

# **Academus<sup>®</sup> Style Guide**

**Theme & Skin**

**Version 2.1**

***CONFIDENTIAL***

# Notices

This Unicon® Academus® Theme and Skin Guide contains important system and operational information related to Academus.

Use of Academus in conjunction with other products and services, other than those expressly authorized by Unicon, Inc. is the user's sole responsibility.

Due to continuing product development, the specifications and capabilities described in this document are subject to change without notice.

## **DISCLAIMER**

Unicon, Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing or use of this document.

## **CONFIDENTIAL**

All information contained herein is confidential and proprietary to Unicon, Inc. and constitutes trade secrets of Unicon, Inc. This document shall not be disclosed, duplicated or used, in whole or in part, for any purpose other than its authorized use with the Academus product. The information contained herein shall not be communicated to a third party without prior written consent from Unicon. Any unauthorized use is strictly prohibited.

## **TRADEMARKS AND SERVICE MARKS**

Unicon® is a registered trademark of Unicon, Inc.

All trademarks, service marks and trade names referenced herein are the property of their respective owners. Any use, copying or reproduction of the Unicon, Inc. service marks without prior written permission of Unicon is strictly prohibited.

## **COPYRIGHT**

© 2007 Unicon, Inc., Chandler, Arizona. All Rights Reserved.

## **LICENSE AGREEMENTS**

The Academus product includes software products that are licensed by manufacturers other than Unicon. Your use of the Academus service indicates your acceptance and agreement to comply with the terms of such license agreements provided with the Academus product.

# Table of Contents

Scope .....	1
Summary .....	1
Release 2.1 User Interface .....	2
Manual Organization .....	3
Audience .....	3
Revision History .....	3
References .....	3
Contacting Unicon .....	4
Support Infrastructure .....	4
Web Access .....	4
Phone Access .....	4
Remote Assistance .....	4
<b>Chapter 1. Academus Rendering Process .....</b>	<b>5</b>
1. Structural Transform .....	5
2. Theme Transform .....	6
3. Visual Presentation (Skin) Application .....	6
<b>Chapter 2. Web Standards Considerations .....</b>	<b>7</b>
What Are Web Standards? .....	7
Why Use Web Standards? .....	7
Accessibility, ADA, and Section 508 Compliance .....	8
Accessibility .....	8
ADA and Section 508 Compliance .....	8
Academus 2.1 Web Standards .....	9
<b>Chapter 3. Directory Structure and File Location .....</b>	<b>11</b>
Theme Files .....	11
Skin Registry .....	12
Skin Files .....	12
Skin Subdirectory Definitions .....	12
<b>Chapter 4. Understanding and Customizing the Theme .....</b>	<b>13</b>
Theme Overview .....	13
Theme Variable .....	13
Theme Variable Listing .....	14
Modifying Theme Variables .....	16
Procedure .....	16
Modifying the Variables As Desired .....	17
Syntax .....	17
Warnings .....	18
Impact of Theme Variables on the Theme and Skin .....	19
Variables that Affect the Page Header and Brand .....	19
Variables that Affect Permissions/Access .....	21
Variables that Affect the Layout .....	21
Variables that Affect the Page Content .....	22
Variables that Affect Channel/Portlet Help Links .....	27
Variables that Affect the Footer .....	28

<b>Chapter 5. Understanding, Creating, and Modifying Skins</b> .....	<b>31</b>
Skin Overview .....	31
1. Cascading Style Sheets .....	32
2. Images .....	32
3. Media Objects .....	33
Skin Use in Academus .....	33
Creating A New Skin .....	33
1. Copy the Entire Skin Package Directory .....	34
2. Rename the Skin Package Directory .....	35
3. Add the New Skin to the skinList Registry .....	35
skinList XML Sample .....	35
Understanding the skinList XML .....	35
Editing the skinList XML .....	36
4. Make Modifications to Skin Files .....	36
5. Test New Skin in a Test Environment .....	36
6. Move New Skin Package to Public Environment .....	37
7. Restart the Portal Server .....	37
8. Verify New Skin Within the Portal .....	37
Modifying An Existing Skin .....	38
1. Create a Backup of the Skin Package .....	38
2. Make Modifications to Skin Files .....	38
Modifying Cascading Style Sheets (CSS) .....	38
Modifying Graphic Images .....	39
Modifying Media Objects .....	39
3. Test the Skin in a Test Environment .....	39
4. Move Skin Package to Public Environment .....	40
5. Restart the Portal Server .....	40
6. Verify Skin Changes Within the Portal .....	40
Upgrading Skins from Academus 2.0 to Academus 2.1 .....	41
CSS Files .....	41
Upgrading the Skin .....	41
.....	41
<b>Chapter 6. Examples</b> .....	<b>43</b>
<b>Chapter 7. Terms and Definitions</b> .....	<b>49</b>

---

## About this Manual

This document provides Unicon Academus© users with descriptions of pertinent theme and skin aspects of Academus 2.1. It also equips users with the necessary knowledge to create, customize or modify themes/skins in the Academus 2.1 release.

---

## Scope

The scope of this document is limited to theme and skin usage in Academus 2.1. Unless noted otherwise, all references to “Academus” in this document refer to the 2.1 release. This document will not address skin usage in open-source uPortal. This document assumes the reader has at least a general understanding of HTML, XHTML, CSS, XML, and XSL Web technologies. An advanced understanding of both HTML and CSS technologies is recommended for undertaking the creation or modification of an Academus 2.1 skin.

---

## Summary

This document describes the Academus theme and skin and how it may be customized. The final theme and skin output, including processing from the backend code and database, produces the portal’s user interface and experience.

A description of the Academus theme/skin context is provided to give an understanding of the environment in which theme/skin customizations take place.

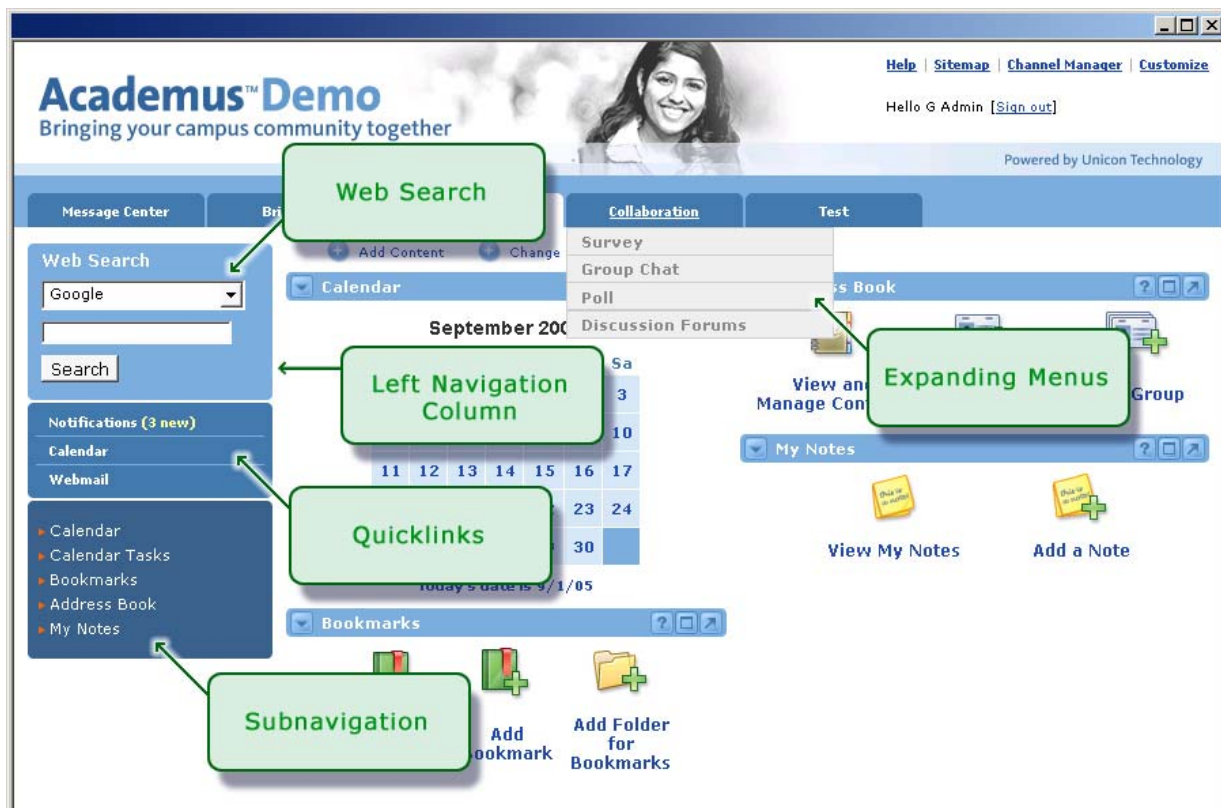
- The Academus rendering process
- Web Standards related to Academus
- Theme overview
- Skin overview
- Related terms and definitions

Detailed instructions are provided for customizing the theme/skin.

- Customizing the theme (configurable user interface components)
- Creating a new skin
- Upgrading an existing skin
- Modifying an existing skin
- Examples of customization

## Release 2.1 User Interface

The Academus 2.1 release has several configurable user interface components:



- **Left navigation column:** Contains other navigational components (Web Search, Quicklinks, main navigation, and subnavigation).
- **Web Search:** Links to third-party public search engines.
- **Callout Area:** Displays links to channels/portlets prioritized by the system administrator; one-click access to popular channels/portlets (“Quicklinks”). Clicking one of the Quicklinks will take the user to a focused view of that channel/portlet.
- **Expanding Menus:** Drop-down and fly-out menus on each of the main navigation tabs. The expanded menu displays a list of links of the channels/portlets on that tab. Clicking one of the menu links takes the user to a focused view of that channel/portlet.
- **Subnavigation Area:** Comprised of a list of links of the channels/portlets on the current tab. Clicking one of these links will take the user to a focused view of that channel/portlet.
- **Spell-checking Utility:** Provided for user-entered text. When spell checking is utilized, spell check links are rendered next to any text area input in the portal interface. Clicking a spell check link launches the spell-checking interface and performs a spell check on the user-entered text of the corresponding text area.

---

## Manual Organization

This Manual is organized as follows:

**Chapter 1: *Academus Rendering Process***, describes the three-step process to generate the final presentation.

**Chapter 2: *Web Standards***, provides a definition of the web standards and the importance of using web standards..

**Chapter 3: *Directory Structure and File Location***, describes the directory structure and file locations pertaining to the theme and skin.

**Chapter 4: *Customizing the Theme***, provides an overview of the Academus theme and provides instruction on how to customize the theme.

**Chapter 5: *Understanding, Creating, and Modifying Skins***, discusses how to create a new skin, as well as modify and existing skin

**Chapter 6: *Examples*** provides instructions that answer questions regarding common theme/skin changes in Academus.

**Chapter 7: *Terms and Definitions***

---

## Audience

This Manual is intended for use by Academus System Administrator's and Unicon, Inc. Cooperative Support.

---

## Revision History

The following table provides the revision history of this User Guide.

Revision	Release	Date	Description
A	2.1	May 2007	Initial Release

---

## References

The following table contains terms which are referenced in subsequent text. The abbreviated form of each term is listed in the table.

Term	Subsequent References
Unicon® Academus 2.1	<ul style="list-style-type: none"><li>• Academus</li><li>• The product</li><li>• The system</li></ul>

Term	Subsequent References
Unicon, Inc.	Unicon

---

## Contacting Unicon

The Academus Support team is located in Chandler, Arizona at Unicon's corporate headquarters facility. The goal of this organization is to deliver world-class support while offering customers the choice and flexibility in the methods of engagement using self-service resources, online support systems, and direct interaction with support engineers.

[www.unicon.net](http://www.unicon.net)

## Support Infrastructure

### Web Access

Support contacts have exclusive access to the Unicon Support web site where they can open, review, and modify their confidential support cases. The site also provides other valuable resources such as frequently asked questions, knowledge base articles, and technical bulletins.

### Phone Access

While the primary method for opening a support case is via the web site, contacts can also open new cases or check on existing cases by talking to a Customer Service agent by phone.

