Arizona Department of Health Services

Bureau of Nutrition and Physical Activity

HANDS Detailed Technical Specification Document

HANDS DTSD

August 25, 2015

Version 1.1

Table of Contents

TABLE OF CONTENTS **2**

REVISION HISTORY **4**

1 OVERVIEW **5**

2 SYSTEM ARCHITECTURE **6**

2.1 System Overview 8
2.2 Network Topography 10
2.3 Reporting Overview 12
2.4 Error Logging Overview 13
2.5 User Experience 14
2.6 Scalability 15
2.7 Security Overview 16

3 FUNCTIONAL DESCRIPTION **18**

3.1 System Administration Layer 18
3.2 Food Management Layer 18
3.3 WIC Services Layer 19
3.4 Commodity Supplemental Food Program (CSFP) Services Layer 19
3.5 Participant Scheduler Layer 19
3.6 Farmers' Markets Layer 20
3.7 Finance Layer 20
3.8 Vendor Management Layer 20
3.9 Program Integrity Layer 21
3.10 Batch Processing Layer 21

4 EXTERNAL INTERFACES **24**

5 STATE DEPLOYMENT MINIMUM REQUIREMENTS **25**

6 CLINIC DEPLOYMENT MINIMUM REQUIREMENTS **26**

7 DEVELOPMENT DETAILS **28**

7.1 Coding Standards Highlights 28
7.2 System Architecture 28

7.3 Development Process and System Deployment 31
7.4 Unit Testing 31

8 APPENDIX A: DATA DICTIONARY AND DATA MODELS 35

REVISION HISTORY

The chart below indicates revisions made to this document:

Version	Name	Brief Description of Change	Published
1.0	Chris Walker	Initial Draft	02-23-2015
1.1	Chris Walker	Maximums Review Edits	04-21-2015

1 OVERVIEW

HANDS is an acronym for Health And Nutrition Delivery System. HANDS is designed to coincide with a state's WIC Clinic flow and Local Agency caseload management, and is used to:

- Collect and process WIC and CSFP data
- Create Letters, Reports, Forms and Labels based upon the data collected
- Allow for access by any computer with an internet connection
- Streamline the WIC and CSFP processes
- Meet USDA Federal Requirements

The HANDS MIS is a modern web server based application built using industry leading programming technologies and design techniques. To operate HANDS, only Internet Explorer 9 or higher is needed. In order to access the Letters and Reports functionality featured in HANDS, the Adobe Reader program is required. MICR printers are used for the printing of Food Benefit's (FB's).

This document is a detailed technical specification document (DTSD) describing the HANDS MIS system. Topics covered within include:

- System Architecture
- Functional Description
- External interfaces
- State Deployment Minimum Requirements
- Clinic Deployment Minimum Requirements
- System Development Details
- System Deployment Details

2 SYSTEM ARCHITECTURE

The HANDS MIS is a modern web server based application built using industry leading programming technologies and design techniques. HANDS was built on top of the .NET Framework. The .NET Framework is a technology that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The design of HANDS follows the MVC model for web applications with 3 logical layers:

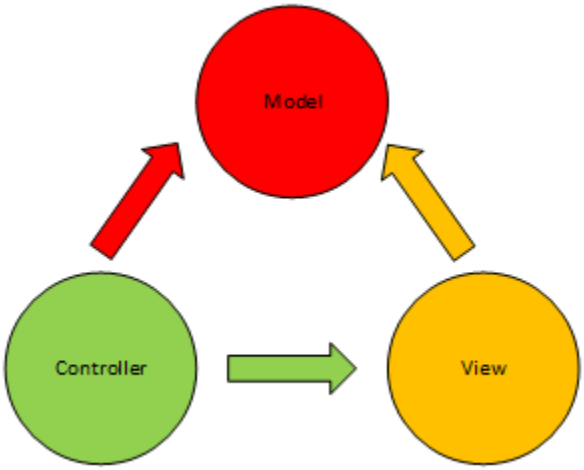

Figure 1: 3 Logical Layers of the MVC Model

- **Model Layer –** The Model layer is the part of the application that handles the logic for the application data.
- **View Layer –** The View layer consist of the parts of the application that handle the display of the data. Views are most often generated from the model data.

- **Controller Layer –** The Controller layer is the part of the application that handles user interactions, serving as a mediator between the Model Layer and the View Layer.

By following the MVC modeling approach when designing HANDS, the application is easily maintained and offers flexibility for adding future feature enhancements without a need for a large scale architectural redesign. The following illustration provides a high level overview of the HANDS design using the MVC model.

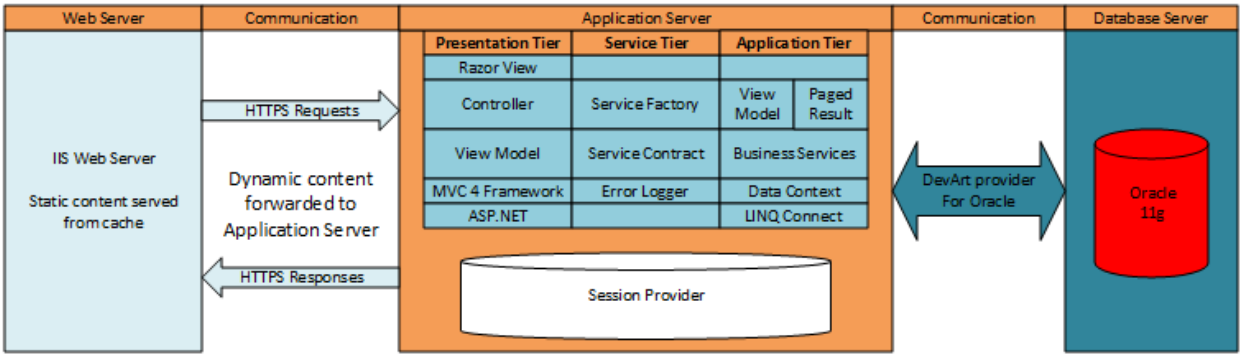

Figure 2: HANDS Component / Package Diagram

The technologies utilized to build HANDS included:

- **ASP.NET –** Is a unified Web development model that includes the services necessary to easily build enterprise-class Web applications within the .NET Framework.
- **MVC 4 Framework –** A unified Web development framework that is part of the ASP.NET Web development model. The framework is a lightweight, highly testable presentation framework that is integrated with existing ASP.NET features, such as master pages and membership-based authentication.
- **jQuery –** a feature-rich JavaScript library that facilitates client-side document manipulation, event handling, animation, and Ajax calls with an easy-to-use API that works across a multitude of browsers.
- **Razor Engine –** Provides the view engine to the MVC 4 framework. Razor is a markup syntax that supports embedding server-based code (C#) into web pages. It uses a code-focused templating approach, with minimal transition between HTML and code.
- **C# –** A simple, modern, general-purpose, object-oriented programming language designed by Microsoft within its .NET initiative.
- **LINQ –** Language-Integrated Query (LINQ) extends powerful query capabilities to the language syntax of C#.
- **DevArt dotConnect –** An enhanced database connectivity solution built over ADO.NET architecture that offers a complete solution for developing database-related applications and web sites.

2.1 System Overview

HANDS is deployed onto three separate server tiers each with a specific and separate function from the others.

- Web Server – Manages the User Interface (UI) and UI session management
- Application Server – Manages the business logic associated with HANDS services
- Data Server – The database server provides a persistent data store for all data within the system

The following HANDS System Overview diagram depicts the flow of communications into and out of HANDS as well as the internal communications among the different parts of the system.

