Voyager® 10.2
WebVoyáge Architecture Overview and Configuration Models

October 2019

CONFIDENTIAL INFORMATION

DISCLAIMER

TRADEMARKS

Document released: October 2019

Web address: http://www.exlibrisgroup.com

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

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 reason:

- Replaced `BMD1000` with `BMD1005` in Figure 12-3 on page 12-5.

Document Summary

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

Chapter 2 "Architecture"
Chapter 2 provides an overview of the WebVoyáge architecture.

Chapter 3 "PDS"
Chapter 3 describes the full implementation of PDS with Voyager WebVoyáge.

Chapter 4 "Search"
Chapter 4 describes search capabilities in Voyager WebVoyáge 7.x and later.

Chapter 5 "Linking/OpenURL"
Chapter 5 describes the OpenURL capability available in Voyager WebVoyáge 7.x and later.

Chapter 6 "Bibliographic Record Linking"
Chapter 6 describes bibliographic record linking available in Voyager WebVoyáge 7.x and later.

Chapter 7 "Display Codes"
Chapter 7 describes display codes used in Voyager WebVoyáge 7.x and later.

Chapter 8 "How Do I Build A Separate Display For Serials?"
Chapter 8 describes how to build a separate display for serials.

Chapter 9 "How Do I Add Static Links To The Header Or Footer?"
Chapter 9 describes how to add static links to the header/footer area.

Chapter 10 "How Do I Remove Information From A Page?"
Chapter 10 describes how to remove information from a page.

Chapter 11 "How Do I Add A Map Or Other Information To A Location?"
Chapter 11 describes how to add a map or other information to a location.

Chapter 12 "How Do I Create An External Search From A Bibliographic Record Display?"
Chapter 12 describes how to create an external search.

Chapter 13 "How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?"
Chapter 13 describes how to dynamically disable limits and change search tips based on the selected search index.

Chapter 14 "How Do I Disable AutoComplete?"
Chapter 14 describes how to disable AutoComplete.

Chapter 15 "How Do I Display A Favicon?"
Chapter 15 describes how to create an icon for a browser tab or title bar display.

Chapter 16 "How Do I Hide Limits On The Advanced Search Page?"
Chapter 16 describes how to hide the limit options on an advanced search while, optionally, a user may click to see them.

Chapter 17 "How Do I Build And Display A Persistent Link To A Bibliographic Record?"
Chapter 17 describes how to dynamically build and display a persistent link to a bibliographic record.

Chapter 18 "How Do I Change The Format Of The Record Display Page?"
Chapter 18 describes how to add class attributes to the Detailed Display Page for improved formatting control.

Chapter 19 "How Do I Add Tracking Codes?"
Chapter 19 describes how to add tracking codes.

Chapter 20 "How Do I Implement Google Book Search?"
Chapter 20 provides instructions regarding the implementation of Google Book Search.

Chapter 21 "How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?"
Chapter 21 provides instructions for implementing Syndetics enhancements.

Chapter 22 "How Do I Implement Geospatial Search?"
Chapter 22 provides instructions for implementing geospatial search.

Chapter 38	"How Do I Suppress Patron Barcode/ID Prompts on Request Forms?" Chapter 38 provides instructions for how to suppress the barcode/ID prompt on the patron request page.
Chapter 39	"How Do I Customize Time Format for Media Equipment and Media Item Booking Forms?" Chapter 39 provides instructions for how to customize the time format for the Media Equipment Booking and Media Item Booking forms.
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

1

Purpose of this Chapter

The purpose of this chapter is to describe what you need to get started to effectively work with/customize WebVoyáge 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.

Architecture

2

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.

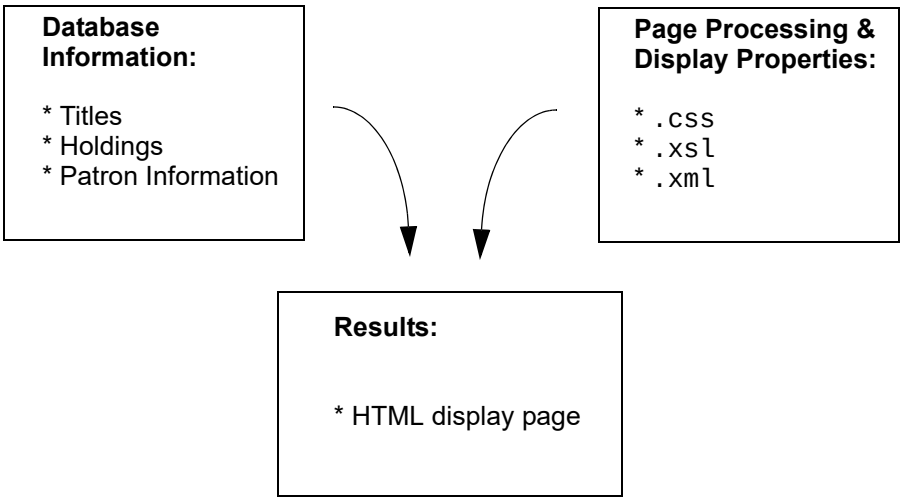

Figure 2-1. Display page build overview

Flowchart Example - myAccount Page

The purpose of this section is to describe and illustrate the relationship of the WebVoyáge 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 `/m1/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:

/m1/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, **myAccount**Links.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.

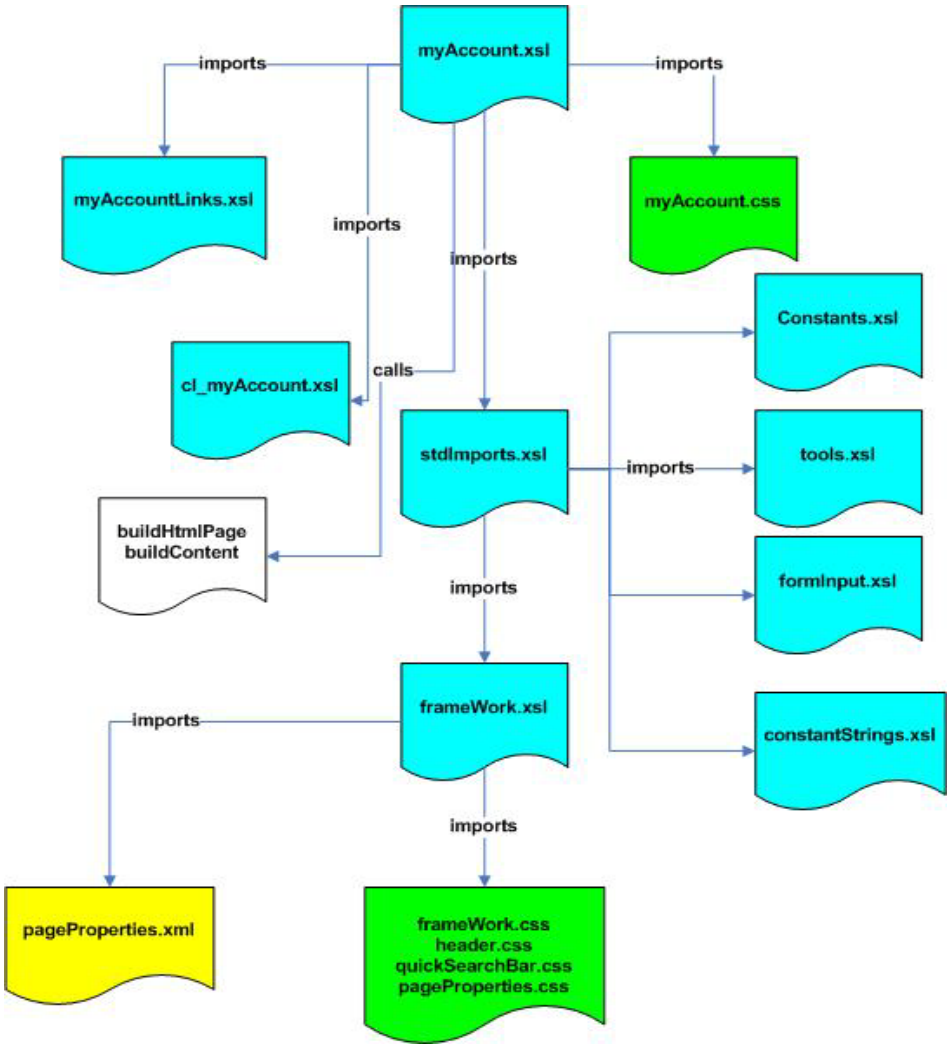

Figure 2-2. Flowchart example for myAccount page

.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 WebVoyáge, 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.

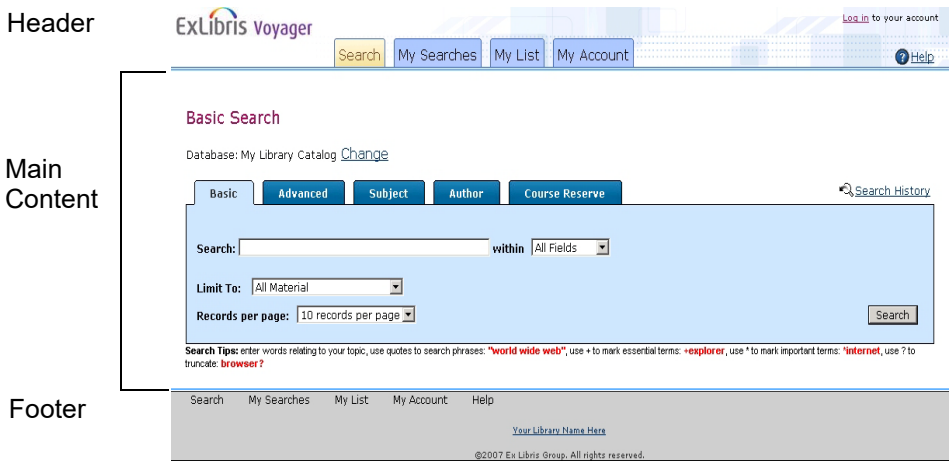

Figure 2-3. Page components

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/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 - 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 `/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 - Subject 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 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 `/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 - Author 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 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/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 - 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`.

