Voyager® 7.2
WebVoyáge Architecture Overview
and Configuration Models

July 2010
## Contents

### About This Document
- Purpose  
- Intended Audience  
- Reasons for Reissue  
- Document Summary  
- Conventions Used in This Document  
- Document Reproduction/Photocopying  
- Comment on This Document

### 1 Getting Started
- Purpose of this Chapter  
- Prerequisite Skills and Knowledge  
- Before You Begin

### 2 Architecture
- WebVoyage Architecture Overview  
- Flowchart Example - myAccount Page  
  - Flowchart Example Description  
  - .css Processing  
- Page Components Example - Basic Search  
- Page Components Example - Advanced Search  
- Page Components Example - Subject Search  
- Page Components Example - Author Search  
- Page Components Example - Course Reserves  
- Page Components Example - Geospatial Search
Contents

3 Search
• Search Overview 3-1
• Search Types 3-1
  Configuration 3-2
  Keyword 3-6
  Left-Anchored/Headings-Browse 3-7
• Limits 3-8
• LOCAL/Remote Database Search 3-11
• Display Records 3-14

4 Linking/OpenURL
• Overview 4-1
• OpenURL Standard 4-1
• OpenURL Requests 4-2
• OpenURL/LinkResolver Configuration 4-3

5 Bibliographic Record Linking
• Overview 5-1
• Defining Record Relationships 5-2
  Displaying Related Records 5-2
  Maintaining Related Records 5-2
• Configuring Voyager for Bibliographic Record Linking 5-3
  System Administration 5-3
  Cataloging 5-3
  WebVoyage 5-4

6 Display Codes
• displaycfg.xml 6-1
• displayHoldings.xml 6-2
• Enable Redirect 6-3
• display.xsl 6-3
Contents

7 How Do I Build A Separate Display For Serials?
• Description For “How Do I Build A Separate Display For Serials?” Example 7-1
• Files 7-1
• Instructions 7-1

8 How Do I Add Static Links To The Header Or Footer?
• Description For “How Do I Add Static Links To The Header Or Footer?” Example 8-1
• Files 8-1
• Instructions 8-2

9 How Do I Remove Information From A Page?
• Description For “How Do I Remove Information From A Page?” Example 9-1
• Files 9-1
• Instructions 9-1

10 How Do I Add A Map Or Other Information To A Location?
• Description For “How Do I Add A Map Or Other Information To A Location?” Example 10-1
• Files 10-1
• Instructions 10-1

11 How Do I Create An External Search From A Bibliographic Record Display?
• Description For “How Do I Create An External Search From A Bibliographic Record Display?” Example 11-1
• Files 11-1
Contents

- Instructions 11-2

12 How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?
  - Description For "How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?" Example12-1
  - Files 12-1
  - Instructions 12-2

13 How Do I Disable AutoComplete?
  - Description For “How Do I Disable AutoComplete?” Example13-1
  - Files 13-1
  - Instructions 13-1

14 How Do I Display A Favicon?
  - Description For “How Do I Display A Favicon?” Example14-1
  - Files 14-1
  - Instructions 14-2

15 How Do I Hide Limits On The Advanced Search Page?
  - Description For “How Do I Hide Limits On The Advanced Search Page?” Example 15-1
  - Files 15-1
  - Instructions 15-1

16 How Do I Build And Display A Persistent Link To A Bibliographic Record?
  - Description For “How Do I Build And Display A Persistent Link To A Bib-
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>How Do I Change The Format Of The Record Display Page?</td>
<td>16-1, 17-1, 17-2</td>
</tr>
<tr>
<td>18</td>
<td>How Do I Add Tracking Codes?</td>
<td>18-1, 18-2</td>
</tr>
<tr>
<td>20</td>
<td>How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?</td>
<td>20-1</td>
</tr>
</tbody>
</table>
Contents

- Syndetic Solutions Implementation 20-2
  - pageProperties.xml 20-2
  - resultsFacets.xsl 20-3
  - resultsTitles.xsl 20-4
  - imageUtils.js 20-5
  - displaycfg.xml 20-6
  - display.xsl 20-6
  - displayRecord.xsl 20-7
- Syndetic Solutions Information 20-7
  - What is a Query String? 20-7
  - Sample URLs 20-8

21 How Do I Implement Geospatial Search?

- Description For “How Do I Implement Geospatial Search?” 21-1
- Files 21-1
- Instructions 21-2

22 How Do I Enable External Authentication?

- Description For “How Do I Enable External Authentication?” 22-1
- Files 22-1
- Instructions 22-1

23 How Do I Modify Page Messages?

- Description For “How Do I Modify Page Messages?” 23-1
- Files 23-2
- Instructions 23-2

24 How Do I Remove the Course Reserve Tab?

- Description For “How Do I Remove the Course Reserve Tab?” Example 24-1
### Table of Contents

#### Contents

- Files 24-1
- Instructions 24-1

25 How Do I Add A New Search Tab?
- Description For “How Do I Add A New Search Tab?” Example 25-1
- Files 25-1
- Instructions 25-1

26 How Do I Add A New Header Tab?
- Description For “How Do I Add A New Header Tab?” Example 26-1
- Files 26-1
- Instructions 26-1

27 How Do I Create Additional Record Views?
- Description For “How Do I Create Additional Record Views?” Example 27-1
- Files 27-1
- Instructions 27-1

28 How Do I Implement DOI and URN Handling?
- DOI/URN Overview 28-1
- Files 28-1
- DOI/URN Implementation 28-1
  - webvoyage.properties 28-2

29 How Do I Implement Hook to Holdings (Citation Server)?
- Hook to Holdings Implementation 29-1
Contents

30 How Do I Implement HTTP Post to Link Resolver?
  • HTTP POST to Link Resolver Overview 30-1
  • Files 30-1
  • HTTP POST to Link Resolver Implementation 30-1
    voyager.ini 30-2
    linkresolver.properties 30-4
    OpenURL Standard 30-6

31 How Do I Display Media Bookings in MyAccount?
  • Media Bookings Overview 31-1
  • Files 31-1
  • Media Bookings Implementation 31-2

32 How Do I Implement ImageServer in WebVoyâge?
  • WebVoyâge ImageServer Overview 32-1
  • Files 32-1
  • ImageServer Implementation 32-2
    display.xsl 32-3
    resultsFacets.xsl 32-3

33 How Do I Implement Messages for Status Patron Groups?
  • Overview of Messages for Status Patron Groups 33-1
  • Files 33-2
  • Instructions 33-2

34 How Do I Add/Modify Search Results Page Icons?
  • Add/Modify Search Results Page Icons Overview 34-1
  • Files 34-1
Contents

• Instructions 34-1

35 How Do I Modify the Renewal Status Messages?

• Modify the Renewal Status Messages Overview 35-1
• Files 35-1
• Instructions 35-3

IN Index IN-1
Contents
About This Document

Purpose

The purpose of WebVoyáge Architecture Overview and Configuration Models is to describe WebVoyáge files, their relationship, and configuration options by example.

Given the programming design used for the new user interface, there is considerable flexibility in customizing the online public access catalog (OPAC) to your preferences. Key to this customization is experience with coding cascading style sheets (CSS), XSL, XML, and/or JavaScript.

This guide implements a learn-by-example format. As a result, WebVoyáge Architecture Overview and Configuration Models incorporates several chapters of specific examples and “how to” instructions. Optionally, you may copy/paste examples as you choose.

⚠️ CAUTION:
Examples provided in this guide may require additional editing to meet your site-specific requirements. If you copy/paste code from this guide, be aware that long lines of code that wrap within the left/right edges of the illustration may contain line breaks resulting from the PDF build that you need to remove for successful processing.
Intended Audience

This document is intended for programmers who are customizing WebVoyáge in CSS, XSL, XML, and/or JavaScript.

Reasons for Reissue

This guide incorporates and is being reissued for the following reasons:

- Update to Limits on page 3-8 resulting from Pivotal item #16384-1969.
- Addition of Figure 3-7.

Document Summary

Chapter 1  “Getting Started”  Chapter 1 describes the prerequisites for working with and customizing Voyager WebVoyáge 7.x.

Chapter 2  “Architecture”  Chapter 2 provides an overview of the WebVoyáge architecture.

Chapter 3  “Search”  Chapter 3 describes search capabilities in Voyager WebVoyáge 7.x.

Chapter 4  “Linking/OpenURL”  Chapter 4 describes the OpenURL capability available in Voyager WebVoyáge 7.x.

Chapter 5  “Bibliographic Record Linking”  Chapter 5 describes bibliographic record linking available in Voyager WebVoyáge 7.x.

Chapter 6  “Display Codes”  Chapter 6 describes display codes used in Voyager WebVoyáge 7.x.

Chapter 7  “How Do I Build A Separate Display For Serials?”  Chapter 7 describes how to build a separate display for serials.

Chapter 8  “How Do I Add Static Links To The Header Or Footer?”  Chapter 8 describes how to add static links to the header/footer area.

Chapter 9  “How Do I Remove Information From A Page?”  Chapter 9 describes how to remove information from a page.
Chapter 10 “How Do I Add A Map Or Other Information To A Location?”
Chapter 10 describes how to add a map or other information to a location.

Chapter 11 “How Do I Create An External Search From A Bibliographic Record Display?”
Chapter 11 describes how to create an external search.

Chapter 12 “How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?”
Chapter 12 describes how to dynamically disable limits and change search tips based on the selected search index.

Chapter 13 “How Do I Disable AutoComplete?”
Chapter 13 describes how to disable AutoComplete.

Chapter 14 “How Do I Display A Favicon?”
Chapter 14 describes how to create an icon for a browser tab or title bar display.

Chapter 15 “How Do I Hide Limits On The Advanced Search Page?”
Chapter 15 describes how to hide the limit options on an advanced search while, optionally, a user may click to see them.

Chapter 16 “How Do I Build And Display A Persistent Link To A Bibliographic Record?”
Chapter 16 describes how to dynamically build and display a persistent link to a bibliographic record.

Chapter 17 “How Do I Change The Format Of The Record Display Page?”
Chapter 17 describes how to add class attributes to the Detailed Display Page for improved formatting control.

Chapter 18 “How Do I Add Tracking Codes?”
Chapter 18 describes how to add tracking codes.

Chapter 19 “How Do I Implement Google Book Search?”
Chapter 19 provides instructions regarding the implementation of Google Book Search.

Chapter 20 “How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?”
Chapter 20 provides instructions for implementing Syndetics enhancements.

Chapter 21 “How Do I Implement Geospatial Search?”
Chapter 21 provides instructions for implementing geospatial search.

Chapter 22 “How Do I Enable External Authentication?”
Chapter 22 provides instructions for enabling external authentication.
Chapter 23  “How Do I Modify Page Messages?”
Chapter 23 provides instructions for modifying page messages.

Chapter 24  “How Do I Remove the Course Reserve Tab?”
Chapter 24 provides instructions for removing the Course Reserves tab.

Chapter 25  “How Do I Add A New Search Tab?”
Chapter 25 provides instructions for adding a new search tab.

Chapter 26  “How Do I Add A New Header Tab?”
Chapter 26 provides instructions for creating a new header tab.

Chapter 27  “How Do I Create Additional Record Views?”
Chapter 27 provides instructions for creating additional record views such as brief and full.

Chapter 28  “How Do I Implement DOI and URN Handling?”
Chapter 28 provides instructions for DOI/URN handling.

Chapter 29  “How Do I Implement Hook to Holdings (Citation Server)?”
Chapter 29 provides instructions for implementing hook to holdings.

Chapter 30  “How Do I Implement HTTP Post to Link Resolver?”
Chapter 30 provides instructions for implementing HTTP POST to link resolver.

Chapter 31  “How Do I Display Media Bookings in MyAccount?”
Chapter 31 provides instructions for implementing media bookings.

Chapter 32  “How Do I Implement ImageServer in WebVoyage?”
Chapter 32 provides instructions for implementing ImageServer.

Chapter 33  “How Do I Implement Messages for Status Patron Groups?”
Chapter 33 provides instructions for implementing messages for Status Patron Groups.

Chapter 34  “How Do I Add/Modify Search Results Page Icons?”
Chapter 34 provides instructions for implementing icons on the search results page.

Chapter 35  “How Do I Modify the Renewal Status Messages?”
Chapter 35 provides instructions for how to modify renewal status messages.

Index  The Index is an alphabetical, detailed cross-reference of topics.
Conventions Used in This Document

The following conventions are used throughout this document:

• Names of commands, variables, stanzas, files, and paths (such as /dev/tmp), as well as selectors and typed user input, are displayed in constant width type.

• Commands or other keyboard input that must be typed exactly as presented are displayed in constant width bold type.

• Commands or other keyboard input that must be supplied by the user are displayed in constant width bold italic type.

• System-generated responses such as error messages are displayed in constant width type.

• Variable portions of system-generated responses are displayed in constant width italic type.

• Keyboard commands (such as Ctrl and Enter) are displayed in bold.

• Required keyboard input such as “Enter vi” is displayed in constant width bold type.

• Place holders for variable portions of user-defined input such as ls -l filename are displayed in italicized constant width bold type.

• The names of menus or status display pages and required selections from menus or status display pages such as “From the Applications drop-down menu, select System-wide,” are displayed in bold type.

• Object names on a window’s interface, such as the Description field, the OK button, and the Metadata tab, are displayed in bold type.

• The titles of documents such as Acquisitions User’s Guide are displayed in italic type.

• Caution, and important notices are displayed with a distinctive label such as the following:

NOTE:
Extra information pertinent to the topic.

IMPORTANT:
Information you should consider before making a decision or configuration.

CAUTION:
Information you must consider before making a decision, due to potential loss of data or system malfunction involved.
TIP:
Helpful hints you might want to consider before making a decision.

RECOMMENDED:
Preferred course of action.

OPTIONAL:
Indicates course of action which is not required, but may be taken to suit your library’s preferences or requirements.

Document Reproduction/Photocopying
Photocopying the documentation is allowed under your contract with Ex Libris (USA) Inc. It is stated below:

All documentation is subject to U.S. copyright protection. CUSTOMER may copy the printed documentation only in reasonable quantities to aid the employees in their use of the SOFTWARE. Limited portions of documentation, relating only to the public access catalog, may be copied for use in patron instruction.

Comment on This Document
To provide feedback regarding this document, use the Ex Libris eService or send your comments in an e-mail message to docmanager@exlibrisgroup.com.
Getting Started

Purpose of this Chapter 1-1
Prerequisite Skills and Knowledge 1-1
Before You Begin 1-2
Getting Started

Purpose of this Chapter

The purpose of this chapter is to describe what you need to get started to effectively work with/customize WebVoyage and use this guide.

Prerequisite Skills and Knowledge

To use this document effectively, you should have a working knowledge of the following:

- Microsoft Windows operating environment.
- CSS.
- XML.
- XSL.
- JavaScript.
- Text editor(s) for working with CSS, XML, XSL, and so on.
- UNIX operating system commands and file system (depending on your environment).
- Basic MARC records formats.
- Local procedures.
Before You Begin

Before you can begin, you need to do the following:

• Have the Voyager WebVoyáge 7 (or higher) and corresponding Voyager integrated library system software installed.

• Have access to an internet browser on your PC.
  
  Refer to the Ex Libris Documentation Center for Microsoft® and Mozilla browser support information.

• Obtain the URL and/or the IP and port address for accessing your instance of Voyager WebVoyáge 7 (or higher).

• Obtain your user ID and password for logging in to Voyager WebVoyáge 7 (or higher). For logon steps, refer to the WebVoyáge Basic User’s Guide.

• Set up your PC to display Unicode-specific data as needed. See the WebVoyáge Basic User’s Guide for instructions.
WebVoyáge Architecture Overview

The purpose of this section is to provide an overview description of the architecture of WebVoyáge for displaying information.

WebVoyáge has a modular design to control formatting for the broadest number of page displays.

To display search results, patron information, and other dynamically generated information, WebVoyáge combines information from the Voyager database (or from an outside resource like Google™ Book Search) with formatting properties from the WebVoyáge .css, .xsl, and .xml files to render a display page in HTML. See Figure 2-1 on page 2-2.

For a flowchart example of how these files work together, see Flowchart Example - myAccount Page on page 2-2.

For an example and description of the HTML page components, see Page Components Example - Basic Search on page 2-5.
Flowchart Example - myAccount Page

The purpose of this section is to describe and illustrate the relationship of the WebVoyage files used for building and displaying content using the myAccount page as an example.

Flowchart Example Description

The content of this section describes the the flowchart example highlighted in Figure 2-2 on page 2-4.

To build the display page for myAccount page, the primary .xsl file is used. For this example, it is the myAccount.xsl file that is used. The name of the .xsl file used for this process generally represents the page being built such as displayRecord.xsl. See the .xsl files located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name
that is used at your site. The default skin path provided is en_US as in the following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

The myAccount.xsl imports the following:

- stdImports.xsl.
  This is an important part of the page processing. Every page uses it. It imports frameWork.xsl.
- cl_myAccount.xsl.
  This is a key component. The cl stands for content layout.
- myAccountLinks.xsl.
- myAccount.css.

NOTE:
The file naming convention used is intentional to show relationships between files used to build the HTML page as with myAccount.xsl, myAccountLinks.xsl, cl_myAccount.xsl, and myAccount.css.

The myAccount.xsl calls the following templates which are used to construct every page:

- buildHtmlPage.
- buildContent.

The frameWork.xsl takes input from the following components:

- .xml files.
- .js files.
- .css files.
- buildHtmlPage template.
- Timeout mechanism.

And subsequently, the frameWork.xsl generates HTML output with the following page components:

- Header.
- Main content based on the buildContent template.
- Footer.

For more information regarding page components, see Page Components Example - Basic Search on page 2-5.
WebVoyage Architecture Overview and Configuration Models

Figure 2-2. Flowchart example for myAccount page

**.css Processing**

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document. CSS is commonly used to apply style to web pages or more specifically colors, fonts, layout, and other aspects of document presentation. These are processed in a specified priority scheme or hierarchy that determines which style rules apply. As a result, pieces of one stylesheet may be overridden by another stylesheet.
In WebVoyage, the baseline defaults are provided by the following .css files that are used by almost every page that is generated:

1. pageProperties.css.
2. quickSearchBar.css.
3. header.css.
4. frameWork.css.

This list illustrates the order of precedence for each of these .css files. In this hierarchy, pageProperties.css provides overriding characteristics to #2 through #4. Specific page .css files like myAccount.css always override the baseline defaults.

Variations in a specific page .css file only apply to that page. For more global changes like a font change for all pages, the files controlling the baseline defaults need to be modified.

TIP:
A tool like Mozilla® Firebug enables you to view these variations/interactions and edit/debug CSS, JavaScript, and so on live in any web page.