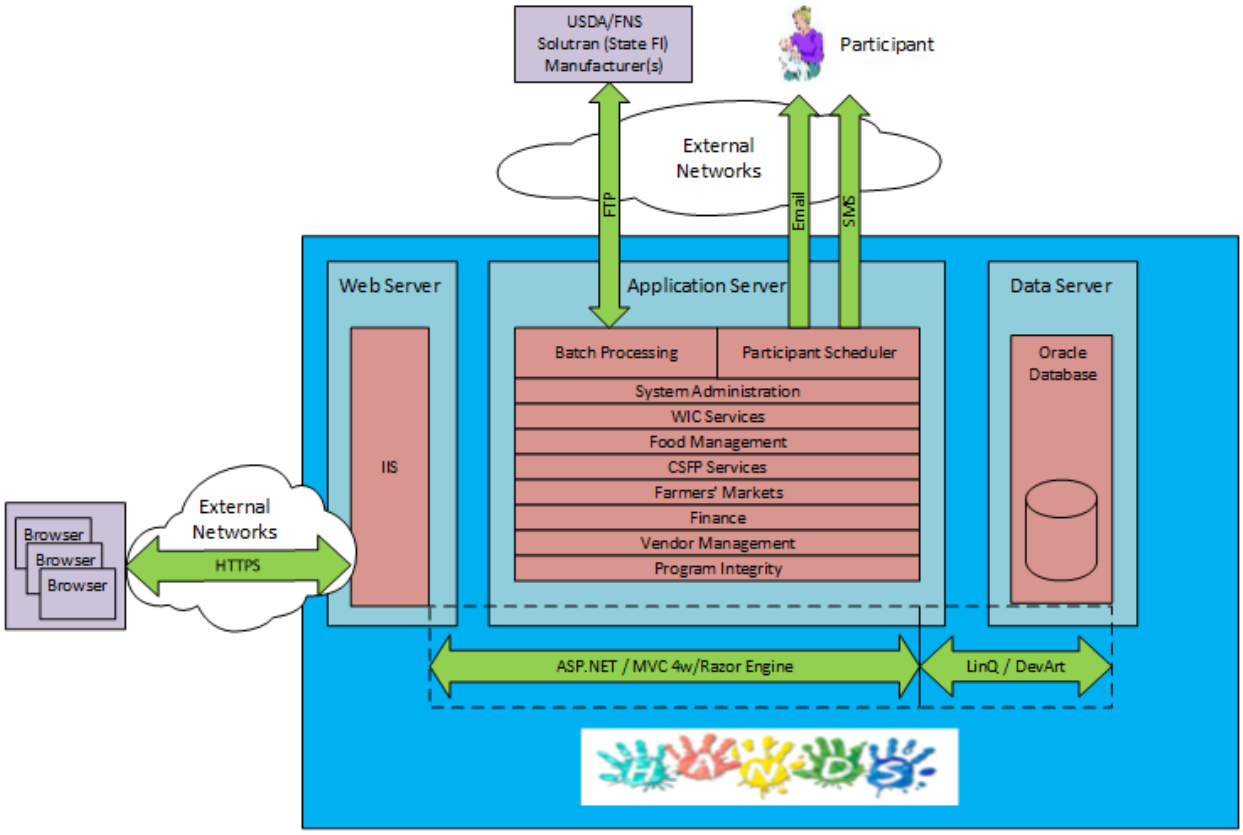

Figure 3: HANDS System Overview

Note – Figure 3 depicts a logical separation of Service Tiers (Web Server, Application Server and Data Server). All three Service Tiers may be deployed to a single physical server, a single physical server per Service Tier or any combination of the three Service Tiers to multiple or single physical servers depending on the size of the state agencies deployment of HANDS. The Arizona WIC program is deployed with the Web Server and Application Server hosted on a single physical machine and the Oracle 11g database hosted on another.

2.1.1 Web Server

Clinic administrators gain access to HANDS through a local browser connected to an IIS Web Server over an external network. Internet Information Services (IIS) for windows Server is a flexible, secure and manageable Web server for hosting on the Web. IIS is a scalable and open architecture capable of handling the most demanding tasks. The IIS Web Server presents HANDS web pages to the clinic administrator's web browser and directs user commands to the HANDS services available within the application server.

Security Policy Application

The Web Server services access security services for both authentication and authorization of user requests. HANDS users are associated with User Role(s) defined by the state administrators, which govern the user's access to available features of the HANDS application.

Session Management

IIS provides session management services to maintain user session information, such as security roles, sign on data, and other data that must be retained for a user session.

2.1.2 Application Server

As part of the MVC design approach, services are organized separately from one another into logical layers, offering the flexibility to add new services or modify existing services without the need of a large scale system architectural redesign. The application server consists of business services, shared services and database access services.

2.1.2.1 **Business Services**

Business services contain the rules used to accomplish WIC business-related tasks. They combine clinic administrator user actions and automated end of day actions with business rules that govern the processes for successfully completing the desired actions.

2.1.2.2 **Shared Services**

Shared services provide a system-wide standardization of functional areas to the business services.

Security

The security services provide authentication capabilities to the HANDS system. Security is role-based and roles dictate the access to business objects and/or tasks that a user is granted access to. A users access to state, local agency and clinic data is granted based on a combination of the roles assigned to the user and the state, agency and/or clinic they are assigned to.

Error Logging

Error logging services provide a mechanism for logging error conditions encountered to other system services. These services ensure a system-wide standardized approach to

error reporting and logging, governing how errors are reported to users and logged within the database for historical purposes.

Reporting Services

The reporting services utilize the SQL Server Reporting Services (SSRS) toolset. The reporting services govern how reporting services gain access to the data within the database and how that data is presented to end users.

2.1.2.3 Data Integration Services

The Devart dotConnect tool provides other system services with consistent access to the HANDS data stores within the Oracle 11g database.

2.1.3 Data Server

The Data server consists of an Oracle 11G database server. The Oracle database provides a persistent data store for all data within the system and is responsible for securely organizing, storing, tracking and serving information to the application services in the most expedient fashion while maintaining referential integrity throughout a properly normalized logical data model.