- `searchGeospatial.css`.
- `pageInputFocus.js`.

Specific to the Geospatial Search 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 Geospatial Search 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

PDS

3

Introduction

Full PDS (Patron Directory Services) support has been integrated into WebVoyáge. PDS is provided as a standard component of the Voyager installation that is usually set up during the installation or upgrade process.

With PDS you have the option to enable Single Sign-On (SSO) capabilities across all Ex Libris products.

PDS is a flexible application that enables the interchange of patron authentication and patron attribute information between a variety of sources across a mix of topologies. For a complete description of PDS implementation options, refer to the *Patron Directory Services Guide* located in the Cross Products folder of the Ex Libris Documentation Center.

PDS Implementation

To activate PDS, you need to configure the `web.xml` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/WEB-INF/` (see Figure 3-1) and, if you use PINs, the `web.xml` file that is located in `/m1/voyager/xxxdb/tomcat/vpds/context/vpds/WEB-INF/` (see Figure 3-2).

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name.

```
    <context-param>
        <param-name>PDSEnabled</param-name>
        <param-value>True</param-value>
        <description>
            Flag to enable or disable PDS Support. (True / False)
        </description>
    </context-param>
    <context-param>
        <param-name>PDSHost</param-name>
        <param-value>[server IP]</param-value>
        <description>The host of the Patron Data Services.</description>
    </context-param>
    <context-param>
        <param-name>PDSPort</param-name>
        <param-value>[server port number]</param-value>
        <description>
            The port at which Patron Data Services can be found
        </description>
    </context-param>
    <context-param>
        <param-name>Institution</param-name>
        <param-value>[institution name]</param-value>
        <description>
            The Institution "tab_service" which to operate under in PDS
        </description>
    </context-param>
```

Figure 3-1. web.xml PDS Parameters (vwebv path)

```
<!--
  <context-param>
      <param-name>DefaultPIN</param-name>
      <param-value>1234</param-value>
      <description>
          Supply a default PIN for patrons that do not yet have PINs in their records.
          PINs can be 4 - 12 characters.
         By not setting a default pin here , patron must have a patron PIN set, if not they
           must contact the circ desk.
          This needs to match the value set in webvoyage.properties in vwebv in
          option.defaultPIN=
      </description>
  </context-param>
-->
```

Figure 3-2. web.xml PDS PIN Parameters (vpds path)

The following `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/WEB-INF/web.xml` parameters need to be configured:

- PDSEnabled

  To activate/enable the PDS software, enter True (or Yes is acceptable) for the value. You must enter the entire word for this value. T is not a recognized value. This value is not case sensitive.

- PDSHost

  Enter the PDS URL location for this value.

- PDSPort

  Enter the port number for the location of PDS for this value.

- Institution

  Identify the institution under which PDS should operate.

  For PDS, institution is an administrative concept that denotes the organization to which patrons and/or resources belong. For Voyager, each institution within a consortium generally has its own Voyager database which is the equivalent of a PDS institution.

If you are implementing PIN security, the following `/m1/voyager/xxxdb/tomcat/vpds/context/vpds/WEB-INF/web.xml` changes need to be made:

- Remove the commenting for the `DefaultPIN` parameter
- Set a default PIN for patrons that do not already have a PIN

IMPORTANT:
The default PIN value that you specify needs to match the value set for `option.defaultPIN=` in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/webvoyage.properties` (where the default `[skin]` is en_US). Do not make any changes to the PIN parameters in the `/m1/voyager/xxxdb/tomcat/vpds/context/vpds/WEB-INF/web.xml` file unless you have also made changes to the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/webvoyage.properties` file.

Login

When PDS is activated for WebVoyáge, the PDS login display replaces the traditional (native) WebVoyáge login display. See Figure 3-3.

Figure 3-3. PDS Login Display

PDS provides the flexibility to customize the login display. If you need to make changes to the login display, the PDS files to modify are:

- `\location\to\pds\html_form\calling_system-voyager`

  This is the location of the templates for the html forms.

- `\location\to\pds\html_form\icon\loginvoyager.jpg`

  This is the file for the background (grandient purple).

- `\location\to\pds\apache\htdocs\PDSVoyager.css`

  This is the stylesheet for text fonts, colors, sizes, and so forth.

Refer to the *Patron Directory Services Guide* located in the Cross Products section of the Ex Libris Documentation Center for more information.

Using the PDS login display, the patron enters information for the following fields:

- Id (required)

  For this field, the patron can enter either a barcode or institution ID.

  This equates to the `bor_verification=` component that is referenced in the PDS guide.

- Last Name (required)

This equates to the `bor_id=` component that is referenced in the PDS guide.

- Home Library

Optionally, the login display may be customized to include a PIN field (see Figure 3-4) that is used as part of the authentication process. To display the PIN option on the login form, you need to remove commenting for the sections displayed in Figure 3-5 and Figure 3-6 that are located in `/<location>/<to>/pds/html_form/calling_system-voyager/login`.

Figure 3-4. PDS Login Display with PIN

```
<!--
<TR>
 <TD class="LoginLabel">Personal ID Number:  </TD>
 <TD><INPUT TYPE="password" NAME=bor_pin class="form" style="width:180px"></
          TD>
</TR>
-->
```

Figure 3-5. PDS Form PIN Label

```
<!--
<input type="hidden" name="check_pin" value="Y">
-->
```

Figure 3-6. PDS Form PIN Field Type

PDS Integration

PDS provides a single point of integration for external authentication and authorization (attribute) systems for Ex Libris products. With PDS, Voyager VPDS acts as an external authentication and authorization system (see Figure 3-7).

Refer to the *Patron Directory Services Guide* for a description of the way that you can combine a variety of external authentication and authorization systems.

NOTE:
PDS has the flexibility to accommodate authentication checks against one server and retrieve user attributes from a different server.

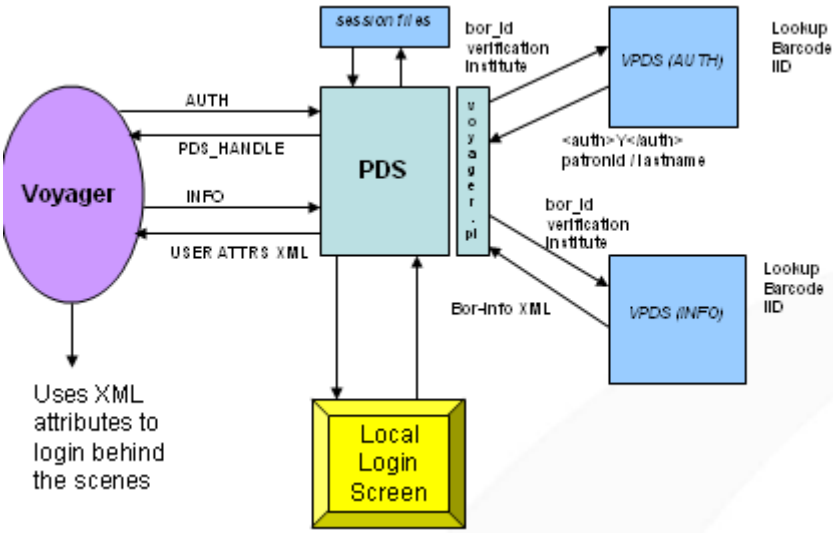

Figure 3-7. VPDS Example

PDS Considerations

This section describes the following PDS considerations:

- PDS/SSO Characteristics
- Clustered/Universal Borrowing Environments

PDS/SSO Characteristics

With PDS and SSO, the login and logout characteristics differ from the traditional (native) WebVoyáge process. For example, the behavior of clicking the Ex Libris Voyager logo (see Figure 3-8) to exit WebVoyáge differs from the traditional (native) WebVoyáge process when PDS/SSO is enabled.

Figure 3-8. Ex Libris Voyager Logo

If a patron clicks the Ex Libris Voyager logo to exit WebVoyáge and is logged in to another Ex Libris PDS SSO-enabled application, the SSO session files (see Figure 3-7) that contain the authentication/authorization session information remain active. When the user returns to WebVoyáge (while still logged in to the other SSO-enabled application), the user immediately enters the WebVoyáge application.

Clustered/Universal Borrowing Environments

In clustered and Universal Borrowing (UB) environments, it is necessary to have a <ubid> parameter. As a result, external authentication systems need to store and return a <ubid> which is the concatenation of `cluster_id` and `db_key`.

Search

4

Search Overview

With WebVoyáge 7.0 and later, it is important to note that the foundation underlying WebVoyáge 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 WebVoyáge Classic. However, for user's just starting with WebVoyáge 7.0 and later, these may be completely new concepts.

IMPORTANT:
Also, remember to use the WebVoyáge Basic User's Guide for key basic concepts regarding the WebVoyáge 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 4-6)
- Left-anchored (see Left-Anchored/Headings-Browse on page 4-7)
- Headings browse (see Left-Anchored/Headings-Browse on page 4-7)

The Basic Search page uses all search types, the Adanced 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 WebVoyáge search function utilizes the index definitions set through the Search component of the Voyager System Administration module. See Figure 4-1.

Figure 4-1. Voyager System Administration Search - Index Settings

The search codes for the WebVoyáge 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 4-2.