Toll Free (US only)	+1-866-386-4266
International	+1-801-715-5507

A support engineer will follow up on the case and get in touch with the customer contact via phone or email.

### Remote Assistance

The support team does not typically require remote access to the systems at the customer site. However, in some cases the support team may need access to the environment to further investigate a particular problem or error. This will happen at the discretion of the individual customers and their preferences related to remote system access. The support team will work directly with the support contact to facilitate this process.

Customers are advised to proactively arrange for remote access for the Unicon support team. This assists in the prevention of any unnecessary delays during a serious production problem.

# Chapter 1. Academus Rendering Process

*Skin Relevance: Abstract, Context*

The Academus rendering process requires three-steps to generate the final presentation:

1. Structural Transform
2. Theme Transform
3. Visual Presentation (Skin) Application

---

## 1. Structural Transform

**Definition:** *Information layer: Transformation of Academus data structures into categorical/metaphorical presentation structures.*

Dynamic data is gathered into an initial XML document composed in a “folder” structure representing generalized containers. Each of the <folder /> and <channel /> containers have multiple attributes that are used to determine what type of container it is. This XML document is transformed against a structural XSLT (traditionally labeled as “tab-column”) that changes the XML structure into nodes that represent a tab-column structure. Much of the previous <folder /> and <channel /> attribute information is carried into the <column /> and <channel /> nodes.

The structural transform is tied to a layout manager within the Academus framework. Until the release of uPortal 2.3, uPortal utilized a singular layout manager referred to as “simple layout manager”. With the release of uPortal 2.3.x, the structural transform can also accommodate aggregated layouts, where aggregated layout-specific structures are included. A portal installation must be registered to one layout manager or the other, either simple or aggregated.

Additional data may be put into this transform. The primary example of this is the inclusion of preferences and channel management interaction into the “integrated modes” theme.

*Note: The version of uPortal that Academus 2.1 is built upon also contains the layout manager known as Distributed Layout Management (DLM). Although, Academus 2.1 contains the DLM code inherited from uPortal, DLM is not supported within the Academus support contract. If you are interested in pursuing DLM for your portal, Contact Unicon Professional Services.*

## 2. Theme Transform

**Definition:** *Interaction layer: Transformation of data in categorical presentation structures into specific media interface.*

Once the XML document has been transformed into a presentation structure (tab-column) by the Structural transform, it is transformed a second time into a specific media type (for example, HTML) using a separate XSLT. This second XSL transform file is traditionally labeled to represent the:

- output structure, or (for example nested-tables or nested-divs)
- interaction type (for example, integrated-modes)

During this transformation, logical decisions are made based on the XML data available and business/usability rules. Decisions are made on considerations such as authentication, user type, permissions, hierarchical order, categorical/sectional grouping, usability, Web conventions, and user conventions. These decisions determine what information and interaction is written to the final output.

As described in Chapter 2, “Web Standards Considerations” on page 7 of this guide, the Theme transformation should output well-formed semantic markup (for example, XHTML) ignorant of visual presentation. While ignorant of the visual presentation, this transformation may add structural hooks (mostly in the form of the *class* attribute) for the CSS to recognize and build upon.

---

## 3. Visual Presentation (Skin) Application

**Definition:** *Visual presentation layer: Application of visual presentation to the theme output.*

After generating the marked-up content of the page, Visual Presentation is applied. Presentation is an extra layer of information on top of a document’s structure, which builds up the (non-visual) structure into something visually appealing.

According to Web standards, as mentioned in Chapter 2, “Web Standards Considerations” on page 7, CSS is the “presentational” layer, and it can take a very simple marked-up document, and turn it into something amazing. CSS can influence both layout and style, and contains the specifications for flow, positioning, sizing, typography, colors, imagery, and even some interaction (rollovers, rendering content visible or hidden, highlighting).

Academus has a concept of visual presentation referred to as a *skin*. A skin is a package containing one or more CSS files and related images. Each Academus user may select (or be designated) a skin. During the theme transformation, the skin associated with the user is specified in the final markup by writing in the path to the skin package location. A skin usually is theme dependent, though there can be more than one skin for any theme.

*Note: CSS does not alter the document markup in any way. It is simply a filter that user-agents (browsers) apply and interpret before displaying the result.*

---

## Chapter 2. Web Standards Considerations

*Skin Relevance: Reference, Methodology*

Web standards are important considerations for the design of an Academus skin. This chapter provides a definition of the web standards and the importance of using Web standards.

---

### What Are Web Standards?

Web standards are technologies and the guidelines and specifications for technologies, established by the World Wide Web Consortium (W3C) and other standards bodies, which are used to create and interpret Web-based content.

The W3C's long-term goals for the web are:

1. *Universal Access*: To make the Web accessible to all by promoting technologies that take into account the vast differences in culture, languages, education, ability, material resources, access devices, and physical limitations of users on all continents.
2. *Semantic Web*: To develop a software environment that permits each user to make the best use of the resources available on the Web.
3. *Web of Trust*: To guide the Web's development with careful consideration for the novel legal, commercial, and social issues raised by this technology.

A document is said to adhere to Web standards when it complies with the guidelines and specifications set forth for the technologies used in the document. According to the W3C's Web standards, a Web document should use:

- XHTML 1.0 strict for structure,
- CSS Level 1 and Level 2 for presentation
- ECMAScript 262 (JavaScript) for scripting

---

### Why Use Web Standards?

A brief summary of the merits and benefits of Web standards includes:

- **Simpler development and maintenance**: Using more semantic and structured HTML makes it easier and quicker to understand code created by somebody else.
- **Compatibility with future Web browsers**: When you use defined standards and valid code you future-proof your documents by reducing the risk of future Web browsers not being able to understand the code you have used.

- Faster download and rendering of Web pages: Less HTML results in smaller file sizes and quicker downloads. Modern Web browsers render pages faster when they are in their standards mode than when they are in their backwards-compatible mode.
- Better accessibility: Semantic HTML, where structure is separated from presentation, makes it easier for screen readers and alternative browsing devices to interpret the content.
- Better search results: The separation of content and presentation makes the content represent a larger part of the total file size. Combined with semantic markup this will improve search results.
- Simpler adaptation: A semantically marked up document can be easily adapted to print and alternative browsing devices, like handheld computers and cellular phones, just by linking to a different CSS file. You can also make site-wide changes to presentation by editing a single file.

---

## Accessibility, ADA, and Section 508 Compliance

*"The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect."* -- Tim Berners-Lee, W3C Director and inventor of the World Wide Web

### Accessibility

In the context of the Web, accessibility is the concept that any user can access all the content of a Web document.

Accessible Web content is an important goal for your institution because of the diversity of user agent (browser) usage, language, culture, or disability within your users.

### ADA and Section 508 Compliance

The U.S. Government has made it law that 'goods and services' should be 'accessible' to Americans with disabilities (comprises 20% of the population), within government institutions. States, who receive funding, are required to abide by federal accessibility standards, and most colleges and universities are therefore subjected. Beyond the legal compliance, design responsibilities necessitate acceptance of accessibility standards to enable 40.8 million individuals the ability to receive and interacted with information despite a disability.

The Americans with Disabilities Act (ADA) was passed on July 26, 1990, as Public Law. It became effective from January 26, 1992. The purpose of the law is to prohibit discrimination on the basis of disability. The Americans with Disabilities Act provided a clear and comprehensive mandate for eliminating discrimination against individuals with disabilities.

The Americans with Disabilities Act was a combination of the Civil Rights Act of 1964 and the Rehabilitation Act of 1973. The Americans with Disabilities Act comprises five titles, where titles II and III are related to accessing information on the Web. These titles prohibit the state and local governments, as well as private enterprises, from discriminating

against people with disabilities in their programs and activities. The Americans with Disabilities Act was written to strike a balance between the reasonable accommodation of citizens' needs and the capacity of private and public entities to respond to these needs.

The Americans with Disabilities Act was strengthened by certain amendments to the Rehabilitation Act of 1973. These amendments were made to Section 508 and Section 504 to include provisions for organizations to comply with the Americans with Disabilities Act. In 1998, the Congress amended Section 508 of the Rehabilitation Act of 1973 and created more stringent rules for access to information in the Federal sector. It mandated that federal agencies should purchase only accessible software and hardware and that federal Web sites be made as accessible as possible.

The requirements stated under Section 508 were complemented by those in Section 504 of the Rehabilitation Act. Section 504 states that any agency receiving federal funds may not exclude or discriminate against individuals with disabilities. In compliance with Section 504, colleges and universities, which receive federal financial assistance, must not discriminate on the basis of disability. They should ensure that their academic programs are accessible, to the greatest extent possible, by all students with disabilities.

Both the Department of Education and the Department of Justice have ruled that the Americans with Disabilities Act and the Rehabilitation Act of 1973 apply to Web-based business as well as institutions extending higher education and distance education. The Department of Justice is enforcing the Americans with Disabilities Act through lawsuits and both formal and informal settlements. In November, the National Federation of the Blind filed a landmark lawsuit against America Online. The suit claimed that America Online violated the Americans with Disabilities Act by failing to provide access, for people with disabilities, to its site. As part of the settlement of this case, America Online had to develop an Accessibility Policy.

The Architectural and Transportation Barriers Compliance Board or the Access Board is an independent federal agency, whose primary mission is to promote accessibility for individuals with disabilities. This agency was given the task of formulating standards for compliance to Section 508. These standards were published in December 2000.

The W3C has been relied on heavily for their expertise and have formed the Web Accessibility Initiative (WAI) to help define priorities for making the Web accessible.

## **Academos 2.1 Web Standards**

Academos currently satisfies all the requirements of section 508 and all but one of the WCAG 1.0 priority 1 items. Because Academos utilizes performance-enhancing features offered by a JavaScript solution, it does not satisfy the 6.3 standard set forth by Web Content Accessibility Guidelines 1.0 as Unicon has interpreted it. There are also a number of WCAG priority 2 standards to which the Academos product family already complies.

Academos release 2.1 adheres to the following Web standards:

- Separation of structure from presentation
- XHTML compliant framework\*
- CSS for presentation
- ECMAScript 262 (JavaScript) for scripting

*Note: Academus may be configured to output XHTML; although, by default is configured to output HTML. The theme framework (stripped of any channel/portlet content) has been coded to be XHTML compliant and validates the W3 XHTML validator. Channel/portlet content may or may not be XHTML compliant dependent upon the provider. A few important issues surround the use of XHTML, especially around user-agent (browser) compliance and interpretation. The biggest of these issues is that Microsoft Internet Explorer (version 6.x or previous) does not handle XHTML documents. Second, all document content (including channel/portlet content) must validate to the XHTML specification or the entire document will fail to validate; therefore, a document declared as XHTML that does not validate on a single point, will not validate at all. Due to these and other issues, Academus recommends and defaults to HTML output.*

## Chapter 3. Directory Structure and File Location

*Skin Relevance: Specific*

This chapter describes the directory structure and file locations pertaining to the theme and skin.

*Note: All file location references will begin from your Academus portal Web application root directory. Since this setup can differ between deployments, the generic “[portal]” convention will be utilized to indicate this root directory.*

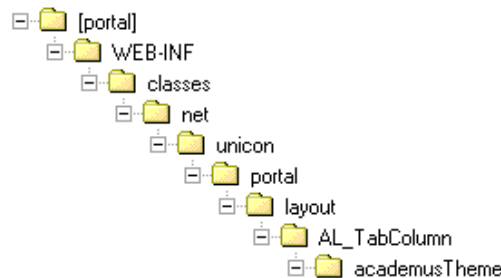
---

### Theme Files

Six files are responsible for the theme transformation:

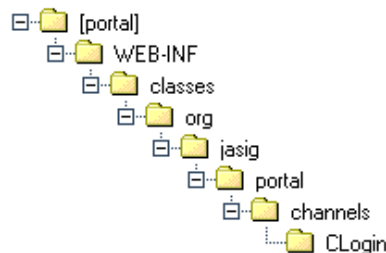
- *academus.xsl*
- *institution\_vars.xsl*
- *login.xsl*
- *navigation.xsl*
- *portlet.xsl*
- *preferences.xsl*.

