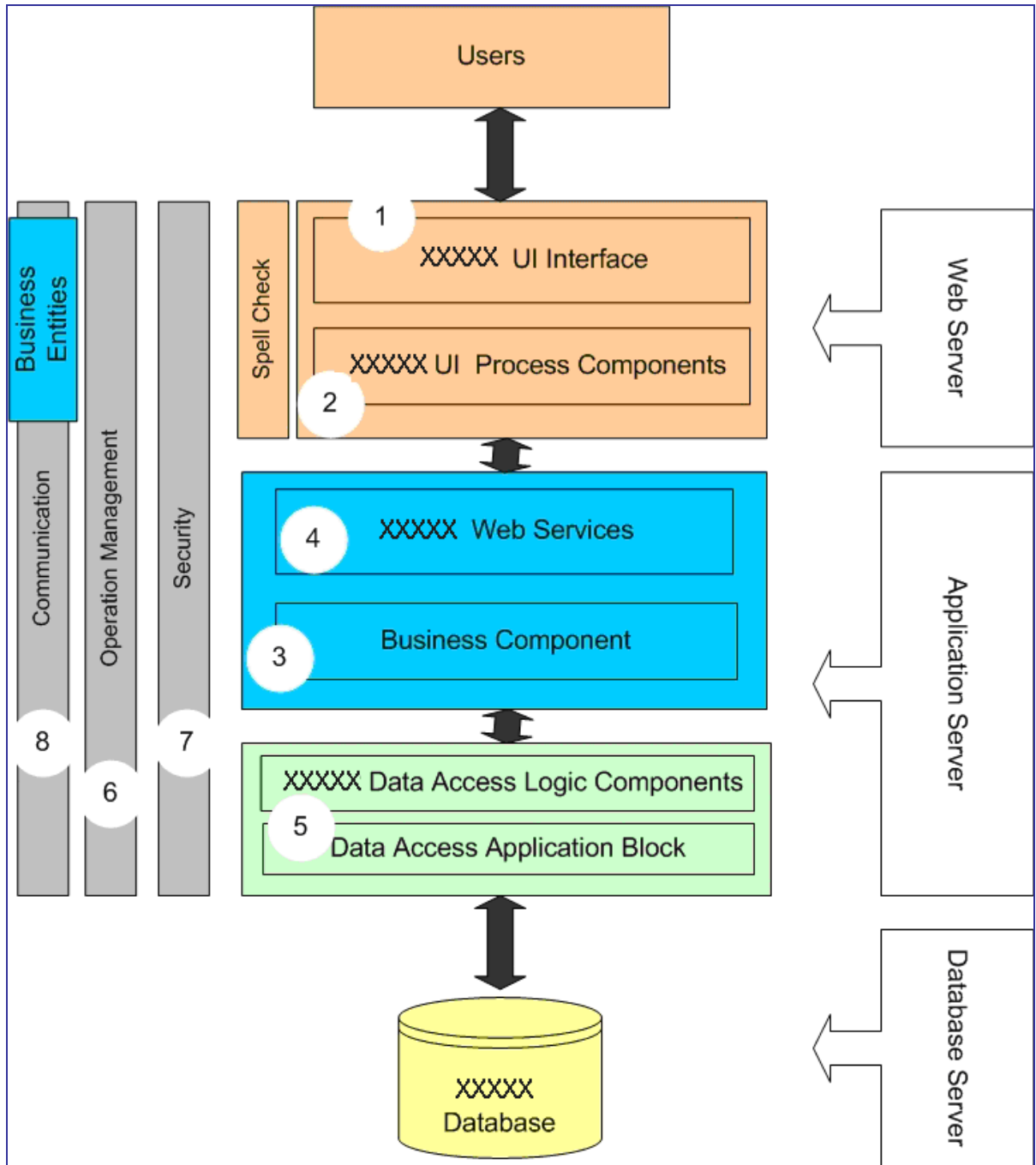


Figure 2: XXXXX I system architecture

User interface (UI)

An application user interface provides the means for users to interact with the application. In the XXXXX I application, a Web browser allows case reviewers to view and update Zzzz zzzzz and YYYY cases and validate the information. User interfaces are implemented using Microsoft

ASP.NET pages and user controls. These interfaces render and format data from the database, as well as acquire and validate new data entered into them.

User process components

In many cases, a user interaction with the system follows a predictable process. XXXXX I will utilize separate user process components synchronize and orchestrate these user interactions. The process flow and state management logic is not hard-coded in the user interface elements themselves, but rather, this code lies behind Web page files and is compiled into the component.

XXXXX I also includes a spell checking component in this application. This component fits in the presentation layer of the application and works along with the user interface and user process components.

Business components

Business components implement the business logic of the application. Business components hold all business logic of the application. Data is always traversed through this component before showing it to user and before sending it to database.

Web services

In the XXXXX I three-tier architecture, business logic components are placed in middle layer server. A means is required to expose the method of this component to other systems, including the front end server. Although.NET Remoting and Web Services are two different methods to achieve this kind of communication, XXXXX I will use Web-services to achieve this communication.

Data access logic components

XXXXX I components and services must access a data store during a business process. Best practices suggest we abstract the logic necessary to access data in a separate layer of data access logic components. Doing so centralizes data access functionality and makes it easier to configure and maintain. This can be illustrated as follows:

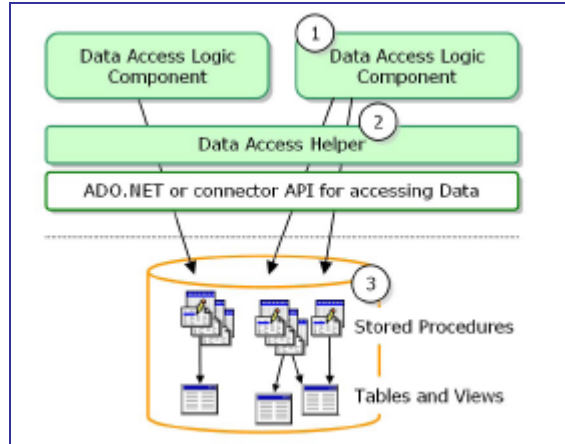


Figure 3: Data access logic component interaction

1. Data access logic components expose methods for inserting, deleting, updating and retrieving data.
2. XXXXX I uses a data access helper component (Data Access Application Block from enterprise library) to centralize connection management and all code that deals with a specific data source.
3. XXXXX I implements queries and data operations as stored procedures to enhance performance and maintainability.

Data Access Application Block

The Data Access Application Block is a .NET component that contains optimized data access code that will help us call stored procedures and issue SQL text commands against a XXXXX I database server.

The Data Access Application Block encapsulates performance and resource management best practices for accessing Microsoft SQL Server™ databases. XXXXX I will use Data Access Application Block of the Enterprise Library for .NET 2.0.

Operation Management

Exceptions

Management

Despite programmers' very best efforts, software code may contain minor undiagnosed flaws. Exceptions can result from either these software flaws or hardware malfunctions.

Exception management encompasses catching and throwing exceptions, designing exception handling policies, flowing exception information, and publishing exception information as appropriate to designated users.

