



Development Guide

Using Passageways Portal 3.1.3.0

Guidelines for extending the portal framework



Development Guide Using Passageways Portal Application

January 2004

por·tal (pôr'tl, pōr'-) *n.*

A secure, single point of access, or window, into a diverse information base; a gateway combining presentation, personalization, frameworks and application integration to support e-business models.

Take Note!

Information in this document, including URL and other Internet Web Site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Passageways, LLC.

© 2003 Passageways, LLC. All rights reserved.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

First Edition (January 2004) Revision 2.5

This edition applies to Passageways Portal Application Version 3.1 for use with Windows 2000

Comments may be addressed to:
Passageways Support Center support@passageways.net

When you send information to the Support Center, you grant Passageways a non-exclusive right to use or distribute the information in any way that it believes appropriate without incurring any obligation to you.

Contents

Preface.....	7
The team that wrote this book.....	7
Part 1. Passageways Portal Framework.....	8
Chapter 1. Architecture and Services	10
1.1 Passageways Portal Application	10
1.4 Passageways Document Indexing Service	15
Chapter 2. Framework Components.....	17
2.1 Portal Environment	17
2.1.1 Portal interface	17
2.1.2 Portal interface features	18
2.1.3 Portal navigation.....	18
2.1.4 Layout manager and portal engine.....	18
2.1.5 Pages.....	19
2.1.6 Themes and skins	19
2.2 Modules	20
2.2.1 Overview.....	20
2.2.2 Module Components	21
2.2.3 Bundled modules and samples	22
2.3 Passageways Assemblies	22
2.4 Workgroups	24
2.5 Roles.....	25
2.6 User profiles.....	25
2.7 Document Search	25
Steps to Index Your Documents.....	27
Scheduling the indexing process.....	28
Document Profiles	29
Document Recommendations	29
2.8 Federated Search.....	30
2.9 Alerts and Notifications.....	31
2.9.1 How to send an alert.....	31
2.9.2 Sending alert techniques.....	31
Part 2. Developing Modules.....	32
Chapter 3. Module Concepts	34
3.1 Overview.....	34
3.2 Module deployment	34
3.3 Technical Form.....	34
3.4 Module Configuration File.....	34
3.5 Installing modules.....	35
Chapter 4. Module API – Basic elements	37
4.1 Island	37
4.2 Development requirements	37
4.3 Option pages	38
4.4 Using island persistence capabilities with PrefString	39
4.5 Images and hyperlinks in islands.....	40
4.6 Using island context objects	40
Public Instance Constructors.....	42
Public Instance Properties.....	42
4.7 Size and layout rules for islands.....	43

4.8 Module Configuration properties	43
4.9 Using themes and style elements.....	44
4.10 Using user information.....	44
4.11 Using workgroup information.....	45
4.12 Storing credential information in the credential vault	45
Chapter 5. Headquarter Items	47
5.1 What are headquarter items?	47
5.2 Building a headquarter item.....	47
Chapter 6. Search Items.....	50
6.1 What are search items?.....	50
6.2 Building a search item	50
Chapter 7. Portal Event Items.....	52
7.1 What are portal event items?.....	52
7.2 Building an event item	52
7.3 Subscribing to an event item	52
Chapter 8. Action Links	54
8.1 What is an Action Link?	54
8.2 Building an Action Link	54
8.3 Action Link Scope.....	54
Chapter 9. Deployment descriptors.....	56
9.1 Portal configuration.....	56
9.2 Module deployment descriptor	56
Chapter 10. Island development issues	60
10.1 Testing islands.....	60
10.2 Island creation guidelines	60
10.3 Namespaces.....	60
10.4 Flash and ActiveX controls within islands	60

Preface

The primary purpose of this book is both to inform and illustrate the use and functionality of the Passageways portal application. Most of the text written is written for a technical audience with special emphasis on Microsoft technologies. These include the .NET Framework Release Version 1.0 This book may change over time however the general guidelines outlined should remain steady throughout the next several versions of the software. We take pride in writing this book and presenting our work to you. The portal is the product of years of work by teams ranging from North America to as far as India. We hope you find this book useful and use it as a guide in building portal applications atop the Passageways portal.

The team that wrote this book

The authors of this book originate from a variety of backgrounds and disciplines. We have tried to describe a small amount of their tremendous work below.

Christopher Beltran is the lead designer and developer of the portal application. He is responsible for product development, enhancements to the portal engine and accompanying management tools. He is also responsible for proposing new and upcoming technologies to the strategy team, which are to be incorporated into future versions of the software. Christopher may be contacted at chris@passageways.net

Part 1

Passageways Portal Framework

Architecture and Services

The Passageways application product family covers a wide set of products. We can find a wide variety of needs fulfilled within the Passageways family products.

This chapter introduces the products that we used in this book as well as products from the Passageways application family that can be implemented.

For more information about the Passageways product family, please refer to the Passageways Web site at <http://www.passageways.net>

Chapter 1. Architecture and Services

1.1 Passageways Portal Application

The main purpose of a portal is to provide a user with a single point of access to multiple types of information. A portal aggregates information in a way preferred by the user, regardless of the location or format of that information.

Passageways contains a number of elements which may not be familiar to you. Firstly, the primary entity of information delivery to users is the island. Islands are small self-contained applications that run within the portal and work to present a user interface for some type of data source.

For example, there may be an island to display group announcements and another island to access company email.

Since islands are typically built around a single data source such as a database or external application, they are packaged into a module. A module is a collection of islands and other items. The main purpose of a module is to effectively package a set of functionality. Each of these entities can be extended and configured as will be detailed in the following chapters.

Passageways runtime process flow

After installation, the portal developer configures and deploys the portal to users. The developer alters the layout and appearance of the default portal login page by using Visual Studio.NET or any text editor. Typically this default page is made to match a corporate standard.

After the login page has been customized, users must be imported or manually entered in. By default, a portal administrator user is created typically with the username "portaladmin" with an identical password.

After the portal contains imported user data, a user can log on to the portal. If a user attempts a logon, the incoming request passes through an authentication layer that provides controlled access to the portal. If the logon is authenticated, the user is presented with the default page, which could be a number of different types of pages, as we will see later in this book. The user's data is stored in a relational database for later retrieval and a new session is started. The Passageways portal processes the layout by generating markup for the portal page and rendering the islands that are activated and accessible to the user.

We will examine this framework in more detail in the following section.

The portal engine

Passageways portal engine provides a pure .NET – driven engine, fully capable of taking advantage of the many benefits provided by the underlying .NET Framework from Microsoft. This includes support for multiple devices, native web-services support, and more. The task of the portal engine is to efficiently manage the aggregation and display of islands on pages. The portal engine manages object instantiation as well as the passing of settings and properties to individual islands.

The process to build a page works as below

Portal content is accessed by clicking on a link in the navigation tree to the user's left. Control is passed to the appropriate Layout Manager, which parses the Url and enters into one of four possible states.

- The first state is when a user clicks on a tree link that leads to an external Url or web site. The layout manager forwards this request and opens up the appropriate web site for the user to view.
- The second state occurs when a user clicks on a tree link that represents a specific file for a group. The layout manager forwards the request to the specific file and hands control back to the web browser who then either opens the file in-line or asks the user to save the file locally. The layout manager does not perform any security checks on these files, as a non-group member could not have activated the tree link. (this feature may or may not be available in all portal installations)
- The third possible state occurs when a user who is not signed in attempts to open a private page for a workgroup. If a guest user attempts to open a secured page, the Layout Manager displays a special message with a warning notifying the user of the page's secured status.
- The fourth and final state is the most complex and the most common. This occurs when a signed-in user requests a page populated with islands on it. The page first creates a context object containing information pertaining to the user, the user's roles, the page properties, and other miscellaneous items. It then loads the appropriate islands for the page and builds a context object for each island. Afterwards, it loads each island and passes to it all the settings and properties that it requires to build itself. These are passed via the context object of type PWIslandContext. The Layout Manager then builds the appropriate matrices and places each island in its allocated area on the page. It then renders markup appropriate to the browser's type and functionality. Finally, it releases the markup containing the islands to the browser.

Each individual island controls caching and not the portal engine itself. Since some islands are dynamic and require immediate refreshes upon every call, the portal refreshes each island on every request. Within each island however, an output cache setting can be set to allow a cached copy of the island to remain in memory for a set period of time.

Constructing the home page

The home page is comprised of several frames and pages, which are used to construct the navigation structures, content area, search area, and main header bar. The top header frame is typically customized to reflect a predetermined corporate image.

The navigation area contains a tree structure that makes it simple and easy for a user to navigate. The navigation area may be hidden at any time to allow for a greater viewing area. The main content area is where most content is displayed including external web sites and island pages.

The home page may be a number of different page types. Firstly it can be the included headquarters page, which provides a unique summary of the user's recent activities along with individual status reports for each workgroup to which the user

belongs. Each item (workgroup and individual) is displayed in the observation-action format.

- Observations are simply points the portal has noticed about the item. For the personal item, this list may contain things like “Last login was 2 hours ago”. For a workgroup item on the headquarters page, this list might be things like “4 announcements waiting” or “8 group members online”.
- Actions are links that allow the user to initiate something against the portal. For a personal item, this might include opening the users personal calendar. For workgroup items, this might include searching for a list of the group’s members.

The home page may also be any existing page within the portal. This includes island pages within a workgroup. You may set the home page using the administrator tool located in the Toolbox.

Important Note: *If you make a particular workgroups island page the home page, every user will see this page and therefore have access to whatever information is located on it so use this feature carefully.*

Portal security overview

Passageways will provide access to many different applications, often at many different sites. Access to these applications must be transparent and seamless without compromising security. Because portals must support access from both inside and outside the firewall, the usual sets of security services are required: Secure Sockets Layer (SSL), Public Key Infrastructure (PKI), Password encryption and one-way hash functions, etc.