Page Components Example - Basic Search

As described, each page includes the following major components:

• Header
• Footer
• Main content

See Figure 2-3.
Any element of WebVoyáge that is used by multiple pages such as the footer component, the search navigation bar, or the login link is defined independent of the pages on which it appears. An independent definition allows the element to be called by any and every page. This provides greater flexibility; but as a result, there isn’t a single .xsl stylesheet for each page that WebVoyáge renders.

The page shown in Figure 2-3 is built with the following files that are used by all pages.

- frameWork.xsl.
- constants.xsl.
- tools.xsl.
- formInput.xsl.
- constantStrings.xsl.
- frameWork.css.
- header.css.
- quickSearchBar.css.
- pageProperties.css.

In addition, the Basic Search page uses the following:

- cl_searchBasic.xsl.
• searchFacets.xsl.
• searchPages.css.
• searchBasic.css.
• pageInputFocus.js.

Specific to the Basic tab, it uses the following:
• The font family is from frameWork.css.
• The font size, color, weight, and alignment of the tab label are from the searchPages.css.
• The font size, color, weight, and alignment of the tab contents such as Limit To are from frameWork.css.
• The font family, size, color, weight, and alignment of the search tips are from pageProperties.css.
• The text values are from webvoyage.properties.
• The images that make up the Basic tab are from the /m1/voyager/xxxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/directory.
• The cursor placement is determined by pageInputFocus.js.

In summary, this example illustrates the hierarchy described in Flowchart Example - myAccount Page on page 2-2 where baseline defaults are used in combination with page-specific, overriding controls for formatting that includes the following:
• Fonts
• Color
• Images
• Content placement on the page
• And so on

Page Components Example - Advanced Search

As described, each page includes the following major components:
• Header
• Footer
• Main content
Any element of WebVoyáge that is used by multiple pages such as the footer component, the search navigation bar, or the login link is defined independent of the pages on which it appears. An independent definition allows the element to be called by any and every page. This provides greater flexibility; but as a result, there isn’t a single .xsl stylesheet for each page that WebVoyáge renders.

The Advanced Search page is built with the following files that are used by all pages.

- frameWork.xsl
- constants.xsl
- tools.xsl
- formInput.xsl
- constantStrings.xsl
- frameWork.css
- header.css
- quickSearchBar.css
- pageProperties.css

In addition, the Advanced Search page uses the following:

- cl_searchAdvanced.xsl
- searchFacets.xsl
- searchPages.css
- searchAdvanced.css
- pageInputFocus.js

Specific to the Advanced tab, it uses the following:

- The font family is from frameWork.css.
- The font size, color, weight, and alignment of the tab label are from the searchPages.css.
- The font size, color, weight, and alignment of the tab contents such as Limit To are from frameWork.css.
- The font family, size, color, weight, and alignment of the search tips are from pageProperties.css.
- The text values are from webvoyage.properties.
- The images that make up the Advanced tab are from the /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/directory.
• The cursor placement is determined by pageInputFocus.js.

In summary, this example illustrates the hierarchy described in Flowchart Example - myAccount Page on page 2-2 where baseline defaults are used in combination with page-specific, overriding controls for formatting that includes the following:

• Fonts
• Color
• Images
• Content placement on the page
• And so on

Page Components Example - Subject Search

As described, each page includes the following major components:

• Header
• Footer
• Main content

Any element of WebVoyage that is used by multiple pages such as the footer component, the search navigation bar, or the login link is defined independent of the pages on which it appears. An independent definition allows the element to be called by any and every page. This provides greater flexibility; but as a result, there isn’t a single .xsl stylesheet for each page that WebVoyage renders.

The Subject Search page is built with the following files that are used by all pages.

• frameWork.xsl.
• constants.xsl.
• tools.xsl.
• formInput.xsl.
• constantStrings.xsl.
• frameWork.css.
• header.css.
• quickSearchBar.css.
• pageProperties.css.
In addition, the Subject Search page uses the following:

- `cl_searchSubject.xsl`.
- `searchFacets.xsl`.
- `searchPages.css`.
- `searchSubject.css`.
- `pageInputFocus.js`.

Specific to the Subject tab, it uses the following:

- The font family is from `frameWork.css`.
- The font size, color, weight, and alignment of the tab label are from the `searchPages.css`.
- The font size, color, weight, and alignment of the tab contents such as Limit To are from `frameWork.css`.
- The font family, size, color, weight, and alignment of the search tips are from `pageProperties.css`.
- The text values are from `webvoyage.properties`.
- The images that make up the Subject tab are from the `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/` directory.
- The cursor placement is determined by `pageInputFocus.js`.

In summary, this example illustrates the hierarchy described in Flowchart Example - myAccount Page on page 2-2 where baseline defaults are used in combination with page-specific, overriding controls for formatting that includes the following:

- Fonts
- Color
- Images
- Content placement on the page
- And so on

Page Components Example - Author Search

As described, each page includes the following major components:

- Header
Any element of WebVoyáge that is used by multiple pages such as the footer component, the search navigation bar, or the login link is defined independent of the pages on which it appears. An independent definition allows the element to be called by any and every page. This provides greater flexibility; but as a result, there isn’t a single .xsl stylesheet for each page that WebVoyáge renders.

The Author Search page is built with the following files that are used by all pages.

- frameWork.xsl
- constants.xsl
- tools.xsl
- formInput.xsl
- constantStrings.xsl
- frameWork.css
- header.css
- quickSearchBar.css
- pageProperties.css

In addition, the Author Search page uses the following:

- cl_searchAuthor.xsl
- searchFacets.xsl
- searchPages.css
- searchAuthor.css
- pageInputFocus.js

Specific to the Author tab, it uses the following:

- The font family is from frameWork.css.
- The font size, color, weight, and alignment of the tab label are from the searchPages.css.
- The font size, color, weight, and alignment of the tab contents such as Limit To are from frameWork.css.
- The font family, size, color, weight, and alignment of the search tips are from pageProperties.css.
- The text values are from webvoyage.properties.
• The images that make up the Author tab are from the /m1/voyager/
  xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/
directory.

• The cursor placement is determined by pageInputFocus.js.

In summary, this example illustrates the hierarchy described in Flowchart
Example - myAccount Page on page 2-2 where baseline defaults are used in
combination with page-specific, overriding controls for formatting that includes the
following:

• Fonts
• Color
• Images
• Content placement on the page
• And so on

Page Components Example - Course
Reserves

As described, each page includes the following major components:

• Header
• Footer
• Main content

Any element of WebVoyáge that is used by multiple pages such as the footer
component, the search navigation bar, or the login link is defined independent of
the pages on which it appears. An independent definition allows the element to be
called by any and every page. This provides greater flexibility; but as a result,
there isn’t a single .xsl stylesheet for each page that WebVoyáge renders.

The Course Reserves page is built with the following files that are used by all
pages.

• frameWork.xsl.
• constants.xsl.
• tools.xsl.
• formInput.xsl.
• constantStrings.xsl.
• frameWork.css.
• header.css.
• quickSearchBar.css.
• pageProperties.css.

In addition, the Course Reserves page uses the following:
• cl_searchCourseReserves.xsl.
• searchFacets.xsl.
• searchPages.css.
• searchCourseReserve.css.
• pageInputFocus.js.

Specific to the Course Reserve tab, it uses the following:
• The font family is from frameWork.css.
• The font size, color, weight, and alignment of the tab label are from the searchPages.css.
• The font size, color, weight, and alignment of the tab contents such as Limit To are from frameWork.css.
• The font family, size, color, weight, and alignment of the search tips are from pageProperties.css.
• The text values are from webvoyage.properties.
• The images that make up the Course Reserve tab are from the /m1/voyager/xxxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/directory.
• The cursor placement is determined by pageInputFocus.js.

In summary, this example illustrates the hierarchy described in Flowchart Example - myAccount Page on page 2-2 where baseline defaults are used in combination with page-specific, overriding controls for formatting that includes the following:
• Fonts
• Color
• Images
• Content placement on the page
• And so on
Page Components Example - Geospatial Search

The Geospatial Search feature is only available if your institution has purchased the Geospatial searching tools.

As described, each page includes the following major components:

- Header
- Footer
- Main content

Any element of WebVoyáge that is used by multiple pages such as the footer component, the search navigation bar, or the login link is defined independent of the pages on which it appears. An independent definition allows the element to be called by any and every page. This provides greater flexibility; but as a result, there isn’t a single .xsl stylesheet for each page that WebVoyáge renders.

The Geospatial Search page is built with the following files that are used by all pages.

- frameWork.xsl
- constants.xsl
- tools.xsl
- formInput.xsl
- constantStrings.xsl
- frameWork.css
- header.css
- quickSearchBar.css
- pageProperties.css

In addition, the Geospatial Search page uses the following:

- cl_searchGeoCorridor.xsl
- cl_searchGeoPolygon.xsl
- cl_searchGeoRadius.xsl
- cl_searchGeoRange.xsl
- cl_searchGeoRectangle
- searchFacets.xsl
- searchPages.css
Specific to the Geospatial Search tab, it uses the following:

- The font family is from `framWork.css`.
- The font size, color, weight, and alignment of the tab label are from the `searchPages.css`.
- The font size, color, weight, and alignment of the tab contents such as Limit To are from `framWork.css`.
- The font family, size, color, weight, and alignment of the search tips are from `pageProperties.css`.
- The text values are from `webvoyage.properties`.
- The images that make up the Geospatial Search tab are from the `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/` directory.
- The cursor placement is determined by `pageInputFocus.js`.

In summary, this example illustrates the hierarchy described in **Flowchart Example - myAccount Page** on page 2-2 where baseline defaults are used in combination with page-specific, overriding controls for formatting that includes the following:

- Fonts
- Color
- Images
- Content placement on the page
- And so on
Search

Search Overview 3-1
Search Types 3-1
• Configuration 3-2
• Keyword 3-6
• Left-Anchored/Headings-Browse 3-7
Limits 3-8
LOCAL/Remote Database Search 3-11
Display Records 3-14
Search

Search Overview

With WebVoyage 7.0 and higher, it is important to note that the foundation underlying WebVoyage in the Voyager system has not changed. The database structure, indexes, integrated system administration configuration, and so on remain the same.

This chapter is intended to describe Voyager search characteristics. You may find that some of this information is familiar to you if you’ve had experience with WebVoyage Classic. However, for user’s just starting with WebVoyage 7.0 and higher, these may be completely new concepts.

⚠️ IMPORTANT:
Also, remember to use the WebVoyage Basic User’s Guide for key basic concepts regarding the WebVoyage search interface.

Search Types

Different search types enable you to access the various indexes provided with your Voyager system. The search types are:

- keyword (see Keyword on page 3-6)
- left-anchored (see Left-Anchored/Headings-Browse on page 3-7)
- headings browse (see Left-Anchored/Headings-Browse on page 3-7)
The Basic Search page uses all search types, the Advanced Search page uses keyword, and the Subject and Author Search pages use one of the types based on what is configured.

To increase your understanding of the relationship between the search types and the available indexes, you may find it helpful to refer to the "Search Definition Tables" Appendix in the Voyager System Administration User’s Guide that contains the following sections:

- Headings Indexes
- Keyword Indexes
- Left-Anchored Indexes

Configuration

The WebVoyage search function utilizes the indexes definitions set through the Search component of the Voyager System Administration module. See Figure 3-1.
The search codes for the WebVoyage search pages are set in the `webvoyage.properties` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/` following a similar pattern as shown in Figure 3-2.

<table>
<thead>
<tr>
<th>Basic Search:</th>
<th>page.search.basic.search.code=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Search:</td>
<td>page.search.subject.search.code=</td>
</tr>
<tr>
<td>Author Search:</td>
<td>page.search.author.search.code=</td>
</tr>
</tbody>
</table>

**Figure 3-2. Example line of code for specifying search code**

**NOTE:**
Since the Advanced Search page only uses keyword searches, it pulls the necessary search code information from the 1) Indexes - Keyword Definitions and 2) Indexes - Holdings Keyword Definitions configured in the Voyager System Administration module. As a result, there is no `page.search.<xxxx>.search.code=` line in the `webvoyage.properties` file for Advanced Search. The Advanced Search configuration options provided in the `webvoyage.properties` file allow you to override the display label (name) for the search codes as defined in the Voyager System Administration module.

Review the comments sections of the `webvoyage.properties` file for additional setup information. See Figure 3-3, Figure 3-4, Figure 3-5, and Figure 3-6 for examples of search code configurations for the search pages.
Figure 3-3. Example configuration for Basic Search page
# Search Index Definitions

Search Index definitions are used on the Advanced search page. This section allows the name of the index to be changed from what is defined in Sysadmin. The order of these entries is controlled by the database.

```plaintext
search.index.GKEY=
search.index.GKEY.display=Keyword Anywhere
search.index.TKEY=
search.index.TKEY.display=Title
search.index.SKEY=
search.index.SKEY.display=Subject
search.index.NKEY=
search.index.NKEY.display=Author Name
search.index.100A=
search.index.100A.display=Personal Name
search.index.260B=
search.index.260B.display=Publisher: Name
search.index.NAUT=
search.index.NAUT.display=Name/Uniform Title
search.index.260C=
search.index.260C.display=Publisher: Date
search.index.ISSN=
search.index.ISSN.display=ISSN
search.index.DEWD=
search.index.DEWD.display=Dewey Call Number
search.index.ISBN=
search.index.ISBN.display=ISBN
search.index.SERI=
search.index.SERI.display=Series
search.index.AUTI=
search.index.AUTI.display=Author/Title
```

Figure 3-4. Example configuration for Advanced Search page (overrides)
Keyword

A keyword search does a search for specified words anywhere within a record. This search type is based on the keyword indexes. For example, Table A-12 in the appendix of the Voyager System Administration User’s Guide defines the keyword search (GKEY) as searching all 010 through 9xx fields and subfields.

The results of a keyword search display on a Titles list page. WebVoyage works in combination with the Voyager System Administration configuration specified by the system administrator to identify what fields such as title, author, and date display and in which order they display for a specific record. The Voyager System Administration configuration also identifies the sort order of the records when there is more than one.

Other characteristics of a keyword search include the following:

• Enter multiple search terms separated by a space.
• AND is the default boolean operator between search terms.
• Enclose terms in parentheses to group them together.
• Surround terms in quotation marks to search for them as a phrase.
• Use the question mark (?) as a wild card or truncation character.
  Examples:
  col?r finds color and colour
  medi? finds medicine, medical, and so on

  NOTE:
  Wild card characters do not work with Z39.50 searches.

• Use the percent sign (%) as a wild card for a specific character.
  Examples:
  g%lf finds golf and gulf
  g%%se finds geese and goose
• Use an exclamation point (!) before a term to indicate that records
  containing the term are not to be displayed.
• Use a plus sign (+) before a term to indicate that it is an essential term.
• Use the asterisk (*) before a term to indicate that it is important but less
  important than a term marked with a plus sign.

Left-Anchored/Headings-Browse

Left-anchored/headings browse searches look for the search terms only at the
beginning of the appropriate field (as configured).

A left-anchored search scans an index and returns every subject, author, title, call
number, or publication date that begins with what you typed. For example, a
headings search on the subject “medi” would return the topics medicaid, medical,
medicare, medici, medicine, medieval, and mediterranean.

The search results are displayed in a browseable Headings List for Subject (see
Figure 3-7) and Author searches. Journal Title and Call Number searches display
in a Title List.
WebVoyáge provides the options to configure search result limits for patrons to use when doing a search. The following broad categories of limits can be configured:

- Quick limits (Basic Search page)
- Advanced limits

You can configure multiple limits to be associated with one quick-limit, drop-down option by using a vertical bar between the valid limit types. For example, page.search.limitTo.text.limit=LANG=ENG|MEDI=v.

Limits are configured in the webvoyage.properties file that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/ (see the section that begins as shown in Figure 3-8), and in the limits.xml file that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/xsl/userTextConfigs/ (see Figure 3-9 for an example section).

Review the comments provided with these files for setup information.
Figure 3-8. Limits section in webvoyage.properties example

```xml
#----------------------------------------#
# Limit To
# ========
#----------------------------------------#
# To specify a default quick limit on the Basic Search page, list its name in \ # the page.search.limitTo option.
# If there is no default quick limit on the Basic Search page, type "none."

<limitGroup type="medium" label="Format:">
    <limit code="all" default="Y">All Formats</limit>
    <limit code="a">Map</limit>
    <limit code="c">Computer File</limit>
    <limit code="d">Globe</limit>
    <limit code="g">Projected Graphic</limit>
    <limit code="h">Microform</limit>
    <limit code="k">Nonprojected Graphic</limit>
    <limit code="m">Motion Picture</limit>
    <limit code="s">Sound Recording</limit>
    <limit code="t">Text (Eye-Readable)</limit>
    <limit code="v">Videorecording</limit>
</limitGroup>
```

Figure 3-9. Example from limits.xml

See Table 3-1 for a description of the limitGroup types (also identified in the webvoyage.properties file's comments).

Table 3-1. Limit Types

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANG</td>
<td>Language</td>
<td>Refer to limits.xml for a list of limit codes such as ENG for English.</td>
</tr>
</tbody>
</table>
Using the limit values that you select, WebVoyage performs search limits according to the following rules:

- The relationship between multiple limit values of the same limit types is considered to be OR. This means that selecting English and French returns only records that are in either English or French.
- The relationship between limit values of different limit types is considered to be AND. This means that selecting English (limitGroup type="language") and Book (limitGroup type="type") returns only records that are books in English.

### Table 3-1. Limit Types

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDI</td>
<td>Medium</td>
<td>This identifies the format such as video.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NOTE:</strong> MEDI is labeled “Format:” in the limits.xml file. See Figure 3-9.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to limits.xml for a list of limit codes such as v for video.</td>
</tr>
<tr>
<td>PLAC</td>
<td>Place</td>
<td>Refer to limits.xml for a list of limit codes such as hiu for Hawaii.</td>
</tr>
<tr>
<td>STAT</td>
<td>Status</td>
<td>Refer to limits.xml for a list of limit codes such as d for ceased publication.</td>
</tr>
<tr>
<td>TYPE</td>
<td>Item Type</td>
<td>Refer to limits.xml for a list of limit codes such as jm for musical recording.</td>
</tr>
<tr>
<td>DATE</td>
<td>Date</td>
<td>Enter dates in the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>DATE=-1990</code> (before 1990)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>DATE=1990-</code> (after 1990)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dates must use the 4-digit format YYYY.</td>
</tr>
<tr>
<td>LOCA</td>
<td>Location</td>
<td>The list of Location limits comes from the System &gt; Location Limit Groups &gt; [group] &gt; Name in the System Administration module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any limit groups in the list that do not have the Suppress in OPAC button pressed are available to select as limits.</td>
</tr>
</tbody>
</table>
LOCAL/Remote Database Search

Voyager provides the ability to connect to a wide variety of databases and electronic resources. Voyager WebVoyáge configuration defines connections to:

- Your LOCAL database
- Remote Voyager databases
- Z39.50-compliant databases
- Citation databases via Voyager or Z39.50

These connections are configured in the `webvoyage.properties` file that is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`. See the section that starts as shown in Figure 3-10.

