ACIA Software

*Really, Not Justanother Computer Industry Acronym.*

BroderickRealty.com Software Requirements Specification

By Kenneth S. Courtney [http://aciasoftware.com](http://aciasoftware.com/) May 16, 2005

C:\Data\Development\Projects\Broderickrealty\Documentation\SRS - Broderickrealty.Com Phase I.Doc

# Table of Contents

[Table of Contents ii](#_TOC_250038)

[Revision History iii](#_TOC_250037)

1. [Introduction 1](#_TOC_250036)
   1. [Purpose 1](#_TOC_250035)
   2. [Document Conventions 1](#_TOC_250034)
   3. [Intended Audience and Reading Suggestions 1](#_TOC_250033)
   4. [Project Scope 1](#_TOC_250032)
   5. [References 1](#_TOC_250031)
2. [Overall Description 2](#_TOC_250030)
   1. [Product Perspective 2](#_TOC_250029)
   2. [Product Features 2](#_TOC_250028)
   3. [User Classes and Characteristics 3](#_TOC_250027)
   4. [Operating Environment 3](#_TOC_250026)
   5. [Design and Implementation Constraints 3](#_TOC_250025)
   6. [User Documentation 3](#_TOC_250024)
   7. [Assumptions 3](#_TOC_250023)
   8. [Dependencies 3](#_TOC_250022)
3. [System Features 4](#_TOC_250021)
   1. [Develop MLS Module(s) 4](#_TOC_250020)
   2. [MLS Data Import Program 5](#_TOC_250019)
   3. [DotNetNuke Custom Skin Suite 5](#_TOC_250018)
   4. [Broker Information Pages 6](#_TOC_250017)
   5. [Lead/Customer Management System 6](#_TOC_250016)
   6. [Integration of 3rd Party DNN Real Estate Module 6](#_TOC_250015)
   7. [Integration CCIM Search in IFrame 6](#_TOC_250014)
4. [External Interface Requirements 7](#_TOC_250013)
   1. [User Interfaces 7](#_TOC_250012)
   2. [Hardware Interfaces 7](#_TOC_250011)
   3. [Software Interfaces 7](#_TOC_250010)

[Communications Interfaces 8](#_TOC_250009)

1. [Other Nonfunctional Requirements 8](#_TOC_250008)
   1. [Performance Requirements 8](#_TOC_250007)
   2. [Safety Requirements 8](#_TOC_250006)
   3. [Security Requirements 8](#_TOC_250005)
   4. [Software Quality Attributes 9](#_TOC_250004)
2. [Other Requirements 9](#_TOC_250003)

[Appendix A: Glossary 9](#_TOC_250002)

[Appendix B: Issues List 9](#_TOC_250001)

[Appendix C: Customer Bill of Rights 10](#_TOC_250000)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
| Draft | 05/16/2005 | Initial Draft | Initia Draft |
| Version 0.9 | 05/18/2005 | Final Draft | Final Draft |

## Introduction

### Purpose

This document will identify the requirements of the Broderick Realty web based enterprise information portal. The system will provide end users the ability to learn more about Broderick & Associates services and brokers search updated daily MLS Listings & photographs, and display Commercial Properties from the CCIM external system within an IFrame. This SRS will only cover the first phase of development of this site.

### Document Conventions

In this document, when referring to the software system being created the term ‘web application’ will be used. This is referring the system, the behavior of the system, functional requirements of the system, or software requirements of the system.

In this document, when referring to the terms ‘PA’ or ‘Module’ the intended meaning is a collection of cooperating objects (and/or types) that solve a given problem or provided a service packaged to form a module within the DotNetNuke framework. A PA is bundled, or packaged, in a manner such that it can be used as a ‘Black Box’ by DotNetNuke (and vice versa). In the context of this PA and .NET, that packaging includes an assembly (\*.dll file) constituent .ASCX files, and a module definition file.

### Intended Audience and Reading Suggestions

This document is intended for stake holders of the project.

### Project Scope

The intent of the Web Application outlined in this document is extending the base services provided by the DotNetNuke Enterprise Information Portal (EIP) Framework 3.0.13 to provide a superior experience to BroderickRealty.com users in the form of Custom MLS Searches with results displayed directly from the BroderickRealty.com web site, integrated Lead Management System providing Broderick Brokers lead management tools, framed CCIM Content from 3rd party provider, bundled within a distinctly unique Broderick Realty Branded web application.