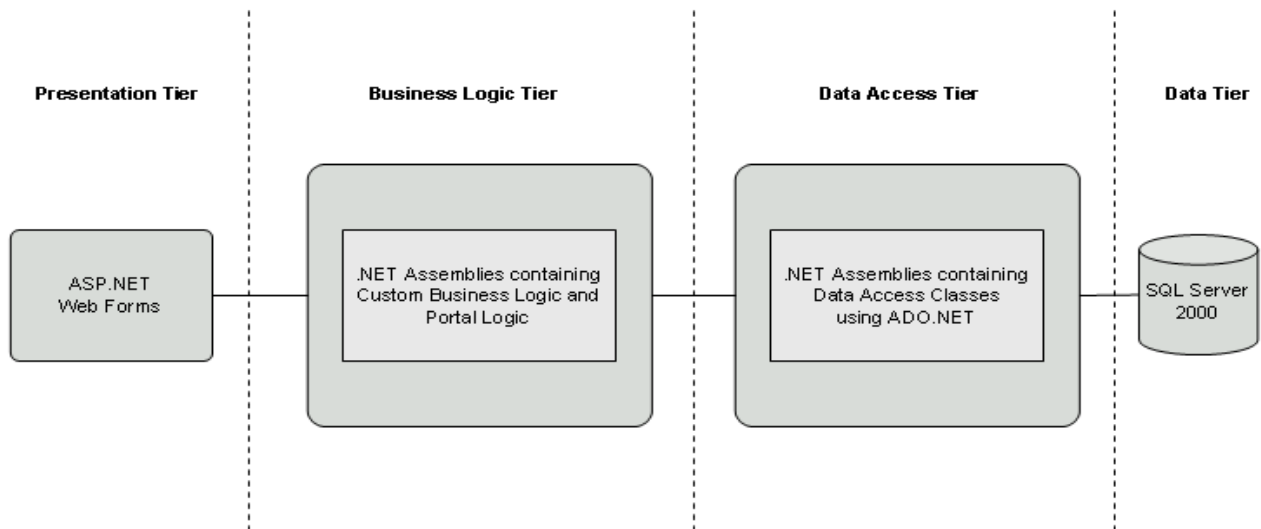
User Authentication

Users are authenticated once at the beginning of their session. Users must enter a valid sign in and corresponding password. If the portal is configured to authenticate against the portal database (default setting), the password is encrypted using a special one-way algorithm, which generates an encrypted string that only that password could have produced. This string is then matched against the one in the database and if they match, the user is granted access and a new session is started.

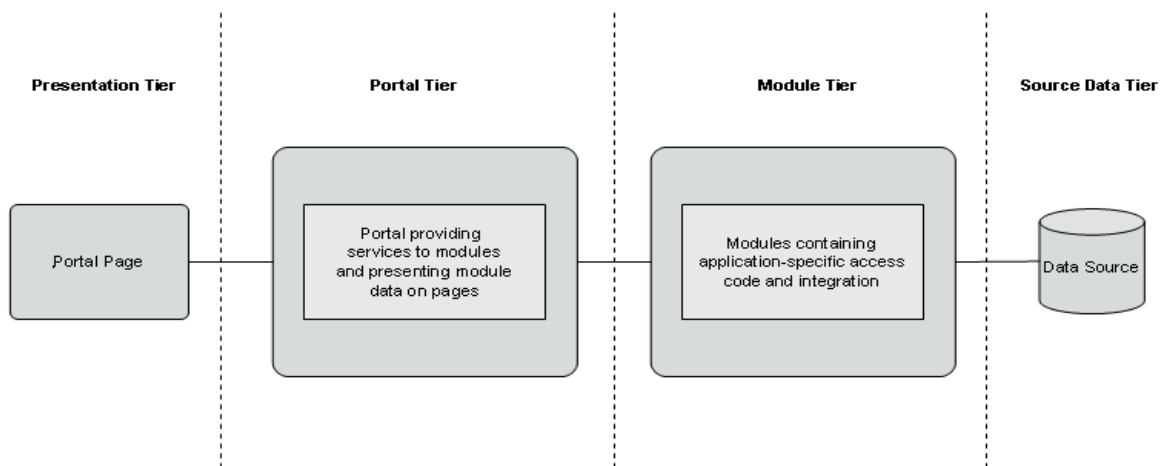
If the portal is set up to authenticate against another source such as a LDAP directory, then the web form tries to authenticates against the source and allows the user in only when the source says the user’s credentials are valid.

If not valid, a message notifying the user of the incorrect sign in information is written to the browser.

The portal framework consists of several dimensions of entities. The entire portal framework resides within the Microsoft .NET Framework. Components developed within the portal environment must conform to .NET specifications as well as portal-specific specifications. The full power of .NET may be leveraged within the portal environment at all levels.



The portal does not provide any functionality specific to applications for users natively. Instead, modules provide the functionality for interacting and presenting data to users from applications. Modules may be bundled with the framework in order to provide an out-of-the-box solution. The portal does however provide various services that support portal functionality and provide useful functions for modules to make use of. A diagram of the basic outline of where modules fit into the big picture can be found below.

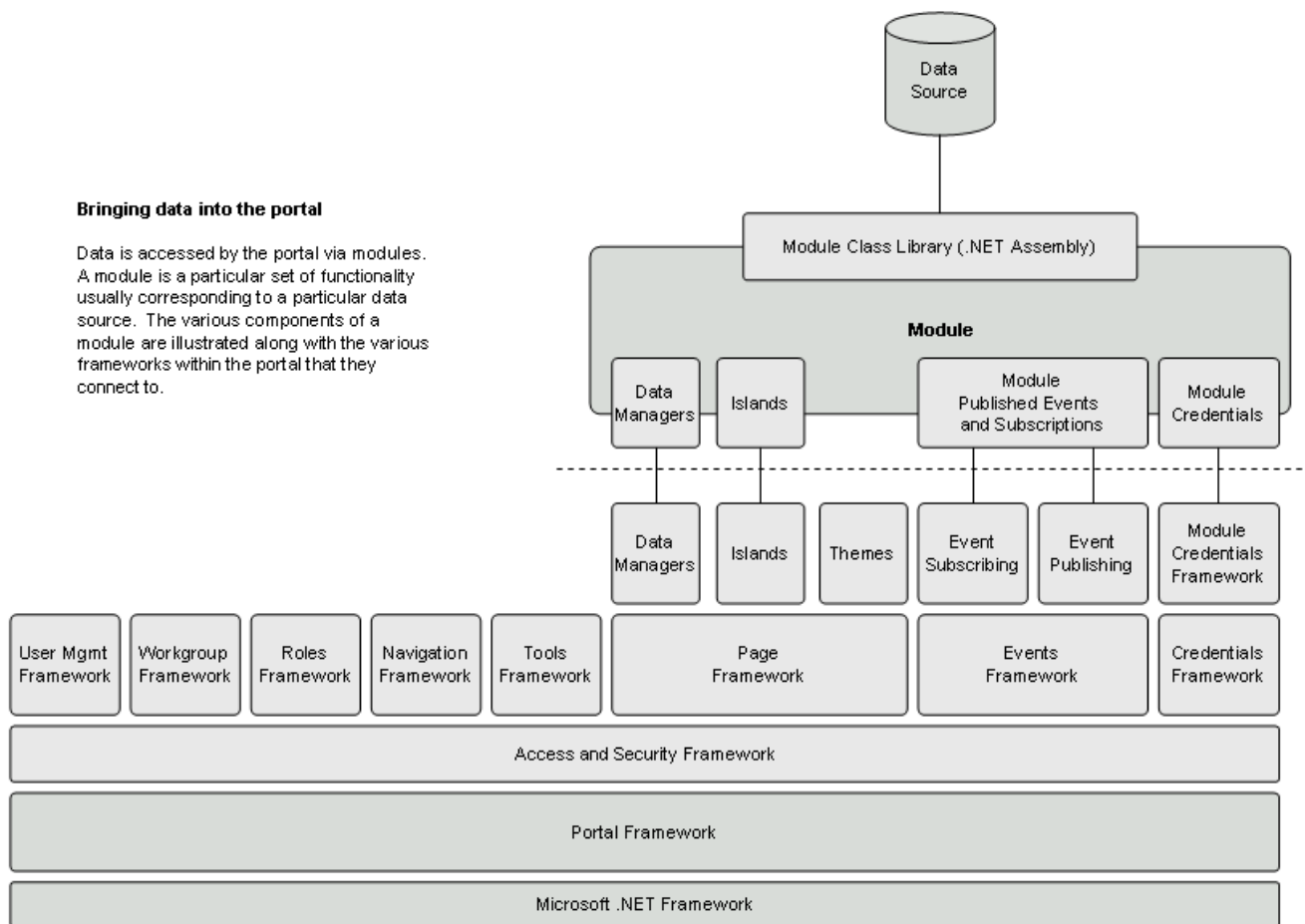


The services included in the portal tier include user management, workgroup management, role management, authentication and authorization, and so forth. The portal is responsible for managing what users gain access to, workgroup memberships, and also brokering data between modules. It also supports an event model that modules can make use of.

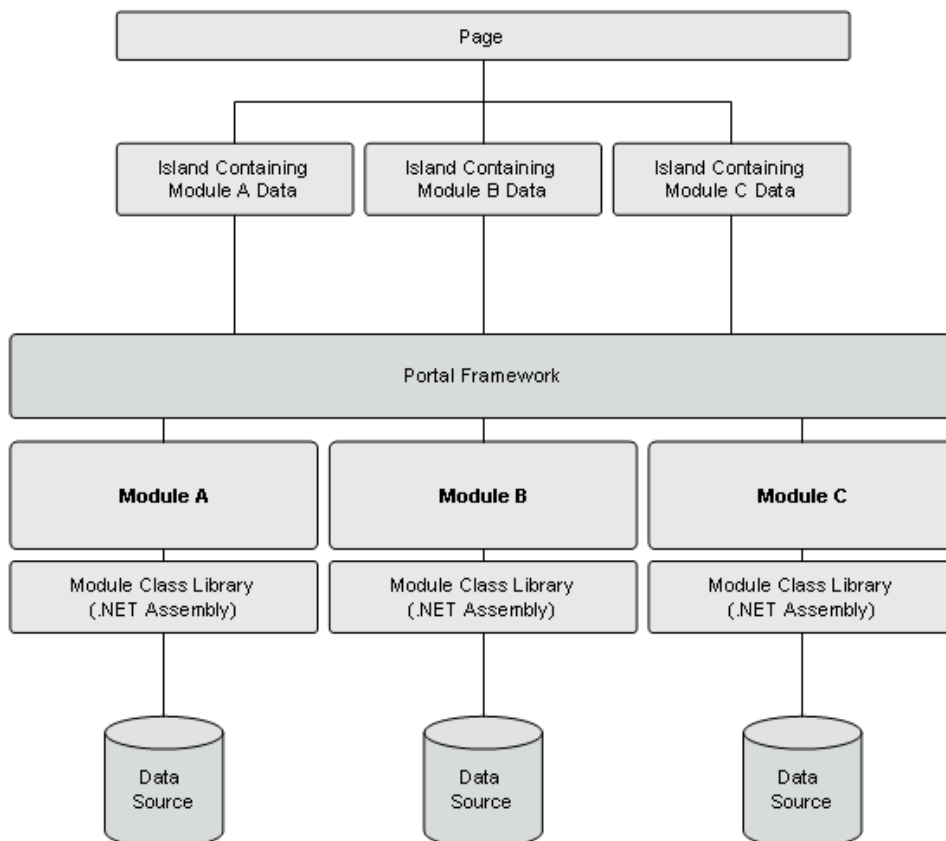
The security and Access Framework is a sub-component of the entire portal framework however its role is to enable access to applications and areas within the portal sites using the user's role membership, workgroup membership, and other factors. This security framework also enables authentication and may incorporate the use of SSL or Digital Certificates when authenticating users and other applications.

Bringing data into the portal

Data is accessed by the portal via modules. A module is a particular set of functionality usually corresponding to a particular data source. The various components of a module are illustrated along with the various frameworks within the portal that they connect to.