Basic Search:

`page.search.basic.search.code=`

Subject Search:

`page.search.subject.search.code=`

Author Search:

`page.search.author.search.code=`

Figure 4-2. 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 comment sections of the `webvoyage.properties` file for additional setup information. See Figure 4-3, Figure 4-4, Figure 4-5, and Figure 4-6 for examples of search code configurations (in the `webvoyage.properties` file) for the Search pages.

```
#=============================================================================#
# Basic search codes
#=============================================================================#
# The default index in the drop down
page.search.basic.search.code.selected=GKEY|*
page.search.basic.search.code.keyAnyAndWith=
page.search.basic.search.code.keyAnyAndWith.code=GKEY^*
page.search.basic.search.code.keyAnyAndWith.order=1
page.search.basic.search.code.keyAnyAndWith.display=All Fields
# page.search.basic.search.code.keyAnyOrWith=
# page.search.basic.search.code.keyAnyOrWith.code=GKEY|*
# page.search.basic.search.code.keyAnyOrWith.order=2
# page.search.basic.search.code.keyAnyOrWith.display=Keyword Anywhere OR with
          Relevance
page.search.basic.search.code.titleKeyAnd=
page.search.basic.search.code.titleKeyAnd.code=TKEY^
page.search.basic.search.code.titleKeyAnd.order=3
page.search.basic.search.code.titleKeyAnd.display=Title
page.search.basic.search.code.subjectKeyAnd=
page.search.basic.search.code.subjectKeyAnd.code=SKEY^
page.search.basic.search.code.subjectKeyAnd.order=4
page.search.basic.search.code.subjectKeyAnd.display=Subject
page.search.basic.search.code.journalTitleKeyAnd=
page.search.basic.search.code.journalTitleKeyAnd.code=JKEY^
page.search.basic.search.code.journalTitleKeyAnd.order=5
page.search.basic.search.code.journalTitleKeyAnd.display=Journal Title
```

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

```
#========================================================================#
# page.search.subject.search.code defines the index to be used
# for searches initiated from the subject search page.
#========================================================================#
page.search.subject.search.code=SUBJ@
page.search.subject.search.display=Subject
```

Figure 4-5. Example configuration for Subject Search page

```
#========================================================================#
# page.search.author.search.code defines the index to be used
# for searches initiated from the author search page.
# Only one search index may be specified for use by this the author search page
#========================================================================#
page.search.author.search.code=NAME+
page.search.author.search.display=Author
```

Figure 4-6. Example configuration for Author Search page

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. WebVoyáge 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 4-7) and Author searches. Journal Title and Call Number searches display in a Title List.

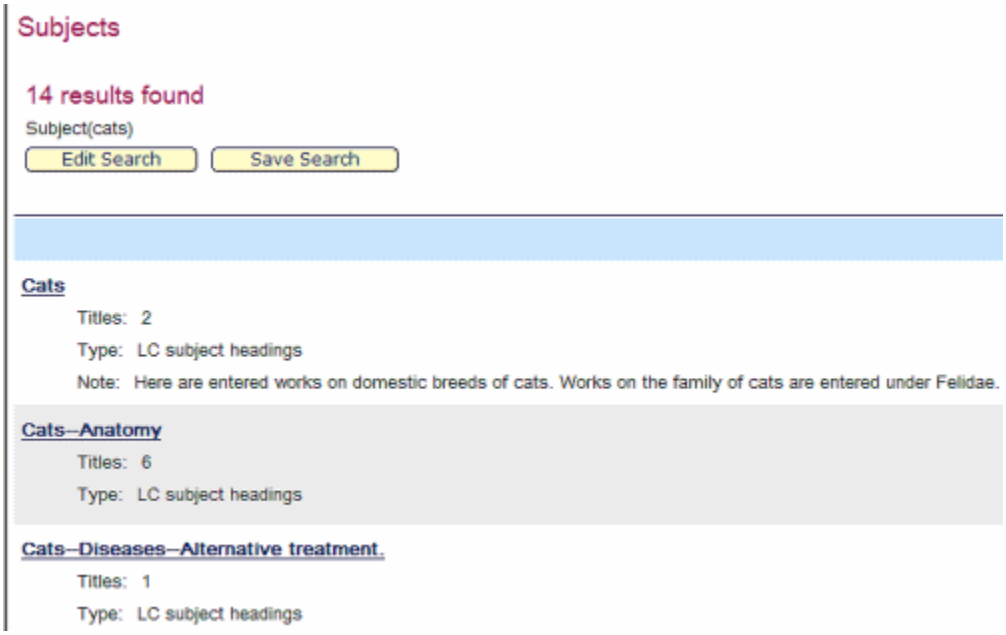

Figure 4-7. Subject Headings display example

Limits

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 `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/` (see the section that begins as shown in Figure 4-8) and in the `limits.xml` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/xsl/userTextConfigs/` (see Figure 4-9 for an example section).

Review the comments provided with these files for setup information.

```
#============================================================================#
#
# 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."
```

Figure 4-8. Limits section in webvoyage.properties example

```
<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 4-9. Example from limits.xml

See Table 4-1 for a description of the limitGroup types (identified in the `webvoyage.properties` file's Limit To comments section). The limit codes identified for each limitGroup is specified in the `limits.xml` file.

Table 4-1. Limit Types

Code	Type	Description
LANG	Language	Refer to `limits.xml` for a list of limit codes such as ENG for English.
MEDI	Medium	This identifies the format such as video. **NOTE:** MEDI is labeled "Format:" in the `limits.xml` file. See Figure 4-9. Refer to `limits.xml` for a list of limit codes such as v for video.
PLAC	Place	Refer to `limits.xml` for a list of limit codes such as `hiu` for Hawaii.
STAT	Status	Refer to `limits.xml` for a list of limit codes such as d for ceased publication.
TYPE	Item Type	Refer to `limits.xml` for a list of limit codes such as `jm` for musical recording.
DATE	Date	Enter dates in the following format: DATE=1990-2000 (between 1990 and 2000) DATE=-1990 (before 1990) DATE=1990- (after 1990) Dates must use the 4-digit format YYYY.
LOCA	Location	The list of Location limits comes from the System > Location Limit Groups > [group] > Name in the System Administration module. Any limit groups in the list that do not have the Suppress in OPAC button pressed are available to select as limits.
CONT	Content	Refer to `limits.xml` for a list of limit codes such as `prm` for performed music.
MEDA	Media	Refer to `limits.xml` for a list of limit codes such as s for audio.

Table 4-1. Limit Types

Code	Type	Description
CARR	Carrier	Refer to `limits.xml` for a list of limit codes such as `ss` for audio cassette.

Using the limit values that you select, WebVoyáge 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.

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 `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`. See the section that starts as shown in Figure 4-10.

Review the comments provided in this file for setup information.

```
#============================================================================#
#
# 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 4-10. 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 4-11 for an example of the configuration code and Figure 4-12 for the display results. Notice also from these examples that two groups have been defined and how they display.

```
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
```

Figure 4-11. Example configuration code

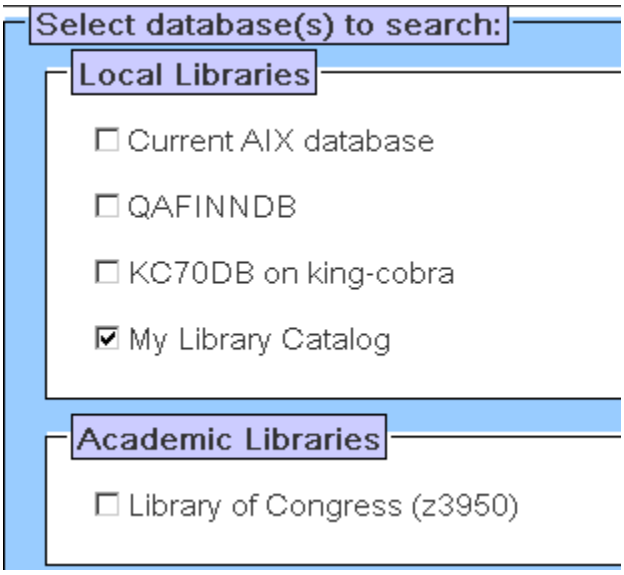

Figure 4-12. Example display of database connections configuration

The user may select one or more databases to search by clicking the Change link from the search page. See Figure 4-13 for an example of the Change link.

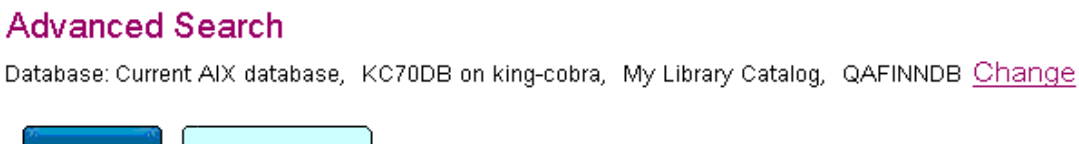

Figure 4-13. Example of Change link

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 4-14.

```
#========================================================================#
#
# Search Pages
# ============
#========================================================================#
page.search.argument.label=Search:
page.search.database.label=Database:
page.search.buttons.quick.button=Quick
page.search.buttons.basic.button=Basic
```

Figure 4-14. Database label example in webvoyage.properties

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 `/m1/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 WebVoyáge 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 7-1 for more information regarding available codes to use in these configuration files and how to enable redirect.

Also check Chapter 9, How Do I Build A Separate Display For Serials? on page 9-1 and Chapter 22, How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions? on page 22-1 for other examples of using the configuration files to display records.

eLink Data

For eLink data from the 856 field to display in the Titles list search results, you need to configure the `option.elink.activate` parameter.