2.2 Network Topography

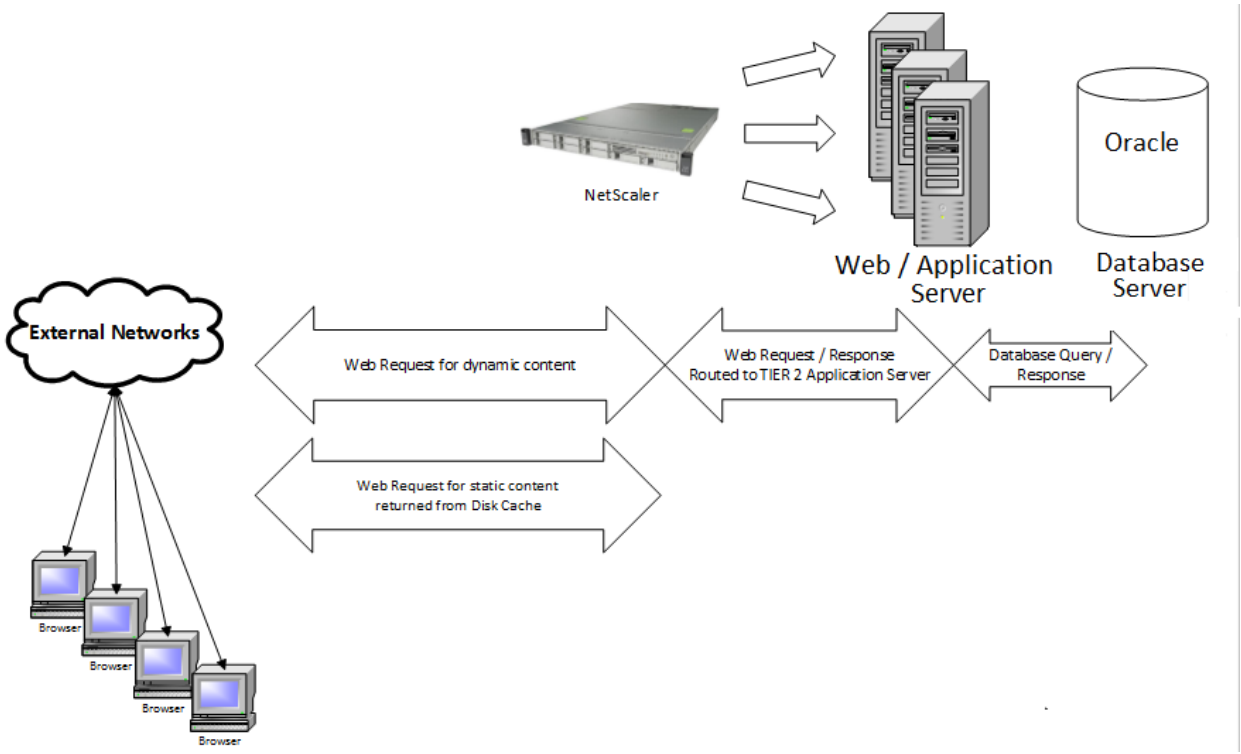

Figure 4: Network Topography

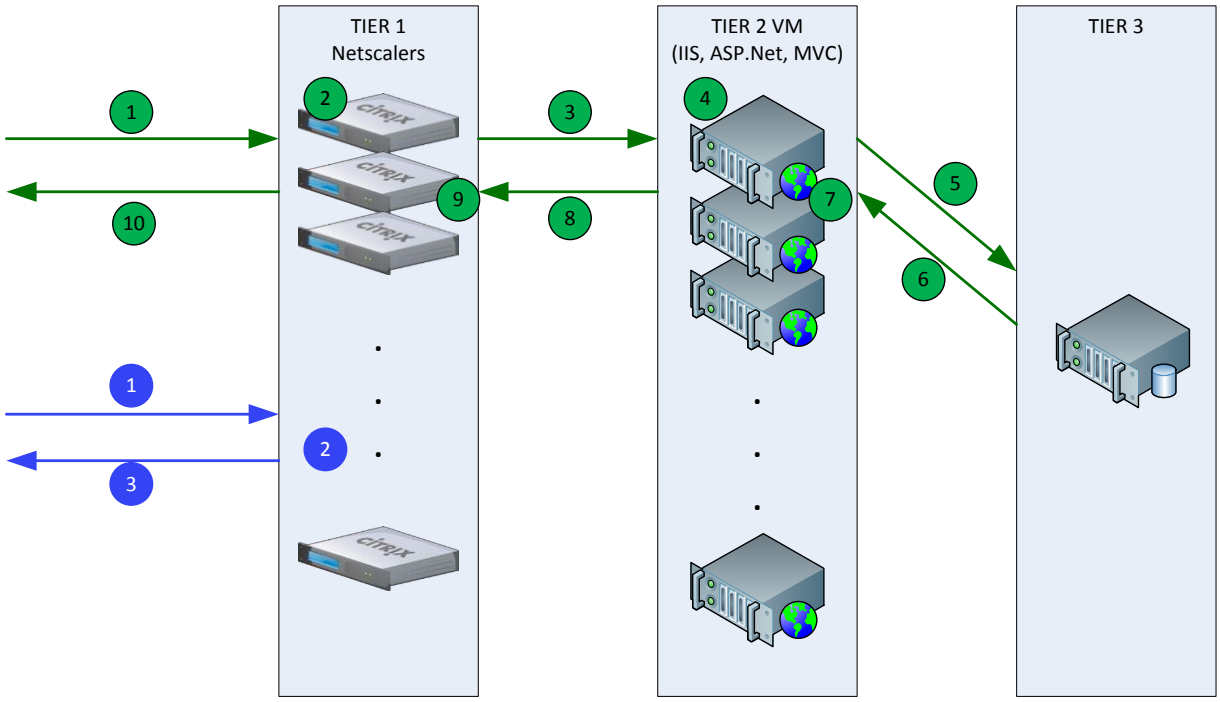

Figure 5: Content Request Application Request Routing

Request for Static Content *(Figure 4 in blue)*

1. Client Requests static data i.e. image, CSS, Javascript.
2. The Netscaler accesses local resource.
3. The Netscaler serves static content from cache.

Request for Dynamic Content *(Figure 4 in green)*

1. Client Requests a HANDS web page.
2. The Netscaler access local resource.
3. Because the request is for a non-static resource, the request is forwarded to the Application server.
4. The Application server ASP.Net MVC framework invokes the appropriate controller in the HANDS application. The controller processes the request and gathers data for a call to an in process HANDS Business Service. A given controller could require data from 1 or more HANDS business services. The in process HANDS service processes the request and invokes 1 or database operations.
5. The database request is sent to the Database server.
6. The Database server's response is returned to the HANDS service on the Application server.
7. The HANDS service composes a service response to be returned to the service caller which in this case is the in process HANDS MVC View Controller. The HANDS MVC View Controller, uses the data returned from the HANDS Business Service to render a view to be returned to the caller.

8. An HTTPS response representing the web page is returned to the Web Server.
9. The HTTPS Response Header is rewritten so it appears the page rendered from the Web server.
10. The web page is returned to the caller's browser.

2.3 Reporting Overview

The HANDS reporting engine is built using the Microsoft SQL Server Reporting Services (SSRS). The SSRS is an enterprise reporting platform that is data-source independent. The SSRS is installed on a report server along with an SQL server database instance which provides the reports services configuration settings. The HANDS Web Server passes reporting request to the report server. The report server retrieves report data from the HANDS Oracle database server and returns the data to the HANDS Web Server report viewer for display to the requesting user. Reports can be exported to the user's local PC in a variety of file formats.

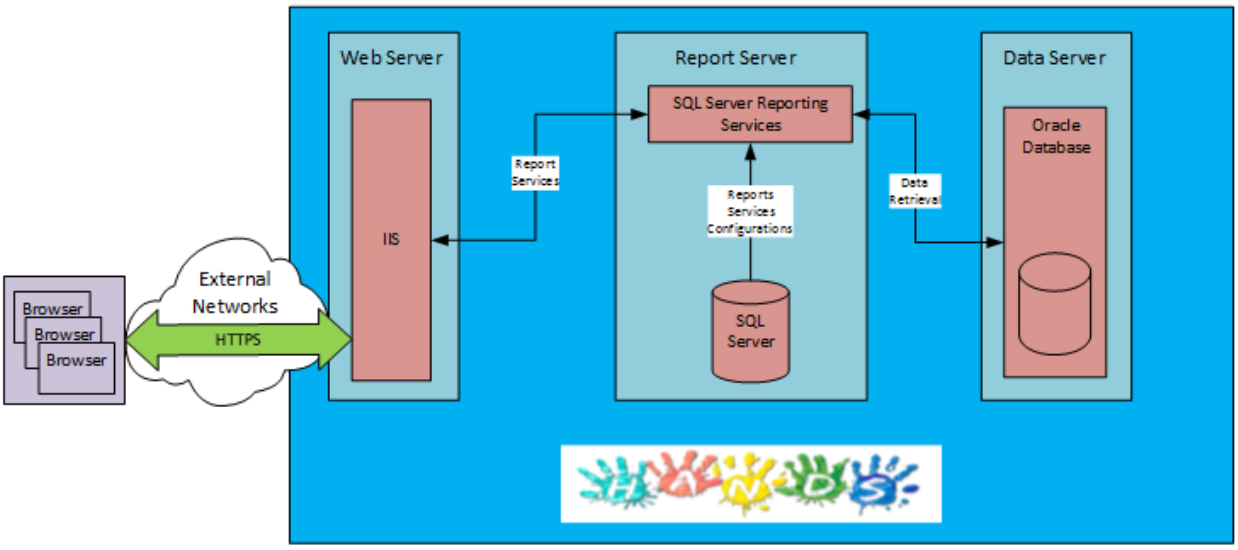

Figure 6: Report Topography

2.3.1 Reporting Categories

An exhaustive list of report capabilities are available within HANDS. Reporting categories include:

- State level / Local agency
- Farmers' Markets
- System administration
- Finance
- Vendor
- Operations management

- WIC services
- CSFP
- Appointment schedule
- Program integrity

2.4 Error Logging Overview

The HANDS error processing philosophy is twofold:

a. Within the context of the error, collect relevant exception and debug related information and log it to the central repository.

b. Display a user friendly message acknowledging the error and providing a Ticket Number to associate with the error. The ticket number can be used during Customer Support calls to link the user's error with the previously collected debug information and exception details.