Below is a general architecture diagram outlining the major components of the portal framework and a module with its connections to the portal outlined. This diagram only depicts the overall architecture with a single module. Each module follows the same design however there may be more than one module in any one portal installation. There are no limits on the number of components a module may have. This means there no limits on the number of islands, events, and credentials each module may contain.



1.4 Passageways Document Indexing Service

The indexing utility synchronizes the portal document index with physical file system folders. It connects to the portal database, retrieves workgroup folders, then proceeds to synchronize each folder. This utility is a windows service application so that it is always running. It contains a configuration file, which allows administrators to configure the way the indexing engine works.

During the indexing process, subscriptions to files and folders are also processed. The frequency of indexing can be adjusted within the configuration file the document indexing service.

The indexing console requires at least version 1.1 of the Microsoft .NET Framework to be installed on the local system before using.

Framework Components

This chapter presents a technical overview of the product itself. It also goes into detail about how to create, build, and customize your portal solution.

Some areas covered in this chapter include

- Portal development
- Island development
- Module development

Each item is explained in detail and code samples provided as needed. In the last several sections of this book, there are real life example solutions, which may provide additional insight into the exact look and feel of the code used in building personalized islands and modules.

Chapter 2. Framework Components

2.1 Portal Environment

Creating a new portal application with Passageways is typically a straightforward process. The portal attempts to provide as much area for customization as possible while still maintaining a stable and efficient execution platform for page delivery. Most of the main user interface components can be adjusted to suit the firm's requirements and better fit the company's corporate style.

The images used in rendering various portal interface elements such as icons and backgrounds may be found in the portal's ImageGallery directory. The image gallery contains a variety of images used throughout the portal. Islands may choose to take advantage of the image gallery and use those images as well. If the firm wishes to change the images in any way they may do so easily by editing the image and replacing the file with an updated one. As long as the filenames remain intact, the portal should render correctly without any difficulties.

2.1.1 Portal interface

The default portal interface consists of a set of frames whose names must remain the same. Their look and feel however may be adjusted using any HTML editor or Microsoft Visual Studio.NET. Most of the business logic contained within the pages is wrapped into the main class library component and only the presentation code remains within the actual file itself. This makes it easy to edit the HTML present on the pages. The only requirement is that the controls present on each page must remain and have the same ID as is already present. You may set any control's visible property to false however it must remain on the page.

A summary of the files used to construct the frame interface is below.

File Name	Explanation
Default.aspx	This is the primary entry point to the application. Any virtual directories set up should point to this page to begin. This is the Sign In page and the first page users will see when they enter the site before signing in. This page is the one that organizations almost always choose to fully customize. It is an ASPX page with very little code. The code is required to log users in however the design and look and feel is entirely customizable.
Start.aspx	This page contains the HTML that builds the appropriate frames including the navigation and the main content area and may be changed around to fit the desired look such as adjusting the height of the top header frame, however frame names should not be altered .
InterfaceTopBar.aspx	This is the topmost page, which typically includes a large horizontal image containing the logo of the site. This page is mostly just an HTML page and is intended to provide a customizable page the organization's needs. It is referred to as the header page.
InterfaceToolBar.aspx	This page contains the search box and search options

drop down menu.

2.1.2 Portal interface features

The portal interface provides several functions to help the user navigate and explore the portal application with as much flexibility as possible. The return to home page button returns the user not to the sign in page but rather the home page set by the administrators. If no home page has been specified, the portal defaults to the headquarters page.

The online help system is a customizable help tree that replaces the navigation tree once the help icon is clicked. The user may return to the navigation tree at any time. The help system by default contains portal-specific topics however this can be altered to contain company-specific help topics as well.

2.1.3 Portal navigation

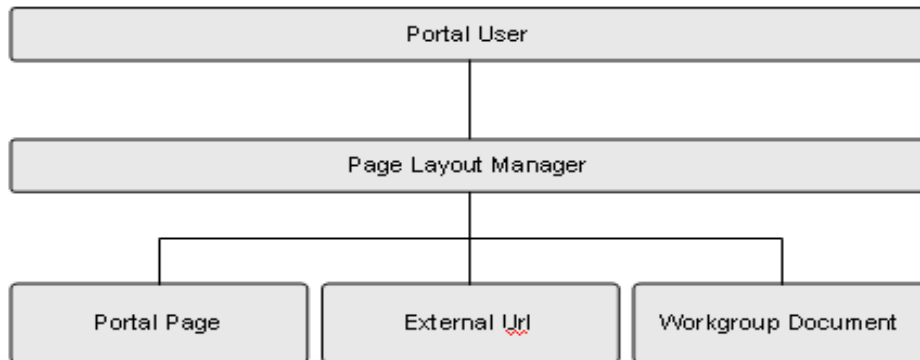
General portal navigation is accomplished via a tree similar to the one found in the Microsoft Explorer application within the Windows operating system. Due to the large amounts of information possible for browsing and organizing, a tree structure is well suited to handle large organizational demands of data and pages. The portal administrator in a variety of ways can customize the tree. The configuration manager application contains options for modifying the behavior of the tree nodes. The administrator can also modify the look and feel of the content by specifying certain properties of the PWNavigationTree control found in the NavigationTree.aspx page in the root directory of the portal.

2.1.4 Layout manager and portal engine

One of the main components of the application is the layout manager. The layout manager's responsibilities include laying islands out on a page correctly, passing appropriate context-level values to each island, and maintaining performance. It also has a secondary responsibility of directing the user to the appropriate type of page such as a web url, file, or islands page.

Portal Layout Manager

Users are directed to different areas depending on the items they click on.



a
ger is customizable to a certain extent although most of its behavior is contained securely within the portal and should not need to be modified at any time.

Islands are laid out on a page in way that island developers should be aware of. The layout manager constructs a single large table spanning the full width and height of the page. An individual island is placed in each cell.

Since islands are part of a larger page, there are some rules to which they must adhere.

Additional logic may be custom coded into the island pages if desired. If you'd like to include special graphics or execute special code within this page you may edit the `LayoutManager.aspx` file directly. While most of the logic is contained within the base class library, some events are exposed so that you may insert custom code. An example is the `Page_Load` event. If you wish, you may add code to this event, which is fired upon each request for a page.

2.1.5 Pages

Pages in the portal are not physical files as is the case in traditional web site development. A large portion of the business value is derived from the fact that business users do not have to manually construct pages but rather only specify the items they value most such as the structure and content of the page.

Pages are stored in a relational database and created dynamically at runtime. Pages themselves are objects and may represent any number of types of content. Some pages act as simply links to existing web urls, while other pages act as links to workgroup files. Still other pages, and the most common, direct the layout manager to construct an islands page according to some predetermined layout. The pages-and-content manager of the group designs this predetermined layout.

The page object simply represents a single page entry. The `PageContext` object however represents a single page within the context of a current user. This includes the user information and workgroup information for that page.

2.1.6 Themes and skins

Various aspects of the interface may be altered using the themes feature in the portal. A theme is defined in the portal as an overall appearance for a particular screen, particularly the color scheme used. Currently, only portal-wide themes are supported but workgroup-level themes should be available soon from the passageways online web site. Themes are made up of a variety of items and files all collected under one folder named by the theme's title. A theme also contains a multitude of image files for which there is a preset naming structure for.

Skins are defined as the borders around individual islands. They include the image files around island headers, icons in tools, and more within the portal. Skins are defined along with a theme. Skins are contained within each theme folder so they travel with themes.

Refer to the Branding Guide available to customers on the passageways online web site.

2.1.7 Portal Events

During the execution of certain activities within the portal, events are raised which may be utilized by custom code. The portal maintains an event directory. Items throughout the framework may subscribe to be notified when certain events occur. The event directory maintains the list of event subscriptions. When an event fires, the portal will make the appropriate call to each function for each item that has subscribed to the fired event. The most common type of item to subscribe to events are modules.

There are many reasons why modules would be interested in particular events in the portal. If a module stores workgroup-specific data or user-specific data, the module may want to be notified when a user or workgroup is deleted. This might be so that the module can run some custom code to clean up its own resources for the deleted item.

The portal is built with a static list of events, which are fired using the event directory. The list of events supported is included below.

Event Name	Description
OnWorkgroupDelete	Occurs when a workgroup is removed. The workgroup ID is passed to subscribers.
OnIslandInstanceDelete	Occurs when an island instance is removed from a page. The island instance ID is passed to subscribers.

In addition to being able to subscribe to an existing event set provided by the portal, modules might also publish events to the event directory. Other modules can then consume the published events. This can be very useful in transferring data between systems, and transferring control and alerts to other applications.

Modules subscribe to events by making entries in the deployment descriptor. Modules publish events using the descriptor as well. In order for modules to trigger custom events, a module must use the PortalEventItem WebControl provided in the WebControls library. For more information on the PortalEventItem WebControl, please refer to the WebControls Development Guide.

In order to consume portal events, modules must contain a file acting as a UserControl with an ascx extension. This control is what the event directory calls. Any code within the file is executed at the time of the indicated event.

2.2 Modules

2.2.1 Overview

Modules are self-contained collections of functionality built for easy distribution and flexible management. They exist as single files in their packed form and as multiple files/folders in their unpacked form. “Packing” a module is analogous to creating a zip file containing multiples files and folders. “Unpacking” a module is identical in nature to the unzipping process with zip files. In fact, a zip program such as WinZip should be able to “unpack” a module file just fine. Module files are the primary means by which functionality is built, moved around, distributed, and installed in Passageways. Module updates are brought in as module files. Full modules are typically installed upon purchase.

Should a module become corrupted at some point, you can always restore it to its original form from the update directory which contains backed up copies and updates that have been applied. (this applies only to module purchased from Passageways and updated using the Passageways LIVE Update system which automatically backs files up and provides update logs)

Before going into details, a module may represent a particular set of industry-specific functionality or conceptual functionality.

For example, there might be a module for interest rates. This module might include islands that allow credit unions to track and manage their rates. In addition, another module might be for Microsoft Office. This module might contain islands, which enable access to Office documents, spreadsheets, and more.

One module may contain more than one island.

2.2.2 Module Components

Each module is comprised of several major components. All individual components are optional. Every module however must have a deployment descriptor for the portal to recognize it as a working module. The descriptor contains information about the module including details on which components the module contains.

A summary of the basic components of a module is below.