The `option.elink.activate` parameter is configured in the `webvoyage.properties` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`. See the `webvoyage.properties` file section that is shown in Figure 4-15.

```
#=======================================
# Option to activate (True) or deactivate (False) elink thumbnails
# Default is deactivate elink thumbnail
#=======================================
option.elink.activate=False
```

Figure 4-15. eLink Data Parameter

To enable eLink data identification (with an eLink icon) in the Titles list (see Figure 4-16), set the `option.elink.activate` parameter to `True`.

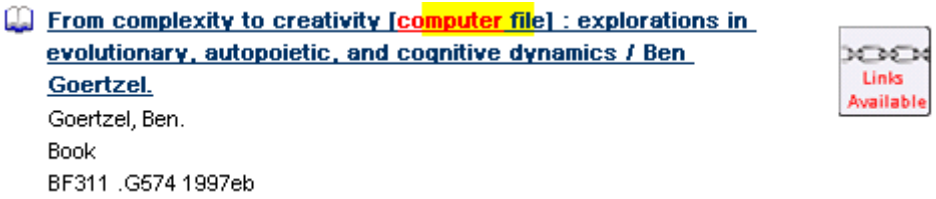

Figure 4-16. eLink Example

Similar to the Classic WebVoyáge, the Titles list in the Tomcat WebVoyáge provides an indicator of that eLinks are available. Separately, the eLink URL and text are provided in the XML.

Linking/OpenURL

5

Overview

WebVoyáge uses the OpenURL standard (see OpenURL Standard on page 5-1) to generate links and obtain information from external sources. From the WebVoyáge 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. Voyager supports both the 1.0 and 0.1 standard. Refer to the comments in the `linkresolver.properties` file (also described in How Do I Implement HTTP Post to Link Resolver? on page 32-1) for additional information.

The OpenURL standard is designed to support mediated linking from sources (MARC databases) to library targets like online 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 5-1.

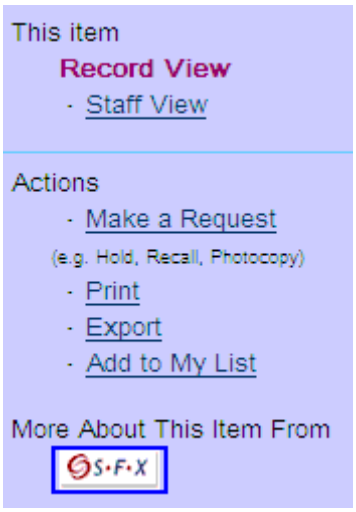

Figure 5-1. Action Box - OpenURL (SFX) request button example

Or an OpenURL-formatted item request can be initiated when staff click the LinkResolver option as shown in Figure 5-2.

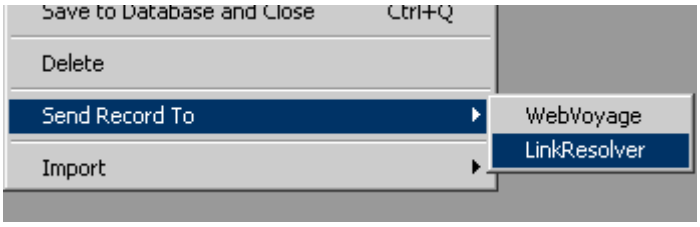

Figure 5-2. LinkResolver option example

OpenURL/LinkResolver Configuration

The following WebVoyáge 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 5-2)

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 32-1.

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

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

In addition, there are also genre parameters such as numGenres, genre1, genre2, and so forth.

See Table 5-1 for a description of these parameters.

Table 5-1. Variables Description

Element	Description	Possible Values
key	Meta-tag for the field to parse.	Enter meta-tag.
tag	MARC field to match.	Enter valid MARC field number.

Element	Description	Possible Values
subfield	MARC subfield to match.	Enter valid MARC subfield letter.
len1	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.	Enter the number of characters or 0.
parseStart	Text of field/subfield after which extraction is to begin.	Enter the text of the field/ subfield after which to parse such as Issue: for example (see Figure 5-3).
parseEnd	Text of field/subfield at which extraction is to end.	Enter the text of field/ subfield at which to end the parse such as Date: (see Figure 5-3).
len2	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.	Enter the number of characters or 0.
numGenres	Identifiy the number of genres defined.	Enter the numeric value for the number of genres defined.
genre1, genre2, and so forth	Multiple genres may be defined/ configured using a combination of leader 6 and 7 values and a genre text string for the URI when there is a match on the leader values. The possible genres are: • Journal • Book • Conference • Article • Preprint • Proceeding • Bookitem	Enter leader 6 and 7 values or an asterisk (as a wildcard character) for matching and the genre text after a colon that is placed in the URI string when there is a successful match (see example below) . Example: `openUrl.cfg.LOCAL. genre1=*m:book` See How Do I Implement HTTP Post to Link Resolver? on page 32-1 for more information.

```
#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 5-3. Example of OpenURL variables configuration (holdingsInfo.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 5-4.

```
actionBox.openUrl.image.linkText=SFX
actionBox.openUrl.image.link=ui/en_US/images/sfx.gif
actionBox.openUrl.urlRoot=http://host:port/SFXinstance
```

Figure 5-4. OpenURL request button configuration example (holdingsInfo.vbib.properties)

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 5-5.

```
#LOCAL
actionBox.openUrl.cfg.LOCAL.openUrlDisplay=Y
```

Figure 5-5. Database identified example

In Figure 5-5, 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

6

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 WebVoyáge
- 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 Voyager System Administration.

NOTE:
Theleft-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 WebVoyáge, 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
- WebVoyáge
- 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 WebVoyáge. 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 WebVoyáge.

WebVoyáge

The configuration settings for WebVoyáge are located in the `webvoyage.properties` file and the `displaycfg.xml` file. Refer to the following sections for the details.

webvoyage.properties

The setting to enable bibliographic record linking on the holdings page is in the `webvoyage.properties` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`. See Figure 6-1.

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

Figure 6-1. Enable bibliographic record linking on holdings page example

Related records holdings display configuration settings are also located in the `webvoyage.properties` file. See Figure 6-2.

```
#========================================================================#
# Holdings info text to display when bibliographic record linking feature is
#  turned on.
# NOTE option.loadRelatedRecords must be Y
# NOTE the pretext, linktext, and posttext are used to display the text
# for the related items link.
# The default is "Click (pretext) here (linktext has underlining URL to the
# related items link) for related items" (posttext).
# One link may be defined per bibliographic record linking profile, as defined
#  in Sysadmin.
#========================================================================#
holdingsInfo.relatedRecords=Related Records
holdingsInfo.relatedRecords.maxPerPage=10
holdingsInfo.relatedRecords.profile.ISSNPREC=
holdingsInfo.relatedRecords.profile.ISSNPREC.link.pretext=Click
holdingsInfo.relatedRecords.profile.ISSNPREC.link.text=here
holdingsInfo.relatedRecords.profile.ISSNPREC.link.posttext=for related items
holdingsInfo.relatedRecords.profile.ISSNSUCC=
holdingsInfo.relatedRecords.profile.ISSNSUCC.link.pretext=Click
holdingsInfo.relatedRecords.profile.ISSNSUCC.link.text=here
holdingsInfo.relatedRecords.profile.ISSNSUCC.link.posttext=for related items
holdingsInfo.relatedRecords.profile.ISSNCOMP=
holdingsInfo.relatedRecords.profile.ISSNCOMP.link.pretext=Click
holdingsInfo.relatedRecords.profile.ISSNCOMP.link.text=here
holdingsInfo.relatedRecords.profile.ISSNCOMP.link.posttext=for related items
```

Figure 6-2. Related records settings

The label setting for the source record is in the `webvoyage.properties` file as shown in Figure 6-3.

```
#========================================================================#
# Label for displaying source record link in the holdings record page
# NOTE option.loadRelatedRecords must be Y
#========================================================================#
holdingsInfo.sourceRecord=Source Record
```

Figure 6-3. Source record label example

displaycfg.xml

There are display settings in the `displaycfg.xml` file that need to match the settings in the `webvoyage.properties` file and Voyager System Administration (Search - Bibliographic Record Linking). Refer to Figure 6-4 for an illustration of the section in the `displaycfg.xml` file where these settings need to match.