Review the comments provided in this file for setup information.

```plaintext
#------------------------
#
# Connections
#
# ===========
# This section contains customizations for remote database connections.
#------------------------
#
# To create one or more groups of remote database connections, do the following:
# 1. Create an empty numbered group variable.
# 2. Provide a name for the group.
#
```

Figure 3-10. Example from Connections section of webvoyage.properties

For each connection, there are multiple lines of code that define:

- The database code (from Voyager System Administration)
- Name to display
- The display configuration file to reference
- The order in which to display the connections

These lines are repeated for each database connection. See Figure 3-11 for an example of the configuration code and Figure 3-12 for the display results. Notice also from these examples that two groups have been defined and how they display.
Figure 3-11. Example configuration code

```plaintext
connect.db.group1=
connect.db.group1.name=Local Libraries
connect.db.group1.order=1
connect.db.group1.AIXCURR.dbCode=AIXCURR
connect.db.group1.AIXCURR.name=Current AIX database
connect.db.group1.AIXCURR.config=holdingsInfo.vbib.properties
connect.db.group1.AIXCURR.order=1
connect.db.group1.QAFINNDB.dbCode=QAFINNDB
connect.db.group1.QAFINNDB.name=QAFINNDB
connect.db.group1.QAFINNDB.config=holdingsInfo.vbib.properties
connect.db.group1.QAFINNDB.order=2
connect.db.group1.KC70DB.dbCode=KC70DB
connect.db.group1.KC70DB.name=KC70DB on king-cobra
connect.db.group1.KC70DB.config=holdingsInfo.vbib.properties
connect.db.group1.KC70DB.order=3
connect.db.group1.LOCAL.dbCode=LOCAL
connect.db.group1.LOCAL.name=My Library Catalog
connect.db.group1.LOCAL.config=holdingsInfo.vbib.properties
connect.db.group1.LOCAL.order=4
connect.db.group2=
connect.db.group2.name=Academic Libraries
connect.db.group2.order=2
connect.db.group2.LOCZ.dbCode=LOCZ
connect.db.group2.LOCZ.name=Library of Congress (z3950)
connect.db.group2.LOCZ.config=holdingsInfo.zbib.properties
connect.db.group2.LOCZ.order=1
```
The user may select one or more databases to search by clicking the Change link from the search page. See Figure 3-13 for an example of the Change link.

Preceding the Change link are the active databases displayed after the Database: label. This label is defined in the Search Pages section of the webvoyage.properties file. See Figure 3-14.
Display Records

When a user selects to display a record’s detail, the system looks for data in the MARC bibliographic record, the MARC holdings record, the line item in the purchase order, and the item record. Whenever any of these records are present, the system displays data based on a variety of factors including:

- Definitions in the `displaycfg.xml` and `displayHoldings.xml` configuration files (located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/xsl/contentLayout/configs`)
- Available data in the MARC bibliographic record
- Available data in the linked MARC holdings record(s)
- Available data in the linked line item copy or copies of a purchase order(s)
- Available data in the linked item record(s)

However, before the system constructs a display from this data, it first considers whether or not records have been set for suppression from WebVoyage display. This is accomplished by the manual or automatic setting of the Suppress from OPAC values in either the MARC bibliographic record or the MARC holdings record or both.

Review the comments provided in the `displaycfg.xml` and `displayHoldings.xml` configuration files to use as a template/guidelines to configure these files. See Display Codes on page 6-1 for more information regarding available codes to use in these configuration files and how to enable redirect.

Figure 3-14. Database label example in webvoyage.properties
Also check Chapter 7, *How Do I Build A Separate Display For Serials?* on page 7-1 and Chapter 20, *How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?* on page 20-1 for other examples of using the configuration files to display records.
Linking/OpenURL

Overview 4-1
OpenURL Standard 4-1
OpenURL Requests 4-2
OpenURL/LinkResolver Configuration 4-3
Linking/OpenURL

Overview

WebVoyage uses the OpenURL standard (see OpenURL Standard on page 4-1) to generate links and obtain information from external sources. From the WebVoyage Record View page, Voyager can generate a URI in OpenURL format from information in the MARC record. When sent to an OpenURL-compliant linking server, the information in the URI is used to locate corresponding full-text records.

OpenURL Standard

OpenURL is a type of Uniform Resource Locator (URL) that contains resource metadata. It is a defined NISO standard described in the “ANSI/NISO Z39.88 - The OpenURL Framework for Context-Sensitive Services” document that is available at www.niso.org.

The OpenURL standard is designed to support mediated linking from sources (MARC databases) to library targets like on-line academic journals. A link resolver parses the elements of an OpenURL and provides links to appropriate targets such as full-text repositories and so on.
OpenURL Requests

OpenURL-formatted item requests can be initiated when a patron clicks the OpenURL request button (SFX, for example) from the Action Box of the Record View page. See Figure 4-1.

Figure 4-1. Action Box - OpenURL (SFX) request button example

Or an OpenURL-formatted item request can be initiated when a patron clicks the LinkResolver option as shown in Figure 4-2.

Figure 4-2. LinkResolver option example
OpenURL/LinkResolver Configuration

The following WebVoyâge files provide configuration options for OpenURL processing:

- holdingsInfo.vbib.properties (for Voyager bibliographic databases)
- holdingsInfo.vcit.properties (for Voyager citation databases)
- holdingsInfo.zbib.properties (for Z39.50 bibliographic databases)
- holdingsInfo.zcit.properties (for Z39.50 citation databases)
- linkresolver.properties (see Figure 4-2)

Open these files and review the comments provided for additional information regarding setup.

An example of configuring linkresolver.properties can be found in How Do I Implement HTTP Post to Link Resolver? on page 30-1.

In all the configuration files, you see the following variables repeated for each key:

- Field name
- Tag/field number
- Subfield
- Length 1 (number of positions)
- Parse start
- Parse end
- Length 2 (number of positions)

See Table 4-1 for a description of these variables.

Table 4-1. Variables Description

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Meta-tag for the field to parse</td>
<td>Enter meta-tag</td>
</tr>
<tr>
<td>tag</td>
<td>MARC field to match</td>
<td>Enter valid MARC field number</td>
</tr>
<tr>
<td>subfield</td>
<td>MARC subfield to match</td>
<td>Enter valid MARC subfield letter</td>
</tr>
</tbody>
</table>
### Table 4-3. Configuration Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>len1</td>
<td>Specify a fixed number of characters to extract from a field, or use this element in conjunction with parseStart to identify the starting point for extraction</td>
<td>Enter the number of characters or 0</td>
</tr>
<tr>
<td>parseStart</td>
<td>Text of field/subfield after which extraction is to begin.</td>
<td>Enter the text of the field/subfield after which to parse such as Issue: for example (see Figure 4-3)</td>
</tr>
<tr>
<td>parseEnd</td>
<td>Text of field/subfield at which extraction is to end</td>
<td>Enter the text of field/subfield at which to end the parse such as Date: (see Figure 4-3)</td>
</tr>
<tr>
<td>len2</td>
<td>Specify a fixed number of characters to extract from a field, or use this element in conjunction with parseEnd to identify the end point for extraction.</td>
<td>Enter the number of characters or 0</td>
</tr>
</tbody>
</table>

```java
#Issue=773/g/0/Issue:/Date:/0
actionBox.openUrl.cfg.LOCAL.key4.key=Issue
actionBox.openUrl.cfg.LOCAL.key4.tag=773
actionBox.openUrl.cfg.LOCAL.key4.subfield=g
actionBox.openUrl.cfg.LOCAL.key4.len1=0
actionBox.openUrl.cfg.LOCAL.key4.parseStart=Issue:
actionBox.openUrl.cfg.LOCAL.key4.parseEnd=Date:
actionBox.openUrl.cfg.LOCAL.key4.len2=0
```

**Figure 4-3. Example of OpenURL variables configuration**

(lookInfoInfo.vbib.properties)

The OpenURL request button such as SFX that displays on the Record View page is defined by the lines of code as shown in **Figure 4-4**.
For each database that can be accessed, you need to create another set of configuration specifications. Notice that each set begins with a definition of the database being referenced/turned on (Y/N) as shown in Figure 4-5.

In Figure 4-5, LOCAL is the database being referenced. Following the same format, simply replace LOCAL with the name of another database to be defined in the next set of configuration specifications. Multiple database configurations can be saved in the properties files concurrently.

**IMPORTANT:**
The database name you use in the configuration must match the database Code as defined in Search - Database Definitions in Voyager System Administration.
Bibliographic Record Linking

Overview 5-1
Defining Record Relationships 5-2
• Displaying Related Records 5-2
• Maintaining Related Records 5-2
Configuring Voyager for Bibliographic Record Linking 5-3
• System Administration 5-3
• Cataloging 5-3
• WebVoyáge 5-4
Bibliographic Record Linking

Overview

Bibliographic records are related to each other for different reasons. You can use bibliographic record linking to relate a new serial title with its preceding title. Conversely, you can relate an old serial title with its succeeding title. Those are just two types of related records.

Bibliographic Record Linking provides sites with a method for:

- Defining relationships between bibliographic records using profiles created in the Voyager System Administration module
- Displaying those relationships in WebVoyage
- Maintaining those relationships with templates in the Voyager Cataloging module

The bibliographic record linking profiles are created so that related records can be identified using data from a single source, the source record. In each profile, the data from the source record is expressed in MARC format by pairing coordinated tags and subfields with a left-anchored index.

For example, the 773 ‡x tag and subfield might be paired with the left-anchored index for searching ISSN numbers. The tag/subfield/index combination that links the source record to its related records can be any tag/subfield in the source record and any left-anchored index defined in System Administration.
NOTE:
The left-anchored indexes available for bibliographic record linking are those defined in the Search component of Voyager System Administration.

Left-anchored indexes which are not available for bibliographic record linking include headings searches as defined in Search Definitions and composite searches as defined in Composite Definitions of Voyager System Administration.

Defining Record Relationships

Bibliographic records can be linked in different directions. For example, a vertical relationship can be used to relate the main bibliographic record of a serial title to its article level bibliographic records and then to relate the article level bibliographic records back to the main record. In addition to vertical relationships, the following relationships exist:

- Sibling relationships between children records that share the same host or parent record
- Chronological relationships between records that are predecessors and successors
- Horizontal relationships between records that reflect different versions of the same bibliographic item

Complementary relationships are defined separately. In this type of relationship, the tag/subfield that links a source record to related records does not automatically specify the complementary tag that links the related records back to the source record. Each complementary relationship must be explicitly defined and all tag/subfields must be available in a left-anchored index.

Displaying Related Records

When a source record displays in WebVoyage, the related record information is displayed by clicking a hyperlink. Patrons view related records by clicking the hyperlink that leads directly to the detailed record view of the related record.

Each site can specify the maximum number of related records to display in the detailed record view.

Maintaining Related Records

Related records can be maintained using the Voyager Cataloging module. You can view and edit existing related records as well as create new related records quickly and easily using templates that contain derivation fields.
Derivation fields allow you to map data from the source record into the new, target record. Standard template functionality remains in place so you can create new records containing mapped data as well as static text and empty tags and subfields.

### Configuring Voyager for Bibliographic Record Linking

Bibliographic Record Linking integrates the functionality of several Voyager modules.

- System Administration
- WebVoyâge
- Cataloging

#### System Administration

In the Voyager System Administration module, you determine the tag, subfields, and indexes that link a source record to one or more related records. This is accomplished through the Bibliographic Record Linking Profile. This profile allows you to create multiple profiles for multiple record relationships. See the Bibliographic Record Linking section in the Search Configuration chapter of the Voyager System Administration User’s Guide for more information.

#### Cataloging

In the Cataloging module, staff can search for, review, and edit the related records using current cataloging functions. If the active bibliographic record contains any of the subfields that define a bibliographic record relationship, a new menu option displays on the menu bar.

The Record Relationships menu lists any profiles that include tags/subfields that exist in the current active record. Selecting one of the profiles returns a list of all related records.

**NOTE:**
Any bibliographic records that have been suppressed do not display in WebVoyâge. However, they do display in the Cataloging module.

Also, if multiple owning libraries exist for records in the database, related records only display per owning library. That is, only the related records belonging to the same owning library as the source record display in WebVoyâge.
The setting to enable bibliographic record linking on the holdings page is in the `webvoyage.properties` file that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/. See Figure 5-1.

```bash
# Option to enable bibliographic record linking on holdings page
# when the related record is available for the bib ID
option.loadRelatedRecords=Y
```

Figure 5-1. Enable bibliographic record linking on holdings page example

Related records holdings display configuration settings are also located in the `webvoyage.properties` file. See Figure 5-2.
Chapter 5: Bibliographic Record Linking

Figure 5-2. Related records settings

The label setting for the source record is in the webvoyage.properties file as shown in Figure 5-3.

Figure 5-3. Source record label example
Display Codes

displaycfg.xml 3-1
displayHoldings.xml 3-2
Enable Redirect 3-3
display.xsl 3-3
Quick Limits 3-4
Display Codes

displaycfg.xml

The codes described in Table 6-1 may be used to format the display of information through the displaycfg.xml file that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/contentLayout/configs/.

Table 6-1. displaycfg.xml display codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>Table of Contents (505 subfields a, r, t, g).</td>
</tr>
<tr>
<td>3000</td>
<td>856 links (linked resources from the 856 field).</td>
</tr>
<tr>
<td>4000</td>
<td>MARC record.</td>
</tr>
<tr>
<td>5000</td>
<td>Database name of the bibliographic record.</td>
</tr>
<tr>
<td>6000, 6010</td>
<td>Electronic resource information.</td>
</tr>
<tr>
<td>7000</td>
<td>Format.</td>
</tr>
<tr>
<td>7106</td>
<td>Includes.</td>
</tr>
<tr>
<td>7107</td>
<td>Physical description.</td>
</tr>
<tr>
<td>9000</td>
<td>Holdings information that is defined in displayHoldings.xml.</td>
</tr>
<tr>
<td>9500</td>
<td>Display holdings summary information.</td>
</tr>
</tbody>
</table>
Many of these codes are defined in `displaycfg.xml`. Comment out the lines of code that you do not want to use for display formatting.

**displayHoldings.xml**

When the 9000 code is specified in `displaycfg.xml`, the codes described in Table 6-2 may be used to format the display of holdings information through the `displayHoldings.xml` file that is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/contentLayout/configs/`.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>OPAC display name for the location.</td>
</tr>
<tr>
<td>1002</td>
<td>Database name for the item.</td>
</tr>
<tr>
<td>1005</td>
<td>OPAC display name for the temporary location (only). This is used in combination with the 1000 code.</td>
</tr>
<tr>
<td>1010</td>
<td>Number of items linked to the MARC holdings record.</td>
</tr>
<tr>
<td>1012</td>
<td>Item status from the item record. If there is only one existing item, its status always displays. If there is more than one item linked to the MARC holdings record, only the items with exceptional statuses (charged, lost, in bindery, and so on) have their statuses displayed. (Exceptional statuses are any status except for Available and Not Charged.)</td>
</tr>
<tr>
<td>1020</td>
<td>Recent issues from serials.</td>
</tr>
<tr>
<td>1022</td>
<td>Supplemental issues from serials.</td>
</tr>
<tr>
<td>1024</td>
<td>Indexes from serials.</td>
</tr>
<tr>
<td>1030</td>
<td>Order status as shown in the line items of purchase orders.</td>
</tr>
<tr>
<td>1040, 1042, 1044</td>
<td>Compressed serials information.</td>
</tr>
<tr>
<td>1050</td>
<td>E-item information. This includes enumeration, chronology, year information, and caption linked to the e-item.</td>
</tr>
<tr>
<td>3000</td>
<td>856 links (linked resources from the 856 field).</td>
</tr>
</tbody>
</table>

Many of these codes are defined in `displayHoldings.xml`. Comment out the lines of code that you do not want to use for display formatting.

**NOTE:**
The XML display for serials functionality is the default functionality.
Enable Redirect

In order to enable a redirect on any field in WebVoyage, you must add redirect information such as the following to a displayTag in either the displaycfg.xml or displayHoldings.xml files.

```xml
redirect="callnumber" redirectOn="hi"
```

See Figure 6-1 for an example from the displaycfg.xml.

```
<displayTags label="Title:">
  <!--
  <displayTag field="130" indicator1="X" indicator2="X"
    subfield="aplskfmnor" redirect="title" redirectOn="apl"/>
  <displayTag field="240" indicator1="X" indicator2="X"
    subfield="aplskfmnor" preText="[" postText="]" redirect="title"
    redirectOn="apl"/>
  <displayTag field="730" indicator1="X" indicator2="X"
    subfield="aplskfmnor" redirect="title" redirectOn="apl"/>
  -->
  <displayTag field="245" indicator1="X" indicator2="X"
    subfield="abcfknps"/>
</displayTags>
```

Figure 6-1. Example of redirect

display.xsl

The display.xsl template identifies all the display codes that can be processed. Locate the lines of code beginning with the code shown in Figure 6-2 to view this information.
WebVoyáge Architecture Overview and Configuration Models

The display.xsl file is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/contentLayout/display/.

Figure 6-2. Example display.xsl code

```xml
<xsl:template name="processDisplayTags">
    <xsl:param name="mfhdID"/>
    <xsl:param name="recordType"/>

    <xsl:for-each select="displayTag">
        <xsl:choose>
            <xsl:when test="@field < '1000'">
                <xsl:call-template name="BMDProcessMarcTags">
                    <xsl:with-param name="field" select="@field"/>
                    <xsl:with-param name="indicator1" select="@indicator1"/>
                    <xsl:with-param name="indicator2" select="@indicator2"/>
                    <xsl:with-param name="subfield" select="@subfield"/>
                    <xsl:with-param name="mfhdID" select="$mfhdID"/>  
                    <xsl:with-param name="recordType" select="$recordType"/></xsl:call-template>
            </xsl:when>
        </xsl:choose>
    </xsl:for-each>
</xsl:template>
```
How Do I Build A Separate Display For Serials?

Description For “How Do I Build A Separate Display For Serials?” Example 4-1
Files 4-1
Instructions 4-1
How Do I Build A Separate Display For Serials?

Description For “How Do I Build A Separate Display For Serials?” Example

By default, WebVoyage uses a single displaycfg.xml file to display all MARC bibliographic records.

The instructions for this example allow you to create and use a separate configuration file for serials based on the MARC leader value.

This model can also be used to create different MARC views for any material type.

Files

The example in this chapter uses the following files:

- sdisplaycfg.xml (new for this example).
- cl_displayRecord.xsl.