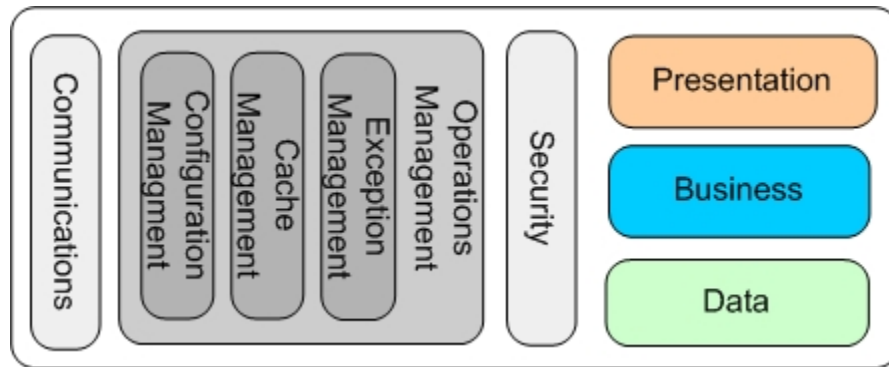


Figure 4: Operational management processes span all architecture layers

XXXXX I will implement Exception Management Application Block of the Enterprise Library for .NET 2.0 to catch runtime errors. Exceptions will be caught and resolved whenever possible. If an error state cannot be resolved, the application will display a meaningful message to the user and provide a means of logging or publishing the exception information for debugging purposes. XXXXX I will use Logging and Instrumentation Application Block from the .NET Enterprise Library to log exceptions.

Logging

When exceptions occur, XXXXX I will log the following information concerning the exception:

1. General Info :
 - Date & Time of Exception
 - Machine Name
 - System User
 - Logon User
 - Client IP-Address
 - Host IP-Address
 - Web-URL
2. Assembly Info
 - NET Runtime Version
 - Assembly Name
 - Assembly Path
 - Assembly Version
 - Assembly Build Date
 - Application Domain
3. Exception Info
 - Exception Number
 - Exception Message
 - Exception Type
 - Exception Source
 - Exception Target Site
 - Stack Trace

4. Inner Exception Info
 - Inner Exception Number
 - Inner Exception Message
 - Inner Exception Type
 - Inner Exception Source
 - Inner Exception Target Site
 - Inner Stack Trace

This information will be logged first into database, then second, logged into Windows local event log and finally, emailed to the designated system administration staff person.

Parameters required for logging (such as database name, database server name, to and from email IDs, SMTP server name, and so on) will be configurable by a separate configuration files distinct from the web.config file, as per the architecture of Enterprise Library.

The Windows server operating system provides a set of event logs. For example, the operating system places error details in the system event log, while an application event log is provided by Windows for applications' use. The event log provides a consistent and central storage repository for error, warning, and informational messages on a single machine. The XXXXX I application's log entries will be entered in a separate log to be called XXXXXlog. Using event log as a source for application log entries makes it easy to compare the sequence of system and application events. Event logs can be configured to maximum size and number of days to maintain messages. (This task will be done by a system administrator outside the application development process.) They can be viewed and manipulated by Event Viewer which is a familiar system administration tool. All exceptions will be logged in a single, centralized location which is accessible remotely.

Although it is possible to write logs from different servers to one server; this approach increases the risk of failure. Therefore, we will write event logs to individual servers, even though this makes diagnosis more cumbersome. In order to minimize this problem, we will use a centralized database to log all errors sequentially. Database tools can then be used for querying and reporting error data.

As our strategy already specifies that we are writing exceptions to a local event log, we minimize risk of losing data while storing it to the database due to system failure or network failure. In addition, we have specified an email notification of exceptions, thus further ensuring no exception data will be lost.

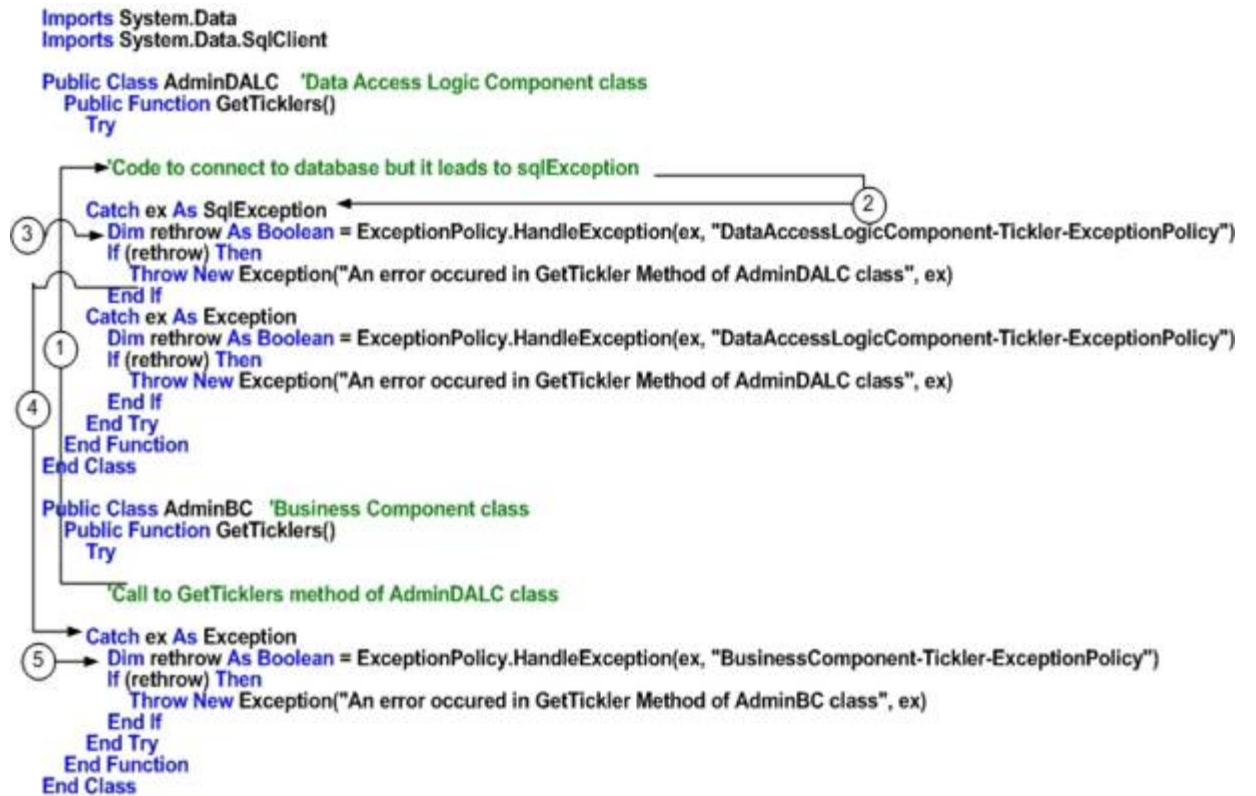
Using three different logging strategies (event log, database, email) ensures that in the event of the failure of any one logging strategy, essential exception data will not be lost.

Propagating

Exceptions will be logged as soon as they are captured. Each exception will be wrapped in a new exception with a user friendly message and thrown to caller method. Each new exception will be of type System.Exception. Exceptions may be logged multiple times, but each time an exception is logged it will contain additional relevant information about where it occurred and what happened.