```
<!--  These profiles should match the profiles set in the webvoyage.properties as well
       as setup in sysadmin -->


<displayTags label="Related Records">
    <displayTag field="3500" profileMatch="ISSNCOMP"/>
</displayTags>

<displayTags label="Preceding Titles:" notFound="No preceding titles found.">
    <displayTag field="3500" profileMatch="ISSNPREC"/>
</displayTags>

<displayTags label="Succeeding Titles:" notFound="No succeeding titles found.">
    <displayTag field="3500" profileMatch="ISSNSUCC"/>
</displayTags>
```

Figure 6-4. displaycfg.xml Settings

Specifically, the `profileMatch=` value (such as ISSNCOMP shown in Figure 6-4) needs to match the profile code configured in Voyager System Administration and the `webvoyage.properties` file as described in webvoyage.properties on page 6-4.

Display Codes

7

displaycfg.xml

The codes described in Table 7-1 may be used to format the display of information through the `displaycfg.xml` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/`.

Table 7-1. displaycfg.xml display codes

Code	Description
2000	Table of Contents (505 subfields a, r, t, g).
3000	856 links (linked resources from the 856 field).
4000	MARC record.
5000	Database name of the bibliographic record.
6000, 6010	Electronic resource information.
7000	Format.
7106	Includes.
7107	Physical description.
9000	Holdings information that is defined in `displayHoldings.xml`.
9500	Display holdings summary information.

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 7-2 may be used to format the display of holdings information through the `displayHoldings.xml` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/`.

Table 7-2. displayHoldings.xml display codes

Code	Description
1000	OPAC display name for the location.
1002	Database name for the item.
1005	OPAC display name for the temporary location (only). This is used in combination with the 1000 code.
1010	Number of items linked to the MARC holdings record.
1012	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.)
1020	Recent issues from serials.
1021	Recent issues from serials diplayed according to the WebVoyage Serials for XML feature
1022	Supplemental issues from serials.
1023	Supplemental issues from serials diplayed according to the WebVoyage Serials for XML feature
1024	Indexes from serials.
1025	Indexes from serials diplayed according to the WebVoyage Serials for XML feature
102X_config	Configuration options loaded by `102X_display.xsl` for the 1012,1023, and 1025 codes set up in the `displaycfg.xml` called by the `display.xsl`
102X_display	Display logic behind displaying the WebVoyage Serials for XML feature
1030	Order status as shown in the line items of purchase orders.

Table 7-2. displayHoldings.xml display codes

Code	Description
1040, 1042, 1044	Compressed serials information. This is taken from the MFHD 853/863/866, 854/864/867, and 855/865/868 fields. Refer to “Serial Issues Collapsing” in the *Voyager Acquisitions User’s Guide* for more information.
1041	Compressed serials information diplayed according to the WebVoyage Serials for XML feature for 853 / 863 pairs
1043	Compressed serials information diplayed according to the WebVoyage Serials for XML feature for 854 / 864 pairs
1045	Compressed serials information diplayed according to the WebVoyage Serials for XML feature for 855 / 865 pairs
104X_chron Values	This file is used to convert numeric values defined in MARC for month and season into chronological names
104X_config	Configuration options for displaying compressed serials
104X_display	Display logic behind displaying the WebVoyage Serials for XML feature for compressed serials
1050	E-item information. This includes enumeration, chronology, year information, and caption linked to the e-item.
3000	856 links (linked resources from the 856 field).

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.

Serials Display

The 1040, 1042, and 1044 codes display serials information in WebVoyáge. Information from the holdings records 86X (863, 864, and 865) fields display if the first indicator is a 3, 4 or 5. Refer to the 104X files in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/`.

Information in the chronology subfields displays according to its numeric designator mapping (1-12 for the months, 21-24 for the seasons). If there is no corresponding mapping, the literal contents of the subfield display.

Enable Redirect

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

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

See Figure 7-1 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 7-1. 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 7-2 to view this information.

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

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

Figure 7-2. Example display.xsl code

```
    <xsl:for-each select="displayTag">

        <xsl:choose>
            <xsl:when test="@field &lt; '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>
```

The `display.xsl` file is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/display/`.

Patron Self-Registration

8

Patron Self-Registration Overview

The Patron Self-Registration feature allows patrons to enter the information required to create a patron record. The system displays a confirmation message to the patron after the registration information has been successfully submitted. Patrons can then go to the Circulation Desk to have a barcode added and the purge date changed.

NOTE:
The purge date of these records is the same day that they are created. The purge date will not display in theHistory tab of the records in the Voyager Circulation Module, but can be viewed by clicking the calendar expansion button in the Purge field when editing the patron records.

This facility enables your library to reduce staff time creating patron records, while maintaining control of the information required and barcodes attached to the patron records.

Usage Example: You might set up a separate workstation where patrons input their own patron information requiring your Circulation staff only to verify the patron information and assign barcodes. If your library has a large number of public users viewing library materials for short periods of time, use Patron Self-Registration to quickly create temporary patron records.

Patron Self-Registration resides on a separate port and runs independently of WebVoyáge. You can create a link to your **Patron Self-Registration** page from WebVoyáge.

IMPORTANT:
Make sure the AutoComplete Settings of the web browsers on library computers using Patron Self-Registration are set to not save data input into forms. AutoComplete stores previous entries in fields and suggests matches. This might allow patrons to use a previous patron's record information. Disable these settings when the browser is being used to input information. The procedure for disabling this feature depends on your browser, so if you need help, consult your browser's documentation.

After successful creation of a record through Patron Self-Registration, the original form is cleared for security purposes. If the **Back** button is clicked in the browser, the previously entered patron information will not be available

The steps for configuring patron self-registration are available in the following sections:

- Specifying the IP Address and Port Number of the OPAC Server on page 8-2
- Customizing the Patron Self-Registration Messages on page 8-3
- Customizing the Patron Self-Registration Page on page 8-3
- Adding a Link to Patron Self-Registration from the Login Page on page 8-12

Specifying the IP Address and Port Number of the OPAC Server

To enable Patron Self-Registration, you must first specify the IP address and port of your OPAC server. Do this in the `voytomcat.ini` file, in the `/m1/voyager/`*`xxx`*`db/tomcat/PSR/context/webvoyage/ini/` directory. Refer to Table 8-1 for a description of the Patron Self-Registration options.

Table 8-1. voytomcat.ini Settings

voytomcat.ini Parameter	Description
Server	Use this parameter to enter the IP address of your OPAC server.
Port	Use this parameter to enter the port specification for your OPAC server.

Table 8-1. voytomcat.ini Settings

voytomcat.ini Parameter	Description
PatronSelfRegistrationURL	Use this parameter to enter a URL specification patrons will connect to after clicking the Exit button when they have successfully submitted patron self-registration information. The Exit button is controlled by the `exit.gif` file in the `/m1/voyager/xxxdb/tomcat/PSR/context/webvoyage/images` directory. With this option, you can redirect the patron back to the WebVoyáge page when the Exit button is clicked.

Customizing the Patron Self-Registration Messages

The messages generated by Patron Self-Registration can be customized. These messages are in the `voytomcat.ini` file, in the `/m1/voyager/`*`xxx`*`db/tomcat/PSR/context/webvoyage/ini/` directory. Use the `Message1`, `Message2`, and `Message3` parameters in the `voytomcat.ini` file for this purpose.

Customizing the Patron Self-Registration Page

The information on the **Patron Self-Registration** page can be customized. This includes the display of the page, the field titles, which fields are required, and which fields display. You can customize the **Patron Self-Registration** page by configuring the files described in Table 8-2:

Table 8-2. Patron Self-Registration Files

File Name	Description
patron.xml	Determines the fields for input on the Patron Self-Registration page. Also switches field display, and required status, on and off.
patron.xsl	Contains the rules for converting the patron.xml file to html, and verifying the input. Change the order of the fields on the Patron Self-Registration page by changing the order of the elements in the <table> sub-element of the <form> element. You can also remove any unwanted fields here, as long as they are not marked required in the patron.xml file.

Table 8-2. Patron Self-Registration Files

File Name	Description
patron.css	Determines some display options of the page, including background color, text alignment, and margins.

These files are in the `/m1/voyager/xxxdb/tomcat/PSR/context/webvoyage/ini/` directory.

CAUTION:
The patron.xml, patron.xsl and patron.css files are configured to properly display and execute the Patron Self-Registration page, in conjunction. Customization beyond the page headings, messages, field elements, and variables detailed above should only be done by personnel familiar with XML, XSL, and Cascading Style Sheets (css).

The patron.xml File

The `patron.xml` is an XML file which determines the fields for input on the **Patron Self-Registration** page. Figure 8-1 shows a sample `patron.xml` file.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="patron.xsl"?>
<!--======= Patron Self Registration =======-->
<Patron>
    <Heading>
        <Visible>Y</Visible>
        <Caption>Patron Self Registration</Caption>
    </Heading>
    <Instructions>
        <Visible>Y</Visible>
        <Caption>Please enter the required fields (bold)</
         Caption>
    </Instructions>
    <id>
```

Figure 8-1. Sample patron.xml File

```
        <Visible>Y</Visible>
        <Idtype>S</Idtype>
        <Caption>SSN</Caption>
    </id>
    <Title>
        <Visible>N</Visible>
        <Required>N</Required>
        <Caption>Title</Caption>
    </Title>
    <FirstName>
        <Visible>Y</Visible>
        <Required>N</Required>
        <Caption>First Name</Caption>
    </FirstName>
    <MiddleName>
        <Visible>N</Visible>
        <Required>N</Required>
        <Caption>Middle Name</Caption>
    </MiddleName>
    <LastName>
        <Caption>Last Name</Caption>
    </LastName>
    <PrimaryPhone>
        <Visible>Y</Visible>
        <Required>Y</Required>
        <Caption>Primary Phone</Caption>
    </PrimaryPhone>
    <OtherPhone>
        <Visible>Y</Visible>
        <Required>N</Required>
        <Caption>Other Phone</Caption>
    </OtherPhone>
    <Address1>
```

Figure 8-1. Sample patron.xml File (Continued)

```
        <Caption>Address</Caption>
    </Address1>
    <Address2>
        <Visible>Y</Visible>
        <Required>N</Required>
    </Address2>
    <Address3>
        <Visible>Y</Visible>
        <Required>N</Required>
    </Address3>
    <Address4>
        <Visible>Y</Visible>
        <Required>N</Required>
    </Address4>
    <Address5>
        <Visible>Y</Visible>
        <Required>N</Required>
    </Address5>
    <City>
        <Visible>Y</Visible>
        <Required>N</Required>
        <Caption>City</Caption>
    </City>
    <State>
        <Visible>Y</Visible>
        <Required>Y</Required>
        <Caption>State</Caption>
    </State>
    <Country>
        <Visible>Y</Visible>
        <Required>Y</Required>
        <Caption>Country</Caption>
    </Country>
```

Figure 8-1. Sample patron.xml File (Continued)

```
    <ZipCode>
        <Visible>Y</Visible>
        <Required>Y</Required>
        <Caption>Zip Code</Caption>
    </ZipCode>
    <EmailAddress>
        <Visible>Y</Visible>
        <Required>Y</Required>
        <Caption>Email Address</Caption>
    </EmailAddress>
    <message>
        <Caption>You have not entered the required fields.</
         Caption>
    </message>
</Patron>
```

Figure 8-1. Sample patron.xml File (Continued)

Most elements of the `patron.xml` file corresponding to the fields in Patron Self-Registration have three variables associated with them. These variables are listed and described in Table 8-4.