The components defined in this document will make use of the existing extensive API provided by DotNetNuke for Security Accounts, authentication, and exception handling, to name but a few of the many services offered.

### References

* + - DotNetNuke Home page:

<http://www.dotnetnuke.com/>

* + - Interview with Shaun Walker founder of DotNetNuke project on DotNetRocks Radio show:

<http://www.franklins.net/fnetdotnetrocks/dotnetrocks.aspx?showid=77>

* + - Official DotNetNuke Forums: <http://www.asp.net/Forums/ShowForum.aspx?tabindex=1&ForumID=90>
    - DotNetJedi Real Estate Module

<http://www.ibuyspysolutions.com/DesktopModules/PA_DigitalStore/ds_MoreInfo.aspx?Mid=328&ItemID=103&TabID=22>

* + - DNNDev XMod 3.1: <http://www.dnndev.com/Default.aspx?tabid=213>

<http://www.snowcovered.com/snowcovered2/Default.aspx?tabid=166&CatalogItemID=1423>

## Overall Description

### Product Perspective

Phase I of the BroderickRealty.com EIP Project is intended to provide BroderickRealty.com’s users the ability to search all active local MLS Listings within the BroderickRealty.com web site, retaining Broderick’s distinctive branding, and allow those users to communicate with Broderick Realty Brokers for further information about all MLS Listings. These listings will be presented to the user in a user friendly tabbed manner similar to that found on Realtor.com, in which each major category is represented by a separate ‘Tabbed’ Link.

Users browsing the Standard MLS Listings will be able to email the designated broker directly. A Lead Management system will be integrated into a separate Employee Only Intranet Portal which will facilitate the addition of new leads purchased by broderick, and assignment of those leads to an agent. Once the lead has been assigned, the broker will be able to submit email messages to the lead based upon several ‘canned’ messages based upon a standard progression.

### Product Features

* Custom DotNetNuke 3.0.13 Skin
  + Enable the configuration of multiple content photos in a header of a fixed size by configuring multiple Image Modules into multiple Content Frames, with an underlying ‘default’ set of images for single view version of skin.
  + A separate Administrative Skin which minimizes the amount of screen real estate used by presentation elements for Admin Portions of application as well as future use on Employee Intranet site.
  + A separate ‘Splash Page’ consisting of a single Content Frame.
* MLS Listing Module
  + Development of a series of Read Only/Searchable MLS Search Modules which will display each listing similarly to the Realtor.com standard listing, based initially upon the listing view provided in the DotNetJedi DNN 3.0 Real Estate module. Each module will search the content type for that listing type
  + Display of MSL Module as appropriate for Separate Search Types as specified by client. This will be an administrator level configuration item on each instance of the real estate module.
* MLS Import Program: MLS Information will be provided via an FTP (File Transfer Protocol) Push Method whereby the Pinellas Realtor Board will transmit a file on a daily basis for import into the web site’s MLS system.
  + Development of a Data Transformation Services Automated Batch Job which will connect import the data provided in the nightly ftp push provided by the Pinellas Real Estate Board. This data will replace existing data on a nightly basis. This file is typically 20 megabytes in size.
  + Development of a .NET Managed Code Component utilizing the ICSharp Zip Lib and Implementing the DotNetNuke Scheduler interface to Import the daily Photograph Uploads. The Board only transmits changes on a daily basis to the photograph collection, and will provide a DVD of the last 6 months of photographs from the system. It is the developer’s job to remove the dead/inactive entries.
* Integration of DotNetJedi DNN 3.0 Real Estate Module:
  + Download, Install & configure the Real Estate Module. This module will be utilized for Non-MLS Listed properties and maintained by Broderick administrative personnel on a regular basis.
* Broker Information Catalog: Broderick Employees will each have a separate entry in a searchable Broker Database. These entries will be accessible from the MLS Listings when that broker is a Broderick Employee via the Broker’s MLS Assigned ID Number contained in the MLS Entry. This broker information will be stored and managed using the facilities provided by the DotNetJedi Real Estate module. Additionally, a broker Search & List Module will be developed for placement on a page under the About Us Page on the top level Page/Tab.

### User Classes and Characteristics

For the purposes of this SRS, a user can be either be a human user or a machine user.