For example: An exception occurs in a data access logic component in one of the modules. The exception type is SQLException. As soon as exception occurs it is captured and logged as described. The exception is wrapped in new exception of type System.exception with an explanatory message, such as, "An error occurred in the <xxx> method of the <yyy> class," and it will be thrown to its caller method. Assuming it has been called by the Business Component method, a new exception of type system.Exception with corresponding explanatory

message is captured by Business Component method. This new exception will be logged with a new explanatory message and the old exception wrapped within. This may be illustrated as follows:



This level of detail is maintained within the error logs. However, this information will not appear to the XXXXX I user. Instead, the user will view an error message similar to the following. Most of the information is generic, with the exception of the Date and Time of Error and the Error Number. This information will be helpful to the Help Desk in locating the error in the logs.

XXXXX I has encountered an unexpected problem

What happened:
An unexpected error occurred in XXXXX I. The current page will not load.

What can you do about it:
Close your browser, then open it again and return to the page you were on. Try to repeat what you were doing when the error occurred.
If the problem repeats, call Help Desk at 123.456.7890 or email xyz@dss.ca.gov and give us the following error information:

Error information:

Date and time of error:	01/01/06 12:00 PM
Error Number:	1201

Numbering

Exception number or error number will be derived from a code number assigned to the component number and module number where the exception occurs, along with an exception type number. The following tables list the code numbers for each component and module:

Note: All of the following tables will be updated during implementation.

Component Name	Number
UI Process Component (UIPC)	1
Business Component (BC)	2
Business Entity Component (BEC)	3
Data Access Logic Component (DALC)	4
Security Component	5

Module Name	Number
Login	01
Log Off	02
Change Password	03
Reset Password	04
Home Page	05
User Export	06
Create User	07
Manage User	08
Manage Role	09
Manage CRA	10
System Configuration	11
Audit Configuration	12
Data Archive	13
Audit Trail Archive	14
View Audit Trail	15
Tickler	16
Delegate	17
Search	18
Review Case	19
Case Load(Supervisor)	20
Case History	21
Data Collection	22

Module Name	Number
Manage Global Tickler	23
Profile Data	24
Load Universal Files	25
Load Sample Files	26
Delete Sample Load	27
Load Sub Sample	28
Load Error	29
Administrator Case Load	30
Federal Export Files	31
List Metadata	32
Manage Metadata Elements	33
Manage UiSections	34
Manage Business Rules	35
Manage Skip Rules	36

The following table shows name of exception policies and their numbers. The exception policy number is formed by combining the component number (first digit) and the module number (second digit).

In order to handle exceptions, XXXXX I will use .NET Enterprise Library. In .NET Enterprise Library we will create policies for each combination of component and module.

Exception Policy Name	Number
UIProcessComponent-Login-ExceptionPolicy	101
UIProcessComponent-Tickler -ExceptionPolicy	116
UIProcessComponent- LoadError-ExceptionPolicy	129
BusinessComponent-Login-ExceptionPolicy	201
BusinessComponent-Tickler -ExceptionPolicy	216
BusinessComponent-LoadError-ExceptionPolicy	229

The exception type names and numbers are listed as follows:

Exception Type Name	Number
System.Exception	00
System.SqlException	01
System.Security.SecurityException	02

Depending on system namespaces used in any particular component, we will include exception types in that exception policy. Since SqlException occurs only in data access logic component, we will only add that exception type in those particular exception policies.

Since all exceptions are derived from System.Exception namespace, we will include this exception type in each policy and it will be always numbered as 00.

For example:

Exception number	Is the:	Digit	Indicates:
20400	Component	2	business component
	Module	04	Reset Password
	Exception type	00	System.Exception
40201	Component	4	Data Access logic component
	Module	02	Log Off
	Exception type	01	System.SqlException

In order for this logging to be successful XXXXX I will require some customization in the Exception Management Application Block of the Microsoft .NET Enterprise Library.

Cache management

XXXXX I will use ASP.NET as an application development language. It provides the following three means of cache management:

- **Data cache** will cache following data.
AllCRAs , AllCounties , AllSampleType , ColorSchemas , GlobalTicklers , FileFormats , SystemParamters , UserDelegatedFunctions , TicklerTypes , UserMenuItems , AllSection
- **Output cache:** content of grid data, Global ticklers page etc
- **Fragment cache:** during data entry, the previous section data/page will be cached when moving to next section

Configuration management

XXXXX I will use the following three different ways to store configurable parameters.

Web.Config—saving the web.config file restarts the application. To avoided frequent restarts we will store some attributes in configurationsections of web.config which points to a secondary configuration file and does not cause an application restart when it is changed. In XXXXX I, parameters which are required by Enterprise Library configuration files are actually placed in configurationsections of web.config file.

<<XYZ>>.config—Enterprise Library creates its component-specific configuration files. For example when we use the Data Access Application Block of Enterprise Library, it creates the dataconfiguration.config file to store parameters related to that block. For some block, it creates multiple configuration files. Since we are using Enterprise Library for operations management of the application we will have several of these configuration files.

Database—any changes to the parameters which determine the application flow must be saved to the database in order to allow them to be audited. This is in contrast to changes to the web.config file which cannot be audited by application. Parameters which determine application flow are as follows:

- Password expiration Notification days
- Password expiration days
- Number of old password
- Shutdown Date
- System Shutdown Message
- Notification Minutes
- Max Number Failed Logins
- Sub Sample Turn Around Days
- Min Years Case Data To Retain
- Max Number Failed Logins
- Audit Log Retain Days
- Audit Log Day of Week
- Audit Log Day of Month
- Audit Log Audit Path
- Audit Log Archival Frequency

Security component

Security components manage authentication, authorization, secure communication, auditing, and profile management for all the layers of XXXXX I.

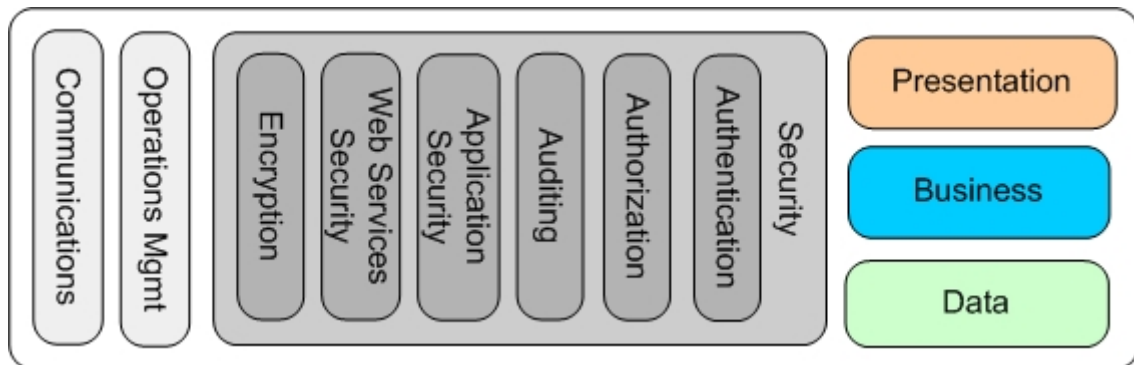


Figure 5: Security component processes span all architecture layers

Authentication

Authentication is the process of obtaining identification credentials, such as name and password from a user and validating those credentials. If the credentials are valid, the user who submitted the credentials is considered an authenticated user. Once a user has been authenticated, the authorization process determines which functions this user has access to.