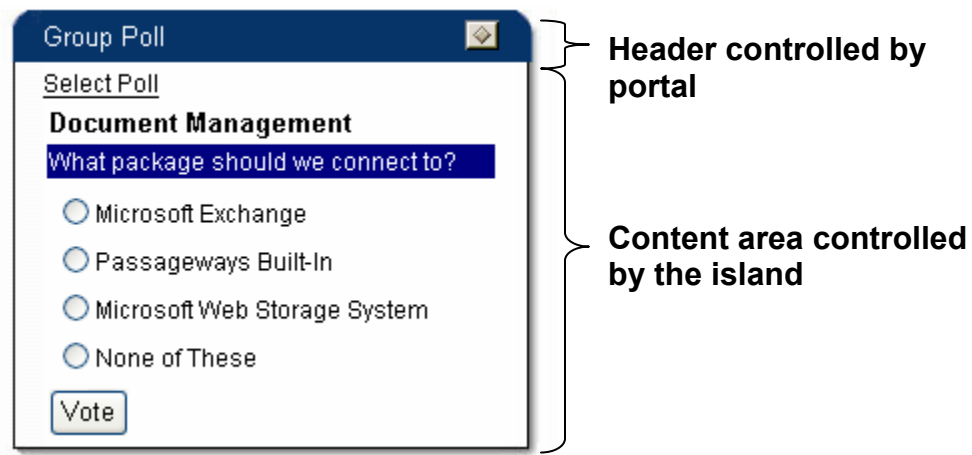
Component Name	Description
Islands	A small application presented to users on pages. Multiple islands can be shown on one page. Islands are the primary information delivery method within the portal.
Event Subscriptions	A collection of events that the module wishes to subscribe to. These events may be portal events or events from other modules. Each event entry must have a corresponding user control file that contains the code that is to be executed when the event fires.
Published Events	A collection of events that the module contains and wishes to publish to the portal. Other modules may then subscribe to these events and be notified when one is fired. The module contains the appropriate methods to fire each event.
Credentials	A collection of credential items the module or any of its components may require. This may include login information for a data source. A particular key identifies each credential. Each credential is user-specific. Only islands within the module may access the credentials for the module.
Properties	A collection of Name-Value pairs, which any component within the module may access. This may include connection strings to module-specific databases and more.
Headquarter Items	These are special text or graphical items that appear on the first page a user sees called their Headquarters Page. These

	items may be warnings, notifications that new mail has arrived, statistics on a data source such as the state of a server and more.
Search Items	These are special User Controls, which search the module's data source based on query parameters passed to it. The results are sent back to the portal for display.
Help Items	These are special tutorials provided by the module which act as the primary help system. The table of contents produced by these help items are merged into the main portal help system at runtime.

2.2.3 Bundled modules and samples

If you are familiar with the .NET architecture and framework, then you may also be familiar with the idea of User Controls in ASP.NET. User controls are small self-contained components, which perform specific actions and present a user interface. They are similar to dynamic include files. However, they can include buttons, forms, links, drop down menus, and so forth and take action according to what events are thrown. A user control for the web, ASP.NET, is a text file with the extension .ascx.

An island is made up of two parts, the island content and its header bar controlled by the portal. The following illustration is a typical island.



2.3 Passageways Assemblies

Most of the core business logic is contained within the Passageways Base Class Library. Most classes are contained within the assembly named PassagewaysBaseClassLibrary.dll and are stored in the applications bin directory.

This library provides many different functions and capabilities. Some classes allow work against the database such as adding and removing workgroups. Others represent various structures such as roles, users, and log files.

There are several major assemblies that make up the portal functionality. Each assembly encapsulates common functionality and usually contains a single namespace however some may contain multiple namespaces.

Some of the assemblies that are bundled with the portal include

- Portal Base Class Library – Main portal components
- Compression Library – Classes for compressing directories
- Search Library – Search functionality
- Code Behind Pages Library – Event code for most portal tools
- Logs Library – contains the business logic necessary to write to the portal event logs
- Data Access Library – Contains classes useful in accessing SQL Server databases easily
- Settings Library – Manages all communication with portal framework configuration files and settings.
- Web Controls Library – Useful controls used throughout the portal

The library is broken down into namespaces very similar to packages in Java. These namespaces are summarized below.

Namespace	Class Members (does not include all classes)	Description of Namespace
Passageways	PWAssistantManager PWHyperLink PWIsland PWIslandContext PWIslandInstance PWIslandProperty PWModule PWPage PWPageContext PWPageTemplate PWPageTemplateIsland PWRole PWRoleElement PWSchedule PWSearch PWSession PWView PWWorkGroup PWWorkGroupDirectory PWWorkGroupType PWWorkGroupTypeProperty RoleCollection ViewPagesCollection WorkgroupCollection	Contains the core classes including user and workgroup classes used to directly operate on the database Certain modules are bundled with every installation and so the classes that perform the work behind these modules are tightly integrated into the library.
Passageways.CodeBehindPages		Contains most of the event logic behind the administration tools and sign in pages
Passageways.Compression	PWDeflator PWDeflatorConstants PWDeflatorEngine	Contains general compression classes

	PWDeflatorHuffman PWDeflatorPending PWInflator PWInflatorDynHeader PWInflatorHuffman PWPendingBuffer PWZipFile	
Passageways.Compression.Zip	PWZipConstants PWZipEntry PWZipInputStream PWZipOutputstream	Contains special classes for creating zip files and extracting zip files to physical disks
Passageways.Streams	PWDeflatorOutputstream PWInflatorInputStream PWOutputWindow PWStreamManipulator	Contains manipulation classes for handling streams
Passageways.Checksums	PWAdler32 PWCrc32 PWChecksum	Contains special classes with checksums used mostly in compression and zip files
Passageways.WebControls	Member HelpTree NavigationTree HeadquartersControl IslandContainer TabbedGrid User Theme PortalEvent	Contains controls used within passageways such as the navigational tree and help menus

2.4 Workgroups

Workgroups are a special grouping entity within the portal. Most systems include the ability to group users. Workgroups are strongly typed entities meaning every group created must be of a particular type. Typical types of workgroups include departments, divisions, branch offices, and more. Types of workgroups are created and set in the Passageways.config file in the root of the application.

Once types have been defined, workgroups can be created. Workgroups, when created, have several resources available to them automatically. The portal enables some organizational features such as a group calendar, announcements, membership managers, polling, pages and content (islands), and more depending upon which modules have been installed. This list of resources is extensible and can be customized.

For instance you could have a group resource named “interest rates” which stores and displays interest rates. You might have one or more islands to display this information but only want certain people to change and adjust the rates themselves.

In this case, an action link for the interest rates island is created in an aspx file and placed in the module’s folder. The workgroup manager then needs to use the assistant manager tool located in “Settings->Workgroups” to delegate management access to the interest rates action item. He may do this per user so ‘Jane’ could be given access to the interest rates.

For more information on how best to organize workgroups, please refer to the Portal Organization Best Practices document included in the implementation packet.

2.5 Roles

Installations typically represent some type of organization, profit or nonprofit. Each organization typically has roles assigned to its employees. These may be officially declared somewhere or, for a smaller business, simply known to everyone by job function. Either way, it is useful to assign roles to each user in the portal as well. Roles can be configured to match your organizations. There is only one role that is required and it is titled “Administrators”. The roles tool should not allow you to remove this role. This role is set-aside for portal administrators. The portal may not function correctly if you remove this role. You may add additional roles to the default ones provided. For instance at a credit union, you might create role titled “Tellers”. Every user may become a member of one or many roles. The administrator only has access to the role membership tools. However, as mentioned in other chapters of this book, this can be changed. Since the role membership tools are Toolbox tools, you may add additional roles to each tool to provide access. For instance, if you wish the “HR Manager” role to also have the ability to adjust user roles, you would indicate that in the Toolbox file located in the root directory of the application.

Roles may have additional custom fields that belong to each user. For example, the “Tellers” role may have a field named “Teller ID” which is a company-specific ID assigned only to tellers. Every user can change his or her Teller ID if they belong to the Tellers role. Directions on how to do this are explained in more detail in the following sections.

2.6 User profiles

Passageways keeps a set of users within the application. These users can be grouped into workgroups to take advantage of collaboration tools and sharing functions. Every user has a unique sign in name and a password they designate. Passwords and usernames are case-sensitive if stored in the portal database. Each role created also has an associated membership list. Workgroup managers may control the group’s membership but not role membership. Only administrators by default can adjust role membership. This can be adjusted however by providing access to the role membership tools in the Toolbox to roles other than the administrators.

Every user, in addition to a sign in name and password, also contains a personal profile. This contains personal items such as the user’s birthday as well as work-related items such as a Teller ID. The basic profile cannot be adjusted however additional custom fields may be added to make the profile company-specific. Each role’s custom fields also become dynamically part of each user’s profile. The user accesses his or her profile by clicking on “Settings” located in the toolbar area typically. If the user’s session has expired, the user may not adjust his or her profile. This is primarily for security reasons and for accidentally leaving a browser window open when the user walks away for a long period of time.

2.7 Document Search

The portal is bundled with a document indexing service that enables large quantities of documents to be profiled and searched. Documents and files are indexed into the portal database, which can then be searched using the included searching library.

Some of the fundamentals behind the portal search service are listed below

- Each workgroup may have a folder(s) where its documents reside.
- Only users who have signed in and are a member of a particular workgroup can search and access the files belonging to that group.
- When a user searches for documents, the result set only displays files for the workgroups that the user belongs to.
- While the document index can be thought of as being company-wide, the indexes are workgroup-based which allows some flexibility in controlling access to documents.

These business rules govern all of the searching capabilities found within the portal. Certain features may have additional rules applied to them. In order to make an organization's documents searchable and easy to retrieve, several things need to be done.

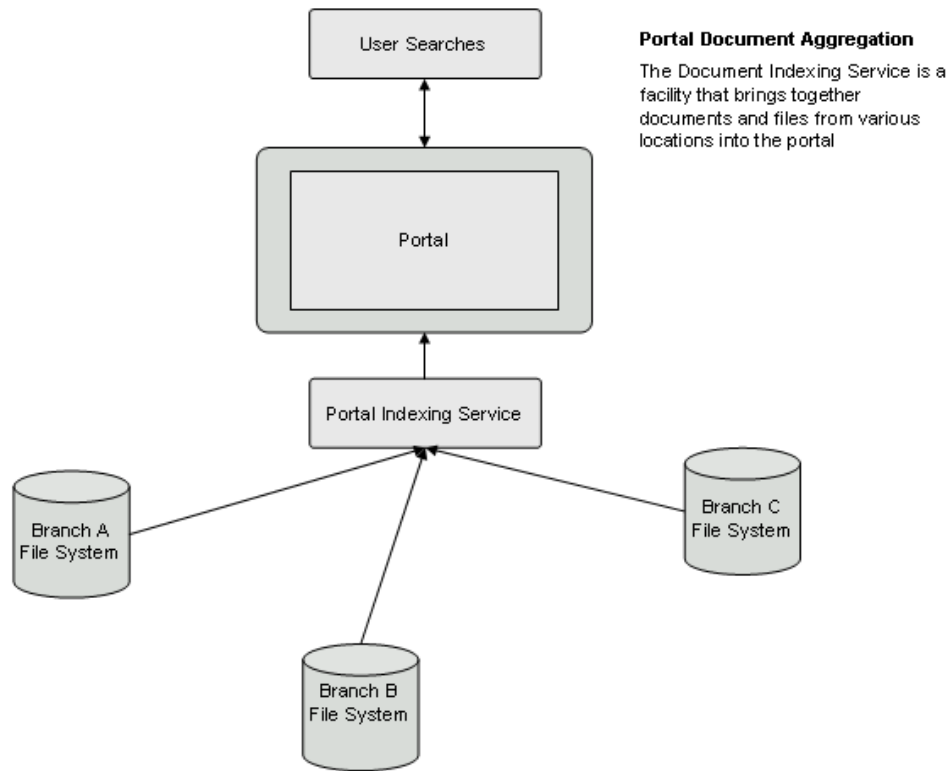
One of the most important concepts in understanding the portal's search capabilities is to understand indexes and how the portal treats them. Indexes are analogous to your telephone book directory. A telephone book is a listing of all the residents with phone number within a given area. The listing is usually order by last name. A telephone contains the three most basic elements of an index. They are summarized below

Basic elements of an index

- Listing of summary data
- Ordered by some field
- Data contained within a particular scope

The indexes built by the portal for documents follow this same design and contain all three of these elements.