* + - Associate: Any associate or employee of the Broderick Realty who requires access to published web applications.
    - Anonymous Internet User: Any unauthenticated user of the system.
    - Client: Any client or non-employee user of the system.
    - Administrative User: An administrative user is a user responsible for managing either the users of the system, the applications of the system, or the mapping of the users to the applications of the system.
    - Pinellas Real Estate Board FTP User: Entity pushing content to the web server.

### Operating Environment

The server portion of the application is depends upon DotNetNuke Enterprise Information Portal (EIP) framework, which requires the Microsoft .NET platform. In addition, Microsoft SQL Server or Microsoft Data Engine is required as the Relational Database Management System for this application because of it’s inherent performance capabilities and ease of development by tight development tools integration within Visual Studio.NET, and the fact that it requires no customization or commercial support.

All modern web browsers supporting DHTML should be able access this site.

### Design and Implementation Constraints

TBD

### User Documentation

Extensive documentation facilities are incorporated into the DotNetNuke 3.0 core product, both for web site users and Administrative tools.

### Assumptions

* + - Subscribing applications will need modification to utilize the VB 6.0 COM Authentication Component.
    - Subscribing applications will be written in a language that can use this COM Component. Any possible additions or omissions from above list should be noted accordingly.

### Dependencies

* + - Developers have appropriate access to modify subscribing applications.
    - Subscribing Applications have access to SQL database of Host portal.

Any possible additions or omissions from above list should be noted accordingly.

## System Features

For the sake of brevity, only features developed for this application will be included in this specification. DotNetNuke provides numerous services which would otherwise need to be presented here such as: User Account Management, Role Membership, Exception Handling, Exception and Event Logging, Application Monitoring, Site Administration Tasks, etc. DotNetNuke in it’s 3.0.13 incarnation consists of in excess of 73,000 lines of code, all of which intended to provide this base set of services. Prior to this set of requirements, it is assumed that DotNetNuke 3.0.13 has been downloaded and configured appropriately at [http://broderick.aciasoftware.com.](http://broderick.aciasoftware.com/)

### Develop MLS Module(s)

* + 1. Description

The MLS Data is provided in a separate file for each type of listing: res, mul, lnd, com, rnt. For each File Type, a separate DotNetNuke module will be developed displaying at a minimum the fields required in any agreements between Broderick Associates and the Realtor Board.

As per MLS Data Usage Requirements, this data is subject to numerous rules & regulations, which essentially state that it cannot be modified in any way by a broker once published to the system. One specific rule is the data is updated at a bare minimum once a week. In the case of this system, the data will be updated daily, which will be covered in a later section.

* + 1. Functional Requirements

R1.1: For Each File Type, Create a new Visual Studio.NET VB.NET Web User Control Project. R1.2: Create a new VB.NET Class Library Business Logic Layer & Associated Data Provider conforming to DotNetNuke Data Access Layer specifications as documented in the DotNetNuke data access layer documentation. Create a separate Helper Class Library. Utilize both of those assets in the construction of each Specific Module.

R1.3: For each listing type’s Module (res, mul, lnd, com, rnt), Add the following Controls: R1.3a Main.ascx: This file is drives the display of controls within the module. This will read a query string parameter for control to load, loading the contol specified in the argument. This file will be identical in each module project, however, a unique file must exist in each project.

R1.3b: ucSearch.ascx: This Control provides the interface for search criteria for each type of search. Since each listing type has different criteria, there must be a separate control for each search type. (Requirement Numbers R1.3b\_res, R1.3b\_mul, R1.3b\_lnd, R1.3b\_com, R1.3b\_rnt)

R1.3c ucList.ascx: This control will provide page able, detailed list of the search results to a user based upon the module level specified number of results per page. There will be one List control representing a the common elements from all search types.

R1.3d ucDetail.ascx: This control will display an individual MLS item. This will be modeled after the existing DotNetJedi Real Estate module used elsewhere in this project. (Requirement Numbers R1.3d\_res, R1.3d\_mul, R1.3d\_lnd, R1.3d\_com, R1.3d\_rnt) R1.3e ucConfiguration.ascx: This control will only be available to users authorized to administer the module (this may be specified at the site, page, or module level). The module will store user settings into the DotNetNuke Module Settings collection, the following settings will be available:

Rows Per Page: The number of listing results to display per page: Because of restrictions associated with the data, there may be limitations on the number of results returned per page.

Broker Module Page: This will display a standard DotNetNuke link control to allow the administrator to point to the page on the site hosting the Broker Information Module.