RAPEP I will use the customized CDSS Authentication approach where the application will have an interface to accept the user name and password and validate them against the database. XXXXX I will also use a class provided by CDSS. This class ensures that each page has been passed by the authentication process.

Authorization

The purpose of authorization is to determine whether a user should be granted the requested type of access to a given functionality.

In XXXXX I, authorization will be implemented by a role-based methodology, in which a particular role has access to one or more functionalities and a user will belong to one or more roles. This approach groups the person with others who do similar functionalities in the application.

Auditing

Auditing refers to monitoring the operations performed on the application, when and by whom.

XXXXX I will have customizable audit functionality. Auditing customization allows the logged in user to turn off auditing of the Category. Every request to XXXXX I will be audited, even if the request is a simple page refresh.

By default, XXXXX I will log following information for each module:

- **IPAddress:** IP address of the web browser making a request to the XXXXX I web site
- **UserID:** User ID of an authenticated user; for unauthenticated requests this data will be stored as "Null."
- **PageID:** XXXXX I will have numerous functionalities as described in Function table in LDM. These functionalities are stored in system with some number. This number will be captured in this field.
- **Action:** Create, Review, Update, Delete
- **KeyValue:** Value which identifies a table name and unique key in the data base:
 - ☞ Pages, such as Ticklers or Case Load which list data have their Action logged as 'V'iew and KeyValue as 'null'
 - ☞ Pages showing specific information have a KeyValue specific to that page; for example, Review Case and Add Tickler, have their Action logged as 'V'iew and their KeyValue as ReviewNumber, or TicklerID
 - ☞ Pages adding or modifying specific information have a KeyValue specific to that page, for example Review Case and Tickler, have their Action logged as 'A'dd or 'M'odify and their KeyValue as ReviewNumber or TicklerID
- **AccessDateTime:** Date and time stamp of the request made to the XXXXX I web site.

Application and Web Services Security

Whenever a user makes a request for data or submits data from a web client, the data transmitted from the web client (that is, the end user) to the web server and vice versa is encrypted with SSL to ensure security.

Data transmitted from the web server to the web service is decrypted before transmittal to the web service (located on the application server.)

To validate a request from the web server to a web service, we will use message level security using an X.509 certificate for authenticating the requestor. This certificate is either purchased from a public certificate authority (CA) or created from a properly configured PKI server. The certificate resides on the application server.

When the web server sends a message, it encrypts the sensitive parts of the data and digitally signs the message using the X.509 certificate with its private key. When a web service receives the message, it uses the public key, which is included with the X.509 certificate, to validate the signature. Additional validation may be required to ensure that the X.509 certificate has not expired and was issued by a trusted CA (certificate authority) service.

The following describes the process of authentication using an X.509 certificate in greater detail:

1. The web server sends message to the web service. The message includes the web server's credentials, signed with the private key that is paired with the public key in the web service's X.509 certificate.
2. The web service validates the certificate by doing the following four verifications:
 - Verifies the certificate has not expired
 - Verifies the certificate contents against the signature of the issuing CA to make sure the certificate is internally consistent and has not been tampered with
 - Verifies the signature of the issuing CA from the web server is equivalent to the certificate from the CA
 - Verifies that the CA has not revoked the certificate
3. The web service uses the public key in the web server's X.509 certificate to verify the web server's signature. This ensures the message was not tampered with after it was signed.
4. The web service sends a confirming message back to the web server.

X509 certification for authentication is illustrated as follows:

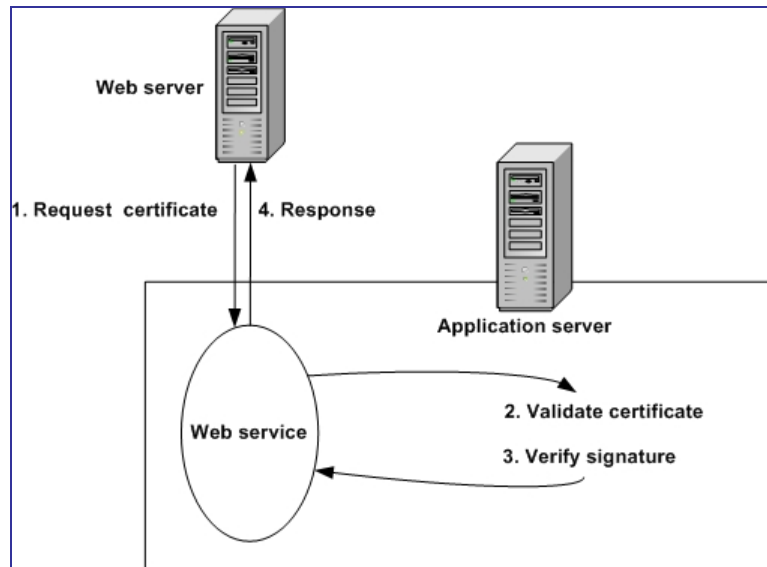


Figure 6: X.509 certificate authentication process

When the user sends a login request with its password, XXXXX I will hash this password at the web server level. The web server will send this hashed password to the database via the application server. This hashed password will be compared through a stored procedure against the hashed password stored in the database for the requested user. If the password matches the stored procedure sends the result as **true**; otherwise it sends **false**.

All sensitive data passed from any layer to layer with XXXXX I will be encrypted or hashed.

Data transmitted within the application server from web service to the business or data access components remains encrypted and the password remains hashed.

Data transmitted from the business and data access components to the database server remains encrypted. This format maintains the ultimate level of security; even the database administrator cannot view the data.

The follow diagram describes the application and web services security for XXXXX I:

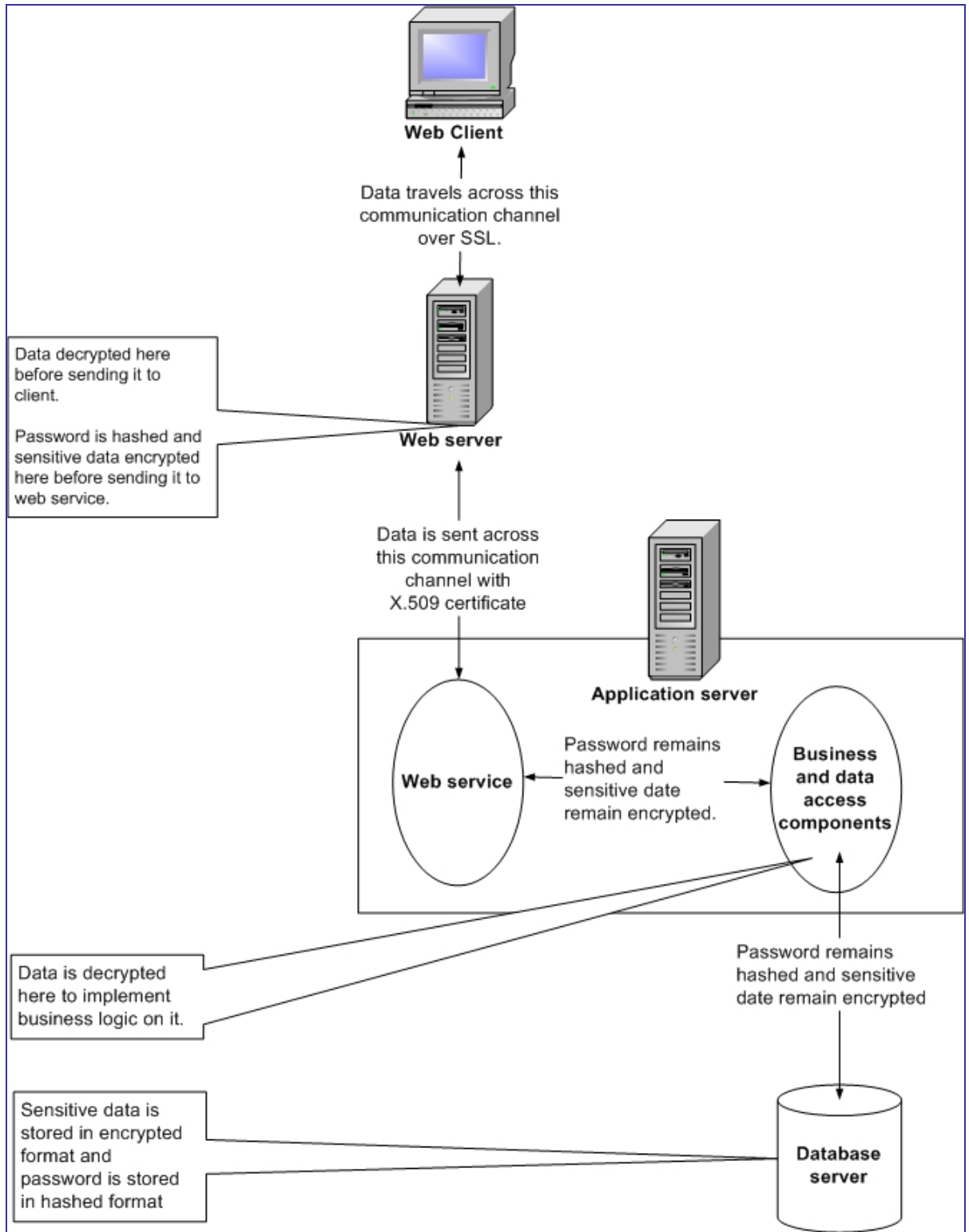


Figure 7: XXXXX I security model

Cryptography

XXXXX I will use the cryptography application block of Enterprise Library to encrypt data and hash passwords.

Hashing

XXXXX I will use SHA256 one-way hash algorithm to hash passwords. Passwords will be hashed at the web server level. Passwords are stored in the database in hashed format only. As depicted in Figure 7, the password will remain hashed through its transfer from layer to layer and component to component.

Encryption

XXXXX I will use the AES algorithm for encrypting sensitive data defined by application. Following are the list of sensitive data:

- SSN
- Zip
- Address
- Case Name
- DOB

The encryption key for encrypting sensitive data will be stored in security related config file. XXXXX I will encrypt security related and database related config files using separate a encryption key with the AES algorithm. This encryption key will be stored in one file. This key file will be protected using DPAPI- user protection mode.

2: Global Modules

Login Module

Identification	Login.aspx
Type	Module
Purpose	Restricts unauthorized use of XXXXX I, requiring authorization to enter
Function	User enters a valid User Name and Password and can enter the XXXXX I application
Dependencies	Section 5.1 Site Characteristics

References

Use Cases	Business/Technical requirements
Login.doc	br-058

Traceability

See Traceability Matrix: **Xxxxx_Business_Rule_Matrix.xls**

Reference Objects

Objects	Methods	Stored Procedures
XXXXXUtilities	Encrypt	ValidUser
User	GetUserMenuItems Authenticate	GetUserMenuItems

Navigation to Web Page

To access the **Login** page, a user does one of the following:

- Types the XXXXX I website URL in the browser.
- Types the complete web address for any other XXXXX I page,
- Clicks on a Favorite, Bookmark or link to any other XXXXX I page, while not logged in to RADAP I; user is automatically routed to the Login page.

Web Page Initial Appearance:



Figure 8: Login page initial appearance

Web Page Description

Login is the first page a user sees when initiating use of the XXXXX I application. Each XXXXX I user must log in to XXXXX I with a valid **User Name** and password before viewing any other XXXXX I pages or perform any tasks in XXXXX I.

The login page contains the following fields and controls:

- **User Name**—field
- **Password**—field
- **Login**—button

Prior to login, the User Name and Password must have been created and assigned to the user by the XXXXX I administrator through the Manage Users module, described on page **Error! Bookmark not defined.**

The user enters a User Name and password in their respective fields and clicks **Login**. XXXXX I does the following:

- Hash the password.
- Validates the User Name and Password against the database to ascertain that the User Name exists in the database and the password is correct.
- If the User Name is valid and the Password is not valid, the page checks the number of consecutive unsuccessful login attempts; if it is less than the variable `MaxNumberFailedLogins` set in the `Systemparameter` table, a message appears: *"You have entered an invalid User Name or Password"*.
- If the User Name is valid and the Password is not valid and the number of failed logins exceeds the `MaxNumberFailedLogins` set in the `Systemparameter` table, the user's account is locked. The user cannot log in until the account is reset by the administrator or clicks on reset link provided on the login page and a locked user message appears:

“You have exceeded the maximum number of unsuccessful login attempts. Your account has been locked. Please contact the system administrator to reset your password.”

- If the User Name and the Password is valid and StatusID is 4(locked) then the *“Your account has been locked. Please contact the system administrator to reset your password.”* Is shown
- If system date is greater than the sum of PasswordCreationDate and PasswordExpirationDays, then the user is redirected to the ChangePassword page and the user must change the password to gain access to the application.
- If the difference between system date and the sum of PasswordCreationDate and PasswordExpirationDays is less than or equal to PasswordExpireNotificationDays then the message appears, *“Your password will expire in <how many> days. You can change your password or continue.”* (see Figure 9.)
- If the log in is successful, resets NoOfFailPasswordAttempts to “0” in the user table and authentication ticket is set and user’s **Home** page appears.
- If the log in is unsuccessful, increments NoOfFailPasswordAttempts in the user table.
- If user makes more than MaxNumberFailedLogins unsuccessful attempts from the same IP address, then login page will be closed and an email message will be sent to the XXXXX I system administrator.

Inputs/Output

Prior to login, the User Name and Password have been created and assigned to the user by the XXXXX I administrator through the Manage Users module, described on page <xxx>. The user enters the following inputs:

Field Name	Description	Attribute (M/E/D)	DB Table	DB Column	Validations / Comments	Type of Control
User Name	User Name of the user	M	User	UserName		Text
Password	Password of the user	M	User	Password	Min length: 7 must contain at least one upper case, one lower case, one numeric and one special character Cannot reuse last 4 passwords	Password Text
StatusID	Status of the user	M	User	StatusID	Possible values are 1 – Active 2 – Inactive 3 – Pending 4 - Locked	For validation purpose only
NoOfFail Password Attempts	No Of unsuccessful login Attempts	M	User	NoOfFail PasswordAttempts		Internally updated by stored procedure

Field Name	Description	Attribute (M/E/D)	DB Table	DB Column	Validations / Comments	Type of Control
Password Expiration Days	Max Number OF Failed Login attempts	D	System Parameter	PasswordExpirationDays		For calculation only
Password ExpireNotificationDays		D	System Parameter	PasswordExpireNotificationDays		For calculation only
MaxNumberFailedLogins		D	System Parameter	MaxNumberFailedLogins		For calculation only



Figure 9: Login page with a message displayed

Password Expiration Notification

The password's lifetime is defined in XXXXX I by a password expiration date which determines when the password will expire. A variable PasswordExpireNotificationDays in the Systemparameter table specifies how many days before password expiration a warning message will appear to the user when the user attempts to log in. The module calculates the number days remaining before expiration and compares this number to the value of PasswordExpireNotificationDays. If the calculated value is less than the variable, a message warns the user: "Your password will expire in <calculated number> days. You can change your password or continue" and a **Change Password** button and a **Continue** button appear. The user can do either of the following:

- Click **Continue**; the user's home page appears.
- Click **Change Password**; the Change Password page appears.