The HANDS application code utilizes the standard C# **try-catch** blocks to provide structured exception handling. When an exception is caught, the exception handling code in the given **catch** block will:

a. Collect the exception details and debug context data including:
   i. Date and time the exception occurred
   ii. The name of the server where the exception originated
   iii. The user id of the user that received the exception
   iv. Available browser information related to the call context
   v. The exception type and exception message
   vi. The call stack at the time of the exception
   vii. Associated object data that was in scope at the time of the exception. Typically this will be a view model or entity
b. Log the exception details and debug context
   i. Store exception details and debug context to the database
   ii. If an error occurs while accessing the database, save the exception information to the application event log.
c. Generate an Error ticket to be returned to the user
   i. If the error was stored in the database the ticket number will be linked to the database record.
   ii. If the error was stored in the application event log the ticket number will contain identifiers that will associate it with the server on which the error was logged.
d. Using the Error Ticket Number, create and throw a custom Error Ticket Exception
e. Catch the Error Ticket Exception at the Controller level and use it to render the HANDS standard error page.

2.5 User Experience

The user experience is an important design consideration for any software application. A software application could be the most feature rich application on the market, but if the end user finds it confusing or cumbersome to use and/or the response time of user actions fall short of the user's expectations, the application will not be judged favorably. The HANDS design team is cognizant of the value of the user experience and as such considered both the user interface experience and user action response times when designing the application.

2.5.1 User Interface

The HANDS UI web pages present a standardized look and feel to ensure users can quickly and easily achieve their goals within the system. Standardized features include:

- Categorized organization of system functions
- Easily switch between main menu functions via the title bar displayed on all web pages
- Drop down sub-menu navigation located in upper left hand corner
- A uniform layout for data searches
- List of values generated for entered text
- Tabbed pages for easy navigation between multiple pages associated with the same service.
- Required fields highlighted in yellow
- Action buttons located in the lower right hand corner
- User prompts to ensure unsaved data is not lost when navigation away from a page will result in loss of unsaved changes.
- A uniform feel to data entry field types

2.5.2 Performance Optimization

In order to keep user action response times at a minimum, optimizations are baked into the interface design to the benefit of the overall user experience.

- **Cache Frequently Used data**
  Static table data is cached at the web server providing quick retrieval of the data without the need to pass requests through the application server tier to the database server tier in order to retrieve the data. The result is faster access to static data. Base tables are loaded from the database into the web server cache the first time they are accessed. From that point forward they are accessed from the cache until the cache expires. The cache expires in the following conditions:
    - If the application is reset or the server is rebooted
    - If the application goes idle for an extended period of time
    - If a base table is altered

- **Limit the amount of data transferred from the database to the application servers**
  A performance logging feature is built into the framework. The HANDS administrative support team use this tool to actively monitor the database communications in order to identify database access methods that exceed an acceptable threshold for data transference between the application servers and the database servers (hot spots). Once a hot spot is identified the development team is involved and a solution is scheduled into a future release of the application.
- **Limit the amount of data transferred from the web servers to the browsers**
- **Optimize database queries**

2.6 Scalability

Scalability is the ability of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth. Scalability is a foundational requirement of HANDS, considered in the design of each component of HANDS from its inception to the present. The resultant design supports the flexibility to right size the HANDS system components to meet the varied demands it might face with each deployment of the system. Simply put each component of HANDS may be run in duplication to meet the unique throughput needs of a State's deployment. Coupling HANDS with an Oracle cluster ensures that no matter how demanding the needs of the HANDS components are on the database, the database services will be reliably provided.

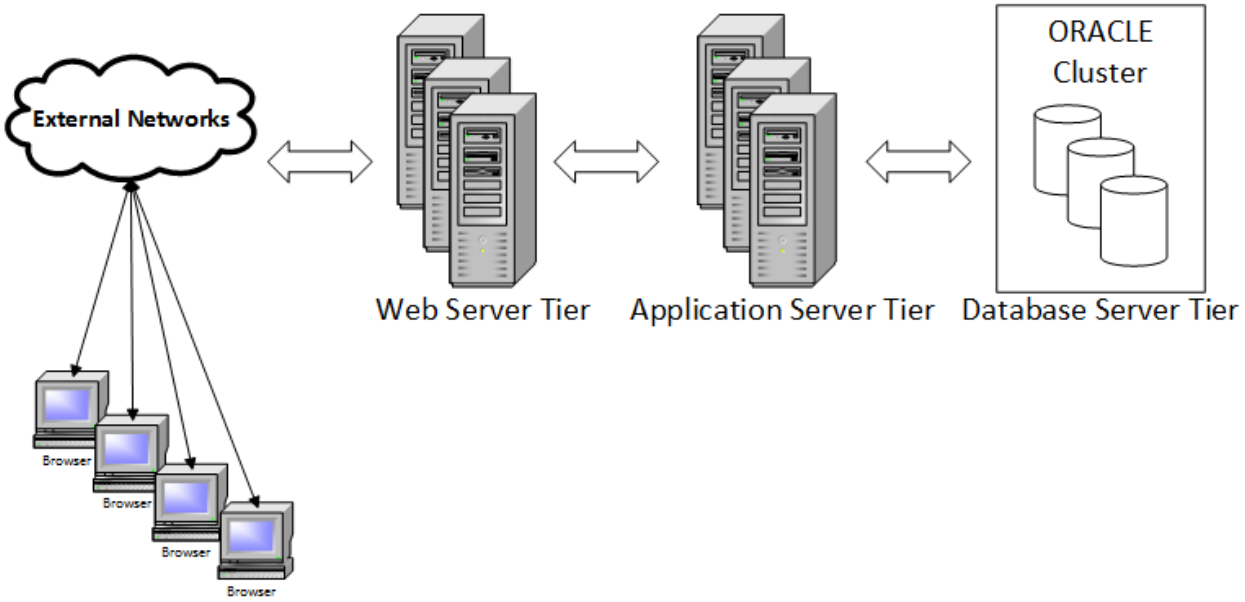

Figure 7: Scalability Overview Diagram

The production environment configuration may vary from state-to-state. All three Service Tiers may be deployed to a single physical server, a single physical server per Service Tier or any combination of the three Service Tiers to multiple or single physical servers depending on the size of the state agencies deployment of HANDS. The Arizona WIC program is deployed with the Web Server and Application Server hosted on a single physical machine and the Oracle 11g

database hosted on another. CMA Consulting Services will work with each state to determine the appropriate configuration of HANDS for their caseload and number of concurrent users.

2.7 Security Overview

Signed SSL Certificates with 2048 bit encryption are used to protect the communications between the clinic administrator's web browser and the HANDS Web Server. An Active Directory (AD) domain controller is used to authenticate and authorize users, assigning and enforcing the security policy configured for the HANDS environment.

Once a user has successfully completed the sign-on process, there are four additional layers associated with the user that govern the user's access to data and HANDS system features, (Secured Objects).

- **Organization Unit –** The Organization layer controls access to data associated with an organization and program. An organization can be setup at the Stage Agency Level, Local Agency Level or Clinic Level. A user may be associated with multiple Organization Units.
    - **Stage Agency –** Limits access to data associated with the State Agency, Local Agencies associated with the State Agency and Clinics associated with the Local Agencies.
    - **Local Agency –** Limits access to data associated with a Local Agency and Clinics associated with the Local Agency.
    - **Clinic –** Limits access to data associated with a Clinic.
- **Title(s) –** Titles provide a mechanism for grouping together the roles associated with the title. Titles limit access to the secured objects associated with the roles assigned to the title. A user may be associated with multiple titles.
- **Roles –** Roles grant access to the secured objects required by a user to perform the activities associated with the role. Multiple roles may be associated with a Title.
- **Secured Objects –** Secured Objects are the web pages, reports, modules and features that make up the HANDS system. Multiple Secured Objects may be associated with a Role.