There is one large primary document index that serves all workgroups in the portal. The index can handle thousands or even millions of documents and resides in the form of a relational database table. This index is contained within the portal database.



realize that the actual file system where documents reside is never searched explicitly. Any search mechanism within the portal searches against the index, not the actual file system.

Managing and controlling what folders are to be indexed is done at the workgroup level. By default, only portal administrators should designate physical file system folders to be indexed. This is to prevent business users from mapping other folders to which they do not have access. The indexing process, that is the process of gathering files and entering them into the master index, is done via a set of tools which can be scheduled to run or executed via the portal administration tools.

The basic steps in indexing a document repository are outlined below.

Steps to Index Your Documents

Use the included Document Indexing Service application to scan in the documents. You must designate physical folders for the workgroups before using this utility. The indexing service will retrieve the list of folders from the portal database to be scanned and then execute a synchronization procedure on each document.

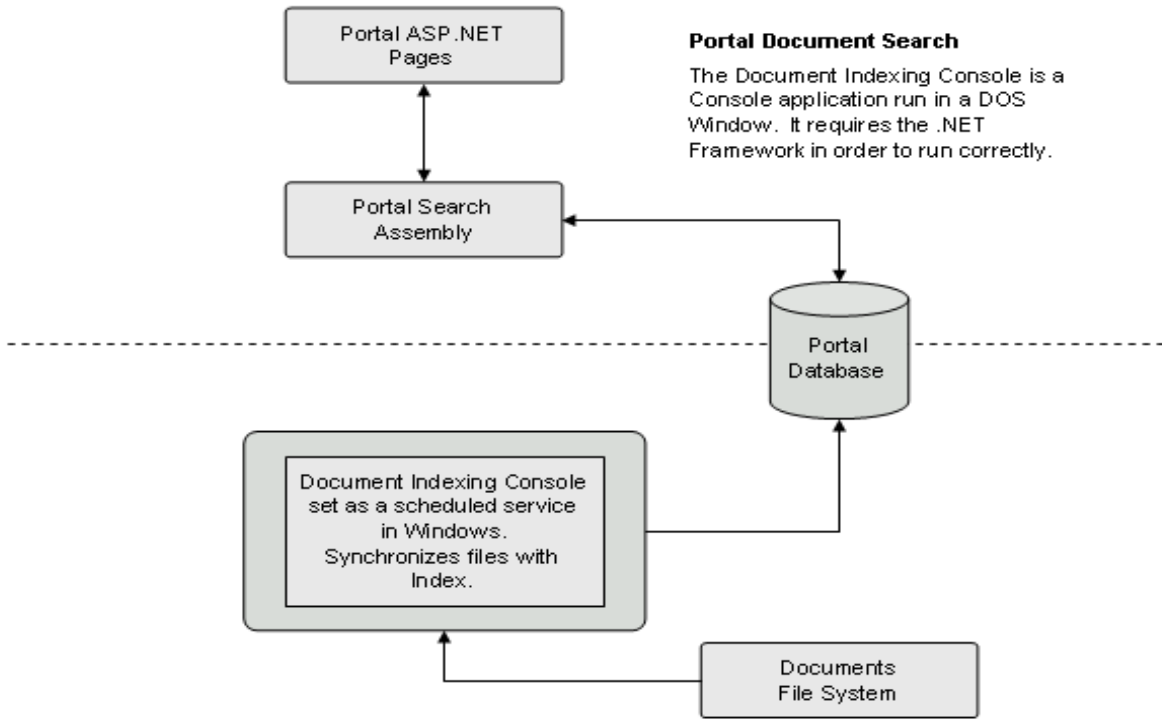
Important Note: You must set the document indexing windows service to log on as a network user who has read access at least to all workgroup folders that are mapped, typically this may be a domain admin account.

New documents found in the workgroup's directory will be scanned into the index. Fields such as the document's summary and author may be added as well depending on what the application finds within the file.

Documents in the index not found on the physical drive are assumed to be deleted and are marked as such in the index. They are not removed from the index. There is a cleanup utility found in the document indexing web tools that allows a workgroup manager to clean up deleted documents from the index.

Once documents have been scanned in, they are immediately available for use in search queries. The search libraries will access only the portal database index and never the physical files. Result sets may include links to the web-accessible location for a particular document.

It is important to note that a web server or a virtual directory does not need to be set up on the file servers where files reside. There is never any link to a particular document present in the portal that everyone may use. There may be links to documents however those links lead to a special portal page, which actually checks the user's permissions (portal permissions only, not their network file permissions at this time) and session state. The page then scans in the document from the remote location and sends the file to the user's browser. This method is done to prevent users from getting a hold of urls leading to documents they do not have access to.



Scheduling the indexing process

The indexing process can be scheduled to occur at preset intervals or at particular times. Since the indexing utility is a typical windows service, it can be started on a web server with minimal effort.

The easiest way to schedule the indexing process is to edit the configuration file of the document indexing service. The service can typically be found in the Program Files\Passageways\Services folder on the web server. There should be a .config file with the documents service .exe file. Open the config file and edit the connection string to the portal database as well as the frequency of synchronization. It is typical to synchronize about every 30-45 minutes. The value entered into the config file is in milliseconds.

Document Profiles

Each document stored in the index contains a profile describing the document. The full text of the document may be stored in the index and is used for full-text searches however this is not recommended due to the size of the resulting databases that might be created. The document profile captures information such as the following:

- Document Summary
- Author
- Indexed Date and Time
- File Extension
- Last Modified Date Time
- File Size
- Keywords
- Full Text (in some cases only)
- Number of Times Accessed

Document profiles may be searched using the bundled search engine. Workgroup managers may adjust profiles. When documents are scanned in, some profile elements are populated to assist in populating profiles for large quantities of documents.

When a document is removed from the physical disk, it is not removed from the index, even after the next synchronization, it is only marked as deleted. However, once a document is removed from the index, its profile is removed as well.

Document Recommendations

A common type of search query is one in which the user is not sure of exactly what to query for, yet knows what he or she is looking for. In order to assist users in finding more relevant documents, workgroup managers may associate keywords with particular documents. For instance, the keyword 'loan' might be associated with a document titled "Loan Application Form". Whenever a portal user searches for 'loan', or any variation of 'loan' such as 'loa' or 'loans', they are presented with not only the typical result set of documents but also a recommendations list at the very top. This list would contain the document titled 'Loan Application Form'.

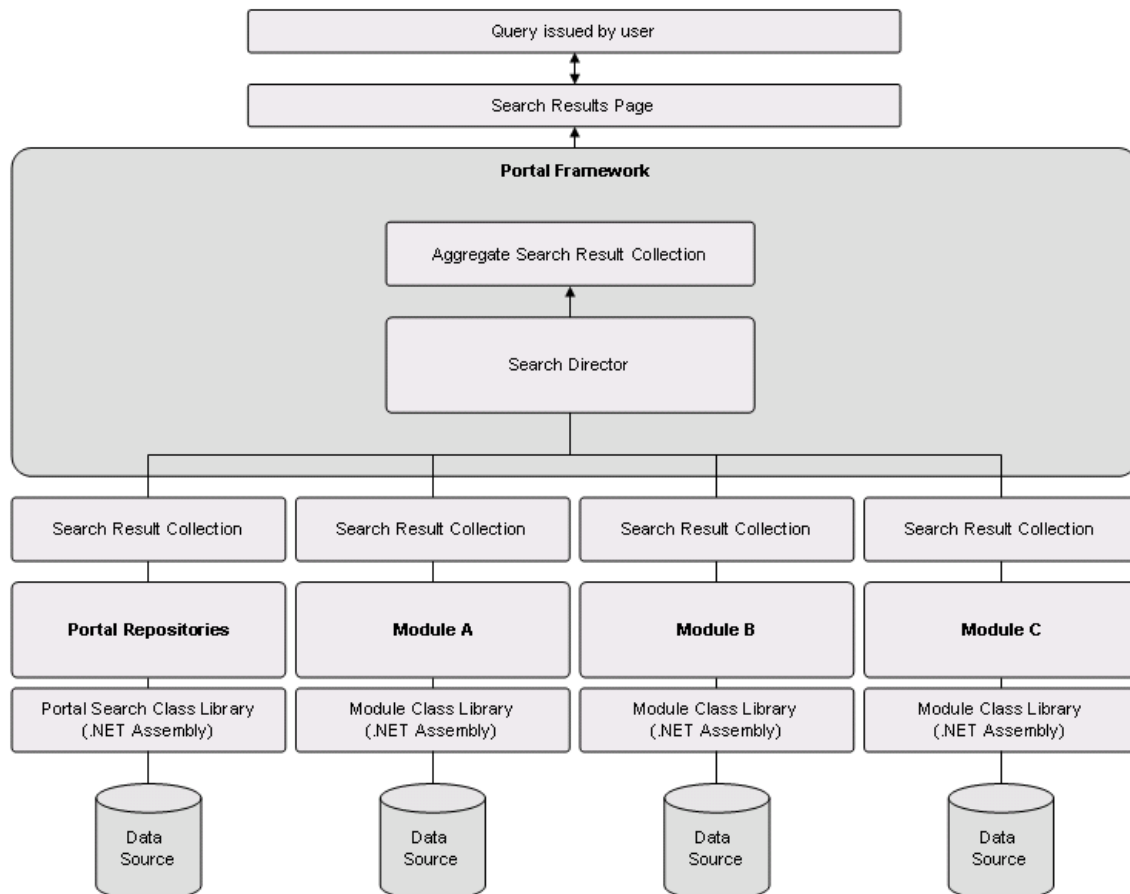
While this seems somewhat basic at first, it can fast become a powerful tool to push documents to users searching for particular items. Oftentimes, a user might search for 'loan' and receive upwards of 800 different documents. The loan application might be number 659 and never be reached. However, if it is associated with the keyword 'loan', then it will be located at the very top for the user to get to easily.