These files are located in the following directory:

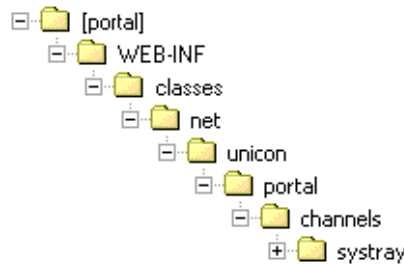


Two channels are included at the theme level.

- **CLogin**: Responsible for rendering the login component (username and password fields, form submission information, and login/submit button). The CLogin channel is located in the following directory:



- **System Tray:** Responsible for rendering the notifications Quicklink (a highlighted link to the notifications channel with an indicator of new notifications). The System Tray channel is located in the following directory:



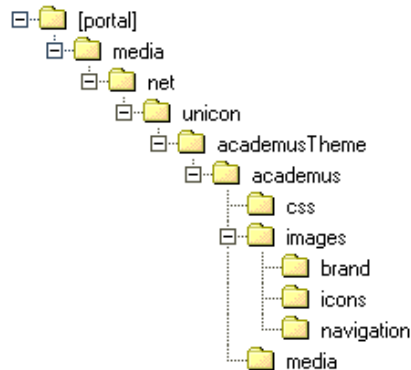
## Skin Registry

Academus utilizes an XML data file to register skins for use within the portal. This registry file is named `skinList.xml` and is located in the theme directory `academusTheme` as noted in “Theme Files” above.

For information on adding a new skin to the registry or modifying existing skin registry information see Chapter 5, “3. Add the New Skin to the skinList Registry” on page 35.

## Skin Files

Skin packages consist of a top-level directory named to match the skin internal name registered within Academus portal. Each skin should contain an identical sub-directory structure of `css` and `images`, with `images` containing further sub-directories of `brand`, `icons`, and `navigation`. The skin package is located in the following location:



## Skin Subdirectory Definitions

Subdirectory	Description	Context
Css	CSS style sheets	Global
Images > Brand	Images related to brand, sponsors, or partners	Portal header and footer
Images > Icons	Channel header and content icons	Channel or portlet
Images > Navigation	Images for styling main navigation	Main navigation/tabs
Media	Media objects; audio, video, Macromedia Flash, etc.	Not specific

---

## Chapter 4. Understanding and Customizing the Theme

*Skin Relevance: Overview, Theme Options*

This chapter provides an overview of the Academus theme and provides instruction on how to customize the theme.

---

### Theme Overview

As defined in Chapter 1, “Academus Rendering Process” on page 5, the Theme is the interaction layer. This layer makes logical decisions on information processed by the structural transform, processes variables, and produces the final (X)HTML document. The theme sets layout and interaction for the tab-column views, focused specific views or content, but does not control the channel containers within the portal framework.

Theme output is governed by a single XSL file that imports additional XSL files.

The Academus theme is customizable without altering the theme XSL files. This is accomplished by using theme variables to set institution-specific configurations of the theme. It is recommended that the theme variables be used rather than making modifications to the theme XSL files. If the theme variables do not satisfy desired customizations; however, the theme file may be modified.

*Important: Due to the complexity of the theme and special skills needed to make modifications, it is advised to contact Unicon for Academus theme customization services.*

### Theme Variable

Standard customization of the theme is accomplished through the use of Theme Variables. Theme Variables are optional institution-specific data that may be configured to suit the needs of an individual institution. At the time of rendering, the theme references the theme variables, altering content by logical code decisions depending upon the configuration of the variables.

## Theme Variable Listing

The Theme Variables are referenced by variable name. Some variables have multiple numbered listings (*related*, *contact*, *footer*) where each variable name contains a number. All of the numbered variables perform the same function, but allow for multiple inputs or listings. In the following table (and impact of theme variable on the theme and skin) the variables will be grouped under one listing with the number in the variable names listed as [1] (*contact[1]Url*, for instance).

Variable Name	Expected Value	Description
<b>Variables that affect the page header and brand</b>		
portalHtmlTitle	Text string	Sets HTML page <code>&lt;title&gt;</code>
portalMetaDescription	Text string	Sets HTML <code>&lt;meta type="description"&gt;</code>
portalMetaKeywords	Text string	Sets HTML <code>&lt;meta type="keywords"&gt;</code>
portalShortcutIcon	Absolute or relative file path (use forward slashes "/")	Sets alternate (non-default) location for Web document icon
institutionPortalName	Text string	Sets and displays the portal title
related[1]Label	Text string	Sets and displays a heading in the header of the page layout
related[1]Url	URL (http://www.domain.com)	Sets a URL link for the related label
userLogin	Text string *(includes variable call)	Sets the user welcome message
<b>Variables that affect the permissions/access</b>		
lockdownPreferences	true/false	Sets access to preferences link
<b>Variables that affect the layout</b>		
tabWidth	Positive integer or percentage (including the % symbol)	Sets the tab width used to calculate page widths
tabWidthBuffer	Positive integer or percentage (including the % symbol)	Sets the additional padding added to the calculated tab width used to calculate overall page width
pageMinWidth	Positive integer or percentage (including the % symbol)	Sets the minimum page width
leftColWidth	Positive integer or percentage (including the % symbol)	Sets the width of the left navigation column
Col[1]NarrowWidth	Positive integer or percentage (including the % symbol)	Sets the width of the narrow column in this position
Col[1]WideWidth	Positive integer or percentage (including the % symbol)	Sets the width of the wide column in this position
Col[1]EvenWidth	Positive integer or percentage (including the % symbol)	Sets the width of the even column in this position
Col[1]NarrowMinWidth	Positive integer or percentage (including the % symbol)	Sets the minimum width of the narrow column in this position
Col[1]WideMinWidth	Positive integer or percentage (including the % symbol)	Sets the minimum width of the wide column in this position
Col[1]EvenMinWidth	Positive integer or percentage (including the % symbol)	Sets the minimum width of the even column in this position
<b>Variables that affect the content</b>		
useDate	true/false	Sets usage of today's date
useLeftColumn	true/false	Sets usage of left navigation column
useAltNav	true/false	Sets usage of alternate navigation
useFlyoutMenus	true/false	Sets usage of flyout menus
useWebSearch	true/false	Sets usage of Web Search

webSearchLocation	left/header	Sets the location of Web Search
useSpellCheck	true/false	Sets usage of spellchecking
useQuicklinks	true/false	Sets usage of Quicklinks
quicklinksLocation	left/header	Sets the location of Quicklinks
useTabChannelNav	true/false	Sets usage of subnavigation
<b>Variables that affect Channel/Portlet help links</b>		
helpUrlPrefix	URL (http://www.domain.com)	Sets URL prefix for external help links
helpTargetNameDefault	Text string (valid window name)	Sets default window name for external help. Will be used unless otherwise overridden.
helpWindowPropertiesDefault	Window properties string (used to specify specs in window.open command)	Sets default window properties for external help windows. Will be used unless otherwise overridden.
<b>Variables that affect the page footer</b>		
useContact	true/false	Sets usage of contact information
institutionName	Text string	Sets and displays the institution title
institutionUrl	URL (http://www.domain.com)	Sets a URL link for the institution title
contact[1]Url	URL (http://www.domain.com)	Sets and displays a contact URL link
contact[1]Label	Text string	Sets and displays a contact label
contact[1]Email	Email address (person@domain.com)	Sets and displays a contact email address and link
contact[1]Text	Text string	Sets and displays contact text
institutionLocation	Text string (address)	Sets and displays institution's physical address
footer[1]Label	Text string	Sets and displays a label in the footer of the page layout
footer[1]Url	URL (http://www.domain.com)	Sets a URL link for the footer label
copyrightLabel	Text string	Sets and displays copyright text
copyrightUrl	URL (http://www.domain.com)	Sets a URL link for the copyright text
legalLabel	Text string	Sets and displays legal text
legalUrl	URL (http://www.domain.com)	Sets a URL link for the legal text
legalDescription	Text string	Sets and displays a legal description
sponsorLabel	Text string	Sets and displays a label in the footer of the page layout
sponsorUrl	URL (http://www.domain.com)	Sets a URL link for the sponsor text
displayVersion	true/false	Sets whether to display version information.

---

## Modifying Theme Variables

This section provides detailed procedures on how to modify theme variables.

*Warning: Unicon recommends making modifications to the theme variables prior to designing a new skin or modifying an existing skin, as the theme variables will alter the content that is being formatted by the skin.*

### Procedure

To modify theme variables, perform the following steps:

1. Locate the file. The theme variables are located in a file called *institution\_vars.xsl*.  
*Note: To locate the file, see Chapter 3, "Directory Structure and File Location" on page 11.*
2. Make a backup copy. The advantages are:
  - There will be a valid, working copy to fall back on in case modifications cause the theme to be corrupted.
  - There will be a valid, working file to use as a reference as changes and modifications are made to the customized file.
3. Store the backup in a safe location.
4. Open the file in an editor. Since the *institution\_vars* file is in XSLT format, it can be opened into any text editor, XML/XSL editor, and most HTML editors.
5. Modify the variables as desired. For details, see "Modifying Theme Variables" on page 16.
6. Save the changes, ensuring that the file keeps the same filename and proper format (.xsl extension).
7. View theme changes in a test environment. If changes were not made within an existing test environment, move the changed *institution\_vars.xsl* file to a test environment. View the changes and make revisions as necessary.

*Important: At this time, skin changes have not been made. The content may have the wrong visual presentation, which is acceptable (skin changes will be done in the next step). Verify the content is valid. If the content cannot be verified because of skin issues, go on to the next step.*

8. Modify the skin as necessary. Depending upon what changes were made to the theme variables, changes may be necessary for the skin to properly render the modified content.

*Note: For further details on testing skin changes, see "Modifying An Existing Skin" on page 38.*

9. Test theme and skin combination in a safe environment. If changes are made to the skin as a result of changes made to the theme variables, utilize a private test environment that mimics a production environment to test the combination of skin changes with theme variable changes within the portal. This will produce the most accurate testing conditions to verify that modifications render and behave as expected. Test changes on all supported OS/browser/version combinations. Make adjustments as necessary.

*Note: For further details on testing skin changes, see “Modifying An Existing Skin” on page 38.*

10. Move theme variables file and skin package to public environment. Once testing has been completed and the changes are deemed satisfactory for public release, copy *institution\_vars.xml* and the skin package directory (in its entirety) over to the corresponding directories on the public server. The changed files must overwrite the existing (previous version) files.
11. Restart the portal server. The portal will not recognize the modified theme variables until the server has been stopped and restarted. This action should also clear any cached skin files.
12. Verify theme and skin changes within the portal. Make a final testing pass on the public environment to ensure that changes are present and are behaving as expected. If problems arise, revert the public environment to a backup copy of the skin and theme variables, restart the portal, and troubleshoot changes to the skin and theme variables in the test environment. If problems persist, contact Unicon Professional Services.

## Modifying the Variables As Desired

*Important: Read the following section before making any changes to the theme variables.*

### Syntax

A base level understanding of XSLT and XSLT syntax is recommended before modifying this file. An explanation of the XSLT variable syntax follows.

*Warning: DO NOT alter the XSLT syntax.*

Each variable is contained in XSLT syntax. XSLT syntax for variables is:

```
<xsl:variable name="variableName">variable value</xsl:variable>
```

The basic variable syntax, `<xsl:variable></xsl:variable>` defines it as an XSLT variable. This syntax must exist for the XSLT to recognize it as a variable.

*Warning: DO NOT alter the variable names.*

## Warnings

The variable attribute `name="variableName"`, gives the variable a unique identifier that can be referenced. These unique identifiers have already been registered in the theme.

Theme Variables Sample:

```
<xsl:variable name="institutionName">Unicon, Inc.</xsl:variable>
<xsl:variable name="institutionUrl">http://www.unicon.net</xsl:variable>
<xsl:variable name="institutionPortalName">Academus</xsl:variable>
<xsl:variable name="institutionPortalUrl">http://www.unicon.net</xsl:variable>
```

Do not modify       Modify

The “variable value” is the value for that specific variable, and can be different for each of the variables.