Other than Last Name and Address1, all the fields have an option to be visible or not and an option to be required or not.

ID is required, if visible. This field can have an Idtype value of **S** or **I**. Table 8-3 details the `idtype` element options.

Table 8-3. Idtype element option details

idtype	Description	Field limits
S	Social Security Number	9 numeric characters
I	Institution ID	30 alphanumeric characters

Last Name and Address1 are always visible and required.

Table 8-4. Field element variable descriptions

Element	Description	Valid values
Visible	Determines whether or not the field is visible to patron.	Y= field is visible N= field is not visible
Required	Determines whether or not the field is required.	Y= input into field is required N= input into field is not required
Caption	Caption of the field as displayed on the page	Any alphanumeric

Change the values of the field elements in the `patron.xml` file by changing the text between the appropriate tags. For example, to change the Other Phone field to required, and the caption to `Local Phone`, change the `N` between the `<Required>` tags to a **`Y`**, and change the text between the `<Caption>` tags.

```
<OtherPhone>
    <Visible>Y</Visible>
    <Required>N</Required>
    <Caption>Other Phone</Caption>
</OtherPhone>
```

Figure 8-2. Default Other Phone field

```
<OtherPhone>
    <Visible>Y</Visible>
    <Required>Y</Required>
    <Caption>Local Phone</Caption>
</OtherPhone>
```

Figure 8-3. Revised Other Phone field

Changing the Layout of the Fields in the patron.xsl File

To alter the layout of the fields on the Patron Self-Registration page, change the table specification in the `patron.xsl` file (lines 9 through 28 of Figure 8-5).

The first <table> element containing the page field element tags (see Figure 8-4) corresponds to the left-hand column of fields on the Patron Self-Registration page and the second corresponds to the right-hand column.

```
<xsl:apply-templates select="XX">
```

Figure 8-4. Page Field Element Tag Format

The page field element name (**"*XX*"**) components of the page field element tags correspond to the elements in the patron.xml file. Changing the order of these tags in the patron.xsl file will change the order of the field on the Patron Self-Registration page.

IMPORTANT:
Make sure that you do not delete any page field element tags that are marked as required in the patron.xml file. For example, if the <Required> tag of the <Address2> element is set to Y, a <xsl:apply-templates select="Address2"> tag must appear in a table element specification in the patron.xsl file.

```
Line#
1   <form method="post" action="/webvoyage/servlet/PatronRegistrationServlet"
    name="patronselfreg">
2                        <table width="800" border="0" cellspacing="0" cellpadding="1"
    align="center">
3                        <tr><td><xsl:apply-templates select="message"/>
4                             </td>
5                        </tr>
6                             <tr valign="top">
7                                  <td WIDTH="50%">
```

Figure 8-5. Table Specification Section of the patron.xsl File

```
Line#
8                                   <table>
9                                       <xsl:apply-templates select="id"/>
10                                      <xsl:apply-templates select="Title"/>
11                                      <xsl:apply-templates select="FirstName"/>
12                                     <xsl:apply-templates select="MiddleName"/>
13                                      <xsl:apply-templates select="LastName"/>
14                                      <xsl:apply-templates select="OtherPhone"/>
15                                    <xsl:apply-templates select="PrimaryPhone"/>
16                                  </table>
17                              </td>
18                              <td WIDTH="50%">
19                                  <table>
20                                      <xsl:apply-templates select="Address1"/>
21                                      <xsl:apply-templates select="Address2"/>
22                                      <xsl:apply-templates select="Address3"/>
23                                      <xsl:apply-templates select="Address4"/>
24                                      <xsl:apply-templates select="Address5"/>
25                                      <xsl:apply-templates select="City"/>
26                                      <xsl:apply-templates select="State"/>
27                                      <xsl:apply-templates select="Country"/>
28                                      <xsl:apply-templates select="ZipCode"/>
29                                     <xsl:apply-templates select="EmailAddress"/>
30                                  </table>
31                              </td>
32                          </tr>
33                      </table>
34                      <table width="800" border="0" cellspacing="2" cellpadding="3"
    align="center">
35                        <tr><td colspan="2" valign="middle" align="center">
36                                <input type="button" name="Submit" value="Submit"
    DEFANGED_OnClick="checkValues(document.forms[0].Message.value)"></input>
37                                 <input type="reset" name="Submit2"
    value="Reset"></input>
38                              </td>
39                          </tr>
40                      </table>
41                  </form>
```

Figure 8-5. Table Specification Section of the patron.xsl File (Continued)

For example, if you want the **Primary Phone** field to display above the **Other Phone** field, rearrange the order of the <xsl:apply-templates select="OtherPhone"/> and <xsl:apply-templates select="PrimaryPhone"/> tags in the patron.xsl file.

To move the Other Phone field to the right-hand column on the Patron Self-Registration page, move the <xsl:apply-templates select="OtherPhone"/> tag to the second table element (containing the page field element tags).

Patron Record Creation with Patron Self-Registration

Once a library patron inputs the required information, a patron record is created in the Voyager Circulation Module for that person. This record will *not* contain a barcode, and will have a purge date of the date the record is created.

You must then assign a barcode to the patron record, and change the purge date at the circulation desk. Patron records created using Patron Self-Registration have an Operator ID of `OPAC`.

Configuring Patron Self-Registration in System Administration

Voyager System Administration contains three items which affect Patron Self-Registration. Configure these items in **System Administration>Circulation>Policies**. Table 8-5 describes each item and details the Circulation Policy Field descriptions.

Table 8-5. Circulation Policy Field Descriptions

Configuration Item	Description
OPAC Circ Desk	The Circulation Desk associated with WebVoyáge. Scope not limited to Patron Self-Registration but affects records created by Patron Self-Registration
Set Purge	The purge date is the date that will be used with a special job that will remove patron records from the database. At this time this date is not being used.
Default Patron Group	Default patron group which records created will be assigned to.

Adding a Link to Patron Self-Registration from WebVoyáge

If you are using Patron Self-Registration in conjunction with WebVoyáge, you can provide a link to the Patron Self-Registration page in WebVoyáge. One possible scenario is to provide this link from the **Login** page. Customers using WebVoyáge who must create a patron record can then do so.

Adding a Link to Patron Self-Registration from the Login Page

One way to do this is to add a link to the `logonmsg.htm` file, containing the IP address and port of the Patron Self-Registration page (see line 4 of Figure 8-6 for an example).

```
Line#
1   <CENTER>
2       Please enter your Patron Barcode and Last Name, then click the OK
        button.
3       <BR>
4       If you are not registered, <A HREF="http://
        xxx.xxx.xxx.xxx:xxxx">click here</A> for Patron Self-Registration.
5   </CENTER>
```

Figure 8-6. The logonmsg.htm File with Link

How Do I Build A Separate Display For Serials?

9

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

By default, WebVoyáge 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 9-1. 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:

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

1. Create and save a new file in the `/m1/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 9-1 for this example.

```
<!--
#(c)#============================================================
#(c)#
#(c)#       Copyright 2008 ExLibris Group
#(c)#                       All Rights Reserved
#(c)#
#(c)#============================================================
-->
<!--
**        Product : WebOpac : displaycfg
**        Version : 7.0
**        Created : 17-OCT-2007
**    Orig Author :
**  Last Modified : 11-APR-2008
**Last Modified By: ASP

-->
<display>
        <titleTags>
```

Figure 9-1. Sample line of code for sdisplaycfg.xml

```
                <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:">
```

```
                <displayTag field="780" indicator1="0" indicator2="2" subfield="at" />
                <displayTag field="780" indicator1="0" indicator2="3" subfield="at" />
        </displayTags >

        <displayTags label="Publisher:">
                <displayTag field="260" indicator1="X" indicator2="X" subfield="abc"/>
        </displayTags>

        <displayTags label="ISSN:">
                <displayTag field="022" indicator1="X" indicator2="X" subfield="a"/>
        </displayTags>

        <displayTags label="Format:">
                <displayTag field="000" indicator1="0" indicator2="6" subfield="2"/>
        </displayTags>

        <displayTags label="Subjects:">
                <displayTag field="600" indicator1="X" indicator2="X"
           subfield="abcdefghkjlmnopqrstuvxyz">
                        <subfield value="v" preText="--"/>
                        <subfield value="x" preText="--"/>
                        <subfield value="y" preText="--"/>
                        <subfield value="z" preText="--"/>
                </displayTag>
        </displayTags>

        <displayTags label="Online Journal:">
                <displayTag field="3000"/>
        </displayTags>

    <displayTags label="Holdings Information" notFound="No holdings available -- check at
           the Circulation Desk.">
                <displayTag field="9000"/>
        </displayTags>

</display>
```

Figure 9-1. Sample line of code for sdisplaycfg.xml (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 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 9-2.

```
<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"
            xmlns:hol="http://www.endinfosys.com/Voyager/holdings"
            xmlns:slim="http://www.loc.gov/MARC21/slim">
```

Figure 9-2. New namespace declarations example

   c. Locate `<!-- ## Our Document Holders ## -->`. See Figure 9-3.

   d. Change the variable name `Config` to `MConfig`. See Figure 9-3.

   e. Add a document holders declaration for the serial configuration display for `SConfig` and `recType` as shown in Figure 9-3.

```
<!-- ## Our Document Holders ## -->
<!-- Config changed to MConfig below -->
<xsl:variable name="MConfig" select="document('./configs/displaycfg.xml')"/>
<xsl:variable name="holdingsConfig" select="document('./configs/displayHoldings.xml')"/>
<!-- added for serial config display -->
<xsl:variable name="SConfig" select="document('./configs/sdisplaycfg.xml')"/>
<xsl:variable name="recType" select="substring($bibRecord/hol:marcRecord/slim:leader,8,1)"
        />
<!-- end of add for serial config display -->
<!-- ################## -->
<!-- ## buildRecordForm ## -->
<!-- ############################################################################ -->
```

Figure 9-3. Our Document Holders example

f. Add logic for looking at the record leader by replacing the lines of code seen in Figure 9-4 with the example lines of code in Figure 9-5.