2.8 Federated Search

The portal contains a native framework for searching much more than the document repository alone. The portal may also search any content stored within the data sources used by modules, the user directory, external Internet locations, and more.

Some of the fundamentals behind the portal federated search service are listed below

- Each module may include a search capability enabling it to become 'searchable'
- Results are sent back to the portal in a unified way alleviating the module developer of having to format results
- Search code is custom built for each module to increase integrity and accuracy of the search
- Each module may contain custom return pages which clicks on a search result are directed to
- Search results are grouped by each resource for the end user



Each module may declare more than one Search Item and name each item as it wishes. Each Search Item represents a single resource that is searched and grouped. For example, the group collaboration module that comes with the portal integrates several resources with the portal including group announcements, discussion forums, and calendar events. Each of these resources is marked as a Search Item within this module.

The results of a search on this module will be grouped by each resource such as “Announcement Results”, “Discussion Results” and so on. This is not the required method of structuring the search items, but it is recommended. The speed of each search depends entirely upon the module’s search capabilities. The portal may set time limits or execute each search within its own thread however the actual searching of the data source is done by the module so careful planning must take place whenever introducing a search item into the portal environment.

2.9 Alerts and Notifications

The portal contains a native framework for sending alerts and notifications to users. The delivery method of these alerts is entirely up to the individual user. Several types of delivery mechanisms are provided to users.

2.9.1 How to send an alert

Sending an alert from your code is very simple. It involves making use of the `PWAlertItem` class within the framework. This class provides a few simple methods to send alerts to various users.

2.9.2 Sending alert techniques

Sending an alert from your code requires some discipline. For instance sending alerts to all users every 5 minutes throughout the day may be cumbersome and counter-productive for users. They then may in fact turn their alerts off entirely. To ensure this does not occur, make sure that whatever alert you do send out is meaningful and clear on what the alert message is trying to tell them. For instance an alert saying “Your expense report for June 29th has been approved” is much more useful than one that says “Someone has just sent you an email” which might be more vague and occur much more frequently.

To send an alert to a single user (in this case the current user)

```
string UserID = PWSession.GetUserID();  
PWAlertItem.Send(UserID, "Alert Subject", "Alert Message", PWAlertType.Normal);
```

Part 2

Developing Modules

Module Concepts

The Passageways application product family covers a wide set of products. We can find a wide variety of needs fulfilled within the Passageways family products.

This chapter introduces the products that we used in this book as well as products from the Passageways application family that can be implemented.

For more information about the Passageways product family, please refer to the Passageways Web site at <http://www.passageways.net>

Chapter 3. Module Concepts

3.1 Overview

Building a custom module is very easy and is highly encouraged. You may find modules available on the passageways.net web site or create your own.

3.2 Module deployment

Modules are typically distributed as simple zipped up files. They can be expanded into the Modules folder of a portal installation easily. Module updates from Passageways are downloaded in a special compressed format that can be uncompressed using the WinZip program if need be however the Live Update service will unpack and install any module update automatically.

3.3 Technical Form

A module is present on the portal server as a folder with files within it. The files inside it contain functionality which communicate with a data source such as an email server or a database. A module may also have class libraries in the form of DLLs which are present in the portal's bin directory. Each module contains or more islands, which are applications that users may place on pages.

An island is a .NET User Control. This is similar in nature to an include file in traditional HTML. However, different from an include file is the fact that islands are treated as objects at runtime and so take on a much more robust form. Each island inherits from the portal class PWIslandContext. This class in turn inherits from the User Control Class in .NET

Each island represents only a small portion of the entire web page that gets built at runtime. It is important to recognize this fact as it does place certain restrictions on the markup that islands generate.

Islands should not contain any html form elements. The page that includes the island will already have form elements. The encoding type will be such that file uploads are available for islands if need be.

Also, images and fixed width tables should be avoided as they may increase the width of the html table and not give the user the expected result. For instance, very wide images may stretch the entire page and affect content in other islands by accident.

3.4 Module Configuration File

Each module has a configuration file in XML format that describes the functionality and behavior of the components contained within the module. The descriptor tells the portal how to treat components and may contain settings that configure a module for a particular installation including connection strings and more.

A module descriptor is a single configuration file placed in the main module directory. It describes the general purpose of the module file along with some special properties. The name of the file is module.config and it may not be accessed directly from a web browser for security reasons.

The descriptor is the way that a module communicates its functionality to the portal.

Each island contained within the module must also have an entry within this file. Properties that your islands may want to access can also be included as Module Properties to hold connection strings and so forth.

There are a host of configuration tags that may be included in this file to indicate how the module should behave and define settings.

3.5 Installing modules

1. Contact your Passageways Client Account Manager who can arrange for the purchase and installation of a new module. If you are developing your own module or purchased a module from a third party, proceed to step 2.
2. Copy the module folder to the Modules folder within the portal application. Also copy any dlls into the portal bin directory if need be. There may be other items that require installation such as databases or windows services which need to be set up using the provided instructions by the module provider.
3. Sign into your portal installation as a Portal Administrator.
4. Navigate to the Manage Portal Framework tool located in the "Tools" area.
5. Within the tool, click "Refresh configuration" to refresh the cache and have the portal detect the new module.
6. Click the "Manage Modules" tool and go to "Module Properties" to adjust the properties of the module, which may include database connection strings and more.
7. Click the "Manage Modules" tool and go to "Manage Island Permissions". Set the permissions on the appropriate islands to allow them to be used within the portal application.
8. Your new module should be available for immediate use.

NOTE: If you are not able to immediately access the islands, try using the cache utilities available in the Tools area or restarting the web server IIS to clear the cache entirely. Islands should appear after either of these actions is performed.

Module API – Basic elements

This chapter presents the module and its various building blocks. This chapter walks through the creation of an island and provides samples of code for building islands and declaring them properly with the module so that the portal will recognize and treat the island as intended.

Chapter 4. Module API – Basic elements

4.1 Island

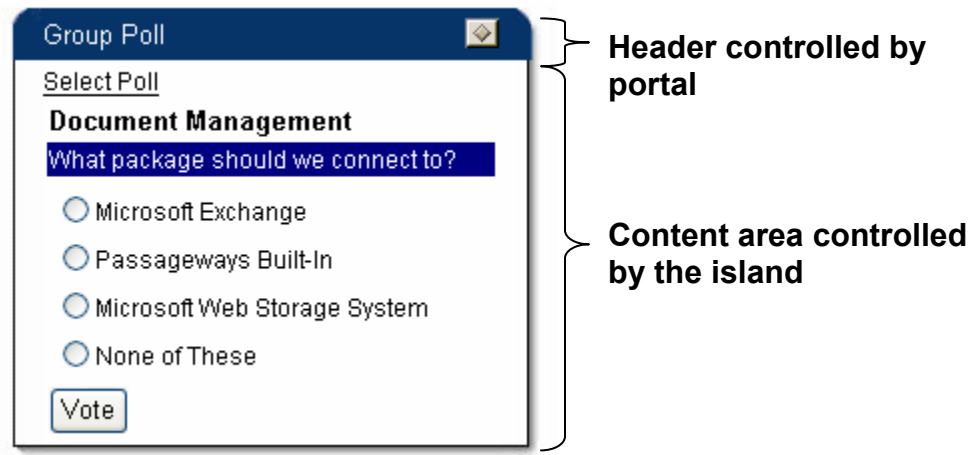
The Passageways portal allows companies to build their own custom portal web site to serve the needs of employees, business partners, and customers. Users can sign on to the portal and receive personalized pages providing access to the information, people, and applications they need. This personalized single point of access to all necessary resources reduces information overload, accelerates productivity, and increases web site usage.

Every island must have at least one .ascx file that serves as the entry point to the island. From here, the island may open up new pages within the island directory if desired.

This file is seen as an object to the portal, specifically one of type PWIslandContext, which inherits from the UserControl type.

To write an island a simple text editor is all that is required however a more suitable editor such as Microsoft Visual Studio.NET is recommended.

An island is made up of two parts, the island content and its header bar controlled by the portal. The following illustration is a typical island.



4.2 Development requirements

In order to fully develop your own Islands, there is a small list of things to make sure and do.

Make sure the Island you create is a Microsoft .NET User Control, usually indicated by an ascx file extension for the entry file (Pop up windows within the island may have any extension you wish). This is required because the Pages place your Islands by dynamically adding controls. This means of doing things is much more efficient and programmable than simple screen scraping.

Due to the nature of the page that displays the islands, there are several restrictions on island HTML markup. Firstly, the entire page of islands is contained within one form so islands are not required and should not contain <form> tags. Also, islands should not contain <HTML> or <BODY> tags either. It is important to remember that an entire island will be placed within a table cell. Knowing this, leaving an open <TD> tag may result in a disrupted page. A good idea is to contain all the islands content in one table defined fully in the island file.

Below are some of the more technical requirements as far as the code is concerned. Islands are .NET user controls so they can be written in any .NET-compliant computer language.

File Requirements	
Folder Name	Should be something unique like [CompanyName][IslandName]
File Name	The entry page should be the same name as the folder to ease installation. [CompanyName][IslandName] is usually fairly safe. The filename must be unique so no two islands can have the same filename for the entry page. Other files within the islands folder can be named as you wish.
File Extension	.ascx (.NET User Control Extension)
DLLs	DLLs are allowed for an island. These DLLs are to be included in the module folder and copied to the portal's bin directory upon installation. Upon installation, the framework will not do the appropriate copying to the correct bin folder. There is no need to create bin folder within the module folder.
SQL Scripts	None. The portal will not install island or module-specific databases. These resources must be installed separately by the system administrators or by a module setup project.
Comments	All relevant files and scripts are contained within a single Island folder.
Code Requirements	
Required public property	[none]
Required Inheritance	Must inherit from the PWIslandContext class
Required public methods	The portal framework requires no methods.
Comments	The inherited class is required. If the island does not inherit from the PWIslandContext in the exact case and name as above, the framework will display a friendly error message that the island could not load.

4.3 Option pages (Actions)

Every island is built into two distinct parts when displayed on a page. The main area is the actual island content which is the entire lower portion. The topmost portion is a special header bar that is constructed. This bar includes the title of the island along with a special icon placed at the upper right hand corner. The icon may be clicked and a page will appear with special options for that island. Several of these options are identical for all islands however each island may include its own set of options as well. A user would be presented with those options after clicking on the icon, and then clicking on the appropriate option link. This provides a handy way of giving users access to pages other than the primary island page. This, in effect, makes an island a small application, which may contain more than just one file. Option pages may be secured so to only allow access if the user is the manager of the current page or group that is being viewed.

Option pages are declared in the islands descriptor within the main module descriptor.

4.4 Using island persistence capabilities with PrefString

Each island, besides customizing itself at runtime, may also choose to persist values to a permanent storage facility. This can be done in two ways. Firstly, the island may already be configured to store values in an external database or system such as Microsoft Exchange. In this case, the island developer will specify how this is done. If the island would like to use the portal's persistent storage facility, the island developer uses the PWIslandContext and PWIslandInstance objects available to every island. The island context object contains methods for retrieving and storing the preference string. This string is an allocated storage area for a single string value. Several methods have been developed for making use of this area.