Variable values may be changed, but variable values must NOT contain the following characters: `<` `>` `&` `'` `“`

These are illegal characters and must be replaced by entity references. There are five predefined entity references in XML/XSL:

Entity	Character	Description
&lt;	<	Less than
&gt;	>	Greater than
&amp;	&	Ampersand
&apos;	'	Apostrophe
&quot;	“	Quotation mark

*Example:* The variable value “Unicon & Academus” must be changed to “Unicon &amp; Academus”, or the value “>> Go” must be changed to “&gt;&gt; Go”, whereas the value “http://www.unicon.net” is valid (colons, back and forward slashes, and periods are all valid characters).

Be cautious of special characters (© ® ™) that may get pasted from a text editor, which can also be invalid characters.

*Important:* When in doubt, remove all special characters. DO NOT add or delete variables from the document and do not alter the following XSLT document identifiers.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

*Warning:* Improper XSLF syntax will cause a parsing error and result in the entire portal failing to render.

If changes are contained to only the variable values and do not use illegal characters, there should not be any problems with modifying the theme variables. It may also be helpful to use an XML/XSL editor with a code validator. Such programs can check to ensure your document is valid XSL and will report errors in the syntax.

*Note:* XML Cooktop® is a free XML/XSL editor with a good validator.

## Impact of Theme Variables on the Theme and Skin

### Variables that Affect the Page Header and Brand

Variable Name	Definition
<b>portalHtmlTitle</b>	<p>Sets HTML Page &lt;title&gt;</p> <p><i>Note: If the portal has CSCR enabled (ask your portal administrator), the portal utilizes a static HTML frameset page. According to the HTML specification, this frameset page has a &lt;title&gt;. This frameset &lt;title&gt; is not shared with the theme HTML page, and thus is not affected by the theme portalHtmlTitle variable. By default, the title of this page is "Academus Portal". To change this page title, locate the frameset file at:</i></p> <p><i>[portal]/main.html</i></p> <p><i>Open the file, edit the &lt;title&gt;, and save the changes. A portal restart is necessary to see the changes.</i></p>
<b>PortalMetaDescription</b>	Sets HTML <meta type="description">
<b>portalMetaKeywords</b>	Sets HTML <meta type="keywords">
<b>portalShortcutIcon</b>	<p>Sets alternate (non-default) location for Web document icon. Giving this variable a value will signal the Theme to render the following HTML into the &lt;head&gt; element of the document:</p> <pre>&lt;link rel="icon" href="portalShortcutIcon" /&gt; &lt;link rel="shortcut icon" href="portalShortcutIcon" /&gt;</pre> <p>Where portalShortcutIcon is the value of the variable, which should be a file path location to the icon file (.ico).</p>

Variable Name	Definition
<p><b>portalHtmlTitle</b></p>	<p>Sets HTML Page &lt;title&gt;</p> <p><i>Note: If the portal has CSCR enabled (ask your portal administrator), the portal utilizes a static HTML frameset page. According to the HTML specification, this frameset page has a &lt;title&gt;. This frameset &lt;title&gt; is not shared with the theme HTML page, and thus is not affected by the theme portalHtmlTitle variable. By default, the title of this page is "Academus Portal". To change this page title, locate the frameset file at:</i></p> <p><i>[portal]/main.html</i></p> <p><i>Open the file, edit the &lt;title&gt;, and save the changes. A portal restart is necessary to see the changes.</i></p>
<p><b>institutionPortalName</b></p>	<p>Sets and displays the portal title. Assigning this variable a value will generate the following HTML in &lt;div id="header"&gt;:</p> <pre>&lt;h1 id="mainlogo"&gt;&lt;a href="{home}"&gt;&lt;span&gt;institutionPortalName&lt;/span&gt;&lt;/a&gt;&lt;/h1&gt;</pre> <p>This variable essentially renders the main logo element. Use CSS to target <code>id="mainlogo"</code>, and then format the text or replace the image as desired.</p> <p><b>related[1]Label, related[1]Url</b></p> <p>Sets and displays a heading in the header of the page layout. There are variables for up to five related headings, each variable name containing the number (displayed in the names and examples in this document with [1]). Each of the related[1]Label variables does the same function, adding the following HTML to the header section of the page, &lt;div id="header"&gt;:</p> <pre>&lt;h2 id="related[1]"&gt;&lt;span&gt;related[1]Label&lt;/span&gt;&lt;/h2&gt;</pre> <p>Assigning a value to this variable (assuming a value exists for the corresponding label variable) will generate the following HTML:</p> <pre>&lt;h2 id="related[1]"&gt;&lt;a href="related[1]Url" target="_blank"&gt;&lt;span&gt;related[1]Label &lt;/span&gt;&lt;a/&gt;&lt;/h2&gt;</pre> <p>This allows the related headings to be linked.</p> <p><i>Note: All external links will be opened in a new browser window.</i></p> <p>The idea behind these headings is to give elements in the HTML that can be used to represent additional brand information (taglines or images) or partner representation (names, linked names, or images/logos). Utilize the corresponding related[1]Url to link these headings. Each heading has a unique <code>id</code> attribute that can be targeted for CSS formatting or image placement.</p>
<p><b>userWelcome</b></p>	<p>Sets the user welcome message. This text is rendered into the header and displayed to the user when logged in to the portal. This variable has a value that prepends to the user's portal username, which is filled in by the portal. A few examples:</p> <pre>Accessed by: {username} Welcome, {username} Customized for {username}</pre> <p>Where, in those examples, "Accessed by:", "Welcome," or "Customized for" are the value of the userWelcome variable.</p> <p><i>Note: A space is necessary at the end of the userWelcome value to have a space between the prepended userWelcome string and the username when they are concatenated.</i></p> <p>Alternatively, set the variable to null (no value) if no welcome message is desired. This will result in just the username being displayed.</p>

## Variables that Affect Permissions/Access

Variable Name	Definition
<b>lockdownPreferences</b>	Sets access of the preferences link within the portal. This link exists in the persistent header links and gives access to the preferences mode of the portal. The value is set to false by default, enabling access to preferences to all portal users. If lockdownPreferences is set to true, the preferences link will not be rendered to any user, except those who are members of the authorized channel publisher group. Refer to the Academus System Administrator Guide for more information on the authorized channel publisher group.

## Variables that Affect the Layout

Variable Name	Definition
<b>tabWidth</b>	<p>Sets the tab width used to calculate the minimum width of the html page to be displayed in the browser window when the default, horizontal navigation tabs are used. If a fixed width is assigned to the navigation tabs, this fixed width should be used for this variable value (including any padding or margin between the tabs). If a variable width is available to the tabs (tab width determined by tab name string length), then an average tab width value should be estimated (including any padding or margin between the tabs). The theme uses the following logic to determine the page's minimum width:</p> <p>A detached view has no minimum width.</p> <p>Does the theme use horizontal navigational tabs?</p> <ul style="list-style-type: none"> <li>• If no (<code>useAltNav = 'true'</code>), then use page default minimum width (<code>pageMinWidth</code>).</li> <li>• If yes (<code>useAltNav = 'false'</code>), does the tabs' total width exceed the default minimum width?</li> </ul> <p>Using the formula <code>{numberOfTabs} x {tabWidth}</code> to calculate the total tab width and compare to the value of <code>pageMinWidth</code>.</p> <ul style="list-style-type: none"> <li>• If no (calculated total tab width &lt; <code>pageMinWidth</code>), then use page default minimum width (<code>pageMinWidth</code>).</li> <li>• If yes, calculate the necessary width to accommodate all of the tabs and override the default page minimum width with this calculated value.</li> </ul> <p>Using the formula: <code>calculated minimum page width = ({numberOfTabs} x {tabWidth}) + {tabWidthBuffer}</code></p> <p><i>Note: Refer to the other variables that effect the layout for more information on the <code>tabWidthBuffer</code> and <code>pageMinWidth</code> variables.</i></p>
<b>tabWidthBuffer</b>	Sets the width used in the calculation of the page minimum width when the width necessary for display of horizontal navigational tabs exceeds the default page minimum width. The value of <code>tabWidthBuffer</code> is used to add margin to the calculated page minimum width if desired. Refer to <code>tabWidth</code> above for more information.
<b>pageMinWidth</b>	Sets the default page minimum width. The default value is 800 pixels.
<b>leftColWidth</b>	Sets the width of the left navigation column.
<b>col1Width</b>	Sets the width of the column in a single-column layout. The default value is 100%.
<b>col1MinWidth</b>	Sets the minimum width of the column in a single-column layout. The default value is 800 pixels.

<b>col21NarrowWidth</b> <b>col21WideWidth</b> <b>col21EvenWidth</b> <b>col22NarrowWidth</b> <b>col22WideWidth</b> <b>col22EvenWidth</b> <b>col2NarrowMinWidth</b> <b>col2WideMinWidth</b> <b>col2EvenMinWidth</b>	Sets the width and minimum width of the columns in a two-column layout, where [21] is the left column and [22] is the right column.
<b>col31NarrowWidth</b> <b>col31WideWidth</b> <b>col31EvenWidth</b> <b>col32Width</b> <b>col33NarrowWidth</b> <b>col33WideWidth</b> <b>col33EvenWidth</b> <b>col31NarrowMinWidth</b> <b>col33NarrowMinWidth</b> <b>col3WideMinWidth</b> <b>col3EvenMinWidth</b>	Sets the width and minimum width of the columns in a three-column layout, where [31] is the left column, [32] is the center column, and [33] is the right column.

### Variables that Affect the Page Content

Variable Name	Definition
<b>useDate</b>	Sets usage of a script to display the current date. The current date is calculated from the user's system clock. When set to true, this variable flags the theme to render the following html in the content area.
<i>Example:</i> <pre> &lt;div id="datedisplay"&gt;   &lt;span class="hide"&gt;Today's Date: &lt;/span&gt;   &lt;script type="text/javascript" language="JavaScript"&gt;currentDateDisplay();&lt;/script&gt; &lt;/div&gt;                     </pre>	
<b>useLeftColumn</b>	Sets usage of the left navigation column. The default value is true. When used, a left column is written into the html content table. This column may contain the following components: <ul style="list-style-type: none"> <li>• User Login (unauthenticated)</li> <li>• Web Search (authenticated, optional) [<i>useWebSearch</i> = "true" and <i>webSearchLocation</i> = "left"]</li> <li>• Quicklinks (authenticated, optional) [<i>useQuicklinks</i> = "true" and <i>quicklinksLocation</i> = "left"]</li> <li>• Alternate main navigation (authenticated, optional) [<i>useAltNav</i> = "true"]</li> <li>• Subnavigation (authenticated, optional) [<i>useTabChannelNav</i> = "true"]</li> </ul> This variable must be set to true in order for the alternate navigation and subnavigation components to render. It must also be set to true in order for the Web Search or Quicklinks to render if either of those components has a location set to left.