R1.3f: ucAgentContact.ascx: This will be a single file in a common module for use by ALL MLS Modules providing Contact Information & Feedback Form (Separate from Broker information page, this control will integrate into ucDetail.ascx.

### MLS Data Import Program

* + 1. Description

The MLS Data is provided in a separate file for each type of listing: *res, mul, lnd, com, rnt*. This functionality will leverage both the Microsoft .NET Framework and Microsoft SQL Server Data Transformation Services.

As per MLS Data Usage Requirements, this data is subject to numerous rules & regulations, which essentially state that it cannot be modified in any way by a broker once published to the system. One specific rule is the data is updated at a bare minimum once a week. In the case of this system, the data will be updated daily, which will be covered in a later section.

* + 1. Functional Requirements

R2.1: For Each File Type, Create a new table in the Microsoft SQL Server Database. (Requirement Numbers R2.1res, R2.1mul, R2.1lnd, R2.1com, R2.1rnt). The Realtor Board has provided definitions for each of these tables.

R2.2: For Each File Type, Create a new DTS Package to import that file type into the SQL Server Database table for that file type. (Requirement Numbers R2.2res, R2.2mul, R2.2lnd, R2.2com, R2.2rnt).

R2.3: Develop a Microsoft .NET Managed Code Library using Vb.NET and Visual Studio.NET 2003 which will be configured to read the inbound FTP Directory for new photograph archive files provided in a standard ZIP file format. Utilize the free, open-source SharpZLib component to extract these files for placement within a writeable directory in the DotNetNuke Portal Directory. For each MLS Photograph, create separate Thumb, and mid- detail image, which will be specified in a configuration option to be determined.

R2.4: Develop a Microsoft .NET Managed Code Library using VB.NET and Visual Studio.NET to delete expired/orphan listings. If the listing is not present in the system, remove it.

R2.5: Create a Custom DotNetNuke Schedule task, referencing components created in R2.3 & R2.4 to enable automation of daily photo import task and orphan photo removal.

### DotNetNuke Custom Skin Suite

* + 1. Description

Create a custom DotNetNuke skin (see DotNetNuke Skinning Guide for details/definitions of skin/containers). Three separate Skin Files will be created for this project, and one single blank container will be used.

* + 1. Functional Requirements

R3.1: Create Master Skin File for use on site informational pages. This skin will consist of multiple content areas to enable dynamic display of pre-formatted & sized images from a directory for randomization.

R3.2: Create Skin file for use by administrative page views. This skin will include a subset of features and retain the basic theme of the master skin, but will maximize screen real estate for use by web applications.

R3.3: Create Splash Page: Create a blank, single Container Skin File for use as the web site’s Splash Page.

R3.4: Create an empty, no control container for use by all skins.

### Broker Information Pages

* + 1. Description

Create a module providing a list & detail view for each broker. This module will be configured using DNNDev.com’s XMod V3.1 for Dotnetnuke 3.0.1x.

* + 1. Functional Requirements

R4.1: Purchase, download, and install DNNDev.com XMod V3.1. Configure a new form using XMod for realtor Information.

### Lead/Customer Management System

* + 1. Description

Create a module providing a list & detail view for each broker. This module will be configured using DNNDev.com’s XMod V3.1 for Dotnetnuke 3.0.1x.

* + 1. Functional Requirements

R4.1: Purchase, download, and install DNNDev.com XMod V3.1. Configure a new form using XMod for lead management Information.

### Integration of 3rd Party DNN Real Estate Module

* + 1. Description

The DotNetJedi (IBuySpy Solutions) DotNetNuke Real Estate Module has been purchased in Source Code format such that it

* + 1. Functional Requirements

R6.1: Purchase, Install & Configured the DotNetJedi Real Estate PA 3.0 Module

### Integration CCIM Search in IFrame

* + 1. Description

Display the CCIM Search in an IFrame Module.

* + 1. Functional Requirements

R7.1: Configure a DotNetNuke Page with an IFrame Module to load the Foreign System’s content.

## External Interface Requirements

### User Interfaces

General

* + - Project specific standards will be used to ensure consistency across the tools.
    - The pages can be viewed with IE5+.
    - The pages will be designed for screen resolution of 800x600pixels.
    - Cookies should be enabled in the browser setting for proper functionality of the application.
    - Verdana, Arial, Helvetica fonts should be used across the tools.

Readability

* + - Terminology used in the tools should be easily understandable by the user.