Single variable approach

If all that is needed is to store the value of one variable, for example a zip code, then the single variable approach is recommended. This involves simply setting the pref string to the variable's value and retrieving the value at runtime.

Multivariable approach

In some cases, the island may need to store more than one value such as a zip code and a city name as well as the display color preferred. This can be done in a multitude of ways however xml is usually an easy and convenient method. Xml involves placing xml tags into one string including the variable names and values as key-value pairs. Then by parsing the xml at runtime, each variable value may be retrieved.

Example

```
<myIsland>
  <variable name="myVariable" value="1"/>
  <variable name="myVariable2" value="2"/>
</myIsland>
```

The entire xml string can be stored with the portal and retrieved upon each usage. If the preference string is missing, the island knows it is being run for the first time and should prompt the user for the variable values.

Another approach to storing several variable values in the pref string is to use a simple comma-separated string in which your code knows the location of each variable value within the string.

4.5 Images and hyperlinks in islands

Islands may contain images and hyperlinks within themselves. There are special governing the use of these two items because of the context in which islands are displayed.

Hyperlinks that lead to external URLs can be coded the same way as any page. The user will not leave the portal but will see the web site inside the content frame. If the island for some reason needs to force the user to exit the portal and see the web site in full, the link should include the standard target="_top" tag. This is strongly discouraged however as most users will not appreciate leaving their portal site. Hyperlinks may also open up new browser windows with pages contained in the island folder. Building the path from a series of items creates the path to the page. Firstly, the application root must be found. The PWSettings class has the portal settings in it, one of which includes the application root URL. The module name also must be attached to the URL and finally the island folder and filename. The string building technique for a typical page looks like the following:

```
[C#]
String Url = PWSettings.ApplicationRoot + "Modules/" + this.ModuleFolderName +
"/Islands/myFile.aspx";
```

4.6 Using island context objects

Each Island must inherit from the PWIslandContext class. This class in turn inherits from the UserControl class and also includes virtually all the information the island might need in order to run. Using this object, an island can get the current user's profile elements, workgroup lists, and more. The Object passed is of type PWIslandContext.

This object is created at runtime and represents the current user who is signed in along with the context of the page on which the island is being hosted. The page may belong to a workgroup or possibly an individual. Also, the page might be currently seen publicly and not a specific user at all.

Island Context				
Page	Workgroup	User		Island Instance
Layout	Resources	Profile	Roles	Island

In addition to several context objects, the above information is also available to each island. The combination of a user, workgroup, page, and island instance makes a specific island context.

The page properties may be accessed directly.

For example in C#, the ViewID is retrieved with "this.ViewID;"

The island instance information is accessed via the Instance property. This includes the persistent data store, preference string, and island information such as the island name and settings from the island deployment descriptor.

[C#] Example

How to inherit the context object

```
<%@ Control Inherits="Passageways.PWIslandContext" %>
```

Workgroups the Current user belongs to

```
WorkgroupCollection groups = PWWorkGroup.GetForUser(PWSession.GetUserID());
```

Looping through Workgroups

```
foreach (PWWorkGroup group in PWWorkGroup.GetByTypeID("department"))  
    {string groupName = group.Name;}
```

Get the collection of properties for this island as defined in the configuration file

```
IslandPropertiesCollection props = this.Instance.Properties;
```

Get the value for a property named MyProperty as defined in the configuration file

```
string myProp = this.Properties.Get("MyProperty").Value;
```

NOTE: If the property is not found due to an error in loading the island config file or a mistyped property name, a null value will not be returned or else a `NullReferenceException` will be thrown using the above statement. Instead, an object of type `PWIslandProperty` will still be returned so the `Value` property will exist. However, if the property is not found the value for the "Name" and "Value" properties of the returned object will be "NOTFOUND" in all capital letters. If you must check for the existence of the property, check this value to ensure the property is being read correctly.

Get the PWIslandInstance object for the current island

```
PWIslandInstance inst = this.Instance;
```

Get the Workgroup ID of the group on which the island is currently placed

```
String groupId = this.ObjectID;
```

Basic Island Example

```
<%@ Import Namespace="Passageways"%>
```

```

<%@ Import Namespace= "System.Net"%>
<%@ Control Inherits="Passageways.PWIslandContext" %>

<SCRIPT Language= "C#">
void Page_Load(Object src, EventArgs E)
{
    myIsland.Text = "Hello World";
}
</SCRIPT>

<asp:Label ID="myIsland" runat= "server"/>

```

Members available to islands at runtime as part of the PWIslandContext class

PWIslandContext: Inherits from the UserControl Class

Public Instance Constructors

PWIslandContext Constructor	Initializes a new instance of the PWIslandContext class.
-----------------------------	--

Public Instance Properties

Instance	Returns the PWIslandInstance object for the current island. PWIslandInstance represents special page-specific information such as the position of the island and ref string manipulation.
IsActivated	Indicates whether island is safe to be shown on a public page. This value is set in islands descriptor.
IsWorkgroupEnabled	Tells the portal whether the island will run on a group page or not.
IsPersonalEnabled	Tells the portal whether to execute the island on personal user's pages. Some islands may depend on the page being a group page.
ShowSkin	Tells the portal whether or not to display the borders and header around the island.
UserID	The current UserID
IsManager	bool value indicating whether the current user is the view manager or not
PageColumn	The column number the island

	resides in
ColumnOrder	The vertical order on the page
ViewID	The current ViewID
PageID	The current PageID
User	Returns the PWUser object for the current user
Properties	Returns a populated IslandPropertiesCollection containing a set of PWIslandProperty objects for the current island.

4.7 Size and layout rules for islands

In order to provide structured flexibility for page creation and building, islands are not guaranteed to be shown in any particular size on a page. This is mainly due to the fact that a user may construct a single or multi-columned page and the user may have a large or small monitor.

The height of Islands depends on their width:

Examples of typical island layouts to illustrate the meaning of half and full widths are below.

4.8 Module Configuration properties

Whenever you install a module, you may allow the local administrator to set certain properties that apply to the module. These properties will be set for all instances on the installation meaning the setting applies every time any island within the module is placed on a page. The properties are name-value pairs and can be used for anything the module developer wishes. For instance, if one property sets the color of a table to blue, then every time an island using the property is used on a page, that table color is blue.

Setting properties

Properties and their values are stored in the module descriptor file explained in the following section. The Module.config file located in the root directory of the module acts as the islands descriptor. Each module may have as many properties as it requires.

Example code from Module.config

This example pertains to an island with an ID of "myIsland". It has three properties referred to by their Names of "FontSize", "BackColor", and "ConnectionString". The island itself will look for these properties at runtime and attempt to use their values. This example illustrates using styling properties however properties can be utilized as the island wishes so they may hold other values other than colors and fonts. The third

property contains the connection string information used to connect to an external database.

```
<Module>
  <ModuleProperty Key="FontSize" Value="10pt"/>
  <ModuleProperty Key="BackColor" Value="Black"/>
  <ModuleProperty Key="ConnectionString" Value="server=IP;uid=sa;"/>
</Module>
```

Accessing module properties

Properties are made available to each island in the module via the PWModule object. In order to access the properties, the island uses the "GetModuleProperty" method of the PWModule object. Properties are returned in the form of a string. The GetModuleProperty method requires both the ID of the module you want to look at and the property key or name that you would like.

```
//This example will attempt to get the connection string stored within the module's
//config file
string conn = PWModule.GetModuleProperty("MyModuleID", "ConnectionString");
```

NOTE: Whenever using properties, it is always a good idea to check for its existence within the island and hard-code a default inside the island itself. The administrator may incorrectly set the property or not set it at all and the island should catch this.

4.9 Using themes and style elements

When an island is presented on a page, it can define its styling elements itself or choose to take advantage of the provided portal's current theme. Islands are strongly recommended to follow the theme structure as it allows the island to better conform to the user's styling preferences. A single CSS stylesheet is placed on the page where an island is present by default. The sheet varies based upon the theme selected by the portal administrators or workgroup managers. In order to make use of the theme elements, islands mark their text and link items with certain CSS classes, which correspond to the items found within the CSS stylesheet that will be placed on a page. All portal theme stylesheets use the same classes so the actual theme sheet that will be used is not relevant to an island developer.

Elements prefixed with 'portal' are meant for the portal itself however islands may take advantage of any of the styling elements available on the page.

Css Class Name (common styles only, there are more styles available)	Description
<code>.portalbodytext</code>	Normal body text of an island. Usually small and normal in weight
<code>.portallink</code>	Represents a hypertext link with underlining style properties set

4.10 Using user information

The Passageways portal allows companies to build their own custom portal web site to serve the needs of employees, business partners, and customers. Users can sign on to the portal and receive personalized pages providing access to the information, people, and applications they need. This personalized single point of access to all necessary resources reduces information overload, accelerates productivity, and increases web site usage.

Retrieving User's Profile Elements

First and Last Name

```
PWUser user = UserDirectory.LoadUser(PWSession.GetUserID());  
string FullName = user.FirstName + " " + user.LastName;
```

Roles the Current user belongs to

```
RoleCollection roles = PWRole.GetForUser(PWSession.GetUserID());
```

Find out if Current user is the View Manager for the current View

```
bool IsManager = this.IsManager;
```

Find out if Current user's session is active and alive

```
bool IsSessionActive = this.IsSessionActive;
```

Retrieve the unique UserID assigned to this user by the framework (GUID)

```
string UserID = PWSession.GetUserID();
```

4.11 Using workgroup information

The Passageways portal allows companies to build their own custom portal web site to serve the needs of employees, business partners, and customers. Users can sign on to the portal and receive personalized pages providing access to the information, people, and applications they need. This personalized single point of access to all necessary resources reduces information overload, accelerates productivity, and increases web site usage.

4.12 Storing credential information in the credential vault

The Passageways portal allows companies to build their own custom portal web site to serve the needs of employees, business partners, and customers. Users can sign on to the portal and receive personalized pages providing access to the information, people, and applications they need. This personalized single point of access to all necessary resources reduces information overload, accelerates productivity, and increases web site usage.

Headquarter Items

This chapter details the process and concepts behind creating headquarters items. Headquarter items are items that appear within the Headquarters WebControl within the portal framework.

Chapter 5. Headquarter Items

5.1 What are headquarter items?

Headquarters is a special control within the portal. It contains a summary or bird's-eye view of the user's portal experience at a glance. This area contains a box for each workgroup the user belongs to as well as a personal area. Each box, personal or workgroup, contains two sets of items, observations and actions.

Observations are things the portal or module observes about the user. This is equivalent to a notification. This can be something as simple as "you have 5 emails waiting".