<p><b>Example:</b></p> <pre>&lt;td id="leftNav" width="{leftColWidth}"&gt;   &lt;div id="leftNavCol"&gt;      [components]    &lt;/div&gt; &lt;/td&gt;</pre>	
<p><b>useAltNav</b></p>	<p>Sets usage of alternate navigation. The default value is false. Setting the value of useAltNav to true signals the theme to change from the standard navigation to an alternate navigation. Standard (default) navigation for the theme is a row of tabs across the bottom of the header section in the page layout.</p> <p>To accomplish this, the theme generates an unordered list (<code>&lt;ul id="navigation"&gt;</code>) of the "tabs" (which are list items, <code>&lt;li&gt;</code>) in the header section (<code>&lt;div id="header"&gt;</code>) of the HTML.</p>
<p><b>Example:</b></p> <pre>&lt;div id="navcontainer"&gt;   &lt;ul id="navigation"&gt;     &lt;li id="navLi1" class="nav-li-selected first"&gt;&lt;span&gt;Tab Label 1&lt;/span&gt;&lt;/li&gt;     &lt;li&gt;&lt;a class="nav-li"&gt;&lt; href="tab2href"&gt;&lt;span&gt;Tab Label 2&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;     &lt;li&gt;&lt;a class="nav-li"&gt;&lt; href="tab3href"&gt;&lt;span&gt;Tab Label 3&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;     &lt;li&gt;&lt;a class="nav-li last" href="tab4href"&gt;&lt;span&gt;Tab Label 4&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;   &lt;/ul&gt; &lt;/div&gt;</pre>	
<p>CSS then dictates the styling of the list into visual tabs by referencing the id and class attribute names (<code>#navcontainer</code>, <code>#navigation</code>, <code>#navLi1</code>, <code>.nav-li-selected</code>, <code>.nav-li</code>, etc. in <code>navigation.css</code>).</p> <p>Switching alternative navigation on (<code>useAltNav = "true"</code>), the theme will render the same unordered list (<code>&lt;ul id="navigation"&gt;</code>), but will place it in the left navigation column (<code>&lt;td id="leftNav"&gt;</code>) of the main layout table (<code>&lt;table id="content-table"&gt;</code>) of the content section (<code>&lt;div id="content"&gt;</code>) of the HTML. All of the id and class attributes are changed to have an "alt" prefix.</p> <p><i>Note: These altnav CSS classes may then be used to render the alternate left navigation in a different manner (using <code>#altnavContainer</code>, <code>#altNavigation</code>, <code>#altNavLi1</code>, <code>.altnav-li-selected</code>, <code>.altnav-li</code>, etc. in <code>navigation.css</code>).</i></p>	
<p><b>Example:</b></p> <pre>&lt;div id="altnavContainer"&gt;   &lt;ul id="altNavigation"&gt;     &lt;li id="altNavLi1" class="altnav-li-selected first"&gt;&lt;span&gt;Tab Label 1&lt;/span&gt;&lt;/li&gt;     &lt;li&gt;&lt;a class="altnav-li"&gt;&lt; href="tab2href"&gt;&lt;span&gt;Tab Label 2&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;     &lt;li&gt;&lt;a class="altnav-li"&gt;&lt; href="tab3href"&gt;&lt;span&gt;Tab Label 3&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;     &lt;li&gt;&lt;a class="altnav-li last" href="tab4href"&gt;&lt;span&gt;Tab Label 4&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;   &lt;/ul&gt; &lt;/div&gt;</pre>	
<p><b>useWebSearch</b> <b>webSearchLocation</b></p>	<p>Sets usage of the Web Search component and its location in the layout. The default value for useWebSearch is true and the default value for webSearchLocation is left. When the usage value is true, the following HTML is written into either the left navigation column (<code>webSearchLocation = "left"</code>) or the header (<code>webSearchLocation = "header"</code>). When the usage value is false, no HTML is written for this component.</p>

*Example:*

```
<script type="text/javascript" language="JavaScript"
src="javascript/framework/search.js">
// Included JS file
</script>
<noscript>
  <div id="webSearchFormContainer">
    <form id="webSearchForm" action="http://www.google.com/search" method="get"
target="_parent">
      <label for="webSearchForm">Web Search</label>
      <input type="text" name="q" value="" id="webSearchInput" />
      <input type="submit" value="Search" id="webSearchSubmit" class="uportal-
button" />
    </form>
  </div>
</noscript>
```

If JavaScript is enabled in the user's browser, the Web Search form is written by the JavaScript. If JavaScript is disabled, the form between the <noscript> tags is used. The HTML form written by the JavaScript follows the same format (HTML tags and id names) as the <noscript> version, only it also writes a dropdown list of search options. This allows the user to select which search engine to query.

The list of search engine types, search engines, and listing order can be configured by modifying the included JavaScript file.

• **Modifying the Web Search Script**

**Important: JavaScript skill is recommended.**

*Note:* A JavaScript change does not require a restart of the web server. However, it may be necessary for users to clean their browser cache to see the updated script results.

To modify the search options, open javascript/framework/search.js. Locate the section shown in the following example:

*Example:*

```
var searchTypes =
{
  "web":
  {
    "label": "Web Search",
    "options":
    {
      "google":{
        "title": "Google",
        "buildUrl": buildUrlDefault,
        "url": "http://www.google.com/search?q="
      },
      "yahoo":{
        "title": "Yahoo",
        "buildUrl": buildUrlDefault,
        "url": "http://search.yahoo.com/search?p="
      }, ...
    }
  }
}
```

This modification defines the JavaScript object structure representing the search options dropdown. The properties defined under searchTypes represent the option groups within the select menu. Initially, these include:

- **web:** For general web search
- **references:** For answer or reference look up
- **news:** For news specific searches
- **books:** For book searches

Each of these option group objects has two properties:

- **label:** For the displayed Option Group text within the dropdown
- **options:** Includes the list of options under this option group

Each of these option objects has three properties:

- **title:** For the displayed Option text in the dropdown
- **buildUrl:** A function reference for how to construct the search url
- **url:** Specifies the first part of the URL for the search (and usually everything required for the search other than the search text itself).

<ul style="list-style-type: none"> <li>• <b>Changing the Text Displayed in the Select Dropdown</b></li> </ul>	<p><b>Important: Basic JavaScript skill is recommended.</b></p> <p>To change the labeling of the option group, change the text within the double-quotes next to “label” for the appropriate option group.</p> <p><i>Example: In this script snippet the option group label is “Web Search” and is intended to include general web search engines. Other predefined option groups include: References, News, and Books.</i></p> <p>To change the labeling of the options, change the text within the double-quotes next to “title” for the appropriate option.</p> <p><i>Example: In this script snippet the option label for Google is “Google”.</i></p>
<ul style="list-style-type: none"> <li>• <b>Adding or Modifying the Options or Option Groups</b></li> </ul>	<p><b>Important: Moderate to advanced JavaScript skill is recommended</b></p> <p>To re-order the options and/or option groups:</p> <ul style="list-style-type: none"> <li>• Cut and paste into the desired place order in the code structure. The displayed order is the same as the structure order.</li> <li>• Make sure to preserve the syntax integrity of the script’s braces and commas (properties must be separated by a comma, but the last property should not have a trailing comma).</li> </ul> <p>To remove option groups or options within the Web Search:</p> <ul style="list-style-type: none"> <li>• Comment out or delete the part of the <i>searchTypes</i> structure that should be removed. To comment out a section use the <i>/* */</i> syntax.</li> <li>• Make sure to preserve the syntax integrity of the script’s braces and commas.</li> </ul> <p>To add new options or option groups:</p> <ul style="list-style-type: none"> <li>• Copy the similar type in the desired order of the script structure.</li> <li>• Make sure the property name is unique. (For example, only one “google”:) and the syntax integrity of the script’s braces and commas are maintained. Then modify the labels/titles as desired.</li> </ul> <p>When adding a new option, the new search engine’s API will need to be understood and captured. Usually, this can be accomplished by doing a sample search on a specific search text with the desired search engine.</p> <p><i>Note: It is suggested to use a single word like “Test”.</i></p> <ul style="list-style-type: none"> <li>• Copy the resulting URL displayed in the address bar of the browser.</li> <li>• Paste this into the “url” property value of the new option.</li> </ul> <p><i>Example:</i></p> <p><i>Replace “url”:“http://www.google.com/search?q=” with “url”:“http://newsearchsite.com?search=Test&amp;lang=en”.</i></p> <p><i>Make sure the parameter within the URL that precedes the sample search text (“Test”), is moved to the end of the URL: (change “url”:“http://newsearchsite.com?search=Test&amp;lang=en” to “url”:“http://newsearchsite.com?lang=en&amp;search=Test”).</i></p> <p><i>Remove the sample search text: (change “url”:“http://newsearchsite.com?lang=en&amp;search=Test” to “url”:“http://newsearchsite.com?lang=en&amp;search=”).</i></p> <p><i>If the URL has the search text somewhere other than after a “?” in the URL, then a custom buildUrl function may need to be written.</i></p>

<ul style="list-style-type: none"> <li>• <b>Changing the Default Search Engine</b></li> </ul>	<p><b>Important: Basic JavaScript skill is recommended.</b></p> <p>The search engine that is selected by default is defined by the variable defaultSearch. The first time a user visits the site it will use the default search engine of "google", regardless of the order of the search engine options. Thereafter, if the user changes the search engine it will save a browser cookie with their preference for search engines. When the page refreshes or the site is visited later it will check the browser cookie and select the same search engine as their last selection.</p> <p>The default search engine can be changed by changing the value of defaultSearch</p> <p><i>Example: defaultSearch = "yahoo"</i></p> <p><i>Note: The text must match the property name for the option.</i></p>
<ul style="list-style-type: none"> <li>• <b>Changing the HTML output of the Web Search component</b></li> </ul>	<p><b>Important: Basic to moderate JavaScript skill is recommended.</b></p> <p>The variable HTMLText contains the HTML that will be written onto the page with the Web Search component. This may be altered as desired so long as the modifications comply with JavaScript syntax and the HTML specification.</p>
<p><b>useSpellCheck</b></p>	<p>Sets usage of the spell-checking component. The default value is true. When the variable value is true, the portal will load a JavaScript file containing the necessary functions for Spell Checker, and insert a spell check link beside each text area. Clicking the spell check link activates spell checking for the text area. When the usage variable is false, the JavaScript file will not load and no spell check links will appear in the portal.</p>
<p><b>useQuicklinks quicklinksLocation</b></p>	<p>Sets usage of the Quicklinks component and its location in the layout. The default value for useQuicklinks is true and the default value of quicklinksLocation is left. When the usage value is true, the following HTML is written into either the left navigation column (quicklinksLocation = "left") or the header (quicklinksLocation = "header"). When the usage value is false, no HTML is written for this component.</p>
<p><i>Example:</i></p> <pre>&lt;div id="qlinkscontainer"&gt;   &lt;span class="hide"&gt;Quicklinks&lt;/span&gt;   &lt;ul id="quicklinks"&gt;     &lt;li class="qlink-li first"&gt;&lt;a id="{qID}" class="qlink-link first"&gt;&lt;span class="qlink-label"&gt;Quicklink 1&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;     &lt;li class="qlink-li"&gt;&lt;a id="{qID}" class="qlink-link"&gt;&lt;span class="qlink- label"&gt;Quicklink 2&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;     &lt;li class="qlink-li"&gt;&lt;a id="{qID}" class="qlink-link"&gt;&lt;span class="qlink- label"&gt;Quicklink 3&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;     &lt;li class="qlink-li last"&gt;&lt;a id="{qID}" class="qlink-link last"&gt;&lt;span class="qlink-label"&gt;Quicklink 4&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;   &lt;/ul&gt;</pre>	
<p><b>useTabChannelNav</b></p>	<p>Sets usage of the subnavigation component. The default value is true. Requires usage of the left navigation column (refer to useLeftColumn above). When the usage value is true, the following HTML is written into the left navigation column. When the usage value is false, no HTML is written for this component.</p>
<p><i>Example:</i></p> <pre>&lt;div id="sidenavcontainer"&gt;   &lt;ul id="sidenav"&gt;     &lt;li class="sidenav-li first"&gt;&lt;a class="sidenav-link first"&gt;Subnav 1&lt;/a&gt;&lt;/li&gt;     &lt;li class="sidenav-li"&gt;&lt;a class="sidenav-link"&gt;Subnav 2&lt;/a&gt;&lt;/li&gt;     &lt;li class="sidenav-li"&gt;&lt;a class="sidenav-link"&gt;Subnav 3&lt;/a&gt;&lt;/li&gt;     &lt;li class="sidenav-li last"&gt;&lt;a class="sidenav-link last"&gt;Subnav 4&lt;/a&gt;&lt;/li&gt;   &lt;/ul&gt; &lt;/div&gt;</pre>	

## Variables that Affect Channel/Portlet Help Links

By default, channels and portlets are expected to display their own help within the channel/portlet. In Academus, if a channel/portlet supports help, there is a “Has Help” channel control that must be set. The theme then translates that into a channel/portlet specific link to activate the channel’s help.

Most Academus channels and portlets, on the other hand, make use of an external help system, displayed as a URL within a pop-up window. External help is used if the publishing parameter of `baseHelpUrl` or `helpUrl` is set. By default, all Academus channels and portlets use `baseHelpUrl` and not `helpUrl`.