Audit Special Requirements:

System will log the following:

- User Name and Password and timestamp for all unsuccessful login attempts

Results of Interaction with Web Page

One of the following:

- The user is logged in
- The user is offered an opportunity to change their password
- The user is not logged in but is offered another opportunity to log it
- The user's account is locked

Home Page

Identification	Home.aspx
Type	Component
Purpose	To display current global ticklers
Function	Allows users to view global ticklers and navigate to other modules
Dependencies	Section 5.1 Site Characteristics

References

Use Cases	Business/Technical requirements
Login.doc	br-003

Traceability

See Traceability Matrix: **Xxxxx_Business_Rule_Matrix.xls**

Reference Objects

Objects	Methods	Stored Procedures
XXXXXCache	IsAuthorized GetGlobalTicklers GetUserMenuItems	GetGlobalTicklers
Tickler	GetTicklers	
Function	GetUserMenuItems	GetUserMenuItems

Navigation to Web Page

Upon successful log in to XXXXX I, the initial Home page appears. Users can return to this page from any XXXXX I page by clicking the **Home** link.

Web Page Initial Appearance

This page displays all current global ticklers. It contains no editable fields. It lists all global ticklers. Each global tickler appears in a list, displaying its date, title and description, as shown below:

Global Ticklers

12/28/05 - Food Stamp Samples Loaded
 Food Stamp samples are loaded into the system and are ready for assigning. The samples were loaded on 12/28/05. All the samples are due by 02/25/05.

01/05/06 - Due date for TANF Samples
 Due date for TANF samples is close. All the TANF cases should be completed by 2 FEB 2006.

01/10/06 - Global Ticklers
 Global ticklers are like news to all RADEP users. Global ticklers will keep the users updated.

Figure 10: Home page typical display

Web Page Description

Global ticklers are messages sent to all users of XXXXX I. They are managed in the Global Ticklers module described on page **Error! Bookmark not defined.**. Each global tickler displays the date it was created, its title and its description.

No tasks are performed by users on this page. From this page, the user can navigate to other modules in the application to which the user has been given access.

Inputs/Outputs

The following data fields are drawn from the database and displayed on this page:

Field Name	Description	Attribute (M/E/D)	DB Table	DB Column	Validations / Comments	Type of Control
Event date	Date global tickler was created	D	Tickler	EventDate Time	Time portion of EvendateTime is trimmed	Text
Subject	Global tickler title	D	Tickler	Subject		Text
Description	Global tickler description	D	Tickler	Description	Rows where IsGlobal is true are selected and displayed	Text

Results of Interaction with Webpage

User views global ticklers.

User Export Data Module

Identification	UserExport.aspx
Type	Module
Purpose	To allow user to export data for analysis
Function	Exports case information in a format that is readable by analysis tools
Dependencies	Section 5.1 Site Characteristics

References

Use Cases	Business/Technical requirements
User Export	TR-024

Traceability

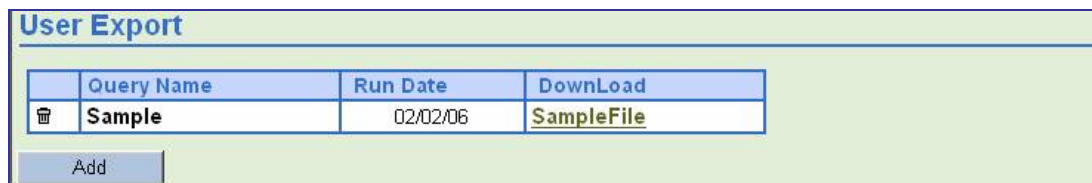
See Traceability Matrix: **Xxxxx_Business_Rule_Matrix.xls**


Reference Objects

Objects	Methods	Stored Procedures
userExport	GetElementBySampleType() RunUserExport() SaveUserExport()	GetElementBySampleType() RunUserExport() SaveUserExport()

Navigation to Web Page

The logged in user clicks the **Export User** link. The **Export User** page appears:



User Export			
	Query Name	Run Date	Download
	Sample	02/02/06	SampleFile

Add

Figure 11: Export User page

Web Page Appearance

The **Export User** page contains a grid listing **Query Name** and **Scheduled Date**, **Delete** icon and **Download**. By default, rows are sorted by **Schedule Date**. An **Add** button appears beneath the grid.

Web Page Description

To edit an existing query, the user clicks the query name. The query definition populates the query definition controls. The user makes the desired changes and clicks **Save**. The query is saved to the database.

To create a new query, the user clicks the **Add** button beneath the **User Export Data** grid.

Additional controls appear for creating or updating user query, as follows:

- **Sample Type (1)**
- **Sample Start Month (1)**—Sample Start month of elements
- **Sample End Month (1)**—Sample End month of elements
- **Query Name (1)**—a name the user assigns
- **Schedule Date (1)**—start date when the query will run
- **Interval (1)**—how often user wants to run a query
- **Select Clause (2)**—displays fields whose values the query will return; multiple selections are possible
- **Where Clause (3)**—defines the filter criteria on which the query should return result; The Where Clause is further described in section **Error! Reference source not found. Error! Reference source not found.**
- **Save (4)**—saves the query; it will run at the **Schedule time** and the query now appears in the grid on the **User Export** opening page.
- **Run Now (4)**—runs the query now, regardless of its scheduled run time
- **Cancel (4)**—cancels the query creation/edit process without saving it

User Export

Query Name:

Sample Type: TANF Primary Active

Start Sample Month: (mm/yyyy) End Sample Month: (mm/yyyy)

Schedule Date: (mm/dd/yyyy) Interval: Days

Select Clause

T7 Zip Code T7 Zip Code
 T8 Funding Stream T8 Funding Stream
 T9 Disposition
 T10 New Applicant

Where Clause

	Grouping	Logical Operator	Element	Compare Operator	Value	Element
			T8	=	1	
	AND		T10		01	
		OR	T10	=	03	
		OR	T1	=	<input type="text"/>	T1

Save Run Now Cancel

Figure 17: User Export

Only one query can be edited at a time.