Instructions

This section provides the instructions for creating the example described in this chapter.
Procedure 7-1. Build Separate Display for Serials

Use the following procedure to build a separate display for serials.

**NOTE:**
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is `en_US` as in the following:

```
/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Create and save a new file in the `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/` directory named `sdisplaycfg.xml` to be used to specify the display fields for your serials records. Use the sample lines of code shown in Figure 7-1 for this example.
Figure 7-1. Sample line of code for sdisplaycfg.xml

```xml
<display>
  <titleTags>
    <displayTag field="245" indicator1="X" indicator2="X" subfield="ab"/>
  </titleTags>

  <displayTags label="Title:"
    <displayTag field="245" indicator1="X" indicator2="X" subfield="abcfknps"/>
  </displayTags>

  <displayTags label="Also Called:"
    <displayTag field="246" indicator1="X" indicator2="X" subfield="abfnp"/>
  </displayTags>

  <displayTags label="Continues:"
    <displayTag field="780" indicator1="0" indicator2="0" subfield="at"/>
    <displayTag field="780" indicator1="0" indicator2="1" subfield="at"/>
  </displayTags>

  <displayTags label="Supersedes:"
```
Figure 7-1. Sample line of code for sdisplaycfg.xml (Continued)
Chapter 7: How Do I Build A Separate Display For Serials?

2. **Make a backup copy of** cl_displayRecord.xsl **that is located in** /ml/voyager/xxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/.

3. **Edit** cl_displayRecord.xsl.

   a. **Locate** the XSL stylesheet element that contains namespace declarations and begins with the following:

   ```
   <xsl:stylesheet version="1.0"
   
   b. **Add two namespace declarations for** hol and slim as shown in **Figure 7-2**.

   ```
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/page"
   xmlns:fo="http://www.w3.org/1999/XSL/Format"
   xmlns:hol="http://www.endinfosys.com/Voyager/holdings"
   xmlns:slim="http://www.loc.gov/MARC21/slim"
   
   ```

   **Figure 7-2.** New namespace declarations example

   c. **Locate** <!-- ## Our Document Holders ## -->. **See** **Figure 7-3**.

   d. **Change the variable name** Config **to** MConfig. **See** **Figure 7-3**.

   e. **Add a document holders declaration for** the serial configuration display for SConfig and recType as shown in **Figure 7-3**.
WebVoyage Architecture Overview and Configuration Models

f. Add logic for looking at the record leader by replacing the lines of code seen in Figure 7-4 with the example lines of code in Figure 7-5.

Figure 7-3. Our Document Holders example

Figure 7-4. Lines to be replaced
Chapter 7: How Do I Build A Separate Display For Serials?

4. Save and test the changes you made to cl_displayRecord.xsl.

Figure 7-5. Replacement lines of code
OPTIONAL:
5. Back out your changes, if necessary, by deploying your backup copy of cl_displayRecord.xsl.
How Do I Add Static Links To The Header Or Footer?

Description For “How Do I Add Static Links To The Header Or Footer?” Example 5-1
Files 5-1
Instructions 5-2
How Do I Add Static Links To The Header Or Footer?

Description For “How Do I Add Static Links To The Header Or Footer?”

Example

Both the header and footer page elements can accommodate links to external web sites, web applications, and so forth.

In this chapter, the header example demonstrates hyperlinks; and the footer example demonstrates adding a new tab.

Files

The examples in this chapter use the following files:

- header.xsl
- footer.xsl
Instructions

This section provides the instructions for creating the examples described in this chapter.

Procedure 8-1. Create Header Links

Use the following procedure to create header links.

**NOTE:**
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is `en_US` as in the following:

```
/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Make a backup copy of the `header.xsl` file that is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/`.

2. Edit the `header.xsl` file to add a new template.

   a. Add your template (see Figure 8-1) immediately before the `<xsl:stylesheet>` element at the bottom of the file.
Chapter 8: How Do I Add Static Links To The Header Or Footer?

b. Call the template that you created in Step a by adding the instruction (see Figure 8-2, line 7) to the buildHeader section located at the top of the header.xsl file.
Procedure 8-2. Create Footer Tabs

Use the following procedure to create a footer tab.

**NOTE:**
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

```
/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Make a backup copy of the footer.xsl file that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/.
2. Edit the footer.xsl file to add a new tab by adding the new tab code (see Figure 8-3, lines 15-19) after the <ul class="navbar"> in the buildFooter section in the file.

```
<?-- # buildFooter # -->
<!-- # buildFooter # -->
<-- # buildFooter # -->
<xsl:template name="buildFooter">
  <xsl:for-each select="/page:page/page:pageFooter">
    <div id="pageFooter">
      <xsl:for-each select="page:tabs[@nameId='page.footer.buttons']">
        <div id="footerTabs" title="{$footerText/footerTabs}">
          <a name="navFooter"></a>
          <h2 class="navFooter"><xsl:value-of select="{$footerText/footerTabs}" /></h2>
          <ul class="navbar">
            <!-- extra footer tabs -->
            <li>
              <a href="http://www.exlibrisgroup.com/" target="_newWin">Ex Libris</a>
            </li>
            <!-- end extra footer tabs -->
            <xsl:for-each select="$Configs/pageConfigs/footerTabDisplayOrder/tab">
              <div id="footerTabs">
                <a name="navFooter"></a>
                <h2 class="navFooter"><xsl:value-of select="{$footerText/footerTabs}" /></h2>
                <ul class="navbar">
                  <!-- extra footer tabs -->
                  <li>
                    <a href="http://www.exlibrisgroup.com/" target="_newWin">Ex Libris</a>
                  </li>
                  <!-- end extra footer tabs -->
                  <xsl:for-each select="$Configs/pageConfigs/footerTabDisplayOrder/tab">
```

Figure 8-3. Footer tab code example

3. Save and test your changes to footer.xsl.

OPTIONAL:

4. Back out your changes, if necessary, by deploying your backup copy of footer.xsl.
How Do I Remove Information From A Page?

Description For “How Do I Remove Information From A Page?” Example 6-1
Files 6-1
Instructions 6-1
How Do I Remove Information From A Page?

Description For “How Do I Remove Information From A Page?” Example

There may be some page elements you want to disable or prevent from displaying. This example describes how to remove the Item Type column from the Charged Items table on the My Account page.

NOTE:
It's important to remove all the relevant pieces of a page element. In this example, the instructions comment out both the heading and table cell pieces of the Item Type column. Commenting out only one isn't sufficient.

Files

The example in this chapter uses the cl_myAccount.xsl file.

Instructions

This section provides the instructions for completing the example described in this chapter.
Procedure 9-1. Remove Information From a Page

Use the following procedure to remove information from a page.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Make a backup copy of cl_myAccount.xsl that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/.

2. Edit cl_myAccount.xsl.
   a. Find the displayChargedItems template section marked by the comment shown in Figure 9-1.

   <!-- ###################### -->
   <!-- ## displayChargedItems ## -->
   <!-- ########################################################### -->
   <xsl:template name="displayChargedItems">

   Figure 9-1. Section comment to locate

   b. Comment out the table heading and table cell lines of code relevant to item type. See Figure 9-2 and Figure 9-3.

   <!-- <th id="cellChargedType"><xsl:value-of select="page:element/
   page:heading[@nameId='type']"/></th> -->

   Figure 9-2. Table heading example
Chapter 9: How Do I Remove Information From A Page?

3. Save and test your changes.

**OPTIONAL:**

4. Back out your changes, if necessary, by deploying your backup copy of cl_myAccount.xsl.

Figure 9-3. Table cell example
How Do I Add A Map Or Other Information To A Location?

Description For “How Do I Add A Map Or Other Information To A Location?”

Example

This example allows you to direct your patrons to add a map to a location and/or other applicable information like the hours of the reading room. The information offered is based on the MFHD location (852 |b).

Files

The example in this chapter uses the following files:

- local_locMapLink.xsl (new for this example).
- display.xsl.

Instructions

This section provides the instructions for creating the example described in this chapter.
Procedure 10-1. Add a Map to a Location and/or Other Applicable Information

Use the following procedure to add a map to a location and/or other applicable information.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

```
/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Create and save a new file in the /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/ directory named local_locMapLink.xsl to define how to build the hyperlink. Use the sample lines of code shown in Figure 10-1 for this example.

NOTE:
Notice the template name locMapLink in this example.
Chapter 10: How Do I Add A Map Or Other Information To A Location?

Figure 10-1. Example code for building hyperlink
2. Make a backup copy of `display.xsl` that is located in `/ml/voyager/xxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xl/contentLayout/display/`.

3. Edit `display.xsl`.
   a. Add a statement at the top of the `display.xsl` file that contains the path to the `local_locMapLink.xsl` file. See Figure 10-2.

```xml
<xsl:import href="../display/marc21slim.xsl"/>
<xsl:import href="../configs/104X_display.xsl"/>
<xsl:import href="../local_locMapLink.xsl"/>
```

**Figure 10-2. Example code for adding path statement**

b. Call the `locMapLink` template from within the BMD100 template. See Figure 10-3.

```xml
<!-- ###################################################################### -->
```

**Figure 10-1. Example code for building hyperlink (Continued)**
Chapter 10: How Do I Add A Map Or Other Information To A Location?

4. Save and test your changes.

OPTIONAL:
5. Back out your changes, if necessary, by deploying your backup copy of display.xsl.
How Do I Create An External Search From A Bibliographic Record Display?

Description For “How Do I Create An External Search From A Bibliographic Record Display?” Example  8-1
Files  8-1
Instructions  8-2
How Do I Create An External Search From A Bibliographic Record Display?

Description For “How Do I Create An External Search From A Bibliographic Record Display?” Example

It is common to use the ISBN as a parameter to a new search in a different application such as Amazon.com®, WorldCat®, and so on.

The instructions in this chapter describe how to parse out the ISBN from a bibliographic record display and insert it into a URL.

The URL is displayed in the Action Box on the item display page.

This model can also be used to extract different pieces of the MARC record and construct different or multiple URLs.

Files

The example in this chapter uses the following files:

- isbnSearch.xsl (new for this example).
- displayFacets.xsl.
Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 11-1. Create External Search From Bibliographic Record Display

Use the following procedure to create an external search from a bibliographic record display.

NOTE: Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Create and save a new file in the /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/ directory named isbnSearch.xsl to be used to specify how to extract the ISBN from the bibliographic record. Use the sample lines of code shown in Figure 11-1 for this example.

NOTE: Notice the template name recordIsbnSearch in this example.
<?xml version="1.0" encoding="UTF-8"?>
<!--
** Note: Creates a link to WorldCat using the ISBN
** Version : 1.0
** Created : 15-APR-08
** Created By : 
--> 

Figure 11-1. Sample code to extract ISBN from bibliographic record

<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/page"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <!-- ###################################################################### -->
    <xsl:template name="recordIsbnSearch">
        <xsl:variable name="isbn">
            <xsl:call-template name="BMDProcessMarcTags">
                <xsl:with-param name="field" select="'020'"/>
                <xsl:with-param name="indicator1" select="'X'"/>
                <xsl:with-param name="indicator2" select="'X'"/>
                <xsl:with-param name="subfield" select="'a'"/>
                <xsl:with-param name="mfhdID" select="$bibID"/>
                <xsl:with-param name="recordType" select="'bib'"/>
            </xsl:call-template>
        </xsl:variable>
        <a target="_blank">
        </a>
        Check Other Local Libraries</a>
    </xsl:template>
</xsl:stylesheet>
2. **Make a backup copy of** `displayFacets.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/.

3. **Edit** `displayFacets.xsl`.
   a. Define where the `isbnSearch.xsl` can be found with a line of code immediately after the namespace declarations. See Figure 11-2.
   
   ```xml
   <xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/page"
   xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <xsl:include href="./isbnSearch.xsl"/>
   </xsl:stylesheet>
   
   Figure 11-2. Define isbnSearch.xsl location example
   
   b. Call the template defined in `isbnSearch.xsl`.
   
   For this example, it is added to the bottom of the Action Box.
   
   Working from the end of the file, add the call before the final `</div>`. See Figure 11-3.
4. (Optional) Set up trimData XSL code. See Figure 20-3 on page 20-4.

The trimData template resides in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/sandbox/xsl/pageTools/tools.xsl.

This template strips out non-numerical data including punctuation and parenthetical references for Web Services or APIs that require number-only input.

5. Save and test your changes.

**OPTIONAL:**

6. Back out your changes, if necessary, by deploying your backup copy of displayFacets.xsl.
How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?

Description For “How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?” Example
Files 9-1
Instructions 9-2
How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?

Description For “How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?”

Example

The Basic Search page includes a Limit To dropdown list that is compatible with keyword searches. The instructions in this chapter explain how to install a JavaScript that disables the Limit To dropdown list when you select an index that is incompatible with limits such as headings and call number.

The JavaScript also changes the search tips displayed to the user based on the index selected. This provides you with the option to offer hints on improving search strategies.

Files

The example in this chapter uses the following files:

- searchBasic.js (new for this example).
- cl_searchBasic.xsl.
- pageProperties.xml.
Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 12-1. Disable Limits and Change Search Tips

Use the following procedure to disable limits and change search tips dynamically based on the selected search index.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Create and save a new JavaScript file in the /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/jscripts/ directory named searchBasic.js to be used to disable the quick limits dropdown list. Use the sample lines of code shown in Figure 12-1 for this example.
function updateSearchTip ()
{
    // added to searchCode.onchange to customize search tip based on index
    // save the defaul search tip the first time through
    defaultTip = window.defaultTip || document.getElementById('customSearchTip').innerHTML;

    currentSearchCode = document.getElementById('searchCode').value

    switch (currentSearchCode)
    {
        case 'CMD':
        case 'CMD*':
            document.getElementById('customSearchTip').innerHTML = "build a simple Boolean search: <span class="example">(cats or dogs) and therapy</span>";
            break;

        case 'NAME+':
        case 'AUTH+':
```
document.getElementById('customSearchTip').innerHTML = "search by personal or corporate author: last name first <span class="example">pessl marisha</span>, or company name <span class="example">jung seed" break;

    case 'CALLs':
    document.getElementById('customSearchTip').innerHTML = "enter as much of the call number that you know: <span class="example">PR 1297</span>" break;

    default:
    // use the default tip if not otherwise overridden
    document.getElementById('customSearchTip').innerHTML = defaultTip;
    }
}

function searchCodeChanged ()
{
    /*
    ** Disable the limits drop for searches that do not support it.
    */

    currentSearchCode = document.getElementById('searchCode').value.substring(0,4);

    if (!document.getElementById('limitTo').disabled) {
    currentLimit = document.getElementById('limitTo').value;
    }

    switch(currentSearchCode)
    {
    // OPAC HEADING INDEXES
    case 'SUBJ':
    case 'TITL':
    case 'NAME':
    case 'AUTH':
    // MAIN CALL NUMBER INDEX
```
case 'CALL':
    // JOURNAL INDEX WITH PRELIMITS
    case 'JKEY':
    case 'JALL':

        document.getElementById('limitTo').value="none";
        document.getElementById('limitTo').disabled=true;
        break;

    default:
        document.getElementById('limitTo').disabled=false;
        document.getElementById('limitTo').value=currentLimit;
    }
}

function addSearchCodeChanged ()
{
    document.getElementById('searchCode').onchange = function()
    {searchCodeChanged(); updateSearchTip();}
    // run the script to catch default index doesn't support limits or edit search
    searchCodeChanged ();
    updateSearchTip();
}

function addLoadEvent(func) {
    var oldonload = window.onload;
    if (typeof window.onload != 'function') {
        window.onload = func;
    } else {
        window.onload = function() {
            if (oldonload) {
                oldonload();
            }
        }
    }
    func();
2. Make a backup copy of pageProperties.xml that is located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/userTextConfigs/.


Locate the comment <!-- ## Start Search Tips ## --> and identify the <page name="page.searchBasic" position="belowContent"> element.

Add a <div> element as in Figure 12-2.

```
<page name="page.searchBasic" position="belowContent">
  <div class="searchTip">
    <span class="label">Search Tips: </span>
    <!-- special div to enable javascript to swap out help text -->
    <div id="customSearchTip" style="display:inline">
      enter words relating to your topic, use quotes to search phrases: <span class="example">"world wide web"</span>,
      use + to mark essential terms: <span class="example">+explorer</span>,
      use * to mark important terms: <span class="example">*internet</span>,
      use ? to truncate: <span class="example">browser?</span>
    </div>
  </div>
</page>
```

Figure 12-2. Example <div> element

4. Save your changes.
5. Make a backup copy of `cl_searchBasic.xsl` that is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.


   At the end of the `buildBasicSearch` template (see Figure 12-3), load the `searchBasic.js` JavaScript file you created.
Figure 12-3. Locate buildBasicSearch template

```xml
<xs1:template name="buildBasicSearch">
  <div id="searchParams">
    <div id="searchInputs">
      <xs1:call-template name="buildFormInput">
        <xs1:with-param name="eleName" select="'searchArg'"/>
        <xs1:with-param name="size" select="'51'"/>
        <xs1:with-param name="accesskey" select="'s'"/>
      </xs1:call-template>
      <xs1:call-template name="buildFormDropDown">
        <xs1:with-param name="eleName" select="'searchCode'"/>
      </xs1:call-template>
    </div>
    <div id="quickLimits">
      <xs1:call-template name="buildFormDropDown">
        <xs1:with-param name="eleName" select="'limitTo'"/>
      </xs1:call-template>
    </div>
    <!-- load javascript file for handling limits enable/disable -->
    <script type="text/javascript" src="{$jscript-loc}searchBasic.js"/>
  </div>
  <xs1:call-template name="buildSearchButtons"/>
</xs1:template>
```

12-8
OPTIONAL:
8. *Back out your changes, if necessary, by deploying your backup copies of pageProperties.xml and cl_searchBasic.xsl.*
How Do I Disable AutoComplete?

Description For “How Do I Disable AutoComplete?”
Example 10-1
Files 10-1
Instructions 10-1
How Do I Disable AutoComplete?

Description For “How Do I Disable AutoComplete?” Example

AutoComplete is a feature of certain web browsers that stores information on the computer's hard drive that a user types into web page forms. When you begin filling in another form, the browser suggests possible answers from information stored on the hard drive.

Particularly at public workstations, you may want to disable the browser’s AutoComplete capability. This is a feature of both Internet Explorer® and Firefox®.

Files

The example in this chapter uses the following files:

- login.xsl.
- searchFacets.xsl.

Instructions

This section provides the instructions for creating the example described in this chapter.
Procedure 13-1. Disable AutoComplete

Use the following procedure to disable AutoComplete.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Make a backup copy of searchFacets.xsl that is located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/.

2. Edit searchFacets.xsl.
   a. Locate the buildTheSearchForm section near the top of the file.
   b. Add autocomplete="off" to the <form action> element. See Figure 13-1.