*Note: The `helpUrl` will override a `baseHelpUrl` if both are set for a channel.*

The help URL, window name, and window properties can be controlled by a combination of setting theme variables and channel/portlet publishing parameters, as described in the following table:

Variable Name	Definition
<b>helpUrlPrefix</b>	<p>Sets the URL prefix to use when constructing a URL for channels/portlets with a publishing parameter of <code>baseHelpUrl</code>. Academus channels and portlets are published with their specific <code>baseHelpUrl</code> set. The theme will construct the final URL that will be called by appending the <code>baseHelpUrl</code> value to the <code>helpUrlPrefix</code> value. This allows the external help to be hosted in various locations and be easily configured.</p> <p><i>Example: For example, the Notifications Portlet help set up looks like the following:</i></p> <pre>PORTLET.baseHelpUrl = help_Bookmark_Channel.html</pre> <p>The default value of <code>helpUrlPrefix</code> is “<code>help_frameset.jsp?helpUrl=http%3A/wcm.unicon.net/help</code>”.</p> <p>The resulting link would have a href attribute of:</p> <pre>href="help_frameset.jsp?helpUrl=http%3A/wcm.unicon.net/help/help_Notifications_Portlet.html"</pre> <p>If a different default value is desired, simply change <code>helpUrlPrefix</code>. For example, if the desire is to display the help without a frameset, then set the value to “<code>http://wcm.unicon.net/help</code>”. All resulting help links would then be displayed without the frameset.</p> <p><i>Note: By default the Academus channels and portlets will point to the frameset (<code>help_frameset.jsp</code>) and will display an HTML file (<code>html/help/helpBanner.htm</code>) as a banner at the top of each help window. This help banner is expected to be customized by the institution. If this customization is not desired, then it is recommended the frameset be removed from <code>helpUrlPrefix</code>.</i></p>

<b>helpTargetNameDefault</b>	<p>Sets the help link target value when the external help is being used, and when no helpTargetName parameter is set.</p> <p><i>Note:</i> If the helpTargetName is set for the channel/portlet, that value will take precedence over helpTargetNameDefault.</p> <p><i>Example:</i> The Notifications Portlet help link would have a target of: target="HelpWindow". There is no helpTargetName defined that would override the default target name.</p>
<b>helpWindowPropertiesDefault</b>	<p>Sets the help window's properties when the external help is being used, and when no helpWindowProperties channel/portlet parameter is set.</p> <p><i>Note:</i> If helpWindowProperties is set for the channel/portlet, that value will take precedence over helpWindowPropertiesDefault.</p> <p><i>Example:</i> The Notifications Portlet help link would set the windows properties with the following code:</p> <pre>onclick="window.open('','HelpWindow','location=yes,menubar=yes,scrollbars=yes,status=yes,toolbar=yes,resizable=yes,location=no,height=600,width=800').focus();"</pre> <p>There is no helpWindowProperties defined that would override the default window properties.</p>

### Variables that Affect the Footer

Variable Name	Definition
<b>useContact</b>	<p>Sets usage of contact information. Default value is true. Contact information is displayed in the footer section of the page, &lt;div id="footer"&gt;. Setting useContact to false will signal the Theme to exclude rendering contact information elements into the HTML.</p>
<b>institutionName</b> <b>institutionUrl</b>	<p>Sets and displays the institution title. useContact must be set to true. The Theme will render the institutionName variable value as text in &lt;div id="footer"&gt; &lt;div id="institution"&gt; &lt;div id="institutionName"&gt;.</p> <p>Giving institutionUrl a value will add a URL to the institution title:                  &lt;a href="institutionUrl" target="_blank"&gt;institutionName&lt;/a&gt;</p>
<b>institutionLocation</b>	<p>Sets and displays the institution's physical address. useContact must be set to true. The Theme will render the institutionLocation variable value as text in &lt;div id="footer"&gt; &lt;div id="institution"&gt; &lt;div id="institutionLocation"&gt;.</p>
<b>contact[1]Url</b> <b>contact[1]Label</b> <b>contact[1]Email</b> <b>contact[1]Text</b>	<p>Sets and displays contact information in the footer section &lt;div id="footer"&gt; &lt;div id="institution"&gt; &lt;div id="institutionContact"&gt;. useContact must be set to true for the resultant HTML of these variables to be included and displayed in the Web document. If any of these variables has a non-null value, the Theme will be signaled to create &lt;div id="contact1"&gt;, and place contact information in it as described by the contact variable values as follows:</p>
<p><i>Example:</i></p> <pre>&lt;div id="contact1"&gt;   &lt;div id="contact1Url" target="_blank"&gt;&lt;a href="contact1Url"&gt;contact1Url&lt;/a&gt;&lt;/div&gt;   &lt;span id="contact1Label"&gt;contact1Label&lt;/span&gt;   &lt;a id="contact1Email" href="mailto:contact1Email"&gt;contact1Text&lt;/a&gt;   &lt;div id="contact1Text"&gt;contact1Text&lt;/div&gt; &lt;/div&gt;</pre>	

<p><i>Note:</i> The function is exactly the same for the <code>contact[2]</code> set of variables, only with name changes:</p> <pre>&lt;div id="contact2"&gt;   &lt;div id="contact2Url" target="_blank"&gt;&lt;a href="contact2Url"&gt;contact2Url&lt;/a&gt;&lt;/div&gt;   &lt;span id="contact2Label"&gt;contact2Label&lt;/span&gt;   &lt;a id="contact2Email" href="mailto:contact2Email"&gt;contact2Text&lt;/a&gt;   &lt;div id="contact2Text"&gt;contact2Text&lt;/div&gt; &lt;/div&gt;</pre> <p>Setting values for any of the contact variables in both contact sets ([1] and [2]) will cause both <code>&lt;div id="contact1"&gt;</code> and <code>&lt;div id="contact2"&gt;</code> to be displayed. All URLs will open in a new window.</p>	
<p><b>footer[1]Label</b> <b>footer[1]Url</b></p>	<p>Sets and displays a label in the footer of the page layout. The first footer label (<code>footer1Label</code>) must have a non-null value to signal the Theme to create the following HTML in <code>&lt;div id="footer"&gt;</code>:</p> <pre>&lt;ul id="footerlinks"&gt;   &lt;li&gt;&lt;a href="footer1Url" target="_blank"&gt;footer1Label&lt;/a&gt;&lt;/li&gt; &lt;/ul&gt;</pre> <p>Each additional footerLabel (2-10) given a value will create another list item (<code>&lt;li&gt;</code>) in the list, with the corresponding footerUrl value determining the URL (<code>&lt;a href&gt;</code>). All URLs will open in a new window.</p>
<p><b>copyrightLabel</b> <b>copyrightUrl</b> <b>legalLabel</b> <b>legalUrl</b> <b>legalDescription</b></p>	<p>Sets and displays copyright and legal information. If <code>copyrightLabel</code> or <code>legalLabel</code> has a non-null value, the following HTML will be rendered in <code>&lt;div id="footer"&gt;</code>, with each of the <code>&lt;p&gt;</code> lines rendered when a non-null value is found for <code>copyrightLabel</code>, <code>legalLabel</code>, or <code>legalDescription</code>, respectively:</p>
<p><i>Example:</i></p> <pre>&lt;div id="legal"&gt;   &lt;p id="copyright"&gt;&lt;a href="copyrightUrl" target="_blank"&gt;copyrightLabel&lt;/a&gt;&lt;/p&gt;   &lt;p id="legalStatement"&gt;&lt;a href="legalUrl" target="_blank"&gt;legalLabel&lt;/a&gt;&lt;/p&gt;   &lt;p id="legalDescription"&gt;legalDescription&lt;/p&gt; &lt;/div&gt;</pre>	
<p><b>sponsorLabel</b> <b>sponsorUrl</b></p>	<p>Sets and displays sponsor information. If <code>sponsorLabel</code> has a non-null value, the following HTML will be rendered in <code>&lt;div id="footer"&gt;</code>:</p> <pre>&lt;div id="sponsor"&gt;   &lt;a href="sponsorUrl" target="_blank"&gt;&lt;span&gt;sponsorLabel&lt;/span&gt;&lt;/a&gt; &lt;/div&gt;</pre> <p>This is intended to render variations of the “sponsored by” or “powered by” component of a Web document.</p>
<p><b>displayVersion</b></p>	<p>Sets whether the build version of Academus will display in the footer.</p> <p>If this is set to true, it will appear as:</p> <pre>&lt;div id="version"&gt; Academus 2.1.0, Academus build number 2.1.10, built on uPortal 2.5.3 &lt;/div&gt;</pre> <p>If you change the value to false it will hide it in a comment on the page. It will be visible only in the page source.</p> <p>This exists for support and maintenance reasons.</p>

**Notes**

A large, empty rounded rectangular box with a thin black border, intended for taking notes. The box is positioned below the 'Notes' header and occupies most of the page's vertical space.

---

## Chapter 5. Understanding, Creating, and Modifying Skins

*Skin Relevance: Overview, Specific*

This chapter discusses how to create a new skin, as well as modify and existing skin.

*Warning: Prior to modifying a skin, note that skin changes are considered customizations; therefore, changes are not supported under your Unicon support contract. If you have questions about your support contract or require assistance in this process, contact Unicon.*

---

### Skin Overview

As defined in Chapter 1, “Academus Rendering Process” on page 5, the Skin is an application of visual presentation to the theme output. Visual presentation is governed by three mechanisms:

1. Cascading Style Sheets (CSS)
2. Images
3. Media Objects

Each of these mechanisms may have multiple associated files. The collection of these associated files is a Skin Package, and refers to the full set of files used to determine a Skin.

## 1. Cascading Style Sheets

A Cascading Style Sheet (CSS) is a simple mechanism for adding style to web content.

*Example: Fonts, colors, spacing*

CSS controls the bulk of the visual presentation in Academus by importing style sheets into the portal web documents. References in the HTML document to import the style sheets are permanent (without changing the theme itself), but the contents and specifications of the style sheets may be modified.

In the Academus architecture, four CSS files are used for presentation on the screen and one CSS file is used for presentation of paged (print) format. For locating the CSS files in the skin directory structure, see Chapter 3, "Directory Structure and File Location" on page 11.

CSS Files	Definition
Screen Presentation CSS	Screen presentation specifications are broken down into six CSS files, separated by category. Given the cascading nature of the CSS specifications from one style sheet can affect or override the specifications in another. The style sheets listed in this table are listed in order of cascading priority, from highest priority to least priority.
Navigation	The navigation CSS file (navigation.css) formats all of the navigation elements in Academus, including default navigation (tabs), alternate navigation (left-side), flyout menus, Quicklinks, and subnavigation menus.
Preferences	The preferences CSS file (preferences.css) formats the preferences/customization areas of Academus, including adding content, changing layout, and changing skins.
Main	The main CSS file (main.css) contains the majority of the theme level specifications. This CSS file covers the default views of the portal, sets branding specifications, page layout, font specifications, and color specifications.
Academus Deprecated	This CSS file (academus_deprecated.css) is a stripped-down version of the Academus 1.3 CSS file (academus.css), included to cover presentation specifications of Academus channels already in existence from past releases.
Portlet	To support portlets in Academus, this style sheet (portlet.css) covers the CSS specifications for portlets as set forth in the Java Community Portlet Specification JSR-168.
uPortal	To support open-source channels that utilize the uPortal CSS specification, Academus includes this style sheet (uportal.css) from open-source uPortal.
Print Presentation CSS	Print presentation specifications for paged media are designated in a single style sheet (print.css).

## 2. Images

Images are graphics displayed within a Web document for such uses as branding, visuals, diagrams, icons and controls. These images may be referenced directly in the HTML document via an `<img>` tag, or may be referenced from the CSS via a `background-image: url("directoryLocation/filename.ext");` property within a CSS element.

Image references in the HTML document (the call for the browser to look for an image) are permanent (without changing the theme itself), but the image source (the specific instance of that image) may be altered in graphic style, color, size, or shape by locating the appropriate image file and editing the graphic in an image editor. Image references within CSS elements are non-permanent, where not only the image source but also the image reference itself may be modified (pointed to another image source, removed, etc.).