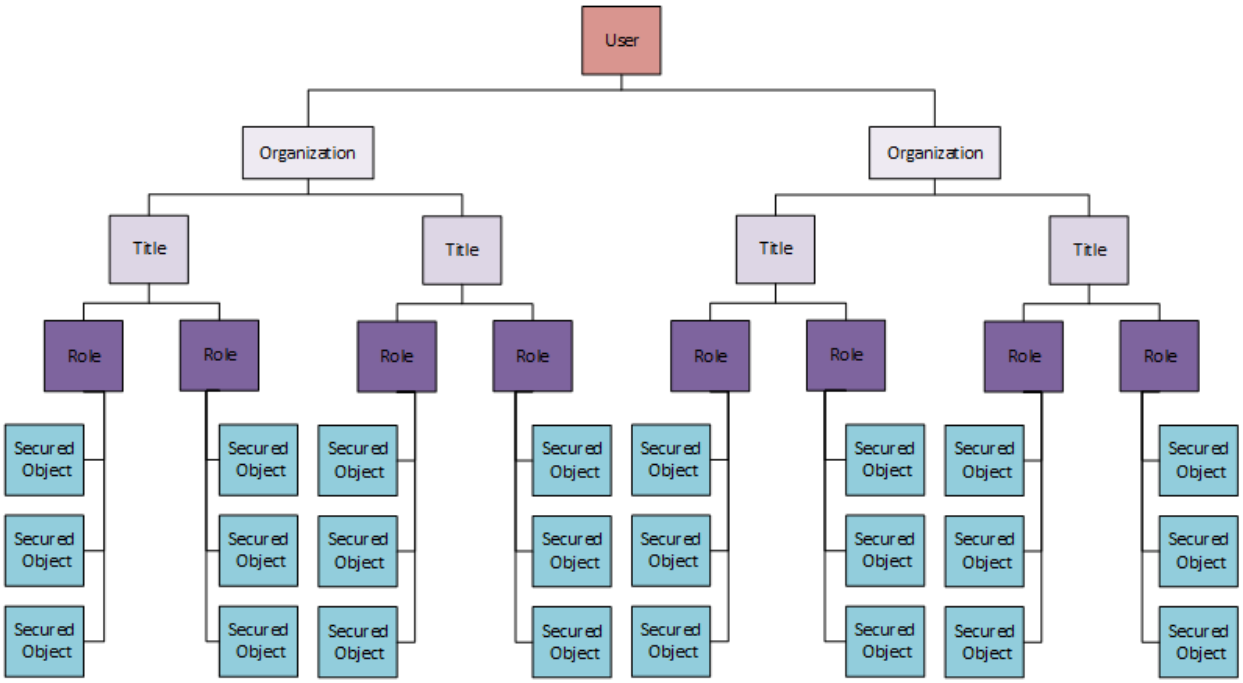

Figure 8: User Security

3 FUNCTIONAL DESCRIPTION

HANDS is a full featured WIC Management Information System (MIS). HANDS Services are organized separately from one another into logical layers as a function of the HANDS MVC design. The logical separation of services provides the flexibility to add new services or modify existing services without the need of a large scale system architectural redesign.

3.1 System Administration Layer

The System Administration layer enables the authorized user to configure the HANDS system for the state. Configurable options include:

- State specific feature options and settings
- Configuration of base table items
- Check printing test tools
- User Administration
- Viewing system logs

3.2 Food Management Layer

The Food Management layer enables the authorized user to configure food packages available for prescription to participants. Configuration options include:

- Client categories
- Container types
- Formula base types
- Food groups and sub-groups
- Food items and maximum allowed of the food item
- Food package categories
- Food packages
- Proration schedules
- Cash Value Amounts
- Food benefit lookup
- Void / Reissue food packages
- Void un-issued food benefits
- Un-Void food benefits
- Available food benefit void reasons

3.3 WIC Services Layer

The WIC Services layer encapsulates all functionality necessary to bring new WIC families into the states WIC program, as well as to re-certify existing participants. Services provided by this layer include:

- Family setup
- Client registration and assessment
- Client certification, recertification and termination
- Care Plan recommendation
- View / Scheduling appointments
- Search for existing families/clients
- Food benefit prescription and issuance
- In-State transfers
- Out-of-State transfers

3.4 Commodity Supplemental Food Program (CSFP) Services Layer

The CSFP Services layer encapsulates all functionality necessary to bring new families into the states CSFP program as well as to re-certify existing participants. Services provided by this layer include:

- Family setup
- Client registration and assessment
- Client certification, recertification and termination
- Search for existing families/clients
- View / Schedule appointments

3.5 Participant Scheduler Layer

The Scheduler encompasses a set of services used by clinic administrators to manage appointments for WIC families. Enhanced scheduling features include:

- Customizable hours of operation on a per clinic basis
- Customizable services duration
- Automatic grouping of family appointments
- Walk-in appointment support (check-in, check-out)
- Template based scheduling with support for detailed conflict warnings
- Interactive conflict resolution

- Quick preview of availability for the month
- Simple identification of booking status (fully booked, heavily booked and lightly booked)
- Color coded appointment types (certification, nutrition discussion, high risk)
- Automatic check-out when EBT benefits are issued
- Customizable appointment notifications
- Automatic synchronization of nutrition topics from nutrition groups discussions to client records

3.6 Farmers' Markets Layer

The Farmers' Markets layer enables the configuration of farms and markets as well as the association of farms with markets. Features supported include:

- Add / Edit farms
- Tracking farm findings and follow up checks
- Add / Edit markets
- Tracking market finding and follow up checks
- Issued Cash Value Voucher (CVV) lookup
- CVV inventory per clinic
- Adding CVVs to a clinic inventory
- Reports
    - Issued vs. Redeemed tracking
    - Monthly participation by category and ethnicity
    - Checks and CVV paid

3.7 Finance Layer

The Finance Layer services are used to provide support in the financial area. Features supported include:

- Grant management
- Cash flow analysis
- Rebate management
- Generation of the FNS-798 report

3.8 Vendor Management Layer

The Vendor management layer enables the authorized user to configure and maintain retailer vendors available within the HANDS system for the state. Configurable options include:

- Add/Edit vendors
- Configuration of banking details
- Configuration of food items sold
- Association of wholesalers with vendors
- Tracking vendor communications
- Mass vendor communications
- Tracking education classes attended by the vendor
- Food Benefit redemption lookup
- Resubmission of rejected food benefit checks
- Calculation of vendor's competitive status in relation to other vendors.
- Configuration of maximum payment amount for a food item either by setting override percent and standard deviation in relation to other vendors in the peer group or statically setting the maximum amount.

3.9 Program Integrity Layer

The program integrity layer provides services to help the authorized user to ensure the program integrity of the states WIC program. Supported features include:

- Disqualifications
- Resolve Dual Enrollment
- Create Dual Enrollment file extracts

3.10 Batch Processing Layer

Batch Processing consists of a group of software routines dedicated to a particular batch processing service. Batch Processing services include:

- Food Benefit Issuance Processing
- Transmission of Files to Financial Institution
- Food Benefit Redemption Processing
- Automated Category Change and Termination Processing
- Appointment Scheduler Processing
- Caseload Determination
- Financial Summary Processing

3.10.1 Batch Processing Topography

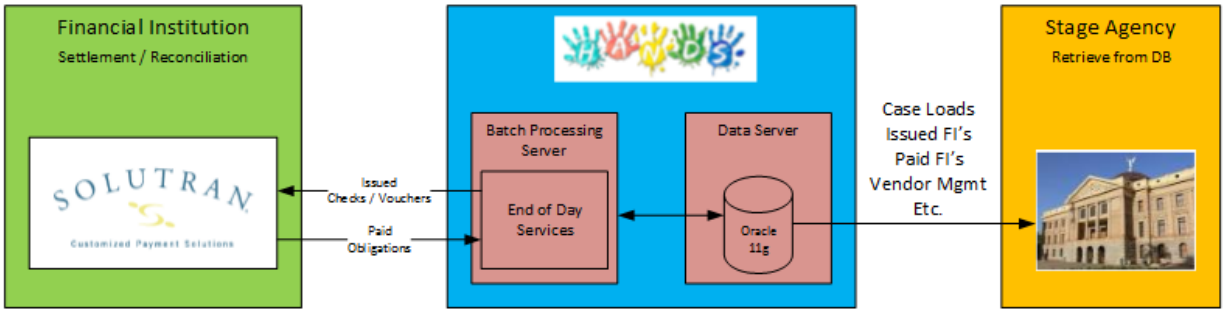

Figure 9: Batch Processing Topography

Note – The Batch Processing engine is typically installed on its own physical server to ensure peaks in the consumption of server resources do not compete with and possibly adversely affect the performance of other time critical services such as the UI services.