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

Figure 9-4. Lines to be replaced

```
<!-- ## Bibliographic Data ## -->
<xsl:choose>
               <xsl:when test="$recType='s'">
                   <xsl:for-each select="$SConfig">
                       <div class="bibliographicData">
                           <xsl:call-template name="buildMarcDisplay">
                               <xsl:with-param name="recordType" select="'bib'"/>
                           </xsl:call-template>


                           <br />Record type: <xsl:value-of select="$recType"/>

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

               <xsl:otherwise >
                   <xsl:for-each select="$MConfig">
                       <div class="bibliographicData">
                           <xsl:call-template name="buildMarcDisplay">
                               <xsl:with-param name="recordType" select="'bib'"/>
                           </xsl:call-template>


                           <br />Record type: <xsl:value-of select="$recType"/>

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

               </xsl:otherwise>
</xsl:choose>
```

Figure 9-5. Replacement lines of code

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

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?

10

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 10-1. 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:

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

1. Make a backup copy of the `header.xsl` file that is located in `/m1/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 10-1) immediately before the `<xsl:stylesheet>` element at the bottom of the file.

```
<!--  ################
  -->
<!--  ## buildHeaderLinks ##
  -->
<!--  ##############################################################-->
<xsl:template name="localHeaderLinks">
<div id="headerLinks" style="font-family:Arial;font-size:.70em;">
  <span style="color:#9F175E;padding: 0 .20em;margin:0 .2em;">University Library Catalog</
          span>
  <a href="#" style="padding: 0 .20em;margin:0 .2em;">My library</a>
  <a href="#" style="padding: 0 .20em;margin:0 .2em;">SFX</a>
  <a href="#" style="padding: 0 .20em;margin:0 .2em;">Primo</a>
  <a href="#" style="padding: 0 .20em;margin:0 .2em;">Chat with a librarian</a>
  </div>
  </xsl:template>
<!-- ##############################################################-->
<!-- ############################################################## -->
```

Figure 10-1. Header links template

b. Call the template that you created in Step a by adding the instruction (see Figure 10-2, line 7) to the `buildHeader` section located at the top of the `header.xsl` file.

```
Line#
  1    <!-- ################# -->
  2    <!-- ## buildHeader ## -->
  3    <!-- ################################################################ -->
  4
  5    <xsl:template name="buildHeader">
  6       <xsl:for-each select="/page:page/page:pageHeader">
  7         <xsl:call-template name="localHeaderLinks" />
  8          <div id="headerRow">
  9             <div id="logo" title="{$headerText/logo}">
```

Figure 10-2. Call instruction

3. Save and test your changes to `header.xsl`.

OPTIONAL:

4. *Back out your changes, if necessary, by deploying your backup copy of* `header.xsl`.

Procedure 10-2. 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:

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

1. Make a backup copy of the `footer.xsl` file that is located in `/m1/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 10-3, lines 15 -19) after the `<ul class="navbar">` in the `buildFooter` section in the file.

```
Line#
 1  <!-- ################# -->
 2  <!-- ## buildFooter ## -->
 3  <!-- ###################################################################-->
 4
 5  <xsl:template name="buildFooter">
 6
 7     <xsl:for-each select="/page:page/page:pageFooter">
 8        <div id="pageFooter">
 9           <xsl:for-each select="page:tabs[@nameId='page.footer.buttons']">
10
11              <div id="footerTabs" title="{$footerText/footerTabs}">
12                 <a name="navFooter"></a>
13                 <h2 class="navFooter"><xsl:value-of select="$footerText/
                footerTabs"/></h2>
14                 <ul class="navbar">
15  <!--  extra footer tabs  -->
16    <li>
17    <a href="http://www.exlibrisgroup.com/" target="_newWin">Ex Libris</a>
18    </li>
19  <!--  end extra footer tabs  -->
20                    <xsl:for-each select="$Configs/pageConfigs/
                footerTabDisplayOrder/tab">
```

Figure 10-3. 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?

11

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 11-1. 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:

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

1. Make a backup copy of `cl_myAccount.xsl` that is located in `/m1/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 11-1.

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

Figure 11-1. Section comment to locate

   b. Comment out the table heading and table cell lines of code relevant to item type. See Figure 11-2 and Figure 11-3.

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

Figure 11-2. Table heading example

```
<!-- <td id="tableCell" headers="cellChargedType"><xsl:value-of select="page:itemType"/
          > </td> -->
```

Figure 11-3. Table cell example

3. Save and test your changes.

OPTIONAL:

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

How Do I Add A Map Or Other Information To A Location?

12

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 12-1. 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:

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

1. Create and save a new file in the `/m1/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 12-1 for this example.

   NOTE:
   Notice the template name `locMapLink` in this example.

```
<!--
**                Note: sample link to map based on loc code
**                Version : 1.0
**                Created : 16-Nov-2007
**                Created By :
-->

<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="locMapLink">
<xsl:param name="mfhd"/>
               <xsl:variable name="locCode">
                    <xsl:call-template name="BMDProcessMarcTags">
                     <xsl:with-param name="field" select="'852'"/>
                     <xsl:with-param name="indicator1" select="'X'"/>
                     <xsl:with-param name="indicator2" select="'X'"/>
                     <xsl:with-param name="subfield" select="'b'"/>
                     <xsl:with-param name="mfhdID" select="$mfhd"/>
                     <xsl:with-param name="recordType" select="'mfhd'"/>
                  </xsl:call-template>
               </xsl:variable>



               <!-- you must create your web site to display maps -->
               <xsl:variable name="baseURL">http://www.exlibrisgroup.com/?loc=</
          xsl:variable>


                        <div class="locationMap">
                           Show me a <a id="locMap" href="{$baseURL}{$locCode}"
          target="_new">map</a>.
                        </div>
```

Figure 12-1. Example code for building hyperlink

```
</xsl:template>

<!-- ########################################################################## -->

</xsl:stylesheet>
```

Figure 12-1. Example code for building hyperlink (Continued)

2. Make a backup copy of `display.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/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 12-2.

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

Figure 12-2. Example code for adding path statement

   b. Call the `locMapLink` template from within the `BMD100` template. See Figure 12-3.

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

<xsl:template name="BMD1005">
<xsl:param name="mfhdID"/>

    <xsl:for-each select="$HoldXML/hol:holdingsRecord/hol:mfhdCollection/
           mfhd:mfhdRecord[@mfhdId = $mfhdID]/mfhd:mfhdData[@name='locationDisplayName']">
        <xsl:if test="string-length(.)">
            <xsl:value-of select="."/>
```

Figure 12-3. Example of call for locMapLink

```
                <!-- ## add a map link ## -->
                <xsl:call-template name="locMapLink" >
                <xsl:with-param name="mfhd" select="$mfhdID"/>
                </xsl:call-template>
                <br/>
        </xsl:if>
    </xsl:for-each>
</xsl:template>

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

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?

13

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 13-1. 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:

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

1. Create and save a new file in the `/m1/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 13-1 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 13-1. 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">
   <xsl:attribute name="href">http://worldcatlibraries.org/XXXX/isbn/<xsl:value-of
          select="$isbn"/>&amp;loc=united+states</xsl:attribute>
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 13-2.

```
<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">

                <xsl:include href="./isbnSearch.xsl"/>
```

Figure 13-2. 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 13-3.

```
                    <!--/ul-->
                    <!-- ## add the other libraries search ## -->
                        <xsl:call-template name="recordIsbnSearch" />

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


</xsl:template>

<!-- ########################################################################## -->

</xsl:stylesheet>
```

Figure 13-3. Call for isbnSearch.xsl in displayFacets.xsl

4. (Optional) Set up trimData XSL code. See Figure 22-3 on page 22-4.

   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?

14

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 14-1. 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:

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

1. Create and save a new JavaScript file in the `/m1/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 14-1 for this example.

```
/*
#(c)#====================================================================
#(c)#
#(c)# Copyright 2007 ExLibris Group
#(c)# All Rights Reserved
#(c)#
#(c)#====================================================================

** Product : WebVoyage :: disable/enable limits on basic search, adjust search tips
** Version : 7.0
** Created : 23-JAN-2008
** Orig Author :
** Last Modified : 18-APR-2008
```

Figure 14-1. Example code for searchBasic.js

```
** Last Modified By : ASP
*/

////////////////////////////////////////////////////////////////////
function updateSearchTip ()
{
              // added to searchCode.onchange to customize search tip based on index


              // save the default 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</span>"
                break;


                case 'CALL+':
                    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
```

Figure 14-1. Example code for searchBasic.js (Continued)

```
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();
```

Figure 14-1. Example code for searchBasic.js (Continued)

```
    }
  }
}

addLoadEvent(addSearchCodeChanged);
```

Figure 14-1. Example code for searchBasic.js (Continued)

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

3. Edit `pageProperties.xml`.

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

   Add a `<div>` element as in Figure 14-2.

```
<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 14-2. Example element

4. Save your changes.

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

6. Edit `cl_searchBasic.xsl`.

   At the end of the `buildBasicSearch` template (see Figure 14-3), load the `searchBasic.js` JavaScript file you created.