An action is a link that the user may click taking the user somewhere. This could also be a form submission or drop down box. An example might be "click here to check my mail" or "Send Alerts to Users"

5.2 Building a headquarter item

A headquarter item is relatively straightforward to build. Items are created in the form of a .NET User Control just as islands are. They inherit from the HeadquartersItem class which in turn inherits from the .NET UserControl class.

It is strongly recommended that headquarter items render only small amounts of text only. Although they have the flexibility to insert almost any type of content into the user's headquarters, it is important to recognize the user may belong to a number of workgroups and other modules may include items as well. Inserting more than a short one-line string may alter the appearance of the user's headquarters in ways undesirable to the end user.

A headquarter item is an ascx file created in the folder of the module. This is outside of the Islands and Managers folders. The headquarter item must be declared in the module descriptor in order for the headquarters to recognize it. A typical headquarter entry item may look like the example below.

```
<HeadquartersItem  
Type="personal-observation"  
UserControl="HQAnnouncements.ascx">  
</HeadquartersItem>
```

The UserControl attribute declares the filename to be loaded. The type attribute is a special key telling the headquarters control where to place the return value of the item. The following naming convention is to be followed in order to correctly place your item in the control.

scope - type	description
Personal-observation	An observation pertaining directly to the user and not part of any particular workgroup.
Personal-action	An action the user may want to take.
Workgroup-observation	An observation about a particular workgroup the user belongs to.

Workgroup-action

An action the user may want to take for a particular workgroup.

There are two properties provided to all headquarter items that assist in rendering their content. These are UserID and WorkgroupID. The UserID is the id of the user who is viewing the headquarters. The workgroup id contains the workgroup box in which the item is being executed. These properties do not have to be declared in the item file because the item file must inherit from the HeadquartersItem class located in the Passageways namespace.

Search Items

This chapter details the process and concepts behind creating headquarters items. Headquarter items are items that appear within the Headquarters WebControl within the portal framework.

Chapter 6. Search Items

6.1 What are search items?

Each module may make itself 'searchable' to the portal so that it's content may be searched and viewed by a user. In order to make a module's content searchable, the module should contain two things, a declaration of a search item, a simple xml tag found within the module's configuration file, as well as a special file in the form of a .NET User Control which executes the actual search.

6.2 Building a search item

A search item is relatively straightforward to build. Items are created in the form of a .NET User Control just as islands are. They inherit from the SearchItem class which in turn inherits from the .NET UserControl class.

It is strongly recommended that search items be planned carefully as a slow search item may slow an aggregate search for the end user.

A search item is an ascx file created in the folder of the module. This is outside of the Islands and Managers folders. The search item must be declared in the module descriptor in order for the headquarters to recognize it. A typical search entry item may look like the example below.

```
<SearchItem
Title="Announcement Results"
UserControl="SearchAnnouncements.ascx">
</SearchItem>
```

The UserControl attribute declares the filename to be loaded. The title attribute is a special title telling the portal what to call the group of results returned.

There are properties provided to all headquarter items that assist in rendering their content. The main property is the UserID. The UserID is the id of the user who is executing the search. These properties do not have to be declared in the item file because the item file must inherit from the SearchItem class.

Each Search Item file must include a special method titled "Search" which accepts a single string argument. The portal will call this method and pass in the query entered by the user. Inside this method is where the code must search the module's repository. The return value of this method is a SearchResultCollection object containing the search results.

Example signature of the required method:

```
Public SearchResultCollection Search(string query);
```

If the module requires additional information such as what workgroups the user belongs to, it can always ask the portal for those items using the regular portal API objects.

Portal Event Items

This chapter details the process and concepts behind creating headquarters items. Headquarter items are items that appear within the Headquarters WebControl within the portal framework.

Chapter 7. Portal Event Items

7.1 What are portal event items?

When special events occur within the portal such as a new user has been added, a workgroup has been removed and other such events, the portal can let modules know about them. This allows the modules to execute code based on when an event occurs. So if a module maintains expense reports for a set of users, whenever a user is removed from the portal, the module may archive all the expense reports submitted by the user instead of keeping them stored in the regular expense reporting database. This type of requirement is commonly used to clean up data sources or to take special actions when events occur.

7.2 Building an event item

More information on developing and subscribing to events coming in the next release of this guide.

7.3 Subscribing to an event item

More information on developing and subscribing to events coming in the next release of this guide.

Action Links

This chapter introduces data managers. It also goes into detail about building a data manager and what its role in the portal framework is.

Chapter 8. Action Links

8.1 What is an Action Link?

Action links are links contained within the Actions link on any island. Actions links can be controlled by workgroup managers and may contain any aspx page or other platform page such as a jsp page. An action link can also be directed to another web application entirely.

8.2 Building an Action Link

Actions links should be ASP.NET Web Applications however they may also be legacy asp applications or even web applications running on different platforms as long as they can be run within an IIS virtual directory. DLLs required by the action link can be included with the module and will be copied to the portal's bin directory at runtime in order to enable the classes for the data manager to become available for them.

8.3 Action Link Scope

Each action link must declare a specific scope within which it would like to be accessed and run within the portal. Some modules contain functionality specific to workgroups. The scope for this type of action link would be organizationalUnit.

In other cases, a module or individual action link may contain functionality for a set of data important across the organization. This might include interest rates, company information, and more. In this case, it does not make sense for each workgroup manager to be given access to such actions. In this case the scope of the manager should be declared as organizational.

NOTE: If a scope equal to organizational is used, only managers of the workgroups of type organization will be allowed access to these action links.

The scope of each action link is indicated in the Scope attribute of the OptionPage tag within the module.config file for the island in which the action link resides.

Deployment descriptors

Deployment descriptors represent an important part of the portal framework. Descriptors are the bridge between the portal and the module functionality.

Chapter 9. Deployment descriptors

9.1 Portal configuration

The portal application directory structure follows most of the same guidelines as does any ASP.NET application. Passageways expands on that capability and incorporates many new uses for its directory structure. The following diagram illustrates a typical portal installation directory structure. The only folder required by the .NET framework is the bin directory which holds the DLLs necessary to run the application. This includes the entire base class library and any other controls the portal may want to make use of.

9.2 Module deployment descriptor

In order for the portal to correctly recognize a set of functionality as a complete module, a descriptor must be created. Deployment descriptors describe the functionality contained within a module as well as settings that tell the portal how the module should be treated. There are four basic components to a typical module. They are summarized below.

Basic components of a module descriptor

- Module information
- Headquarter items
- Event subscriptions
- Published events
- Credentials
- Islands

Module information is the information specific to each module. This includes general definitions, descriptions, versioning information, and more.

Headquarter items are items that the module would like to appear in the headquarters control for all users. This is a very useful way of pushing content directly to a user. This can include alerts and notifications as well as simple observations. It can also include links or actions the module wishes to make available to users.

Event subscriptions are items the module wishes to be notified about from the portal. Each subscription carries with it a file name that contains the code the modules wishes to execute when the event is fired. Events may be portal events or events coming from other modules.

Modules may publish their own events to the portal event directory. The events the module would like to publish are contained in the module descriptor as published event items.

If any of the islands or data managers would like to store credentials for users such as usernames and passwords, the credentials must be declared in the module's descriptor. Credentials are module-wide so are shared amongst all items within the module.

Islands are the primary delivery mechanism of information to user pages. Islands are self-contained applications that present a UI to users. They are built similar to asp.net pages and can contain virtually any type of web content including html, dhtml, javascript, video, and more.

Data managers are special managers provided to workgroup administrators that enable users to manage a data resource. These are full asp.net applications which run from within the module and have access to portal resources. Access to these managers is handled by the portal.

Below is an example of what a module descriptor might look like.

```
<?xml version="1.0" standalone="yes"?>
<Module>
  <Name>Passageways Sample Module</Name>
  <Description>Sample Module Descriptor</Description>
  <Version>1.0.0.0</Version>
  <Activated>>true</Activated>
  <CLSID>Special Updates ID, don't change manually</CLSID>
  <ID>Passageways.Modules.Sample</ID>
  <Comments></Comments>
  <Credential>
    <Key />
    <DefaultServer />
    <DefaultUsername />
    <DefaultPassword />
    <DefaultToken />
    <DefaultUserID />
  </Credential>
  <ModuleProperty Name="ConnString" Value="myConnString" />
  <PortalEvent
    Event="PortalEvent.OnWorkgroupDelete"
    Assembly="C:\\Inetpub\\wwwroot\\Portal\\bin\\Passageways
    .Modules.MyModule.dll"
    OnEventFire="OnWorkgroupDelete"
  />
  <HeadquartersItem
    Type="personal-observation"
    UserControl="HQItem.ascx"
  />
  <Island>
    <ID>PWIsland</ID>
    <Name>PWIsland</Name>
    <Description>Sample Island</Description>
    <FolderName>PWIsland</FolderName>
    <EntryFileName>PWIsland.ascx</EntryFileName>
    <Activated>>true</Activated>
    <WorkgroupEnabled>true</WorkgroupEnabled>
    <PersonalEnabled>true</PersonalEnabled>
    <ShowSkin>true</ShowSkin>
    <OptionPage>
      <Text>Edit Sample</Text>
      <FileName>editSample.aspx</FileName>
      <Scope>organizationalUnit</Scope>
    </OptionPage>
    <Assemblies>
```

```
    <Assembly FileName="Passageways.Modules.Sample.dll" />
  </Assemblies>
  <Comments></Comments>
  <Property>
    <Text>Sample</Text>
    <Value>>true</Value>
    <PossibleValues>>true,false</PossibleValues>
  </Property>
</Island>
</Module>
```

Island development issues

In this chapter, we address development issues and facts we have come across with customers of the portal.

Chapter 10. Island development issues

10.1 Testing islands

Testing islands is typically a relatively simple process. In order to test an island, first ensure it is part of a module and the module descriptor contains an entry for that island. Install the module on a portal installation and give yourself only permissions to the island. You may then place the island on pages only you may see and test functionality there. You may also log files in a custom log file if you wish to isolate the island's messages.

10.2 Island creation guidelines

Creating an island requires an understanding of the limitations and requirements as well as the benefits and features of writing .NET User Controls.

10.3 Namespaces

The portal is contained within a given set of namespaces. It is recommended that in order to avoid class name conflicts at runtime with the portal or any other modules running, that a namespace contains the module's name and island name be declared for any classes within a module class library.

An example of would be `Passageways.Modules.MyModule.MyIsland`

10.4 Flash and ActiveX controls within islands

Flash animations are perfectly acceptable within an island. Since islands output html to the browser, the typical markup for a flash animation is the exact same type of markup within an island. The only concerns to watch out for are the relative sizes of the animations. Since browsers expand html tables as wide as an animation goes, it is wise to keep animations small and within the size limits of the desired island size, full or half.

ActiveX controls follow the same rules as do flash animations. They are acceptable and work well although the size of the control should be taken into consideration.

For more information regarding development of Passageways components or for answers to any questions you might have contact your Client Account Manager or submit help requests through the support center at <http://support.passageways.net>.