When the logged in user defines a query and selects **Save**, the module validates that all fields are selected, in the following sequence:

- If no **Query Name** has been entered, the message appears “*You must enter a Query Name.*”
- If no **Sample Type** was entered, the message appears, “*You must select a Sample Type.*”
- If no **Start Sample Month** was entered, the message appears, “*You must select a Start Sample Month.*”
- If no **End Sample Month** was entered, the message appears, “*You must select an End Sample Month.*”
- If the Start Sample Month is after the End Sample Month, the message appears, “*The Start Sample Month must be after the End Sample Month.*”
- If no **Select Clause** has been selected, the message appears, “*You must select at least one Select Clause.*”
- If no **Where Clause** has been selected, the message appears, “*You must select a Where Clause.*”
- If no **Operator** has been selected, the message appears, “*You must select an Operator.*”
- If no **Value** has been selected, the message appears, “*You must select a Value.*”
- (For **Save** only) If no **Schedule Start** has been selected, the message appears, “*You must select a Schedule Start.*”
- **Interval**—time period between instances where the query runs

If the user clicks **Run now** the module validates that the following fields are selected, in the following sequence:

- If no **Sample Type** was entered, the message appears, *“You must select a Sample Type.”*
- If no **Start Sample Month** was entered, the message appears, *“You must select a Start Sample Month.”*
- If no **End Sample Month** was entered, the message appears, *“You must select an End Sample Month.”*
- If the Start Sample Month is after the End Sample Month, the message appears, *“The Start Sample Month must be after the End Sample Month.”*
- If no **Select Clause** has been selected, the message appears, *“You must select at least one Select Clause.”*
- If no **Where Clause** has been selected, the message appears, *“You must select a Where Clause.”*
- If no **Operator** has been selected, the message appears, *“You must select an Operator.”*
- If no **Value** has been selected, the message appears, *“You must select a Value.”*

Each field selected in the **Select Clause** will have its own column in the output.

Inputs/output

Field Names	Description	Attribute (M/E/D)	DB Table	Validations/Comments	Type of Control
Sample Type	Sample Type	M	SampleType .Description		Dropdown list
Sample Start Month	Sample Start Month	M	SampleProfile		Dropdown list
Sample End Month	Sample End Month	M	SampleProfile		Dropdown list
Query Name	Search result name	M	User Query		Text Box
Interval	How often user wants to run a query	M	User Query		Text Box
Format Type	Format Type for a export type	M	Format	Populated from FileFormatType table	Drop Down List

Field Names	Description	Attribute (M/E/D)	DB Table	Validations/Comments	Type of Control
Select Clause	Multiple Selection of a field	M	Element	List of FedItemNumbet + Field Name frpm Element table for the selected Sample Type List of all columns from CaseSkeleton table	List Boxes.

Results of Interaction with Web Page

A file is created or is scheduled to be created containing data which can be imported into data analysis tools.

Administration Modules

Create User Module

Identification	CreateUser.aspx
Type	Module
Purpose	To create users for the CRA of the logged in user.
Function	Allows new users to be entered into the system at CRA level.
Dependencies	Section 5.1 Site Characteristics

References

Use Cases	Business/Technical requirements
New	BR-053, BR-056, TR-013

Traceability

See Traceability Matrix: **Xxxxx_Business_Rule_Matrix.xls**

Reference Objects

Objects	Methods	Stored Procedures
UserBE,UserBL,UserDAL	AddUser(AuditBE, userBE)Integer	AddUser
CRABE,CRABL,CRADAL	GetCRASampleTypesByCRAID(AuditBE,CRAID) DataSet GetCRARoles(AuditBE, CRAID) DataSet	GetCRASampleTypesByCRAID GetCRARoles
XXXXXUtilBL	Hash(inputSring) String	

Navigation to the Web Page

The logged in user with proper authorization clicks the **Create User** link. The **Create User** page appears, as follows:

Create User

1 Last Name :

Middle Initial :

First Name :

Email :

User Name :

Password :

Confirm Password :

2 Role : Reviewer ▼

Select	County	Sample Type
<input type="checkbox"/>	Alpine	TANF-Federal Primary
<input type="checkbox"/>	Alpine	TANF-Federal Secondary
<input type="checkbox"/>	Alpine	TANF-SSP Federal Primary
<input type="checkbox"/>	Alpine	FS-Federal Primary
<input type="checkbox"/>	Butte	TANF-SSP Federal Primary
<input type="checkbox"/>	Butte	FS-Federal Primary
<input type="checkbox"/>	Calaveras	FS-Federal Secondary
<input type="checkbox"/>	Calaveras	FS-State Primary

4

Figure 12: Create User page

Web Page Appearance

The **Create User** page contains the following controls:

- **Last Name (1)**
- **Middle Initial (1)**
- **First Name (1)**
- **Email (1)**
- **User Name (1)**
- **Password (1)**
- **Confirm Password (1)**
- **Role (2)**—roles appearing in this list are marked as IsCRA
- **Grid (3)** containing:
 - **Select check box**—can select multiple rows
 - **County**
 - **Sample Type**
- **Save (4)**—button

Web Page Description

The user enters data for the new user in all indicated fields **(1)**, including an initial **Role (2)** to be assigned for this user. Roles which appear in this list are indicated **IsCRA** in the database

and therefore they can be assigned to users at the CRA level. User can only be assigned one role. The **County/Sample Type** combinations that appear in this list are associated with the CRA of the logged in user. (They are set in the **Manage CRA** module.) The logged in user assigns the new user permission to work with the cases for one or more county/sample type combinations. The logged in user clicks **Save (4)** and the data is validated and saved to the database.

When the logged in user clicks **Save**; the data is validated as follows:

- The **User Name** is validated to ensure that the input User Name is not already in the data base. If the User Name already appears in the database, a message appears, *“The User Name you entered already exists.”*
- The module confirms that the new password entered into **Password** field and **Confirm Password** field match. If they do not match, a message appears: *“The Password and the Confirm Password you entered are not identical.”*
- The module confirms that the password entered in the **Password** field and the **Confirm Password** field meets the requirements listed in the Validations/Comments column below. If they do not meet these requirements, a message appears: *“Passwords must be at least seven characters long with at least one upper case, at least one lower case, one numeric and one special character.”*
- The email address is validated.

If the data is validated, the proposed new user is saved to the database with a user account status of **Pending**.

Following tables are affected **User, UserRole and UserSampleType**

Inputs

Field Name	Description	Attribute (M/E/D)	DB Table	DB Column	Validations / Comments	Type of Control
User Name	User Name	M	User	User Name	Cannot already exist in table	Text
Password	Password	M	User	Password	Min length: 7 Should contain at least one upper case, one lower case, one numeric and one special character	Text Password
Confirm Password	Confirm password	M			Must match Password entry	Text Password
Email	Email address	M	User	EmailID	Must be a valid email address	Text
LastName	Last Name	M	User	LastName		Text
Middle-Initial	Middle Initial		User	MiddleInitial		Text
FirstName	First Name	M	User	FirstName		Text

Field Name	Description	Attribute (M/E/D)	DB Table	DB Column	Validations / Comments	Type of Control
Role	Role	M	Role	Description	Must select a role. (Populate all roles from Role table which are flagged IsCRARole True)	Drop down list
Select	Select	E	CRASampleType	ID	Must select at least one	Grid check box
County	County	D	Counties	Name		Grid item
Sample Type	Sample Type	D	SampleType	Description		Grid item

Outputs

None

Results of Interaction with Web Page

The new user has been entered in the database with the status Pending.