### 3. Media Objects

Media Objects are video, audio, and related media objects (in this definition not including images or CSS) that can be referenced within a Web document and displayed by a Web browser. Most media objects require a third-party plug-in to the Web browser in order to enable the Web browser to properly read and display the object. Media object references in the HTML document are permanent (without changing the theme itself), but the object source may be modified.

A singular media object – the Collaborative Groupware gradebook channel header (a Macromedia Flash Player object) – is the only media object currently in use in Academus.

---

## Skin Use in Academus

A skin must be registered within Academus for use. Once registered, the skin is available to users of the portal who may select the skin by means of portal customization.

For information on registering a skin within Academus, refer to “3. Add the New Skin to the skinList Registry” on page 35.

Standard customization the theme is accomplished through the use of Theme Variables. Theme variables are optional institution-specific data that may be configured to suit the needs of an individual institution. At the time of rendering, the theme references the theme variables, altering content by logical code decisions depending upon the configuration of the variables. For more information see “Understanding and Customizing the Theme” on page 13

*Note: It is recommended to make modifications to the theme variables prior to designing a new skin or modifying an existing skin, as the theme variables will alter the content that is being formatted by the skin.*

---

## Creating A New Skin

*Note: The Administrator of an institution can determine which skins are available to the institution’s users by customizing the skinList.xml registry file included with the portal installation. Unless some form of lockdown is applied to the portal, each user can select a skin from the available list using Preferences within the portal. For the location of the skinList file, reference Chapter 3, “Directory Structure and File Location” on page 11.*

To add a new skin to Academus, perform the following steps:

*Note: These steps are detailed in this section.*

1. Copy the entire skin package directory.
2. Rename the skin package directory.
3. Add the new skin to the *skinList* file.
4. If applicable, make modifications to the existing skin files.

5. Test new skin in a test environment.
6. Move new skin package (and *skinList* or *skinList* changes) to public environment.
7. Restart the portal server.
8. Verify new skin within the portal.

## 1. Copy the Entire Skin Package Directory

*Important: It is recommended to copy the skin that ships with Academus, or one that has been verified as a valid skin.*

1. Locate the skin directory `academusTheme`, for assistance, see Chapter 3, “Directory Structure and File Location” on page 11.
2. Take an existing skin package and copy it in its entirety to the same directory level.

## 2. Rename the Skin Package Directory

1. Select a name for the new skin.

*Note: Academus will use this name internally and will seek to locate a skin package directory with a matching name.*

2. Ensure the name matches both in spelling and case.

## 3. Add the New Skin to the *skinList* Registry

Academus utilizes an xml file named *skinList.xml* to register skins as available in the portal. In order for a new skin to be recognized as an option for portal users, it must be added to the skinList file. Below is a sample from the file:

### skinList XML Sample

```
<skins>
  <skin>
    <skin>academus</skin>
    <skin-name>Academus</skin-name>
    <skin-description>Academus visual style</skin-description>
  </skin>
  <skin>
    <skin>unicon</skin>
    <skin-name>Unicon</skin-name>
    <skin-description>Unicon visual style</skin-description>
  </skin>
  <skin>
    <skin>new_visual_style</skin>
    <skin-name>New Visual Style</skin-name>
    <skin-description>My Institution's Atheltics visual style</skin-description>
  </skin>
</skins>
```

### Understanding the skinList XML

- `<skins>` node is the parent node and a container for each of the individual `<skin>` nodes.
- `<skin>` node is a container for skin-specific information. A full entry for a skin begins with `<skin>` and ends with `</skin>`, and is the child node directly underneath the parent `<skins>` node. Within the `<skin>` node, are three nodes (`skin`, `skin-name`, `skin-description`) that specify specific information about the skin.
- `<skin>` node is the internal name used by the portal

*Important: This name must match the name of the skin package directory in both spelling and case.*

- `<skin-name>` node is the name that will be displayed to users
- `<skin-description>` node is the description that will be provided with the skin

## Editing the skinList XML

*Note: Any text editor and most html editors can edit xml files.*

To edit the skinList xml:

1. Locate the skinList file and open it in the preferred editor.
2. Add a new skin entry in the skinList file.
  - Copy an existing entry from <skin> to </skin>, including the <skin>, <skin-name>, and <skin-description> nodes in-between.
  - Paste this xml within the parent <skins> node, following the same xml structure as depicted above.
3. Change the specific information to reflect the new skin.

*Note: The <skin> text name must match the skin package directory name in order for Academus to find and recognize the skin package files associated with it.*

4. Save the changes to the file.

## 4. Make Modifications to Skin Files

For detailed instructions on how to modify existing skin files, refer to “2. Make Modifications to Skin Files” on page 38.

## 5. Test New Skin in a Test Environment

1. Utilize a private test environment that mimics a production environment to test skin changes within the portal.

*Note: This will produce the most accurate testing conditions to verify that skin changes render and behave as expected.*

2. Test changes on all supported OS/browser/version combinations. Make adjustments as necessary.
3. Test for errors.
4. Test for usability. Determine whether skin changes have made the user interface more or less usable.

*Note: Usability testing is done by putting designs in front of actual or representative users and getting their feedback. This feedback will represent how a larger user group will receive skin/user interface changes and quickly identify issues before changes are released to the public. Unicon offers usability services using a User-Centered Design (UCD) philosophy and methodology, with skilled professionals to assist you. Contact Unicon if you have questions about UCD, usability, or user interface design.*

## 6. Move New Skin Package to Public Environment

1. Ensure all testing has been completed and the skin is deemed satisfactory for public release.
2. Copy the entire new skin package directory to the skin directory on the public server.
3. Copy the updated skinList.xml file to the location on the public server. This will overwrite the existing file with the new changes.

*Note: An alternative to copying the test skinList.xml file: The existing production/public skinList file may be opened and edited to register the new skin.*

## 7. Restart the Portal Server

1. Stop the Server.
2. Restart the Server.

*Important: The portal will not recognize the new skin registration until the server has been stopped and restarted. This action should also clear any cached skin files.*

## 8. Verify New Skin Within the Portal

1. Once the portal has restarted, login as one of the portal users and verify the new skin displays as a selection in preferences.
2. Select the skin for that user and verify the new skin renders as expected.
  - If problems arise, remove the skin from the skinList, restart the portal, and troubleshoot in the test environment.
  - If problems persist, contact Unicon.

## Modifying An Existing Skin

*Note: The Administrator of an institution can determine which skins are available to the institution's users by customizing the skinList.xml registry file included with the portal installation. Unless some form of lockdown is applied to the portal, each user can select a skin from the available list using Preferences within the portal. For the location of the skinList file, reference Chapter 3, "Directory Structure and File Location" on page 11.*

In modifying an existing skin, use the following steps:

*Note: These steps are detailed in this section.*

1. Create a backup of the skin package.
2. Make modifications to skin files.
3. Test skin in a test environment.
4. Move skin package to public environment.
5. Restart the portal server.
6. Verify skin changes within the portal.

### 1. Create a Backup of the Skin Package

The first step in modifying your skin is to make a backup. Having a backup provides the following advantages:

- There will be a valid, working copy to fall back on in case modifications cause the skin to be corrupted in such a way that it cannot be (or is chosen not to be) recovered.
- There will be valid, working files to use as a reference as changes and modifications are made.

### 2. Make Modifications to Skin Files

Modification of the skin files will fall into one of two categories: CSS or image.

#### Modifying Cascading Style Sheets (CSS)

*Important: Unicon recommends that users who modify a skin should possess a thorough understanding of CSS technology and syntax before making modifications. Any text editor and most html editors can edit CSS files.*

1. Locate the .css file to be modified in the skin package css subdirectory.
2. Make modifications to the CSS classes as desired.
3. Validate your CSS.

## Modifying Graphic Images

- Graphic images in the skin are of one of two formats: .gif or .jpg.
- Graphics can be edited in your preferred image editing software.
- New graphic images may be added at any time to the images directory or brand, icons, or navigation subdirectories and referenced from the style sheet.
- Filename and file format may change, as long as those changes are also made to the image reference in the CSS.
- Academus images are *size agnostic*, meaning that changing the height or width of an image is legal and acceptable and does not require a change in the code.
- Changing graphic images should not require a restart of the portal; although, there may be some temporary issues with image caching at the server or browser level.
  - If you are not seeing a change to a graphic image, clear your browser's local cache.
  - If your change still does not show up, clear the server cache and restart the portal.
  - If you continue to have problems, contact Unicon.

## Modifying Media Objects

1. Locate the Macromedia Flash Player® (.swf) file named tblHdr.swf. It is located in the media subdirectory, along with tblHdr fla, the file in Macromedia Flash authoring format.

*Note: This file is utilized for the column headers of the Gradebook channel within Academus Collaborative Groupware.*

2. Edit the tblHdr fla file.

*Note: Macromedia Flash authoring software is required to edit this file.*

3. Publish the file to a Macromedia Flash Player (.swf) format.

*Note: Flash Player version 5 is recommended.*

4. Overwrite the existing file.

## 3. Test the Skin in a Test Environment

1. Utilize a private test environment that mimics a production environment to test skin changes within the portal.

*Note: This will produce the most accurate testing conditions to verify that skin changes render and behave as expected.*

2. Test changes on all supported OS/browser/version combinations. Make adjustments as necessary.

3. Test for errors.
4. Test for usability. Determine whether skin changes have made the user interface more or less usable.

*Note: Usability testing is done by putting designs in front of actual or representative users and getting their feedback. This feedback will represent how a larger user group will receive skin/user interface changes and quickly identify issues before changes are released to the public. Unicon offers usability services using a User-Centered Design (UCD) philosophy and methodology, with skilled professionals to assist you. Contact Unicon if you have questions about UCD, usability, or user interface design.*

#### **4. Move Skin Package to Public Environment**

1. Ensure all testing has been completed and the skin is deemed satisfactory for public release.
2. Copy the entire new skin package directory to the skin directory on the public server. The changed skin package files must overwrite the existing (previous version) files.

#### **5. Restart the Portal Server**

Some skin modifications will not require a restart of the portal server; therefore, this step may not be a required.

1. Stop the Server.
2. Restart the Server.

*Note: Restarting the server will clear any files cached on the server,*

#### **6. Verify Skin Changes Within the Portal**

1. Perform a final testing pass on the public environment to ensure that changes are present and are behaving as expected.
2. If problems arise, revert the public environment to a backup copy of the skin, restart the portal, and troubleshoot skin changes in the test environment.
3. If problems persist, contact Unicon.

## Upgrading Skins from Academus 2.0 to Academus 2.1

Institutions may opt to use a custom skin for their portal deployment as a way to brand their portal. This customization includes changes to theme color, but may contain other detailed changes. Unfortunately, this type of customization may make it difficult to ensure a smooth upgrade experience.

### CSS Files

Changes made to the default portal CSS files will not be automatically available in the customized skin. Academus 2.1 contains changes to the portal CSS files; for institutions to get the CSS improvements, effort will need to be made to your customized skin files.

### Upgrading the Skin

To upgrade the skin to work with Academus 2.1, Unicon recommends performing the following steps:

1. Create a backup of your skin package and the default Academus 2.1 skin files.
2. Place your skin package into Academus 2.1, as it was in your Academus 2.0 deployment.
3. Test the portal for errors and usability. If errors are detected, Unicon recommends a merger process, similar to the following:
  - Use a software tool (for example, WinMerge) to compare the CSS skin files between the base Academus 2.1 CSS and the base Academus 2.0 CSS. This will identify what exactly has changed between the versions that will need to be carried over into your custom skin.
  - For each change, locate the line within the skin and make the appropriate changes (addition, edit, or deletion). Before adding or deleting, make sure the selector is not located elsewhere in the new 2.1 file. Instead of deleting, just comment it out in the file, until the file is tested and found to be satisfactory.
  - If the change involves a color style, select an appropriate color value to use (for example, look elsewhere in the file for color values being currently used) in place of the default color.
  - If the merger process does not end with the desired result, it will require a CSS professional to further adjust the files to achieve the final goal.

*Important: Before modifying the skin files, read the section “Modifying An Existing Skin” on page 38.*

**Notes**

A large, empty rounded rectangular box with a thin black border, intended for taking notes. The box is positioned below the 'Notes' header and occupies most of the page's vertical space.