3.10.2 Batch Processing Order of Operation

The End of Day (EOD) batch processing routines are executed once per day at a time of day chosen by the state agency. Batch processing consists of a series of sub-routines that follow an order of operation coordinated by the main windows batch file (HNDZ_EOD.BAT) responsible for controlling the entire EOD batch process work flow. The following table details the EOD sub-routines and the order of operation.

Order	Calling Module / Script	Processes	Script Name	LA	Hands	Bank
Start	**WINDOWS Scheduler**	Runs the End of Day batch processing controller (HNDZ_EOD.BAT) at the configured time of day. The HNDZ_EOD.BAT controls the entire central EOD process.	HNDZ_EOD.BAT			
1.0	**HNDZ_EOD.BAT**	Build issuance data files and send to the financial institution (FI)	HEOD_ISSUE.SQL			
1.1	HEOD_ISSUE.SQL	Issuance data preparation	EODADM.HANDS_EOD_ISSUE.EOD_MAIN			
1.1.1	EODADM.HANDS_EOD_ISSUE.EOD_MAIN	Void food instruments that are past their stale date	STALE_DATE_FIS		x	
1.1.2	EODADM.HANDS_EOD_ISSUE.EOD_MAIN	Compile direct payment and revalidated FI information	OBLIGATE_FIS		x	
1.1.3	EODADM.HANDS_EOD_ISSUE.EOD_MAIN	Executes INSERT_INTO_BANK_REPORTS	BUILD_CHECK_ISSUANCE		x	
1.1.4	EODADM.HANDS_EOD_ISSUE.EOD_MAIN	Insert into I_BANK_REPORTS	INSERT_INTO_BANK_REPORTS		x	
1.2	HEOD_ISSUE.SQL	Create bank reports data file	ISSUE_FSMC.SQL		x	
1.3	HEOD_ISSUE.SQL	Create WIC farmers market coupons file	ISSUE_FMFSMC.SQL		x	
1.4	HEOD_ISSUE.SQL	Create WIC farmers market checks file	FARMER_FSMC.SQL		x	
1.5	HEOD_ISSUE.SQL	Executes VENDOR_FSMC.BAT	VENDOR_FSMC.BAT		x	

1.5.1	VENDOR_FSMC.BAT	Create vendor data file	VENDOR_FSMC.SQL		x	
2.0	**HNDZ_EOD.BAT**	Execute file transfers	HEOD_TRANSFER_FILES.BAT			
2.1	HEOD_TRANSFER_FILES.BAT	Transmit the issuance text files to the bank	FTPTRANS.EXE			x
2.2	HEOD_TRANSFER_FILES.BAT	Load the redemption data received from the bank	IMPORT_FILE.BAT			x
2.3	HEOD_TRANSFER_FILES.BAT	Load the farmers market data received from the bank	LOAD_FM_FILE.BAT			x
2.4	HEOD_TRANSFER_FILES.BAT	Send the Vendor Peer Group Averages Data to the bank	PGA_FSMC.SQL			x
2.5	HEOD_TRANSFER_FILES.BAT	Transmits the peer group averages data to the bank	FTPPGA.EXE			x
2.6	HEOD_TRANSFER_FILES.BAT	Create state's peer group averages data	OUTBOUND_PGA_FSMC.SQL		x	
2.7	HEOD_TRANSFER_FILES.BAT	Creates state's vendor data	OUTBOUND_VENDOR_FSMC.SQL		x	
3.0	**HNDZ_EOD.BAT**	End of day completion steps	HEOD_POST.SQL			
3.1	HEOD_POST.SQL	End of day completion steps	EODADM.HANDS_EOD_POST.EOD_MAIN			
3.1.1	EODADM.HANDS_EOD_POST.EOD_MAIN	Create vendor payment records	POST_BANK_DATA_EOD		x	
3.1.2	EODADM.HANDS_EOD_POST.EOD_MAIN	Create consolidated record of all farmers' market coupons issuance and redemption information.	POST_FARMERS_COUPONS		x	
3.1.3	EODADM.HANDS_EOD_POST.EOD_MAIN	Participant termination	EOD_TERMINATE	x	x	
3.1.4	EODADM.HANDS_EOD_POST.EOD_MAIN	Category changes	EOD_CATEGORY_UPDATES	x	x	
3.1.5	EODADM.HANDS_EOD_POST.EOD_MAIN	Category synchronization	EOD_CATEGORY_SYNCS		x	
3.1.6	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Obligations	REBUILD_OBLIGATIONS		x	
3.1.7	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Outlays	REBUILD_OUTLAYS		x	
3.1.8	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Rebate Details	REBUILD_REBATE_DETAILS		x	
3.1.9	EODADM.HANDS_EOD_POST.EOD_MAIN	Store FI Divisibility Records	EOD_FI_DIVISIBILITY		x	
3.1.10	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Active Certifications	REBUILD_ACTIVE_CERTS		x	
3.1.11	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Client Part Base	REBUILD_CLIENT_PART_BASE		x	
3.1.12	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Caseload Details	REBUILD_CASELOAD_DETAILS		x	
3.1.13	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Caseloads	REBUILD_CASELOADS		x	
3.1.14	EODADM.HANDS_EOD_POST.EOD_MAIN	Move data to Caseload Details	MOVE_DATA_TO_CASELOAD_DETAILS		x	
3.1.15	EODADM.HANDS_EOD_POST.EOD_MAIN	Move data to Caseloads	MOVE_DATA_TO_CASELOADS		x	
3.1.16	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Caseloads CSFP	REBUILD_CASELOADS_CSFP		x	
3.1.17	EODADM.HANDS_EOD_POST.EOD_MAIN	Move data to Caseload CSFP	MOVE_DATA_TO_CASELOADS_CSFP		x	
3.1.18	EODADM.HANDS_EOD_POST.EOD_MAIN	Rebuild Peer Group Averages	REBUILD_PEER_GROUP_AVGS		x	
3.1.19	EODADM.HANDS_EOD_POST.EOD_MAIN	Update State Clients	UPD_STATE_CLIENTS		x	
3.1.20	EODADM.HANDS_EOD_POST.EOD_MAIN	Update appointments	UPDATE_APPT_TABLES	x	x	
3.1.21	EODADM.HANDS_EOD_POST.EOD_MAIN	Update Budgets	UPD_BUDGETS		x	

4 EXTERNAL INTERFACES

HANDS was designed to interface with a variety of external interfaces supporting a variety of access methods and file formats. HANDS is currently capable of exchanging data with the external interfaces listed in the table below:

External Interface	Data Description	Exchange Format	Exchange Direction	Exchange Frequency	Exchange Method
Formula Manufacturer	Formula Rebate File	Fixed length (ASCII)	Out	As needed / Monthly	Email
Solutran	Redemption, Farmers' Market, Vendor Peer Group Averages, Issuance	Fixed length (ASCII)	In/Out	Daily	FTP
Other States	Dual Participation data	Fixed length (ASCII)	In/Out	As needed / Monthly	Manual
Vendor Portal	Vendor self service management	Vendor Portal API	In/Out	Real time	Vendor Portal API
Participant Reminders	Appointment Reminders sent once daily	Text	Out	Daily	SMS / Email