Audit Configuration

Identification	AuditConfig.aspx
Type	Module
Purpose	To specify module to be audited
Function	Changes the application audit configuration of the modules
Dependencies	Section 5.1 Site Characteristics

References

Use Cases	Business/Technical requirements
Missing	TR-045, TR-033

Traceability

See Traceability Matrix page: **XXXXX_Business_Rule_Matrix.xls**

Reference Objects

Objects	Methods	Stored Procedures
FunctionBE, FunctionBL,FunctionDA L	GetAuditConfigData() DataSet SaveAuditConfigData() Boolean	GetAuditConfigData SaveAuditConfigData() Boolean
XxxxxAppCache	GetAuditConfigData() DataSet	

Navigation to Web Page

The logged in user clicks the **Audit Config** link. The **Audit Configuration** page appears as follows:

Function	Yes/No
Login Module :	<input type="checkbox"/>
Log off Module :	<input type="checkbox"/>
Change Password Module :	<input type="checkbox"/>
Tickler Module :	<input type="checkbox"/>
Home Page :	<input type="checkbox"/>
Review Cases Module :	<input type="checkbox"/>
Case Load (Supervisor) Module :	<input type="checkbox"/>
Search :	<input type="checkbox"/>
Delegate :	<input type="checkbox"/>
Create User Module :	<input type="checkbox"/>
Manage Users Module :	<input type="checkbox"/>
Manage Roles Module :	<input type="checkbox"/>

Figure 13: Audit Configuration page

Web Page Appearance

The web page contains a check box for each function of the application and a **Save** button. When the page loads, functions currently marked as auditable (IsAuditable set to “true”) in the database are checked.

Web Page Description

Each function corresponds to a row in the Function table of the database. The logged in user checks the functions to be audited and clicks **Save**. The settings are updated in the database. As a result, all user activity in the selected function is logged.

The table affected is **Function** table.

Inputs

Field Name	Description	Attribute (M/E/D)	DB Table	DB Column	Validations / Comments	Type of Control
Function name	Function name	D	Function	Description		Text
Yes/No		E	Function	IsAuditable		Checkbox

Results of Interaction with Web Page

Configuration of functions to be logged are saved.

Delegate

Identification	Delegate.aspx
Type	Module
Purpose	To allow the user with appropriate authority to delegate authority on some functions to an other user for a specified time period
Function	To delegate authority to perform tasks in XXXXX I
Dependencies	Section 5.1 Site Characteristics

References

Use Cases	Business reqs/Tech reqs
DelegateAuthority.doc	br-009, br-020

Traceability

See Traceability Matrix Xxxxx_Business_Rule_Matrix.xls

Reference Objects

Objects	Methods	Stored Procedures
DelegateBE, DelegateBL, DelegateDAL	GetDelegateeByUserID (UserID) DataSet GetSelectedDelegatee(DelegateID) ValidateExistingDelegation(DelegateBE) Boolean AddDelegatee(DelegateBE) Boolean ModifyDelegatee((DelegateBE) Boolean DeleteDelegatee((DelegateID) Boolean	GetDelegateeByUserID GetSelectedDelegatee ValidateExistingDelegation AddDelegatee ModifyDelegatee DeleteDelegatee

Objects	Methods	Stored Procedures
UserBL,UserDAL	GetUsersByCRAID(AuditBE,CRAID) DataSet GetCRAIDfByUser (AuditBE,UserID) integer	GetUserByCRAID GetCRAIDByUser
FunctionBL,FunctionDAL	GetIsDelegableFunctions() DataSet	GetIsDelegableFunctions

Navigation to Web Page

User clicks the **Delegate** link .The **Delegate Authority** page appears, as follows:

Delegate				
	User	Authority	Start Date	End Date
	John	Assign	12/05/05	12/10/05
	John	Approve	12/05/05	12/10/05
	Kevin	Approve	12/05/05	12/10/05
	Sandra	Assign	12/05/05	12/10/05
	Sandra	Approve	12/05/05	12/10/05
<< <		1 2 3 4 5	> >>	
<input type="button" value="Add"/>				

Figure 14: Delegate Authority page opening view

Web Page Appearance

The **Delegate Authority** grid displays information about the users to whom functions have been delegated, if any. Users appear sorted by the **User Name**, in ascending order. Each row names one delegated function for a user, and its start and end date. Users may have multiple rows. An Add button allows the logged in user to delegate a function (which can be delegated) to a user. To delete an assigned function, the user clicks the **Delete** icon for that row. To modify a delegation, the user clicks the **Modify** icon for the corresponding row.

Web Page Description

To delegate an authority, the user clicks **Add**. The following fields appear:

- **User (1)**—displays all users for current CRA
- **Functions (2)** displays all functions which the user can delegate; for example **Review Cases, Search Cases, Assign Cases, Approve Cases, and Reports**
- **Start Date (3)**—date the delegation for selected functions begins
- **End Date (3)**—date the delegation for selected functions expires
- **Cancel (4)**—button
- **Save (4)**—button

The screenshot shows a web form titled "Delegate". It contains the following elements:

- User:** A dropdown menu with "----Select----" as the current selection. A callout circle labeled "1" points to this dropdown.
- Assign Cases:** A checkbox. A callout circle labeled "2" points to this checkbox.
- Approve Cases:** A checkbox. A callout circle labeled "2" also points to this checkbox.
- Start Date:** A text input field with a calendar icon to its right. A callout circle labeled "3" points to this field.
- End Date:** A text input field with a calendar icon to its right. A callout circle labeled "3" also points to this field.
- Save:** A button. A callout circle labeled "4" points to this button.
- Cancel:** A button. A callout circle labeled "4" also points to this button.

Figure 15: Delegate Authority controls visible

The logged in user selects a **User (1)** from the drop down list, checks the functions to delegate **(2)**, enters a **Start Date** and **End Date (3)** for the delegation and clicks **Save (4)**.

- If selected functions are validated, the dates are validated. If they are not, an error message appears: "Start date must be before End date" or "Start date must be today or after."
- If the dates are validated, permissions are entered to the database allowing the selected user to perform the selected tasks.
- If multiple functions are selected for one user, then while saving separate rows are inserted for each function in Table.

Alternatively, the user clicks **Cancel (4)** and data is not saved and the initial Delegate Users page returns.

To remove a delegated authority from a user, the logged in user clicks the **Delete** icon for the respective authority row, and the row is deleted and the authority is rescinded.

To modify a delegated authority, that is, to change its **Start Date** or **End Date**, the logged in user click the **Modify** icon for the respective authority. The controls appear populated with the data. The **User Name** and function are read-only. The logged in user can modify the **Start Date** or **End Date** and clicks **Save**.

- The dates are validated. If they are not, an error message appears: "Start date must be before End date" or "Start date must be today or after."
- If the dates are validated, the Start Date and End Date are updated to the database.

Inputs/ Outputs

Field Name	Description	Attribute (M/E/D)	DB Table	DB Column	Validations / Comments	Type of Control
User	User name	M	User	UserName	Users in the CRA of the logged in user	Drop Down List
Function	Functions with isDelegatable flag true.	M	function	Description	A user must select at least one function.	Check box list
Start Date	Start date for authority	M	Delegation	StartDate	Must be less than End Date.	Date and pop up calendar
End Date	End date for authority	M	Delegation	EndDate	Must be greater than Start date	Date and pop up calendar

Results of Interaction with Web Page

Users are delegated authority to perform specific functions for the indicated time frame, or delegated authorities can be modified or deleted.

Delegate data is saved in Delegation table.