   
   <form action="{$formAction}" method="GET" accept-charset="UTF-8" id="{$formName}" autocomplete="off">

Figure 13-1. Example of autocomplete="off" code

3. Save your changes.

4. Make a backup copy of login.xsl that is located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/.

5. Edit login.xsl.
   a. Locate the buildContent section.
   b. Add autocomplete="off" to the <form action> element. See Figure 13-2.
Chapter 13: How Do I Disable AutoComplete?

6. Save and test your changes.

OPTIONAL:
7. Back out your changes, if necessary, by deploying your backup copies of searchFacets.xsl and/or login.xsl.
How Do I Display A Favicon?

Description For “How Do I Display A Favicon?” Example  11-1
Files  11-1
Instructions  11-2
How Do I Display A Favicon?

Description For “How Do I Display A Favicon?” Example

In Internet Explorer and Firefox, a favicon (favorite icon) displays in the address bar, the favorites menu, bookmarks, and page tabs.

A favicon is a way to brand your catalog for your patrons.

NOTE:
There are multiple methods for installing a favicon. The method described in this chapter is specific to WebVoyage and allows you to use any type of image files such as .jpg, .gif, or .png in addition to favicon .ico files.

For further information regarding favicons, see the following sites:


Files

The example in this chapter uses the frameWork.xsl file.
Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 14-1. Display favicon

Use the following procedure to display your favicon.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Create a 16x16 pixel icon.

2. Save the icon to the /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/ directory.

3. Make a backup copy of frameWork.xsl that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageTools/.

4. Edit frameWork.xsl.
   a. Locate the buildHtmlPage template.
   b. Insert the example code shown in Figure 14-1 after the <head> element.

```
<link rel="shortcut icon" href="{$image-loc}favicon.ico" type="image/x-icon" />
<link rel="icon" href="{$image-loc}favicon.ico" type="image/x-icon " />
```

Figure 14-1. Example favicon code for frameWork.xsl

   c. Replace favicon.ico with the name of the icon file that you created at the beginning of this procedure and saved in .../images/.
Chapter 14: How Do I Display A Favicon?

NOTE:
By default, the \{$image-loc\} notation is a path to the /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/ directory. This path is defined in constants.xsl.

5. Save and test your changes.

OPTIONAL:
6. Back out your changes, if necessary, by deploying your backup copy of frameWork.xsl.
How Do I Hide Limits On The Advanced Search Page?
How Do I Hide Limits On The Advanced Search Page?

Description For “How Do I Hide Limits On The Advanced Search Page?”

Example

This chapter describes how to hide the various limits options on the Advanced Search page until a user clicks a More link to display them.

Files

The example in this chapter uses the following files:

- searchAdvanced.js (new for this example).
- searchAdvanced.css.
- cl_searchAdvanced.xsl.

Instructions

This section provides the instructions for creating the example described in this chapter.
Procedure 15-1. Hide Limits on the Advanced Search Page

Use the following procedure to hide limits on the Advanced Search page.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is `en_US` as in the following:

```
/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Create and save a new JavaScript file in the `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/jscripts/` directory named `searchAdvanced.js` to be used to hide limits on the Advanced Search page. Use the sample lines of code shown in Figure 15-1 for this example.
Chapter 15: How Do I Hide Limits On The Advanced Search Page?

Figure 15-1. Example code for searchAdvanced.js

```javascript
/**
 * Copyright 2007 ExLibris Group
 * All Rights Reserved
 *
 * ** Product : WebVoyage :: disable/enable limits on advanced search
 * ** not an accessible technique
 * ** Version : 7.0
 * ** Created : 23-JAN-2008
 * ** Orig Author :
 * ** Last Modified : 23-JAN-2008
 * ** Last Modified By :
 */

// hide the limitList div
function hideLimits() {
    document.getElementById('limitList').style.display='none';
}

// display the limitList div
function showLimits() {
    document.getElementById('limitList').style.display='';
}

// toggle the limitList div
// we'll check the class of the toggle switch
function toggleLimits () {
    showLimits();
    document.getElementById('limitToggle').style.display='none';
}

function addLoadEvent(func) {
```

15-3
2. **Make a backup copy** of `searchAdvanced.css` **that is located in** `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/css/`.

3. **Edit** `searchAdvanced.css`.  
   Go to the end of the file and add the example code shown in **Figure 15-2**.

4. **Save your changes.**

---

```javascript
var oldonload = window.onload;
if (typeof window.onload != 'function') {
  window.onload = func;
} else {
  window.onload = function() {
    if (oldonload) {
      oldonload();
    }
    func();
  }
}

addLoadEvent{
  function () {
    // hide the the limits on page load and show the toggle switch
    hideLimits();
    document.getElementById('limitToggle').style.display='';
  }
};

/* display link to show limits */
#limitToggle {
  font-size: smaller;
  font-family: Verdana;
  margin:10px 10px 0.5em;
}
```

**Figure 15-1. Example code for searchAdvanced.js (Continued)**

**Figure 15-2. Example code for searchAdvanced.css**
5. **Make a backup copy of** cl_searchAdvanced.xsl **that is located in** /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/.

6. **Edit** cl_searchAdvanced.xsl.
   
   a. **Locate the** buildSearchForm **section.**
   
   b. **Add a call to the** searchAdvanced.js **file immediately before the** <!-- line 6 - year label, radio button & selection box --> **comment. See Figure 15-3.**

```
<!-- add java script to hide/show limits on page -->
<script type="text/javascript" src="{$jscript-loc}searchAdvanced.js"/>

<div id="limitToggle" class="limitsOff" style="display:none"><a href="javascript:toggleLimits();">more</a></div>
<div id="limitList">
<!-- end - other end of limitList div was added below -->
</div>
```

**Figure 15-3. Example coding change for cl_searchAdvanced.xsl**

   c. **Add the closing** <div> **element above the** buildSearchButtons **template at the bottom of the file. See Figure 15-4.**

```
</xsl:for-each>
</div>

<xsl:call-template name="buildSearchButtons"/>

</div>

<!-- search advanced form - end -->
</xsl:template>

</xsl:stylesheet>

**Figure 15-4. Additional coding change to cl_searchAdvanced.xsl**

7. **Save your changes and test.**
8. Back out your changes, if necessary, by deploying your backup copies of 
searchAdvanced.css and cl_searchAdvanced.xsl.
How Do I Build And Display A Persistent Link To A Bibliographic Record?

Description For “How Do I Build And Display A Persistent Link To A Bibliographic Record?” Example
Files
Instructions
How Do I Build And Display A Persistent Link To A Bibliographic Record?

Description For “How Do I Build And Display A Persistent Link To A Bibliographic Record?” Example

Patrons can use the persistent link to bibliographic records for bookmarking, tagging, emailing, blogging, and so forth.

Files

The example in this chapter uses the following files:

- local_PersistentLink.xsl (new for this example).
- cl_displayRecord.xsl.
- displayCommon.css.

Instructions

This section provides the instructions for creating the example described in this chapter.
Procedure 16-1. Build Persistent Link to Bibliographic Record

Use the following procedure to create a persistent link to a bibliographic record.

NOTE:
Directory path references to **xxxdb** implies that you need to substitute your database path name; and where **[skin]** is referenced, substitute the path name that is used at your site. The default skin path provided is **en_US** as in the following:

```
/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Create and save a new file in the `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/` directory named `local_PersistentLink.xsl` to store the template that defines how to build the hyperlink. Use the sample lines of code shown in Figure 16-1 for this example.
Chapter 16: How Do I Build And Display A Persistent Link To A Bibliographic Record?

Figure 16-1. Sample code for local_PersistentLink.xsl
2. Make a backup copy of `cl_displayRecord.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.

3. Edit `cl_displayRecord.xsl`.
   a. Locate the namespace declarations at the top of the file and include a reference to the `local_PersistentLink.xsl` file after the `<xsl:stylesheet>` element. See Figure 16-2.

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/page"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <!-- ## Include Persistent Link template ## -->
    <xsl:include href="../pageFacets/local_PersistentLink.xsl"/>

    <!-- ## Include Persistent Link template ## -->
    <xsl:include href="../pageFacets/local_PersistentLink.xsl"/>

    <xsl:template>
        <div class="persistentLink">
            <a id="persistentLink" href="{$baseURL}{$bibID}" target="_new">
                Persistent Link
            </a>
        </div>
    </xsl:template>

<!-- ###################################################################### -->
</xsl:stylesheet>
```

Figure 16-1. Sample code for `local_PersistentLink.xsl` (Continued)

2. Make a backup copy of `cl_displayRecord.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.

3. Edit `cl_displayRecord.xsl`.
   a. Locate the namespace declarations at the top of the file and include a reference to the `local_PersistentLink.xsl` file after the `<xsl:stylesheet>` element. See Figure 16-2.

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/page"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <!-- ## Include Persistent Link template ## -->
    <xsl:include href="../pageFacets/local_PersistentLink.xsl"/>

    <!-- ## Include Persistent Link template ## -->
    <xsl:include href="../pageFacets/local_PersistentLink.xsl"/>

    <xsl:template>
        <div class="persistentLink">
            <a id="persistentLink" href="{$baseURL}{$bibID}" target="_new">
                Persistent Link
            </a>
        </div>
    </xsl:template>

<!-- ###################################################################### -->
</xsl:stylesheet>
```

Figure 16-1. Sample code for `local_PersistentLink.xsl` (Continued)

b. Call the persistent link template from the Bibliographic Data section. See Figure 16-3.
Chapter 16: How Do I Build And Display A Persistent Link To A Bibliographic Record?

4. Make a backup copy of `displayCommon.css` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/css/`.

5. Add the code in Figure 16-4 to `displayCommon.css`.

```xml
<!-- ## Bibliographic Data ## -->
<xsl:for-each select="$Config">
  <div class="bibliographicData">
    <xsl:call-template name="buildMarcDisplay">
      <xsl:with-param name="recordType" select="'bib'"/>
    </xsl:call-template>
    <!-- ## Use persistentLink template to display the persistent link ## -->
    <xsl:call-template name="persistentLink">
      <xsl:with-param name="bibID" select="$bibID"/>
    </xsl:call-template>
  </div>
</xsl:for-each>
```

**Figure 16-3.** Persistent link template call in Bibliographic Data section

6. Save and test your changes.

**OPTIONAL:**

7. **Back out your changes, if necessary, by deploying your backup copy of cl_displayRecord.xsl and displayCommon.css.**
How Do I Change The Format Of The Record Display Page?

Description For “How Do I Change The Format Of The Record Display Page?” Example 14-1
Files 14-1
Instructions 14-2
How Do I Change The Format Of The Record Display Page?

Description For “How Do I Change The Format Of The Record Display Page?”

Example

For greater formatting control of elements on the record display page, class attributes may be added to the XML and subsequently adjusted in the appropriate style sheet. This allows you, for example, to change the font characteristics of the Title and Author lines without affecting other data on the page.

See Procedure 17-1, Add Class Attributes For Formatting, on page 17-2 for instructions regarding class attributes.

Files

The example in this chapter uses the following files:

- displaycfg.xml.
- display.xsl.
- displayCommon.css.
Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 17-1. Add Class Attributes For Formatting

Use the following procedure to implement class attributes for formatting.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Make a backup copy of displaycfg.xml that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/.

2. Edit displaycfg.xml.
   a. Locate the displayTag to which you want to apply formatting.
   b. Add a class attribute after the label= string. Use a name of your own choosing. See Figure 17-1 for examples.

3. Save your changes to the displaycfg.xml.

4. Make a backup copy of display.xsl that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/display/.

5. Edit display.xsl.
Chapter 17: How Do I Change The Format Of The Record Display Page?

a. Locate the template for buildMarcDisplay.

b. Add the lines shown in Figure 17-2 to assign one of two classes to the bibliographic tag data.
Figure 17-2. Example code to add to buildMarcDisplay
Chapter 17: How Do I Change The Format Of The Record Display Page?

6. Save your changes to `display.xsl`.

7. Make a backup copy of `displayCommon.css` that is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/css/`.

8. Add style directives to the `localcssclass` tags that you defined in `displaycfg.xml` and place at the bottom of the `displayCommon.css` file.

See example code in Figure 17-3.

```css
.tagTitle, .tagAuthor
{
  margin-bottom: .75em;
  font-size: larger;
  color: #ff0000;
}
```

Figure 17-3. Example code for `displayCommon.css`

9. Save and test your changes.

10. Back out your changes, if necessary, by deploying your backup copies of `displaycfg.xml`, `displayRecord.xsl`, and `displayCommon.css`. 
How Do I Add Tracking Codes?

Description For “How Do I Add Tracking Codes?” Example 15-1
Files 15-1
Instructions 15-2
How Do I Add Tracking Codes?

Description For “How Do I Add Tracking Codes?” Example

Various companies offer web page tracking/analytic services such as Google Analytics. The general practice is to include tagging on all the pages you want tracked.

The instructions in this chapter may be use to add the relevant code to one of two .xsl files that are called by all WebVoyâge pages.

NOTE:
You must establish a relationship with the tracking service first and consider use of such a tool in relationship to your institution's privacy policy.

Files

The example in this chapter can apply to either of the following files:

- frameWork.xsl.
- footer.xsl.
Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 18-1. Add tracking codes

Use the following procedure to add tracking codes (in coordination with an outside service).

Specifically, this example inserts the Google Analytics tracking code into the footer.xsl. Coordinate with other services regarding their specific instructions.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Make a backup copy of footer.xsl that is located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/.

2. Edit footer.xsl.
   a. Copy the script snippet the third-party vendor generated for you upon signup. This example uses a Google Analytics script snippet.
   b. Paste the script snippet into the buildFooter template. See Figure 18-1.
Chapter 18: How Do I Add Tracking Codes?

Figure 18-1. Example of script snippet add to buildFooter template

```xml
<xsl:template name="buildFooter">
  <xsl:for-each select="/page:page/page:pageFooter">
    <div id="pageFooter">
      <xsl:for-each select="page:tabs[@nameId='page.footer.buttons']">
        <div id="footerTabs" title="{$footerText/footerTabs}'">
          <a name="navFooter"></a>
          <h2 class="navFooter"><xsl:value-of select="{$footerText/footerTabs}'"></h2>
          <ul class="navbar">
            <xsl:for-each select="$Configs/pageConfigs/footerTabDisplayOrder/tab">
              <xsl:variable name="tempName" select="@name'/">
              <xsl:variable name="newWin" select="$clickOpensNewWindow'/">
              <xsl:call-template name="buildFooterTab">
                <xsl:with-param name="displayTab" select="$tempName'/">
                <xsl:with-param name="newWin" select="$newWin'/">
                </xsl:call-template>
              </xsl:for-each>
            </ul>
          </div>
        </div>
      </xsl:for-each>
    </div>
  </xsl:for-each>
</xsl:template>

```

---

3. Save and test your changes.

**OPTIONAL:**
4. Back out your changes, if necessary, by deploying your backup copy of footer.xsl.
How Do I Implement Google Book Search?

Description For “How Do I Implement Google Book Search?” 16-1

Files 16-1

Google Book Search Implementation 16-2

• googleBooksAvail.js 16-2
• local_googleBooksAvail.xsl 16-2
• displayFacets.xsl 16-3
• displayGoogleBooks.css 16-3

Disable Google Book Search 16-3
Description For “How Do I Implement Google Book Search?”

The Voyager 7.x version of WebVoyage provides code to interface with the Google Book Search API. This enables users of WebVoyage to display Google Book Search information.

This feature is enabled at installation. To disable it, see Procedure 19-1, Disable Google Book Search Feature, on page 19-3.

Files

The Google Book Search feature uses the following files:

- googleBooksAvail.js.
- local_googleBooksAvail.xsl.
- displayFacets.xsl.
- displayGoogleBooks.css.
Google Book Search Implementation

This section describes the WebVoyáge implementation for displaying Google Book Search information in Voyager 7.x.

NOTE: Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:


googleBooksAvail.js

The googleBooksAvail.js script in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/jscripts/ executes the call to the Google Book Search service. (This service is transparent to the user.)

The lines shown in Figure 19-1 define the text that displays in the Action Box.

```
function listBookEntries(booksInfo)
{
    var bookPreviewFull = 'Full text available';
    var bookPreviewPartial = 'Limited Preview';
    var bookPreviewNoView = 'About This Book';
}
```

Figure 19-1. Action Box text

local_googleBooksAvail.xsl

The local_googleBooksAvail.xsl file in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/ defines a template named googleBooksAvail. This template describes how to identify the ISBN, LCCN, or OCLC numbers which are the standard numbers used to do the lookup executed with googleBooksAvail.js.
displayFacets.xsl

The displayFacets.xsl file in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/pageFacets/ calls the googleBooksAvail template. See local_googleBooksAvail.xsl on page 19-2.

displayGoogleBooks.css

The displayGoogleBooks.css file in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/css/ manages the display of the Google information within the Action Box.

Disable Google Book Search

This section provides the instructions for disabling Google Book Search described in this chapter.

Procedure 19-1. Disable Google Book Search Feature

Use the following procedure to disable Google Book Search.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/.

1. Make a backup copy of displayFacets.xsl that is located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/pageFacets/.

2. Edit displayFacets.xsl.

   a. Locate the line of code near the bottom of the file as shown in Figure 19-2.
b. Comment out the lines of code as shown in Figure 19-3.

```xml
<!– # mdp add the google book template -->
    <div id="googleBooksRow">
        <xsl:call-template name="googleBooksAvail"/>
    </div>
<!--– # mdp add the google book template -->
```

Figure 19-3. Comment out code to disable Google Book Search

3. Save and test your changes.

OPTIONAL:

4. Back out your changes, if necessary, by deploying your backup copy of displayFacets.xsl.
Description For “How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?”

Files

Syndetic Solutions Implementation
- pageProperties.xml
- resultsFacets.xsl
- resultsTitles.xsl
- imageUtils.js
- displaycfg.xml
- display.xsl
- displayRecord.xsl

Syndetic Solutions Information
- What is a Query String?
- Sample URLs
How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?

WebVoyage is preconfigured with the capability to display cover images on the results and holdings pages.

The ISBN or ISSN is generally used to do the lookup at the remote service.

NOTE:
You must have a pre-existing relationship or agreement with a service that provides this cover art.

Files

The Syndetics example described in this chapter uses the following files:

- pageProperties.xml.
- resultsFacets.xsl.
- resultsTitles.xsl.
- imageUtils.js.
- displaycfg.xml.
Syndetic Solutions Implementation

This section describes, as an example, the WebVoyage implementation of displaying Syndetics Solutions cover images for results and holdings pages in Voyager 7.x.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:


pageProperties.xml

The pageProperties.xml file located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/userTextConfigs/ defines how the URL to the cover image for the titles results page is built to include the pre- and post-link text and the alternate name for the image. See Figure 20-1.

Figure 20-1. Example of pageProperties.xml

The /SC.gif in Figure 20-1 is the Syndetics Solutions syntax for small cover (SC). If you prefer a medium or large cover image, use MC or LC, respectively.
Chapter 20: How Do I Display Cover Images From Services Like Amazon.com and Syndetics

The resultsFacets.xsl file located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/ defines a template named buildResultsCoverImage. See Figure 20-2.

This template builds on the URL components defined in pageProperties.xml.