5 STATE DEPLOYMENT MINIMUM REQUIREMENTS

A states deployment of HANDS should take into account sizing estimates such as the state case load and the number of concurrent users for the state agency over the next 5 years. Using the projected usage data, CMA Consulting will work with the state agency to identify the appropriate deployment environment of the system. There are many deployment options for HANDS available to a state agency. That being said, the minimum requirements for deployment to a small state agency are:

Hardware

- 16 GB Memory
- 8 IntelXeon Processor
- Windows Server 2008 R2 x 64
- 10.0 Gbps network connection

Software/Features

- IIS7
- .NET Framework 4
- WCF Activation
- devArt dotConnect for Oracle 6.80
- Oracle 11g

6 CLINIC DEPLOYMENT MINIMUM REQUIREMENTS

Clinic PCs used to access the HANDS web portal should meet the following minimum requirements:

- MS IE9+
- Ability to support a resolution of 1024x768 or greater
- Font-sizes set to small
- Food Benefit voucher/check printer connected
- Scanner connected and configured as specified below
- Signature pad connected and configured as specified below

6.1.1 Check Printer Setup

The check printer template is dependent on the financial institution contracted by the state. Check printer settings are dependent on the printer(s) manufacturer and model utilized by the state clinics. CMA Consulting Services will work with each state agency to ensure HANDS is configured properly to interface with the chosen printer type(s) installed at the clinics and to define the check printing template within HANDS as dictated by the financial institution.

6.1.2 Scanner Setup

Scanner setup at the clinic's PCs as follows:

- Internet Explorer 9 or greater must be installed
- IE Security settings
    - Add HANDS web site to the Trusted Sites
    - Make sure that Trusted Sites are configured to have Protected Mode turned off
    - Install Epson Perfection V30 Scanner drivers.
- Apply the provided setup script to update Silverlight registry settings and a establish security trust of the Silverlight plugin. Install digital certificate used to sign Silverlight XAP file on client PCs
    - Install in Trusted Root
        - Select Show Physical stores and store on Local Computer
    - Install in Trusted Publishers
        - Select Show Physical stores and store on Local Computer

6.1.3 Signature Pad Setup

Signature pad setup at the clinic's PCs as follows:

- Connect a Topaz Signature Pad Model: T-LBK66SE-BHSB-R.

- Install SigPlus software for Topaz Signature Pads: http://www.sigpluspro.com/sigplus/index.html
- Install the CMA provided SigPlusWrapper software. . IE Browser Settings | Security Tab
    - Add HANDS production website to list of Trusted Sites
    - Make sure that "Enable Protected Mode (requires restarting Internet Explorer)" is NOT selected.
    - Make sure "Trusted Sites" is selected and then click on the "Custom level…" button
- Scroll down to the "ActiveX controls and plug-ins" section and configure as follows:
    - Allow ActiveX Filtering: Disable
    - Allow previously unused ActiveX controls to run without prompting: Enable
    - Allow Scriptlets: Enable
    - Automatic prompting for ActiveX controls: Enable
    - Binary and script behaviors: Enable
    - Display video and animation on a webpage that does not use external media player: Disable
    - Download signed ActiveX controls: Enable
    - Download unsigned ActiveX controls: Enable
    - Initialize and script ActiveX controls not marked as safe for scripting: Prompt
    - Only allow approved domains to use ActiveX without prompt: Disable
    - Run ActiveX controls and plug-ins: Enable
    - Script ActiveX controls marked safe for scripting: Enable

6.1.4 SMS Text Messaging

HANDS can be configured at the state level to support sending SMS text message appointment reminders to participants. HANDS currently supports an interface to the One Call Now service for transmission of SMS text messages within the continental US. The service was designed to accommodate future expansion of support to alternative SMS text delivery options. Appointment reminders are sent to participants with appointments occurring the next calendar day from when the reminder is initiated. Reminders are typically configured to be sent once per day at a configurable time of day. For example reminders are configured to be sent every morning at 10:00 AM. On January 1st the reminders will be sent to participants with appointments scheduled on January 2nd. On January 2nd the reminders are sent to participants with appointments scheduled on January 3rd.

7 DEVELOPMENT DETAILS

Coding standards and procedures provide a common format for the source code and comments. This supports the development team to share code and ideas expressed within the code in a style and format that is uniform across the team.

7.1 Coding Standards Highlights

- Private variables are camel-cased.
- Public variables, properties, and all methods are Pascal-cased.
- Curly braces each have their own line, except for very short property getters and setters.
- Comments are always on their own lines above code (not to its side.)

7.2 System Architecture

HANDS is divided into three logical tiers: data, domain, and presentation. Data is accessed by LINQ to SQL classes. Domain represents business logic and mapping. The presentation layer is an ASP.NET MVC Framework (version 4) application.

7.2.1 LINQ to SQL Classes

LINQ to SQL classes are mappings of database tables to code in HANDS. The classes are generated by a text generation template that reads the database schema. When a change is made to the database, the text template must be executed to ensure consistency.

To connect to an Oracle 11g database, HANDS employs the third-party library called dotConnect for Oracle by devArt. The library responsible for creating LINQ to SQL classes is called LinqConnect (also by devArt.) HANDS utilizes the Entity Developer designer tool that comes with LinqConnect to add, modify, or remove database tables to or from the LINQ to SQL class diagram.

See http://www.devart.com/linqconnect/ for thorough documentation on these data access components.

7.2.1.1 Performance Consideration

Sometimes it is necessary to avoid parts of the LINQ to SQL code execution in order to squeeze performance out of an intense operation. There are two optimization methods generally employed by HANDS to accomplish such optimizations.

The first level of optimization is known as “efficient projections.” For convenience, the LINQ to SQL classes will make additional round-trips to the data source to fulfill parent and/or child object references. But you may not need the additional data in your operation. To efficiently project a LINQ expression, a developer may map LINQ to SQL object properties into an anonymous object or a defined transient object. In doing so, additional round-trips to the database are not made because relational data is not automatically loaded.

The second level of optimization is to use HANDS' DatabaseFactory to get an instance of a database connection class, which simply uses ADO.NET classes, to map a parameterized SQL statement to a list of LINQ to SQL objects. The code execution will be several times faster than writing LINQ expressions, but it is more error-prone and more difficult to find bugs.

7.2.2 Domain Classes

There are two kinds of domain classes: service and model. Service classes consist of business logic validation and data access via aforementioned LINQ to SQL classes. It is the job of a service class to validate input, form a query and send it to the LINQ to SQL classes, collect the query results and map the results to transient objects. Service classes may be found in the Hands.Domain.Service project.

A transient object is created from any of the domain classes in the model category. These objects are short-lived and do not need to represent tables in a database in a 1:1 mapping. Domain models are created to fulfill the requirements of a screen in HANDS. Domain classes are found in the Hands.Domain.Model project.

7.2.3 ServiceFactory

HANDS consists of over 250 service classes. To normalize access to these class instances, and to provide consistent code instrumentation for all instances created, there is a factory class named ServiceFactory.

Given an interface type, the ServiceFactory will find the registration for the corresponding service class and provide an instance, by proxy, for that class. The proxy is a lightweight object that calls methods on the underlying concrete service instance. This allows the ServiceFactory to provide monitoring for unhandled exceptions and to capture timing metrics for all executing code. This is a powerful tool because it logs critical information so that developers can identify and address bugs quickly.

In order for this to work, all service classes must implement a service contract – a C# interface that lives in the Hands.Domain.ServiceContract project. Additionally, the location of each service must be identified in the ServiceContractMap class (also in the Hands.Domain.ServiceContract project.)