---

## Chapter 6. Examples

This chapter provides instructions that answer the following questions regarding common theme/skin changes in Academus:

- How do I change the main logo of the portal?
- How do I change the colors of a skin?
- How do I change information specific to Academus/Unicon to information specific to my institution?
- How do I change from horizontal tab navigation to vertical left side navigation?

## How do I change the main logo of the portal?

The main logo is image that is specified in the main.css file of the skin. The theme writes out the following HTML for the main logo:

```
<h1 id="mainlogo"><a><span>{institutionPortalName}</span></a></h1>
```

For example, the Academus skin contains the following CSS classes to format the logo:

```
#header h1 {
margin: 0px;
padding: 0px;
}

#header #mainlogo {
background: url("../images/brand/main_logo.gif") no-repeat;
position: absolute;
top: 23px;
left: 18px;
z-index: 5;
}

#header #mainlogo a {
display: block;
height: 29px;
width: 252px;
}
```

To change the image that is used for the main logo, take one of two paths:

1. Edit the existing image file:
  - Follow the directions for Modifying An Existing Skin. Locate the skin directory and find the image file main\_logo.gif.
  - Open the file in an image editor and make desired changes to the image.
  - Save the changes.
  - It may also be necessary to change related CSS classes if the size of the image has changed.
2. Change the CSS class to point to a different image file:
  - Follow the directions for Modifying An Existing Skin.
  - Locate the skin directory and find the file main.css.
  - Edit the background reference of #header #mainlogo:

```
background: url("../images/brand/main_logo.gif") no-repeat;
```

For example,

```
background: url("../images/brand/institution_logo.gif") no-repeat;
```
  - Save the changes.
  - It may also be necessary to change related CSS classes if the size of the image has changed.
  - Follow the directions for Modifying An Existing Skin.

## How do I change the colors of a skin?

*Important: All of the skin colors are controlled by CSS.*

1. Follow the directions for Modifying An Existing Skin.
2. Locate the skin directory and find the CSS file main.css.
3. Open this file in a CSS or text editor. Color specifications in CSS are done with hexadecimal values, and usually appear in specifications for background, background-color, color, or border.

```
body {
font-family: verdana, arial, sans-serif;
font-size: 7pt;
background-color: #FFF;
color: #666;
}
```

- #FFF Specifies a white background color
- #666 Specifies a dark grey text color for the document body.

*Note: To change the colors globally, use an editor with a search and replace feature. Search for the hexadecimal value of the color to change, and replace that value with the desired new color hexadecimal value. For example, search for “#666” and replace with “#000”.*

4. Test changes by viewing the portal skin in a browser (will often require a browser refresh to see the change), or by using a CSS editor with a preview pane.
5. Repeat the process until the desired colors are achieved.
6. Save the changes to the CSS file.
7. Follow the directions for Modifying An Existing Skin.

## **How do I change information specific to Academus/Unicon to information specific to my institution?**

*Important: All institution specific information related to the theme is located in the theme variables.*

1. Follow the directions for Modifying Theme Variables.
2. Read the Syntax, Notes, and Warnings and locate the theme file `instution_vars.xml` and open it in an XSL or text editor.
3. Change any of the variable values as desired.

*Example:*

```
<xsl:variable name="institutionName">Unicon, Inc.</xsl:variable>
```

Might be changed to:

```
<xsl:variable name="institutionName">My Institution</xsl:variable>
```

Or,

```
<xsl:variable name="copyrightLabel">Copyright 1993-2005 Unicon, Inc.</xsl:variable>
```

Might be changed to:

```
<xsl:variable name="copyrightLabel">Copyright 2005 My Institution</xsl:variable>
```

4. Repeat the process until the desired information is achieved.
5. Save the changes to the XSL file.
6. Follow the directions for Modifying Theme Variables.

## How do I change from horizontal tab navigation to vertical left side navigation?

*Important: This procedure provides a change to the theme, accomplished by changing values of the theme variables.*

1. Follow the directions for Modifying Theme Variables.
2. Read the Syntax, Notes, and Warnings, locate the theme file `instution_vars.xml` and open it in an XSL or text editor.
3. Change the following variables to the following values:
 

```
<xsl:variable name="useLeftColumn">true</xsl:variable>
<xsl:variable name="useAltNav">true</xsl:variable>
```
4. Use the alternate navigation ids and CSS classes (see section `useAltNav`) to visually format the navigation component HTML. It is suggested that those CSS declarations be made in the skin file `navigation.css`.

An example might be:

```
/* ((((( ALT Navigation ))))) */
/*-----*/
#altnavContainer {
margin: 2px 0px 2px 0px;
padding-top: 10px;
background: #39638B url("../images/navigation/left_nav_top.gif") no-repeat top left;
}

#altNavigation {
margin: 0px;
padding: 0px;
padding-bottom: 10px;
background: #39638B url("../images/navigation/left_nav_bottom.gif") no-repeat left bottom;
list-style: none;
}

.altnav-li, .altnav-li-selected {
border-top: 1px solid #628CB6;
}

#altNavigation .first {
border: none;
}

.altnav-link:link, .altnav-link:visited, .altnav-link:hover, .altnav-link-selected:link,
.altnav-link-selected:visited, .altnav-link-selected:hover {
display: block;
margin-bottom: 1px;
padding: 4px 10px 4px 16px;
background: #39628A url("../images/navigation/sidenav_arrow.gif") no-repeat;
background-position: 3px 3px;
color: #FFF;
font-size: 8pt;
text-decoration: none;
}

.altnav-link:hover, .altnav-link-selected:hover {
background-color: #325678;
}
```

```
#altsubnavSelected {
margin: 0px;
margin-left: 15px;
padding: 0px;
list-style: none;
}

.altsubnav-selected-li {
margin: 1px;
}

.altsubnav-selected-link:link, .altsubnav-selected-link:visited, .altsubnav-selected-
link:hover {
display: block;
padding: 3px 8px;
border-top: 1px solid #39638B;
color: #D4E4F4;
font-size: 8pt;
}

.altsubnav-selected-link:hover {
background: #477BAD;
color: #FFF;
}

.altsubnav {
margin-top: -21px;
padding: 0px;
list-style: none;
background-color: #EEE;
border-top: 1px solid #AAA;
border-left: 1px solid #AAA;
border-right: 2px solid #AAA;
border-bottom: 2px solid #AAA;
z-index: 10;
position: absolute;
left: 175px;
}

.altsubnav-li {
margin: 1px;
background-color: #CCC;
}

.altsubnav-link:link, .altsubnav-link:visited, .altsubnav-link:hover {
display: block;
width: 200px;
padding: 2px 10px;
color: #000;
font-size: 8pt;
}

.altsubnav-link:hover {
background-color: #B7B7B7;
}
```

5. Follow the directions for “Modifying the Variables As Desired” on page 17 and Modifying Theme Variables and “Modifying An Existing Skin” on page 38.

## Chapter 7. Terms and Definitions

Term	Definition
Academus	Academus Portal from Unicon delivers the framework, software, and services you need to establish your online campus and provide a single, user-friendly Web-based interface. A complete, enterprise-level package, Academus Portal is an ideal solution whether you're developing a new campus portal from the ground up or replacing an existing portal. For more information reference the Unicon Website.
Accessibility	A concept of universal access to the content of a Web document regardless of user disability, language, culture, or user agent (browser). For more information, reference the Web Accessibility Initiative (WAI) from the W3C.
ADA	The Americans with Disabilities Act (ADA) was passed on July 26, 1990, as Public Law. It became effective from January 26, 1992. The purpose of the law is to prohibit discrimination on the basis of disability. The Americans with Disabilities Act provided a clear and comprehensive mandate for eliminating discrimination against individuals with disabilities. The Americans with Disabilities Act was strengthened by certain amendments to the Rehabilitation Act of 1973. These amendments were made to Section 508 and Section 504 to include provisions for organizations to comply with the Americans with Disabilities Act. In 1998, the Congress amended Section 508 of the Rehabilitation Act of 1973 and created more stringent rules for access to information in the Federal sector. It mandated that federal agencies should purchase only accessible software and hardware and that federal Web sites be made as accessible as possible. For more information, reference the Department of Justice Section 508 Website and The Access Board.
Cascading Style Sheets (CSS)	A simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents. For more information, reference the W3C CSS Web page.
ECMAScript	This ECMA Standard is based on several originating technologies, the most well known being JavaScript (Netscape) and JScript (Microsoft). The language was invented by Brendan Eich at Netscape and first appeared in that company's Navigator 2.1 browser. It has appeared in all subsequent browsers from Netscape and in all browsers from Microsoft starting with Internet Explorer 3.0. For more information, reference the ECMAScript Website.
HyperText Markup Language (HTML)	HyperText Markup Language (HTML) is an SGML DTD. In practical terms, HTML is a collection of platform-independent styles (indicated by markup tags) that define the various components of a World Wide Web document. HTML was invented by Tim Berners-Lee while at CERN, the European Laboratory for Particle Physics in Geneva. HTML is currently in its fourth version (4.0). For more information, reference the W3C HTML Web page.
Theme	Specific to Academus Portal, this refers to the processing of portal system data into a user interface. The Theme is the interaction layer of the portal, and produces the structure and layout of Academus Portal user interfaces. For more information, reference Academus Portal Rendering Process: Theme Transform.
Theme Variable	Specific to Academus Portal, this refers to Theme-related variables that can be customized by the client institution. Modifying the variable values produces physical changes in the content and structure of the resultant HTML document. Theme Variables were designed to give the client institution a flexible means of customizing Academus Portal with institution-specific branding and information. For more information, reference Theme Variables.
Skin	Specific to Academus Portal, Skin refers to the visual presentation layer applied to the Theme. A skin does not alter document content or markup, but specifies content presentation options like images, fonts, colors, and spacing. For more information, reference Academus Portal Rendering Process.
Skin Package	Specific to Academus Portal, this refers to the collection of directories and files that comprise a Skin. For more information, reference Skin Files.

Term	Definition
Skin Registry	Specific to Academus Portal, this refers to the XML file that registers a Skin in the portal and makes it available for use by portal users. For more information, reference Skin Registry.
uPortal	uPortal is an open-source portal under development by institutions of higher-education. It is a collaborative development project with the effort shared among several JA-SIG member institutions. For more information, reference the uPortal Web site.
Usability	Usability is the measure of the quality of a user's experience when interacting with a product or system — whether a Web site, a software application, mobile technology, or any user-operated device. Usability is a combination of factors that affect the user's experience with the product or system, including: ease of learning, efficiency of use, memorability, error frequency and severity, and subjective satisfaction. For more information, reference Usability.gov and useit.com.
The World Wide Web Consortium (W3C)	The World Wide Web Consortium (W3C) was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C has over 350 Member organizations from all over the world and has earned international recognition for its contributions to the growth of the Web. For more information, reference the W3C Web site.
Web Standard	Established technologies, guidelines, and specifications for creating and interpreting web-based content produced by Web standards bodies. These technologies are carefully designed to deliver the greatest benefits to the greatest number of Web users while ensuring the long-term viability of any document published on the Web. Designing and building with these standards simplifies and lowers the cost of production, while delivering sites that are accessible to more people and more types of Internet devices. For more information, reference the World Wide Web Consortium (W3C) and the Web Standards Project (WASP).
Extensible HyperText Markup Language (XHTML)	Extensible HyperText Markup Language (XHTML) 1.0 is the first major change to HTML since HTML 4.0 was released in 1997. It brings the rigor of XML to Web pages and is the keystone in W3C's work to create standards that provide richer Web pages on an ever increasing range of browser platforms including cell phones, televisions, cars, wallet sized wireless communicators, kiosks, and desktops. For more information, reference the W3C XHTML Web page.
Extensible Markup Language (XML)	Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. For more information, reference the W3C XML Web page.
Extensible Stylesheet Language (XSL)	Extensible Stylesheet Language (XSL) is a family of recommendations for defining XML document transformation and presentation. It consists of three parts: XSL Transformations (XSLT), XML Path Language (XPath), and XSL Formatting Objects (XSL-FO). An XSLT stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses a formatting vocabulary, such as (X)HTML or XSL-FO. For more information, reference the W3C XSL Web page.