```xml
<xsl:template name="buildResultsCoverImage">
  <xsl:param name="tag"/>
  <xsl:param name="tagType"/>
  <xsl:for-each select="$Configs/pageConfigs/resultsCoverTag[@nameIdMatch=$tagType]">
    <div class="resultListCoverImageCell">
      <img src="{$tagPRE_TEXT}{$tag}{$tagPOST_TEXT}" class="resultListCoverImage" alt="{$altText}" style="display:none" onload="checkImage(this)"/>
    </div>
  </xsl:for-each>
</xsl:template>
```

Figure 20-2. Example of buildResultsCoverImage template in resultsFacets.xsl

The buildResultsCoverImage template is used later in the file for constructing the URL with an ISBN or ISSN.

The trimData template strips parenthetical references from the standard numbers. See Figure 20-3.
The `resultsTitles.xsl` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/` builds the titles results page. It uses a JavaScript file named `imageUtils.js`.

Figure 20-3. Example of trimData template call in resultsFacets.xsl
imageUtils.js

The `imageUtils.js` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/jscripts/` performs the functions shown in Figure 20-4.

```javascript
var setDefaultErrMsg=""; // Default image to be displayed on error
var setDefaultErrMsg=""; // Default text to be displayed on error

function checkImage(obj)
{
    if(obj.complete===true)
    {
        if(obj.width < 2)
        {
            obj.src=setDefaultErrMsg;
            obj.setAttribute("alt",setDefaultErrMsg);
            obj.setAttribute("style","display:none");
        }
        else
        {
            obj.setAttribute("style","display:block");
        }
    }
}
```

Figure 20-4. Example of imageUtils.js

The Syndetics service (and possibly other remote cover services) does not allow you to pre-check whether an image exists before you build the link. Therefore, `imageUtils.js` checks to see if the image has been retrieved and loaded. If not, nothing displays.
The `displaycfg.xml` file located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/contentLayout/configs/` defines how the URL to the cover image for the record display page is built that includes the pre- and post-link text and the bib field used to build the link. See Figure 20-5.

The `infoPRE_TEXT` and `infoPOST_TEXT` syntax build a link to more information about the record. This is used to make the cover image a hyperlink.

The `singleInstance="true"` syntax indicates that the first ISBN or ISSN found is used for building the link.

**NOTE:**
The `displaycfg.xml` file includes a comment reminder to remove the comment markers surrounding `coverTags` to enable cover graphics.

---

```xml
<!-- Cover Tags for MARC Display

Uncomment the <coverTags> block below to add cover graphics and title info link to the MARC display.

Be sure to replace "ENTER_YOUR_CLIENT_ID" below with your Syndetics Client ID.
-->

<!--
<coverTags altText="Cover Image" linkPRE_TEXT="http://www.syndetics.com/hw7.pl?isbn="
infoPOST_TEXT="/index.html&client=ENTER_YOUR_CLIENT_ID&type=rn12"
singleInstance="true">
  <displayTag field="020" indicator1="X" indicator2="X" subfield="a"/>
</coverTags>
-->
```

Figure 20-5. Example displaycfg.xml code

The `singleInstance="true"` syntax indicates that the first ISBN or ISSN found is used for building the link.

**NOTE:**
The `displaycfg.xml` file includes a comment reminder to remove the comment markers surrounding `coverTags` to enable cover graphics.

---

**display.xsl**

The `display.xsl` file located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/contentLayout/display/` defines the following templates for retrieving cover images:

- `buildCoverImage`.
• `buildCoverImageLinks`.

These templates provide function similar to the templates described in `resultsFacets.xsl` on page 20-3.

**displayRecord.xsl**

The `displayRecord.xsl` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/` uses the JavaScript file named `imageUtils.js` while building the record display page.

**Syndetic Solutions Information**

Syndetic Solutions provides the information in this section regarding “What is a Query String?” and sample URLs.

⚠️ **CAUTION:**

This information is current as of its receipt. However, over time Syndetic Solutions may change/update its instructions. Should you have any questions, contact Syndetic Solutions Customer Support. See www/syndetics.com.

💡 **TIP:**

The "What is a Query String?" section from Syndetic Solutions references URLs that include the use of ampersands (&). In configuration files, the ampersands need to be escaped as in the following:

```plaintext
```

**What is a Query String?**

Consider a Syndetics URL:

```plaintext
http://www.syndetics.com/index.aspx?isbn=0002154129/SC.GIF&client=clientcode&showCaptionBelow=t&caption=Click+for+more+info
```

The part before the "?" is the website address, the part after the "?" is the Query String. In this case there are 4 parameter/value pairs, separated by "&":

```plaintext
```
The order of the parameters within the Query String does not matter. What matters is that they be separated by ampersands and that they have a Name, Equal Sign "=" and Value. There should be no spaces around the "?", "&" or "=". If there are spaces in the Value, replace them with the plus sign "+".

Sample URLs

These strings have no actual line-breaks or spaces.

**Default caption:**


**Superimposed caption:**


**Caption with Medium size Cover:**


**Caption with Custom Message, Medium size Cover, and Background color:**


**You can also use the RGB value (must be base 10):**


*In order to find more RGB colors, you can search the internet for "web safe RGB colors."*
With a Small size Cover if your message is longer than 19 characters (20 or more - including spaces), then the default caption is used:


With a Medium size Cover if your message is longer than 49 characters (50 or more - including spaces), then the default caption is used:


SC.GIF (Small Cover)

MC.GIF (Medium Cover)
How Do I Implement Geospatial Search?

Description For “How Do I Implement Geospatial Search?” 18-1
Files 18-1
Instructions 18-2
How Do I Implement Geospatial Search?

Description For “How Do I Implement Geospatial Search?”

Geospatial Search is a feature that you optionally set to enable map searching when you install WebVoyage.

NOTE:
This feature is only available if your institution has purchased the Geospatial Searching tools.

WebVoyage provides you with a variety of options when searching for map-related items in your database. You can conduct a search for geospatial items by specifying a region which must be covered, in part or in whole, by the item. This region can be a rectangle, a polygon, a point or circle, a corridor or route, or a range. See Figure 21-1 on page 21-3 for an example.

Files

The Geospatial Search feature described in this chapter uses the following files:

- webvoyage.properties.
- pageProperties.xml.
Instructions

This section provides instructions for implementing Geospatial Search in Voyager 7.x.

NOTE:
Directory path references to $xxxdb$ implies that you need to substitute your database path name; and where $[skin]$ is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:


See Procedure 21-1, Geospatial Search Implementation for the steps to provide access to Geospatial Search.

Procedure 21-1. Geospatial Search Implementation

Use the following steps to implement Geospatial Search.

1. Make a backup copy of $webvoyage.properties$ that is located in $/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/$.

2. Open the $webvoyage.properties$ file and locate $option.geospatialSearch$.

3. Set this option to $Y$ to have WebVoyáge display a Geospatial Search tab. See Figure 21-1 for a display example.
Figure 21-1. Geospatial Search tab

4. Save the updated `webvoyage.properties` file.

5. Open the `pageProperties.xml` file in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/xsl/userTextConfigs/` and locate

```
<!-- enable these lines if site has geospatial searching.
See Figure 21-2. -->
```

See Figure 21-2.
6. Remove the comment lines to enable Geospatial Search.

In Figure 21-2, the comment lines are lines 8 and 11.

7. Save your changes.
How Do I Enable External Authentication?

Description For “How Do I Enable External Authentication?”

External authentication is an optional setting in WebVoyage to enable patron authentication from an external system.

Files

The external authentication settings described in this chapter use the webvoyage.properties file.

Instructions

This section provides instructions for implementing external authentication in Voyager 7.x.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/.
See Procedure 22-1, External Authentication Implementation for the steps to enable external authentication.

### Procedure 22-1. External Authentication Implementation

Use the following steps to implement external authentication.

1. Make a backup copy of `webvoyage.properties` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/`.

2. Open the `webvoyage.properties` file and locate the option `extAuthSystemEnabled=N`. See Figure 22-1.
Chapter 22: How Do I Enable External Authentication?

# Should WebVoyage users use an external authentication system when logging in?
# If Y, WebVoyage uses the external authentication system as configured below
# If N, WebVoyage displays the native logon form

option.extAuthSystemEnabled=N

# URL to the external authentication system

option.extAuthSystemURL=

# Should WebVoyage bypass the logon form if using an external authentication system?
option.extAuthBypassLoginScreen=N

Figure 22-1. External authentication settings example

# use of the external authentication link is optional,
# this line will have no effect if option.extAuthSystemEnabled=N
page.logIn.extAuth.linkText=Go to External Patron Login System

# Number of record display per page

Figure 22-1. External authentication settings example
5. Change `option.extAuthBypassLoginScreen=N` as needed.

6. Change `page.logIn.extAuthlinkText=Go to External Patron LoginSystem` as needed.

7. Save the updated `webvoyage.properties` file.

8. Update the redirect URL that the adaptor uses to return patrons to WebVoyage using the following format:

   ```
   http://[host]:[port]/vwebv/externalLogin.do?[redirect string]&authenticate=[status]
   ```

   This URL indicates to WebVoyage whether or not the patron is successfully authenticated.

   **TIP:**
   Review the comments provided in the `webvoyage.properties` file for additional information.
How Do I Modify Page Messages?

Description For “How Do I Modify Page Messages?” 20-1
Files 20-2
Instructions 20-2
How Do I Modify Page Messages?

Description For “How Do I Modify Page Messages?”

When a `<page:message>` block from the server occurs, a customized message may be displayed.

In the XML, a page message is identified by `blockCode`, `errorCode` and/or `requestCode`. See Figure 23-3 on page 23-4.

When an `errorCode` such as “searchResults.noHits” is received, for example, a customized No Hits message may be displayed. The `<pageMessages>` code in WebVoyáge provides the capability to modify the messages that display. See Figure 23-1 for an example of an `errorCode` identified for a customized message.

```
<pageMsg errorCode="searchResults.noHits">
```

Figure 23-1. `errorCode` example

See Figure 23-2 for a `blockCode` example used when an SSN login error occurs.
NOTE:
You can view the XML by enabling and clicking the Show XML button. The parameters for enabling the Show XML button are located in frameWork.xsl that can be found in the following path:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xml/pageTools/

Files

The page messages settings described in this chapter use the pageProperties.xml file.

Instructions

This section provides instructions for modifying page messages in Voyager 7.x.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:


See Procedure 23-1, Modify Page Messages for the steps to modify page messages.

Procedure 23-1. Modify Page Messages

Use the following steps to modify page messages.
Chapter 23: How Do I Modify Page Messages?

1. **Make a backup copy of** pageProperties.xml **that is located in** /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/userTestConfigs/.

2. **Open the** pageProperties.xml **file and locate** <!-- ## Override [blockCode]. See Figure 23-3. **[blockCode]. See Figure 23-3.**
<!-- Override [blockCode] [errorCode] or [requestCode] messages

The following XML is a place to define what text appears in the interface when we get a <page:message> block

from the server (enable debug in framework.xsl to reveal the showXML button in the interface to view XML data)

At this point the following are just overrides to the text that comes back in the XML

For Example given the following XML block

```
<page:message>
    <page:message blockCode="" errorCode="searchResults.noHits" requestCode="">No hits.<page:message>
</page:messages>
```

we have an errorCode of 'searchResults.noHits'

so we create a <pageMsg> either an errorCode attribute matching what we want to override with a block of text or HTML

```
<pageMsg errorCode="searchResults.noHits">Search resulted in no hits.</pageMsg>
```

Hopefully this makes life a little easier, in the future I would like all pageMessages to be defined here

-->

    <pageMessages>
    <pageMsg blockCode="PATRONMSG">
        <p class="blockMessage">
            You may not have entered your barcode and name correctly.
            <br/>Retry your request or ask for help at the Circulation or Reference Desk.
        </p>
    </pageMsg>
    <pageMsg blockCode="PATRONSOCMSG">
        <p class="blockMessage">
            You may not have entered your social security number and name correctly.
            <br/>Retry your request or ask for help at the Circulation or Reference Desk.
        </p>
    </pageMsg>
    <pageMsg blockCode="PATRONIIDMSG">

Figure 23-3. Modifying page messages example

```
<p class="blockMessage">
    You may not have entered your barcode and name correctly.
    <br/>Retry your request or ask for help at the Circulation or Reference Desk.
</p>
```


3. Define/modify the page message text to match your preference.


5. Test your changes.

**OPTIONAL:**

6. *Back out your changes, if necessary, by deploying your backup copy of `pageProperties.xml`.**
How Do I Remove the Course Reserve Tab?

Description For “How Do I Remove the Course Reserve Tab?” Example
Files
Instructions
How Do I Remove the Course Reserve Tab?

Description For “How Do I Remove the Course Reserve Tab?” Example

Course reserves may be an optional requirement for your site. This example describes how to remove the Course Reserve tab from the Search page.

Files

The examples in this chapter use the following files:

- `pageProperties.xml`
- `index.html`

Instructions

This section provides the instructions for completing the examples described in this chapter.
Procedure 24-1. Remove the Course Reserve Tab From the Search Page

Use the following procedure to remove the **Course Reserve** tab.

**NOTE:** Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is `en_US` as in the following:

```
/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. **Make a backup copy of** `pageProperties.xml` **that is located in** `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/userTextConfigs/`

2. **Edit** `pageProperties.xml`.
   a. **Find the Search Tab Display Order section marked by the comment shown in Figure 24-1.**

```
<searchTabDisplayOrder>
    <tab name="page.search.buttons.basic.button"/>
    <tab name="page.search.buttons.advanced.button"/>
    <tab name="page.search.buttons.subjectHeading.button"/>
    <tab name="page.search.buttons.author.button"/>
    <tab name="page.search.buttons.courseReserve.button"/>
    <!-- enable these lines if site has geospatial searching
    <tab name="page.search.buttons.geospatial.button"/>
    <tab name="page.search.geospatial.button"/>
    -->
</searchTabDisplayOrder>
```

Figure 24-1. **Search Tab Display Order section**

   b. **Comment out the** `courseReserve` **line in this section. See Figure 24-2.**
3. Turn off the reference to course reserves on the default (index.html) page.

4. Make a backup copy of index.html that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/htdocs/.

5. Edit index.html.
   a. Find the More choices paragraph. See Figure 24-3.
b. Comment out the `courseReserve` line in this section. See Figure 24-4.

---

Figure 24-3. More choices paragraph example

- Basic search
- Advanced search
- Course reserve materials
- Log in to use your saved preferences
- Review your account
- Read help for WebVoyage

Figure 24-4. Comment out course reserve href example

6. Save and test your changes.

OPTIONAL:
7. Back out your changes, if necessary, by deploying your backup file copies.
How Do I Add A New Search Tab?

Description For “How Do I Add A New Search Tab?”

Example 22-1
Files 22-1
Instructions 22-1
How Do I Add A New Search Tab?

Description For “How Do I Add A New Search Tab?” Example

This chapter describes how to add a new search tab.

The new search tab requires that you make a cgi or static HTML file that it can point to.

Files

The examples in this chapter use the following files:

- webvoyage.properties.
- internal.properties.
- pageProperties.xml.

Instructions

This section provides the instructions for creating the examples described in this chapter.
Procedure 25-1. Create New Search Tab

Use the following procedure to create a new search tab for new books.

This procedure assumes that a `newBooks.cgi` file has been created.

**NOTE:**
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is `en_US` as in the following:

```
/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Make a backup copy of the `webvoyage.properties` file that is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/`.

2. Add labels for the new tab in the Search Pages section. See Figure 25-1.

   ```
   page.search.buttons.newBooks.button=New Books
   page.search.buttons.newBooks.message=New Books
   ```

   **Figure 25-1. Labels for new search tab example**

3. Make a backup copy of the `internal.properties` file that is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/`.

4. Bind the new tab to an action. Add the code shown in Figure 25-2 to the Search section of `internal.properties`.

   ```
   page.search.buttons.newBooks.action=newBooks.cgi
   ```

   **Figure 25-2. Bind new tab in Search section example**

5. Make a backup copy of the `pageProperties.xml` file that is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xs1/userTextConfigs/`.
Chapter 25: How Do I Add A New Search Tab?

6. Add the XML to create the new tab. Find the `searchTabDisplayOrder` element and add the code shown in Figure 25-3 to `pageProperties.xml`.

   ```xml
   <tab name="page.search.buttons.newBooks.button"/>
   
   Figure 25-3. Example XML for new search tab
   
   Place the code where you would like the new search tab to display. See Figure 25-4 for an example.
   
   ```xml
   <searchTabDisplayOrder>
     <tab name="page.search.buttons.basic.button"/>
     <tab name="page.search.buttons.advanced.button"/>
     <tab name="page.search.buttons.subjectHeading.button"/>
     <tab name="page.search.buttons.author.button"/>
     <tab name="page.search.buttons.courseReserve.button"/>
     <tab name="page.search.buttons.newBooks.button"/>
     <!-- enable these lines if site has geospatial searching
     <tab name="page.search.buttons.geospatial.button"/>
     <tab name="page.search.geospatial.button"/>
     -->
   </searchTabDisplayOrder>
   
   Figure 25-4. Example placement of XML code
   
   NOTE:
The tab name should correspond with the label code such as `newBooks` established in `webvoyage.properties` added as in Step 2.

7. Save and test your changes.

   OPTIONAL:
   8. Back out your changes, if necessary, by deploying your backup file copies.
How Do I Add A New Header Tab?

Description For “How Do I Add A New Header Tab?”
Example 23-1
Files 23-1
Instructions 23-1
How Do I Add A New Header Tab?

Description For “How Do I Add A New Header Tab?” Example

This chapter describes how to add a new header tab.

Files

The examples in this chapter use the following files:

- webvoyage.properties.
- internal.properties.
- pageProperties.xml.

Instructions

This section provides the instructions for creating the examples described in this chapter.
Procedure 26-1. Create New Header Tab

Use the following procedure to create a new header tab to exit the session.

**NOTE:**
Directory path references to `xxxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is `en_US` as in the following:

```
/m1/voyager/xxxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Identify a unique variable that you can use to bind properties to the exit session action across all the files you edit. Choose a variable that clearly communicates the tab you're describing. For this example, the variable used is `exitSession`.

2. Make a backup copy of the `pageProperties.xml` file that is located in `/m1/voyager/xxxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/userTextConfigs/`.