Page design

* + - All Pages performing similar functionality should have consistent Look & Feel.
    - Appropriate titles should be given to each page. The titles should specify the functionality of the Page.
    - Appropriate alternate text should be provided for all the images, to help in navigation and better readability.

Navigation:

* + - Access to the functionality of the application will be restricted based on the user type.
    - Navigation facilities will be provided to navigate from one page to another page with minimum number of clicks.

Messages:

* + - Descriptive and user friendly messages for invalid input should be provided in all the pages.
    - Completion/Confirmation messages should be displayed when the application processes the data successfully.

### Hardware Interfaces

##### <NONE>

### Software Interfaces

The PA Interacts with all other software via the DotNetNuke API, including Exception management, logging, and CRUD operations.

Pinellas Realtor Board MLS data will be ‘pushed’ once daily via FTP. This FTP Data will be imported using Microsoft Data Transformation Services batch process executed by the SQL Server Agent.

Additionally, new photos are uploaded in the same directory on a daily basis, and those photos will be imported using DotNetNuke scheduler job as described in the DotNetNuke scheduler documentation.

### Communications Interfaces

* + - HTTP protocol should be used for communication with the web browser and the web server.
    - An FTP Site will be configured to provide a communications mechanism between the Pinellas County Real Estate Board and the Broderick & Associates Web Application.

## Other Nonfunctional Requirements

### Performance Requirements

No Performance requirements have been identified. Regardless, DotNetNuke provides a high performance framework utilizing Microsoft’s best practices and patterns for Data Caching. DotNetNuke caches both frequently accessed data and User Interface elements, and only updates them when needed by the user. This caching crosses session boundaries providing an almost instantaneous response to users.

### Safety Requirements

NONE

### Security Requirements

* + 1. Database:

Access to the database will be limited to the following two users.

* + - 1. User with Admin privilege will have full access to the database.
      2. One user per client having complete accesses to the client specific schema.
    1. Application:
       1. Access to the tools is restricted to authenticated Users through secure login Id and password.
       2. One admin user will be created during the installation.
       3. Admin User shall have full access whereas the Normal User shall have limited access to the tools.
       4. Each page should be checked for valid user.
       5. The session will expire within set time and the users will be provided with appropriate messages to proceed further.

### Software Quality Attributes

##### NONE

## Other Requirements

##### NONE

# Appendix A: Glossary

###### CRUD: Create Read Update Delete Model.

###### DNN: DotNetNuke

###### Skin: A collection of text files in the form of Web User Controls (\*.ascx) and resources used by that file to create a theme for use by the DotNetNuke EIP framework.

###### Container: A collection of text files in the form of Web User Controls (\*.ascx) and resources used by that file to create a theme for use by the DotNetNuke EIP framework. EIP: Enterprise Information Portal

###### IFrame: An HTML Tag/Construct Enabling the framing of another complete HTML Document.

# Appendix B: Issues List

This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.

# Appendix C: Customer Bill of Rights

The Right to an Overall Plan

#### Customers have the right to a release plan that outlines what can be done, when, and at what cost. They have the right go through the development of that plan with you. How do you get customers and developers to agree on what is included in a project? You develop iteratively and agree in small pieces. This means developing a plan for every iteration of work. The iteration plan has to be detailed enough to describe what is to be done, when, and for how much.

The Right to Make Changes

#### Customers will change their minds—substitute functionality and alter priorities—and shouldn't have to pay exorbitant prices to do so. It is the nature of any Web project that, before the Web site is launched, something will change that. At the beginning of each iteration, give the customer the opportunity to change direction by requesting new stories or reevaluating story priorities.

The Right to Get Value

#### Customers should receive the greatest possible value out of every development hour. While they are paying for your time, they are also paying for your experience and good-quality work. Engage customers in the process—let them see what you are doing.

The Right to See Progress

#### Customers should be able to view a running system at the end of each iteration. At the beginning of each iteration, a set of stories is approved. For each story a provable deliverable is defined, which is what the customer will review at the end of each iteration to determine that the story is complete. This can be done by running the code and having it pass repeatable tests.

The Right to Be Informed

#### The project manager and team members should have regular access to customers in order to keep them informed. Communication is critical to the project, as it gives customers a chance to become more involved in development of the Web site and therefore more able to make key decisions along the way. Don't hide from customers. Keep them involved in everyday project development. If something comes up tell them right away; don't procrastinate. It is their right to determine the impact of a problem, and only they can say what an acceptable solution should be.