To get an instance of a service class (from, say, the presentation layer,) simply call "ServiceFactory.GetServiceFor<InterfaceType>()" where "InterfaceType" is a C# interface that identifies the methods you want to call.

7.2.4 Logging

HANDS logs unhandled exceptions and slow service methods to the database in tables S_ERROR_LOGS and S_OPTIMIZATION_LOGS, respectively.

Each entry in the error log is identified uniquely by a ticket number. When an unhandled error happens in HANDS, the user is alerted with a user friendly error page displaying the ticket number. This process greatly aids help-desk staff and developers in locating and fixing bugs. The error log table primarily displays the error message and associated .NET stack trace at the site of the unhandled exception.

To identify methods that are executing slowly, HANDS will write entries to the optimization table if a given amount of time elapses during a service method call. The amount of elapsed time to trigger an optimization log entry is configurable by the top-level configuration file (typically Web.Config) in the AppSettings configuration section. An optimization log entry shows the name of the executing class, the method called, the amount of time it took to complete, the number of records loaded from the database, and the number of objects created during the call.

7.2.5 Presentation

The presentation layer is divided into functional areas. Each functional area has its own layer supertype – a base class with functionality specific to that functional area.

When a URL is accessed in HANDS, the request is processed by the MVC Framework and routed to its functional area, controller, and action. The action performs field-level validation and requests an instance of a service class from the domain via ServiceFactory. When the results from the domain are returned, the action processes the results and returns HTML to the requesting web browser.

HANDS does not deviate from the suggested Microsoft pattern for developing ASP.NET MVC Framework applications. Therefore, the training material provided by Microsoft at http://www.asp.net/mvc is highly relevant to development in HANDS.

7.2.6 UI Widgets

HANDS boasts a large collection of configurable user interface widgets based on a framework developed specifically for the HANDS system. The widget framework is further based on jQuery UI. By using jQuery UI as the foundation for the user interface, accessibility (http://jqueryui.com/about/#deep-accessibility-support) and visual customization (http://jqueryui.com/about/#flexible-styling-and-themes) are inherently supported in the HANDS system.

Using the HANDS widgets is as simple as calling HTML helpers in an ASP.NET MVC Framework view. A partial list of UI widgets includes: Buttons, Addresses, Accordions, Auto-complete, Check-boxes, Combo-boxes, Data-tables (with paging and nested rows,) Text-boxes, Currency text-boxes (including a financial accounting mode,) Select boxes, Multi-select boxes (with filtering,) Object previews (show a PDF,) Time-pickers, Text-areas, Icons, Graphs and several utility widgets.

7.2.7 Javascript

Javascript is tantamount to creating a desktop-like browsing experience, and therefore receives first-class treatment in HANDS. jQuery version 1.9.1 is used by HANDS to create a consistent experience across several web browsers. Each widget developed for HANDS also uses the jQuery UI widget factory pattern to be compliant with the greater jQuery Foundation and its large community of developers.

The base set of Javascript files that HANDS uses (roughly 45 files) are compressed and minified when delivered to a web browser accessing the production environment. These scripts include general functionality that is used in all HANDS web pages.

Script that is specific to one page is loaded on-demand. The ASP.NET MVC Controller identifies the scripts that need to be delivered along with the HTML for the current page to be fully functional. This ensures that each page in HANDS only loads the minimum amount of Javascript that it needs to execute.

7.3 Development Process and System Deployment

The deployment of HANDS is highly scripted. The deployment scripts provide the necessary automation to ensure that upgrades of the application are deployed with minimal effort and risk of human error.

Deployment is tightly coupled with the software development life cycle. The development team follows an Agile software development approach. The Agile software development approach promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change. Some of the benefits awarded from the Agile software development approach include:

- Build breaks are caught early
- Development efforts are conducted against the latest build available
- Short feedback loop for developers to see the affects their checked-in effort has on the build
- Integration testing occurs early in the process ensuring issues related to how system components are integrated with one another are caught early in the development cycle
- Accountably on each developer is enforced, as each developer is adversely effected when the resultant build of the combined changes is not stable

The build process follows a continuous integration approach. A Team Foundation Server (TFS) is used to manage the build process. TFS is responsible for automating the build process, which includes features such as:

- Merging development branches into the main source image
- Execute Unit tests
- Build deployment scripts
- Reporting build status
- Supporting development team collaboration
- Test Case Management
- Version Control

7.4 Unit Testing

Units tests are performed as part of the continuous integration build process. Developers are responsible for adding new unit tests and maintaining existing unit tests as they introduce new services and features to the application. The developers follow unit test guidelines to ensure a standardized approach to unit testing as well as to ensure suitable test coverage of application features.

7.4.1 Developer Unit Test Checklist

General

- All of the fields specified in the requirements documentation are presented in the user interface
- The fields names presented in the UI exactly match the fields names presented in the requirements documentation. This includes abbreviations, spelling, and capitalization
- Buttons that add records to the system should be named "Add" (not "Add Vendor", "Add Family", etc.)
- Phone Numbers are displayed using standard format
- Dates are displayed using standard format
- Blue expander buttons on Nested Data tables appear in leftmost column of data table
- Cancel Button should be right most button in Form Actions toolbar
- Save Button should be immediately to the left of the Cancel button in the Form Actions toolbar

CRUD Functions

- For Adds, click save without entering any data and confirm that all required fields specified in the BRD are immediately identified
- Verify that clicking the Cancel button returns the form data to state it was in after the last successful save.
- Click the Save button and verify the following:
    - Fires all validation and reports errors in a fashion consistent with the rest of the system. If there are errors on the page, verify that the data was not saved to the database
    - Once all validation errors are addressed, verify that data is saved to the database.
    - Verify that a consistent notification message is displayed indicating the record was successfully saved
- Modify / Add data to page without clicking Save. Verify navigating away prompts user that page has unsaved data.

Child Data Tables

- When adding / editing records to a child data table verify that required field validation fires as well as any other business rules.
- Add / edit / delete data in a child data table without clicking on the page level Save button. Navigate away from the page and verify that the user is prompted that the page has unsaved data.

- When adding / updating / deleting data in a child data table, verify that you are only changing the data in the workspace. The data should not be saved to the database until the page level Save button is clicked.

Search Functions

- Verify that Search Button initiates a search and does NOT fire view model validations
- Verify that Cancel Button clears search parameters
- Verify that search results appear on same screen as search parameters.
- Verify that data in search results displays correctly
    - Dates should be displayed in standard date format to be consistent with other dates in the system
    - Phone Numbers should be displayed in standard phone number format to be consistent with other phone numbers in the system.

Navigation

- Verify that buttons are in the appropriate tool bar.
    - Toolbar provides primary navigation, including navigation between modes (e.g., in WIC Services navigation between Family, Client, Immun, etc.)
    - Secondary Toolbar provides navigation within a sub area (e.g., in WIC Services Food Package between Assignment, Issuance, etc.)
    - FormActions Toolbar provides actions specific to the current page (e.g. Save, Cancel, etc.)
- Verify that the Save Button navigation is as follows:
    - For Add screens Save button should either navigate back to the previous screen (e.g. when adding to a list) or should navigate to the Edit screen for the given record… whichever is more appropriate for the given function.
    - For Edit Screens the Save button should remain on the current page so that the user can continue to edit
- Verify that the Cancel button navigation leaves the user on the current page

Error Handling

- Verify that if any of your Service calls can throw HandsBusinessExceptions, that these exceptions are handled in the controller and that the exception message is added to the Model State so that it is displayed in the browser just like other validation messages
- Verify that no other exceptions such as system related exceptions are being handled by controller code. These must be handled by the framework.

- Verify that RedirectOut / RedirectIn are setup correctly within your controller. Enter data on your form that will raise a validation error and verify that the validation message is displayed and that none of the entered data is lost when the validation message is displayed

8 APPENDIX A: DATA DICTIONARY AND DATA MODELS

HANDS%20v3%200
%202%20Data%20Di

The imbedded Data Dictionary is posted separately at www.azwic.gov.