3. Locate the `<!-- ## Header Tab Display Order ## -->` comment in `pageProperties.xml`, and add new header tab element between the `<headerTabDisplayOrder>` and `</headerTabDisplayOrder>` tags. Where you place it depends on the order in which you want it to display.

   To add it to the right of the existing tabs, place the new element immediately before the `</headerTabDisplayOrder>` tag. See **Figure 26-1**.

   ```xml
   <!-- The following element will create a new tab in the header -->
   <tab name="page.header.buttons.exitSession.button" />
   ``

   **Figure 26-1.** pageProperties example for new header tab

**NOTE:**
The value of the name attribute needs to match the `.button` variable set in `webvoyage.properties` (see Step 5).

4. Make a backup copy of the `webvoyage.properties` file that is located in `/m1/voyager/xxxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/`. 
5. Set the new header tab label in the Header section of `webvoyage.properties` as in Figure 26-2.

```properties
#Custom "Exit Session" tab labels
page.header.buttons.exitSession.button=Exit Session
page.header.buttons.exitSession.message=Exit the catalog
```

Figure 26-2. Header tab label example in `webvoyage.properties`

The `.button` value is the text that displays on the header tab, and the `.message` value is the alternative text that displays for mouse hovering and screen readers.

6. Make a backup copy of the `internal.properties` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/`.

7. Set the header tab's action in the Header section of the `internal.properties` file as in Figure 26-3.

```properties
#Custom "Exit Session" tab action
page.header.buttons.exitSession.action=exit.do
```

Figure 26-3. `internal.properties` file example changes

The `.action` value is a URL. This can be an existing WebVoyage action or a complete URL such as `http://catalog.loc.gov/`.

8. Save and test your changes.

**OPTIONAL:**

9. Back out your changes, if necessary, by deploying your backup file copies.
How Do I Create Additional Record Views?

Description For “How Do I Create Additional Record Views?” Example 24-1
Files 24-1
Instructions 24-1
How Do I Create Additional Record Views?

Description For “How Do I Create Additional Record Views?” Example

This chapter describes how to create additional record views such as brief and full views.

Files

The examples in this chapter use the following files:

• displayRecord.xsl.
• cl_displayRecord.xsl.
• cl_displayStaff.xsl.
• displaycfg.xml.
• displayFacets.xsl.
• web.xml.

Instructions

This section provides the instructions for creating the examples described in this chapter.
Procedure 27-1. Create Additional Record Views

Use the following procedure to create additional record views.

NOTE:
Directory path references to xxxdb implies that you need to substitute your
database path name; and where [skin] is referenced, substitute the path name
that is used at your site. The default skin path provided is en_US as in the
following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Copy displayRecord.xsl that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/ to displayBriefRecord.xsl.

2. Modify existing code (see Figure 27-1) in displayBriefRecord.xsl with the
code shown in Figure 27-2.

3. Copy cl_displayRecord.xsl that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/ to cl_displayBriefRecord.xsl.

4. Modify existing code (see Figure 27-3) in cl_displayBriefRecord.xsl with the
code shown in Figure 27-4.
5. Modify the existing Action Box code (see Figure 27-5) in cl_displayBriefRecord.xsl with the code shown in Figure 27-6.

<!-- ## Action Box ## -->

```xml
<xsl:call-template name="buildActionBox">
  <xsl:with-param name="pageRecordType" select="'actionBox.recordView.link'"/>
</xsl:call-template>
```

Figure 27-5. Existing Action Box code

<!-- ## Action Box ## -->

```xml
<xsl:variable name="moreActions">
  <li><span class="recordPointer">Brief Record</span></li>
</xsl:variable>
```

Figure 27-6. Example modification of Action Box code
6. Make a backup copy of the `cl_displayRecord.xsl` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.

7. Modify the existing Action Box code (see Figure 27-7) in `cl_displayRecord.xsl` with the code shown in Figure 27-8.

![Figure 27-6. Example modification of Action Box code (Continued)](image)

```xml
<xsl:call-template name="buildActionBox">
  <xsl:with-param name="pageRecordType" select="'actionBox.briefRecordView.link'"/>
  <xsl:with-param name="moreActions" select="$moreActions"/>
</xsl:call-template>

<!-- ## Action Box ## -->
<xsl:call-template name="buildActionBox">
  <xsl:with-param name="pageRecordType" select="'actionBox.recordView.link'"/>
</xsl:call-template>

Figure 27-6. Example modification of Action Box code (Continued)

![Figure 27-7. Existing Action Box code in cl_displayRecord](image)

```xml
<!-- ## Action Box ## -->
<xsl:call-template name="buildActionBox">
  <xsl:with-param name="pageRecordType" select="'actionBox.recordView.link'"/>
</xsl:call-template>

Figure 27-7. Existing Action Box code in cl_displayRecord
Chapter 27: How Do I Create Additional Record Views?

8. Modify the existing Action Box code (see Figure 27-9) in `cl_displayStaff.xsl` with the code shown in Figure 27-10.

Figure 27-8. Example modification to Action Box code in `cl_displayRecord`

```xml
<!-- ## Action Box ## -->

<xsl:variable name="briefRecordURL"><xsl:value-of select="substring-after(//
page:element[@nameId='actionBox.recordView.link']/
page:URL,'holdingsInfo?')"/></xsl:variable>

<xsl:variable name="moreActions">
  <li><span class="recordLinkBullet">·</span><a href="{$briefRecordURL}"><span>Brief Record</span></a></li>
</xsl:variable>

<xsl:call-template name="buildActionBox">
  <xsl:with-param name="pageRecordType" select="'actionBox.recordView.link'"/>
  <xsl:with-param name="moreActions" select="$moreActions"/>
</xsl:call-template>

Figure 27-9. Existing Action Box code in `cl_displayStaff`

```xml
<!-- ## Action Box ## -->

<xsl:call-template name="buildActionBox">
  <xsl:with-param name="pageRecordType" select="'actionBox.staffView.link'"/>
</xsl:call-template>

Figure 27-10. Code snippet for modifying Action Box in `cl_displayStaff.xsl`
```
9. Copy displaycfg.xml that is located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/ to displaybriefcfg.xml

10. Modify displaybriefcfg.xml with your preferences.

11. Make a backup copy of the displayFacets.xsl file that is located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/.

12. Modify existing code (see Figure 27-11) in displayFacets.xsl with the code shown in Figure 27-12.
Chapter 27: How Do I Create Additional Record Views?

13. Add the code shown in Figure 27-13 after the code shown in Figure 27-14 in the displayFacets.xsl file.

```
<xsl:for-each select="page:element">
    <xsl:choose>
        <xsl:when test="@nameId=$pageRecordType">
            <li><span class="recordPointer">&#160;</span><label><xsl:value-of select="page:linkText"/></label></li>
        </xsl:when>
    </xsl:choose>
</xsl:for-each>
```

Figure 27-14. Existing code in displayFacets.xsl
14. Make a backup copy of the web.xml file that is located in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/WEB-INF/.

15. Add the code shown in Figure 27-17 to web.xml after the last <filter-mapping> stanza that references SelectSkin Filter but before any other type of stanza.

```xml
<filter-mapping>
    <filter-name>SelectSkin Filter</filter-name>
    <url-pattern>/briefHoldingsView/*</url-pattern>
</filter-mapping>
```

Figure 27-15. filter-mapping code example

16. Add the code shown in Figure 27-16 to web.xml after the last <servlet-mapping> stanza but before any other type of stanza.

```xml
<!--Figure 27-16. Existing code in displayFacets.xsl (Continued)-->
Chapter 27: How Do I Create Additional Record Views?

17. Add the code shown in Figure 27-17 to web.xml after the last <servlet> stanza but before any other type of stanza.

```
<servlet-mapping>
  <servlet-name>BriefHoldingsInfoServlet</servlet-name>
  <url-pattern>/briefHoldingsInfo/*</url-pattern>
</servlet-mapping>
```

**Figure 27-16. servlet-mapping code example**

```
<servlet>
  <servlet-name>BriefHoldingsInfoServlet</servlet-name>
  <display-name>BriefHoldingsInfoServlet</display-name>
  <servlet-class>
    com.endinfosys.voyager.webvoyage.servlet.PageServlet
  </servlet-class>
  <init-param>
    <param-name>PageCode</param-name>
    <param-value>briefHoldingsInfo</param-value>
  </init-param>
</servlet>
```

**Figure 27-17. Example modification code for web.xml**
Figure 27-17. Example modification code for web.xml (Continued)
Chapter 27: How Do I Create Additional Record Views?

18. Add the code shown in Figure 27-18 at the end of the following file:

```
/m1/shared/apache2/conf/ConfiguredVirtualHosts/
XXXdb.jkmounts.conf
```

The bri in Figure 27-18 matches the beginning of the name of the new brief holdings view (briefHoldingsInfo).

NOTE:
This change requires that Apache be restarted.

19. Save and test your changes.

OPTIONAL:
20. Back out your changes, if necessary, by deploying your backup file copies.
How Do I Implement DOI and URN Handling?

DOI/URN Overview

WebVoyage can display links to URN and DOI resources in MARC records. URN stands for Uniform Resource Name, and DOI stands for Digital Object Identifier.

Unlike the URL (Uniform Resource Locator) address, the URN or DOI in the 856 field of the MARC record does not point directly to a digital item. Instead, they are routed through a handler server that maps them to the physical location of the digital item.

For information about entering URN or DOI links in the 856 field of a MARC record, see the Voyager Cataloging User’s Guide.

Files

WebVoyage uses the webvoyage.properties file to implement the DOI/URN handling feature.

DOI/URN Implementation

This section describes the WebVoyage implementation for DOI/URN handling.
NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:


webvoyage.properties

To implement DOI and/or URN handling, you need to replace the http://hdl.handle.net/ variable with the specific URL you want to replace DOI and/or URN when the MARC 856 field is processed. This variable is located in the webvoyage.properties file located in the following path:


See Table 28-1 for the default code/configuration provided at installation.

Table 28-1. DOI/URN Implementation

<table>
<thead>
<tr>
<th># Print property DOI and URN URL to convert the DOI and URN to the physical location</th>
</tr>
</thead>
<tbody>
<tr>
<td># of the digital item</td>
</tr>
<tr>
<td>===========================================================================#</td>
</tr>
<tr>
<td>property.DOI=<a href="http://hdl.handle.net/">http://hdl.handle.net/</a></td>
</tr>
<tr>
<td>property.URN=<a href="http://hdl.handle.net/">http://hdl.handle.net/</a></td>
</tr>
</tbody>
</table>

NOTE:
If the DOI or URN variable is not configured, the DOI and/or URN is replaced by the default URL http://<host>:<port>/vwebv.

If there is an occurrence where both the DOI and URN exist, the URN takes priority over the DOI.
How Do I Implement Hook to Holdings (Citation Server)?
How Do I Implement Hook to Holdings (Citation Server)?

Hook to Holdings Implementation

The Hook to Holdings functionality enables WebVoyage to display holdings data from the local database when a remote (zcit or vcit) database is searched. See Figure 29-1 for a holdings display example. In Figure 29-1, the portion of the display above the line is from a citation database; and the portion below the line is from the local database.

This is accomplished by comparing specified fields, typically the 022‡a and 773‡x, to determine if the value in the remote bibliographic record's field matches a value in the local bibliographic record's field. When a match exists the holdings data from the local bibliographic record is displayed with the remote citation database bibliographic data.

To implement this feature, the following needs to occur:

- A Hook to Holdings profile in Voyager System Administration needs to be created and linked to the citation database.
- The citation database needs to be defined as a remote database.

For additional information, see Voyager Systems Administration User’s Guide and Citation Server User’s Guide.
Figure 29-1. Hook to Holdings display example
How Do I Implement HTTP Post to Link Resolver?

HTTP POST to Link Resolver Overview 30-1
Files 30-1
HTTP POST to Link Resolver Implementation 30-1
• voyager.ini 30-2
• linkresolver.properties 30-4
  OpenURL Standard 30-6
How Do I Implement HTTP Post to Link Resolver?

HTTP POST to Link Resolver Overview

The HTTP POST to link resolver functionality sends a MARC bibliographic record to a designated link resolver.

The workflow for this function can be initiated in any Voyager client that allows MARC viewing. For example, the Voyager cataloging client can invoke link resolving with the Record > Send Record To > Linkresolver menu options when a MARC record is displayed.

Files

The implementation of this feature uses the following configuration files:

- linkresolver.properties.
- voyager.ini.

HTTP POST to Link Resolver Implementation

To implement the link resolver function, you need to configure the linkresolver.properties file that resides on the WebVoyage server and the voyager.ini file that resides on the client PC.
To configure the voyager.ini file, edit the [MARC POSTing] stanza. See Figure 30-1 for an example.

```ini
[MARC POSTing]
WebVoyage="http://<host>:<port>/vwebv/holdingsInfo"
Redirect to link resolver="http://<host>:<port>/vwebv/linkResolver"
```

**Figure 30-1. voyager.ini configuration example**

The [MARC POSTing] stanza is designed to display the MARC record in WebVoyage when Record > Send Record To > WebVoyage is clicked or pass the record to a link resolver when Record > Send Record To > Linkresolver is clicked. See Figure 30-2.

**NOTE:**
The link resolver function assumes that when a user accesses Record > Send Record To > Linkresolver from the staff client that the user is connected to the LOCAL database as defined in Voyager System Administration Database Definitions.
Chapter 30: How Do I Implement HTTP Post to Link Resolver?

Figure 30-2. Linkresolver option example
The `linkresolver.properties` file is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/` where `xxxdb` is your database name. In the `linkresolver.properties` file, you need to do the following:

- Specify the root URL of your link resolver such as your SFX instance, for example. See Figure 30-3.
- Identify the fields to be used for the OpenURL construction and how to parse them. See Figure 30-4 for a citation database example.

```plaintext
openUrl.cfg.LOCAL.urlRoot=http://<host>:<port>/<SFX instance>
```

**Figure 30-3. Link Resolver URL example**

For each field identified (as in Figure 30-4), the following keys need to be specified:

- Field name.
- Tag/field number.
- Subfield.
- Length 1 (number of positions).
- Parse start.
- Parse end.
- Length 2 (number of positions).

In the Figure 30-4 example, the following fields are defined:

- Title.
- Author’s last name.
- Author’s first name.
- Volume.
- Date/Year.
- Pages.
- ISSN.
openUrl.cfg.LOCAL.key1.key=title
openUrl.cfg.LOCAL.key1.tag=245
openUrl.cfg.LOCAL.key1.subfield=a
openUrl.cfg.LOCAL.key1.len1=0
openUrl.cfg.LOCAL.key1.parseStart=
openUrl.cfg.LOCAL.key1.parseEnd=
openUrl.cfg.LOCAL.key1.len2=0

openUrl.cfg.LOCAL.key2.key=aulast
openUrl.cfg.LOCAL.key2.tag=100
openUrl.cfg.LOCAL.key2.subfield=a
openUrl.cfg.LOCAL.key2.len1=0
openUrl.cfg.LOCAL.key2.parseStart=
openUrl.cfg.LOCAL.key2.parseEnd=,
openUrl.cfg.LOCAL.key2.len2=0

openUrl.cfg.LOCAL.key3.key=aufirst
openUrl.cfg.LOCAL.key3.tag=100
openUrl.cfg.LOCAL.key3.subfield=a
openUrl.cfg.LOCAL.key3.len1=0
openUrl.cfg.LOCAL.key3.parseStart=,
openUrl.cfg.LOCAL.key3.parseEnd=
openUrl.cfg.LOCAL.key3.len2=0

openUrl.cfg.LOCAL.key4.key=volume
openUrl.cfg.LOCAL.key4.tag=773
openUrl.cfg.LOCAL.key4.subfield=g
openUrl.cfg.LOCAL.key4.len1=0
openUrl.cfg.LOCAL.key4.parseStart=v.
openUrl.cfg.LOCAL.key4.parseEnd=p.
openUrl.cfg.LOCAL.key4.len2=0

openUrl.cfg.LOCAL.key6.key=date-year
openUrl.cfg.LOCAL.key6.tag=903
openUrl.cfg.LOCAL.key6.subfield=a
openUrl.cfg.LOCAL.key6.len1=4

Figure 30-4. Example of fields/subfields identified
OpenURL Standard

The OpenURL standard identifies the key fields available for use such as title, auLast, auFirst, and so on. WebVoyage follows the OpenURL standard for the HTTP POST to link resolver functionality. See the “ANSI/NISO Z39.88 - The OpenURL Framework for Context-Sensitive Services” standards document available at www.niso.org that describes the OpenURL standard.
How Do I Display Media Bookings in MyAccount?

Media Bookings Overview 28-1
Files 28-1
Media Bookings Implementation 28-2
How Do I Display Media Bookings in MyAccount?

Media Bookings Overview

When configured, Media Scheduling bookings activity displays on My Account page in WebVoyage. A link is provided from Your Items action box to the following:

- Upcoming Bookings.
- Charged Bookings.

With this function, you can display and cancel active bookings.

NOTE:
Charged bookings may not be renewed or cancelled.

Files

WebVoyage uses the webvoyage.properties file to implement the media bookings feature.
Media Bookings Implementation

To implement the media bookings function, you need to configure the `webvoyage.properties` file that resides on the WebVoyage server. The `webvoyage.properties` file is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/` where `xxxdb` is your database name.

In the `webvoyage.properties` file, you need to do the following:

- Set the `option.mediaBookingsPatronInfoDisplay` parameter to `Y` (Yes). See Figure 31-1.

  ![option.mediaBookingsPatronInfoDisplay=Y](image1)

  **Figure 31-1. Media bookings parameter setting in WebVoyage**

- Set the `page.myAccount.mediaBookings.cancelAllowed` parameter to `Y` (Yes). See Figure 31-2.

  ![page.myAccount.mediaBookings.cancelAllowed=Y](image2)

  **Figure 31-2. Media bookings cancel allowed parameter example**

  This is an optional setting.

  This parameter causes the following to be displayed in the Upcoming Bookings section:

  - Individual check box.
  - Cancel All button.
  - Reset button.

- Modify label display and/or media bookings display options. See Figure 31-3 for the default settings.

  ![page.myAccount.mediaBookings.itemInfo](image3)

  **Figure 31-3. Media bookings itemInfo parameter**

  The `page.myAccount.mediaBookings.itemInfo` parameter may be set to the options in Table 31-1.
Chapter 31: How Do I Display Media Bookings in MyAccount?

Figure 31-3. Media bookings display options

Table 31-1. page.myAccount.mediaBookings.itemInfo options

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\t</td>
<td>the item’s title</td>
</tr>
<tr>
<td>\i</td>
<td>the item’s enumeration, chronology and year</td>
</tr>
<tr>
<td>\n</td>
<td>the item’s copy number</td>
</tr>
<tr>
<td>\c</td>
<td>the item’s call number</td>
</tr>
<tr>
<td>\b</td>
<td>the item’s barcode</td>
</tr>
<tr>
<td>\l</td>
<td>the item’s location</td>
</tr>
<tr>
<td>Options</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------</td>
</tr>
<tr>
<td>\a</td>
<td>the item’s author</td>
</tr>
</tbody>
</table>
How Do I Implement ImageServer in WebVoyage?

WebVoyage ImageServer Overview 29-1
Files 29-1
ImageServer Implementation 29-2
• display.xsl 29-3
• resultsFacets.xsl 29-3
WebVoyage Architecture Overview and Configuration Models
How Do I Implement ImageServer in WebVoyage?

WebVoyage ImageServer Overview

For locations with ImageServer installed, WebVoyage can retrieve and display thumbnail- or actual-resolution-size images. The thumbnail displays on the search results Titles page.

The scanned document (image) link is retrieved based on information stored in 856‡f. The 856‡z provides the description of the image that is highlighted on the search results Titles display.

NOTE:
If you have both ImageServer and a cover titles service installed, the cover image displays on the search results Titles page when there is both an ImageServer thumbnail image and cover image available to display.

Files

WebVoyage uses the following files to implement the ImageServer feature:

- webvoyage.properties.
- display.xsl.
- resultsFacets.xsl.
ImageServer Implementation

To implement the ImageServer function in WebVoyage, you need to configure the webvoyage.properties file that resides on the WebVoyage server. The webvoyage.properties file is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/ where xxxdb is your database name.

In the webvoyage.properties file, you can activate and customize this feature for your site. See Figure 32-1 and Figure 32-2.

---

```properties
# Option to activate (True) or deactivate (False) thumbnail
# Default is deactivate thumbnail
#
# option.thumbnail.activate=False
option.thumbnail.activate=True
```

Figure 32-1. webvoyage.properties ImageServer configuration example

---

```properties
# Image Server configuration for link, thumbnail alter text
#
#imagerServer.scanDoc=http://localhost:classicWebVoyagePort/cgi-bin/Pscandoc.cgi?
#imageServer.scandocAltText=Scan Document
#imageServer.thumbnailAltText=Thumbnail
#imageServer.loginRequiredText=**** Login Required ****
```

Figure 32-2. webvoyage.properties ImageServer configuration example

---

Procedure 32-1. Implement ImageServer Function in WebVoyage

To implement ImageServer function for WebVoyage, do the following:
Chapter 32: How Do I Implement ImageServer in WebVoyage?

1. Confirm that the `option.thumbnail.activate` variable is active and set to the value of `True`. See Figure 32-1.

2. Remove any comment notation (#) to activate the `imageServer.<xxxxx>=` variables.

3. Customize the `imageServer.scanDoc` variable to reflect your site's URL or IP address of the server running scandoc and the port of the Classic WebVoyage.

4. Save your changes to the `webvoyage.properties` file.

---

display.xsl

The `display.xsl` file is used to extract the 856‡f to link to the image.

It is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/xsl/contentLayout/display/` where `xxxdb` is your database name.

resultsFacets.xsl

The `resultsFacets.xsl` file provides the template to create the title result, jump bar, and so on for the search results Titles page.

It is located in `/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/xsl/pageFacets/` where `xxxdb` is your database name.
How Do I Implement Messages for Status Patron Groups?

Overview of Messages for Status Patron Groups 33-1
Files 33-2
Instructions 33-2
How Do I Implement Messages for Status Patron Groups?

Overview of Messages for Status Patron Groups

For each status patron group, you need to create an entry in webvoyage.properties that contains the code of the patron group defined in Voyager System Administration (see Figure 33-1) and the message to be displayed on the title and holdings pages. See Implement Messages for Status Patron Groups on page 33-2 for the steps to do this.

Figure 33-1. Example of Patron Groups in Voyager System Administration
Files

WebVoyâge uses the following file to implement messages for status patron groups:

- webvoyage.properties.

Instructions

This section provides instructions for how to implement messages for status patron groups.

Procedure 33-1. Implement Messages for Status Patron Groups

To implement messages for status patron groups that display on the title and holdings pages, do the following:

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Make a backup copy of the webvoyage.properties file that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/.

2. Locate the status patron groups messages section of the webvoyage.properties file. See Figure 33-2 for an example.

```
# Status Patron Groups messages for display on title and holdings pages
```

Figure 33-2. Example of status patron groups messages section
Chapter 33: How Do I Implement Messages for Status Patron Groups?

3. Edit the status patron groups messages section to include entries for each patron group defined in Voyager System Administration. See Figure 33-3 for an example entry for the CARREL patron group code.

TIP: You may imbed substitution tokens in your messages as in \i shelved at Carrel \l until \d. See Table 33-1 for a list of tokens.

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\i</td>
<td>Standard item information.</td>
</tr>
<tr>
<td>\c</td>
<td>Outstanding number of requests.</td>
</tr>
<tr>
<td>\d</td>
<td>Date.</td>
</tr>
</tbody>
</table>

Figure 33-2. Example of status patron groups messages section (Continued)

Figure 33-3. Example of status patron groups messages customized
### Table 33-1. Substitution Tokens

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\e</code></td>
<td>Date unless date is today.</td>
</tr>
<tr>
<td><code>\t</code></td>
<td>Time.</td>
</tr>
<tr>
<td><code>\u</code></td>
<td>Time if date is today.</td>
</tr>
<tr>
<td><code>\l</code></td>
<td>Location.</td>
</tr>
<tr>
<td><code>\F</code></td>
<td>First name field from the patron record. <strong>NOTE:</strong> Only for patron status groups.</td>
</tr>
<tr>
<td><code>\L</code></td>
<td>Last name field from the patron record. <strong>NOTE:</strong> Only for patron status groups.</td>
</tr>
<tr>
<td><code>\b</code></td>
<td>Number representing the limit for a certain item request/fine.</td>
</tr>
<tr>
<td><code>\p</code></td>
<td>Number representing the amount of fines/requests that you have.</td>
</tr>
<tr>
<td><code>\n</code></td>
<td>Item loan period.</td>
</tr>
<tr>
<td><code>\rsloc</code></td>
<td>Routing Source Location. Routing source is the place where the item was discharged.</td>
</tr>
<tr>
<td><code>\rslib</code></td>
<td>Routing Source Library. Routing source is the place where the item was discharged.</td>
</tr>
<tr>
<td><code>\rtloc</code></td>
<td>Routing Target Location. Routing target is the destination.</td>
</tr>
<tr>
<td><code>\rtlib</code></td>
<td>Routing Target Location. Routing target is the destination.</td>
</tr>
</tbody>
</table>

4. Save your changes to the `webvoyage.properties` file.
How Do I Add/Modify Search Results Page Icons?

Add/Modify Search Results Page Icons Overview 34-1
Files 34-1
Instructions 34-1
How Do I Add/Modify Search Results Page Icons?

Add/Modify Search Results Page Icons
Overview

This chapter describes how to add or modify icons used on the search results page for various item types.

You can customize the labels and images (icons) for all item types with this capability.

Files

WebVoyage uses the following file to implement icons for various item types on the search results page:

- pageProperties.xml

Instructions

This section provides instructions for how to implement icons for various item types on the search results page.
Procedure 34-1. Add/Modify Search Results Page Icons

To implement icons on the search results page, do the following:

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Make a backup copy of the pageProperties.xml file that is located in /ml/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/userTextConfigs/.

2. Locate the <bibFormats> section of the pageProperties.xml file. See Figure 34-1 for an example.

```xml
<!-- This section defines the text and the icon to use (for results list icons) for bibFormat -->
<bibFormats>
  <bibFormat type="aa" icon="icon_book.gif">Book</bibFormat>
  <bibFormat type="ab" icon="icon_serial.gif">Periodical</bibFormat>
  <bibFormat type="ac" icon="icon_pamphlet.gif">Book</bibFormat>
  <bibFormat type="ad" icon="icon_book.gif">Book</bibFormat>
  <bibFormat type="am" icon="icon_book.gif">Book</bibFormat>
  <bibFormat type="as" icon="icon_serial.gif">Periodical</bibFormat>
  <bibFormat type="ba" icon="icon_manuscript.gif">Archival/Manuscript Material</bibFormat>
  <bibFormat type="bb" icon="icon_serial.gif">Periodical</bibFormat>
  <bibFormat type="bc" icon="icon_manuscript.gif">Archival/Manuscript Material</bibFormat>
</bibFormats>
```

Figure 34-1. pageProperties <bibFormats> example
3. Edit the `<bibFormats>` section of the pageProperties.xml file to specify the following (see Table 34-1):

- The item type.
- The icon image file to be used.
- The label to display.

Table 34-1. `<bibFormats>` options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type=</td>
<td>Use to specify the item type.</td>
</tr>
<tr>
<td>icon=</td>
<td>Use to specify the name of the image.</td>
</tr>
</tbody>
</table>

**NOTE:** The images are located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/bibFormat/`.

4. Save your changes to the pageProperties.xml file.
How Do I Modify the Renewal Status Messages?

Modify the Renewal Status Messages Overview 35-1
Files 35-1
Instructions 35-3
How Do I Modify the Renewal Status Messages?

Modify the Renewal Status Messages
Overview

This chapter describes how to modify renewal status messages.

You can customize the success/failure messages that display to patrons when a renewal is requested. See Figure 35-1 on page 35-2 and Figure 35-2 on page 35-3.

Files

WebVoyage uses the following file to store renewal messages for display when patrons request renewals:

- webvoyage.properties..
Figure 35-1. Renewal Status Message (Successful)
Chapter 35: How Do I Modify the Renewal Status Messages?

Figure 35-2. Renewal Status Message (Failed)

Instructions

This section provides instructions for how to modify renewal status messages.

Procedure 35-1. Modify Renewal Messages

To modify renewal status messages, do the following:

NOTE: Directory path references to \xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name.
that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Make a backup copy of the webvoyage.properties file that is located in /m1/voyager/xxxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/.

2. Locate the MyAccount page charged item renewal message section of the webvoyage.properties file. See Figure 35-3 for an example.

```java
# MyAccount page charged item renewal status.
# This status is displayed only after the renewal of item or items is performed
page.myAccount.chargedItem.item.renewStatus.renewed=Renewed
page.myAccount.chargedItem.item.renewStatus.notRenewed=Not Renewed

# MyAccount page charged item renewal message.
# This message is displayed only after the renewal of item or items is performed
page.myAccount.chargedItem.item.renewMsg.success=All items are renewed successfully
page.myAccount.chargedItem.item.renewMsg.fail=One or more items failed to renew (Please see status below for detail)
```

Figure 35-3. Example of renewal status messages

3. Edit the renewal status messages to your preference.

4. Save your changes to the webvoyage.properties file.
# Index

## A

- Advanced Search, 2-7
- architecture overview, 2-1
- Author Search, 2-10
- auto complete
disable, 13-1

## B

- Basic Search, 2-5
- bibliographic record
  - relationship
displaying in WebVoyâge, 5-2
  - maintaining, 5-2
- buildBasicSearch, 12-7
- buildContent, 2-3
- buildCoverImage, 20-6
- buildCoverImageLinks, 20-7
- buildHtmlPage, 2-3, 14-2
- buildMarcDisplay, 17-3
- buildResultsCoverImage, 20-3
- buildSearchButtons, 15-3
- cl_searchGeoCorridor.xsl, 2-14
- cl_searchGeoPolygon.xsl, 2-14
- cl_searchGeoRadius.xsl, 2-14
- cl_searchGeoRange.xsl, 2-14
- cl_searchGeoRectangle.xsl, 2-14
- cl_searchSubject.xsl, 2-10
- constants.xsl, 2-6, 2-8, 2-9, 2-11, 2-12, 2-14
- constantStrings.xsl, 2-6, 2-8, 2-9, 2-11, 2-12, 2-14
- Course Reserves, 2-12
- CSS files
displayCommon.css, 16-1, 16-5, 17-5
displayGoogleBooks.css, 19-1, 19-3
frameWork.css, 2-5, 2-6, 2-7, 2-8, 2-9, 2-10, 2-11, 2-12, 2-13, 2-14, 2-15
header.css, 2-5, 2-6, 2-8, 2-9, 2-11, 2-13, 2-14
myAccount.css, 2-3, 2-5
pageProperties.css, 2-5, 2-6, 2-7, 2-8, 2-9, 2-10, 2-11, 2-13, 2-14, 2-15
quickSearchBar.css, 2-5, 2-6, 2-8, 2-9, 2-11, 2-13, 2-14
searchAdvanced.css, 2-8, 15-1, 15-4, 15-6
searchAuthor.css, 2-11
searchBasic.css, 2-7
searchCourseReserve.css, 2-13
searchGeospatial.css, 2-15
searchPages.css, 2-7, 2-8, 2-10, 2-11, 2-13, 2-14, 2-15
searchSubject.css, 2-10

## D

- default page, 24-3
- display cover images
  - Syndetics Solutions, 20-1
  - display media bookings, 31-1
  - display.xsl, 10-1, 10-4, 10-5, 17-1, 17-2, 17-5, 20-2, 20-6, 32-1, 32-3
displaycfg.xml, 7-1, 17-1, 17-2, 17-5, 20-1, 20-6, 27-1, 27-6
displayChargedItems, 9-2
displayCommon.css, 16-1, 16-5, 17-5
displayFacets.xsl, 11-1, 11-4, 11-5, 19-1, 19-3, 19-4, 27-1, 27-6, 27-7
displayGoogleBooks.css, 19-1, 19-3
displayRecord.xsl, 2-2, 20-2, 20-7, 27-1, 27-2
N
no hits search results message, 23-1

P
page
 remove information, 9-1
page components
 Advanced Search, 2-7
 Author Search, 2-10
 Basic Search, 2-5
 Course Reserves, 2-12
 Geospatial Search, 2-14
 Subject Search, 2-9
pageInputFocus.js, 2-7, 2-8, 2-9, 2-10, 2-11, 2-12, 2-13, 2-15
pageMessages, 23-1
pageProperties.css, 2-5, 2-6, 2-7, 2-8, 2-9, 2-10, 2-11, 2-13, 2-14, 2-15
persistent link, 16-1
prerequisites, 1-1

Q
quickSearchBar.css, 2-5, 2-6, 2-8, 2-9, 2-11, 2-13, 2-14

R
record display page
 change format, 17-1
Related Records
 Cataloging module
 maintaining, 5-2
display in WebVoyage, 5-2
resultsFacets.xsl, 20-1, 20-3, 32-1, 32-3
resultsTitles.xsl, 20-1, 20-4

S
search tips
 dynamically change, 12-1
searchAdvanced.css, 2-8, 15-1, 15-4, 15-6
searchAuthor.css, 2-11
searchBasic.css, 2-7
searchCourseReserve.css, 2-13
searchFacets.xsl, 2-7, 2-8, 2-10, 2-11, 2-13, 2-14, 13-1, 13-2, 13-3
searchGeospatial.css, 2-15
searchPages.xsl, 2-7, 2-8, 2-10, 2-11, 2-13, 2-14, 2-15
searchSubject.css, 2-10
serials
 separate display for, 7-1
stdImports.xsl, 2-3
Subject Search, 2-9
Syndetics Solutions, 20-1

templates
 buildBasicSearch, 12-7
 buildContent, 2-3
 buildCoverImage, 20-6
 buildCoverImageLinks, 20-7
 buildHtmlPage, 2-3, 14-2
 buildMarcDisplay, 17-3
 buildResultsCoverImage, 20-3
 buildSearchButtons, 15-5
displayChargedItems, 9-2
googleBooksAvail, 19-2, 19-3
trimData, 20-3
tools.xsl, 2-6, 2-8, 2-9, 2-11, 2-12, 2-14
 tracking codes
 add, 18-1
trimData, 20-3
WebVoyage Architecture Overview and Configuration Models

U

Upcoming Bookings, 31-1

W

web.xml, 27-1, 27-8, 27-9
WebVoyage
displaying Related Records, 5-2
webvoyage.properties, 2-7, 2-8, 2-10, 2-11, 2-13, 2-15, 21-1, 21-2, 22-1, 22-2, 25-1, 25-2, 26-1, 26-2, 31-1, 31-2, 32-1, 32-2, 33-2, 35-1, 35-4

displayFacets.xsl, 11-1, 11-4, 11-5, 19-1, 19-3, 19-4, 27-1, 27-6, 27-7
displayRecord.xsl, 2-2, 20-2, 20-7, 27-1, 27-2, 27-4
footer.xsl, 8-1, 8-4, 8-5, 18-1, 18-2, 18-4
formInput.xsl, 2-6, 2-8, 2-9, 2-11, 2-12, 2-14
frameWork.xsl, 2-3, 2-6, 2-8, 2-9, 2-11, 2-12, 2-14, 14-1, 14-2, 14-3
header.xsl, 8-1, 8-2, 8-3, 8-4
local_googleBooksAvail.xsl, 19-1, 19-2
login.xsl, 13-1, 13-2, 13-3
myAccount.xsl, 2-2, 2-3
myAccountLinks.xsl, 2-3
resultsFacets, 32-1
resultsFacets.xsl, 20-1, 20-3, 32-3
resultsTitles, 20-1, 20-4
searchFacets.xsl, 2-7, 2-8, 2-10, 2-11, 2-13, 2-14, 13-1, 13-2, 13-3
stdImports.xsl, 2-3
tools.xsl, 2-6, 2-8, 2-9, 2-11, 2-12, 2-14

X

XML files
displaycfg.xml, 7-1, 17-1, 17-2, 17-5, 20-1, 20-6, 27-1, 27-6
pageProperties, 34-1
web.xml, 27-1, 27-8, 27-9

XSL files
cl_displayRecord.xsl, 7-1, 7-5, 7-7, 7-8, 16-1, 16-4, 16-5, 27-1
cl_displayStaff.xsl, 27-1, 27-5
cl_myAccount.xsl, 2-3, 9-1, 9-2, 9-3
cl_searchAdvanced.xsl, 2-8, 15-1, 15-5, 15-6
cl_searchAuthor.xsl, 2-11
cl_searchBasic.xsl, 2-6, 12-1, 12-7, 12-9
cl_searchCourseReserves.xsl, 2-13
cl_searchGeoCorridor.xsl, 2-14
cl_searchGeoPolygon.xsl, 2-14
cl_searchGeoRadius.xsl, 2-14
cl_searchGeoRange.xsl, 2-14
cl_searchGeoRectangle.xsl, 2-14
cl_searchSubject.xsl, 2-10
constants.xsl, 2-6, 2-8, 2-9, 2-11, 2-12, 2-14
constantStrings.xsl, 2-6, 2-8, 2-9, 2-11, 2-12, 2-14
display.xsl, 10-1, 10-4, 10-5, 17-1, 17-2, 17-5, 20-2, 20-6, 32-1, 32-3