```
<!-- ##################### -->
<!-- ## buildSearchForm ## -->
<!-- ########################################################## -->

<xsl:template name="buildBasicSearch">

   <div id="searchParams">
      <div id="searchInputs">
         <xsl:call-template name="buildFormInput">
            <xsl:with-param name="eleName"  select="'searchArg'"/>
            <xsl:with-param name="size"  select="'51'"/>
            <xsl:with-param name="accesskey"  select="'s'"/>
         </xsl:call-template>
         <xsl:call-template name="buildFormDropDown">
            <xsl:with-param name="eleName"  select="'searchCode'"/>
         </xsl:call-template>
      </div>
      <div id="quickLimits">
         <xsl:call-template name="buildFormDropDown">
            <xsl:with-param name="eleName"  select="'limitTo'"/>
         </xsl:call-template>
      </div>

      <!-- load javascript file for handling limits enable/disable -->
      <script type="text/javascript" src="{$jscript-loc}searchBasic.js"/>

   </div>
```

Figure 14-3. Locate buildBasicSearch template

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

</xsl:template>

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

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?

15

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 15-1. 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 15-1.

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

Figure 15-1. 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 15-2.

```
<form action="{/page:page//page:element[@nameId='page.logIn.logIn.button']/
          page:buttonAction}" method="GET" accept-charset="UTF-8"  name="selectDatabases"
          autocomplete="off">
```

Figure 15-2. Additional example of autocomplete=”off” code

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?

16

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

With Internet Explorer (IE) and Firefox, there is the capability to display favicons (favorite icons). These may display in the address bar, favorites menu, page tabs, and bookmarks and provide you with a way to brand your catalog for your patrons.

For the best results using favicons with WebVoyáge and Internet Explorer, use version 8 or later of Internet Explorer.

NOTE:
There are multiple methods for installing a favicon. The method described in this chapter is specific to WebVoyáge 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:

- `http://msdn2.microsoft.com/en-us/library/ms537656.aspx`.
- `http://en.wikipedia.org/wiki/Shortcut_icon`.

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 16-1. 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:

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

1. Create a 16x16 pixel icon.
2. Save the icon to the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/` directory.
3. Make a backup copy of `frameWork.xsl` that is located in `/m1/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 16-1, after the <head> element.

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

Figure 16-1. Example favicon code for frameWork.xsl

   c. Replace `favicon.ico` with the name of the icon file that you created at the begining of this procedure and saved in .../`images/`.

d. Also, add the `favicon.ico` to the `DocumentRoot` for `vwebv`. The `DocumentRoot` is defined as `@VOYAGER_BASE@/@DBNAME@/tomcat/vwebv/context/vwebv/htdocs`.

NOTE:
By default, the `{$image-loc}` notation is a path to the `/m1/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?

17

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 17-1. 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:

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

1. Create and save a new JavaScript file in the `/m1/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 17-1 for this example.

```
/*
#(c)#====================================================================
#(c)#
#(c)# Copyright 2007 ExLibris Group
#(c)# All Rights Reserved
#(c)#
#(c)#====================================================================

** 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) {
```

Figure 17-1. Example code for searchAdvanced.js

```
  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='';
              });
```

Figure 17-1. Example code for searchAdvanced.js (Continued)

2. Make a backup copy of `searchAdvanced.css` that is located in `/m1/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 17-2.

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

Figure 17-2. Example code for searchAdvanced.css

4. Save your changes.

5. Make a backup copy of `cl_searchAdvanced.xsl` that is located in `/m1/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 17-3.

```
<!-- 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  -->
```

Figure 17-3. 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 17-4.

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


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


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

</xsl:stylesheet>
```

Figure 17-4. 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?

18

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 18-1. 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:

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

1. Create and save a new file in the `/m1/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 18-1 for this example.

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
#(c)#=========================================================================
#(c)#
#(c)#       Copyright 2007-2009 Ex Libris (USA) Inc.
#(c)#       All Rights Reserved
#(c)#
#(c)#=========================================================================
-->

<!--
**                        Note: create persistent link based on bib ID
**          Product : WebVoyage :: local_PersistentLink
**          Version : 7.1.0
**          Created : 16-Nov-2007
**      Orig Author : Eric
**    Last Modified : 04-MAY-2009
** Last Modified By : Mel Pemble
-->

<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="persistentLink">
<xsl:param name="bibID"/>

              <!-- :doc: you must set this to match your public URL
              where  xxx = the ip
                and pppp = the port
              -->
              <xsl:variable name="baseURL">http://xxx.xxx.xxx.xxx:pppp/vwebv/
          holdingsInfo?bibId=</xsl:variable>
```

Figure 18-1. Sample code for local_PersistentLink.xsl

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


<!-- ======================================================================== -->


</xsl:stylesheet>
```

Figure 18-1. 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 and a variable definition for `linkID` after the `<xsl:stylesheet>` element. See Figure 18-2.

```
<xsl:stylesheet version="1.0"
              exclude-result-prefixes="xsl fo page"
              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"


<!-- ## Include Persistent Link template ## -->


<xsl:include href="../pageFacets/local_PersistentLink.xsl"/>
<xsl:variable name="linkID" select="//page:page/page:pageBody/page:holdingsCollection/
          hol:holdingsRecord/hol:bibRecord/@bibId"/>
```

Figure 18-2. Include file reference and variable definition example

    b. Add an xmlns attribute to the xsl:stylesheet tag, to define the holdings namespace. See Figure 18-2.

c. Call the persistent link template from the `Bibliographic Data` section. See Figure 18-3.

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

Figure 18-3. Persistent link template call with linkID in the Bibliographic Data section

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 18-4 to `displayCommon.css`.

```
#persistentLink
{
   clear:both;
}
```

Figure 18-4. Sample displayCommon.css code

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?

19

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 19-1, Add Class Attributes For Formatting, on page 19-2 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 19-1. 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:

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

1. Make a backup copy of `displaycfg.xml` that is located in `/m1/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 19-1 for examples.

```
<displayTags label="Title:" localcssclass="tagTitle">
<displayTags label="Author:" localcssclass="tagAuthor">
```

Figure 19-1. Class attribute creation example

3. Save your changes to the `displaycfg.xml`.
4. Make a backup copy of `display.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/display/`.
5. Edit `display.xsl`.

a. Locate the template for buildMarcDisplay.

b. Add the lines shown in Figure 19-2 to assign one of two classes to the bibliographic tag data.

```
                                        </a><br/>

                                    </xsl:for-each>
                                </div>
                            </li>
                        </xsl:if>
                    </xsl:if>



<!-- assign class-->
                    <li>
                        <xsl:attribute name="class">
                            <xsl:choose>
                                <xsl:when test="string-length(@localcssclass)">
                                    <xsl:value-of select="@localcssclass" />
                                </xsl:when>
                                <xsl:otherwise>bibTag</xsl:otherwise>
                            </xsl:choose>
                        </xsl:attribute>
                        <xsl:copy-of select="$bibTag"/>
                    </li>
<!-- assign class dinking -->

<!-- comment out these lines if using localcssclass
                  <li class="bibTag">
                     <xsl:copy-of select="$bibTag"/>
                  </li>
-->
               </xsl:if>
```

Figure 19-2. Example code to add to buildMarcDisplay

```
            </xsl:for-each>


            </ul>
          </div>
        </xsl:when>
```

Figure 19-2. Example code to add to buildMarcDisplay (Continued)

6. Save your changes to `display.xsl`.
7. Make a backup copy of `displayCommon.css` that is located in `/m1/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 19-3.

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

Figure 19-3. 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?

20

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 WebVoyáge 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 20-1. 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 20-1.

```
<!-- ################ -->
<!-- ## buildFooter ## -->
<!-- ################################################################## -->

<xsl:template name="buildFooter">

   <xsl:for-each select="/page:page/page:pageFooter">
```

Figure 20-1. Example of script snippet add to buildFooter template

```
      <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>

         </xsl:for-each>

         <div id="libraryLink">
            <span>
               <xsl:call-template name="buildLinkType">
                  <xsl:with-param name="eleName"  select="'page.footer.library.link'"/>
               </xsl:call-template>
            </span>
         </div>
```

```
         <div id="copyright" title="{$footerText/copyright}"><span><xsl:value-of
           select="page:element[@nameId='page.footer.copyright.message']/page:messageText"/
           ></span></div>


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


<!-- Google Analytics snippet -->
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js'
          type='text/javascript'%3E%3C/script%3E"));
</script>
<script type="text/javascript">
var pageTracker = _gat._getTracker("UA-xxxxxx-x");
pageTracker._initData();
pageTracker._trackPageview();
</script>


</xsl:template>
```

Figure 20-1. Example of script snippet add to buildFooter template (Continued)

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?

21

Description For "How Do I Implement Google Book Search?"

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

This feature is enabled at installation. To disable it, see Procedure 21-1, Disable Google Book Search Feature, on page 21-3.

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:

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

googleBooksAvail.js

The `googleBooksAvail.js` script in `/m1/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 21-1 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 21-1. Action Box text

local_googleBooksAvail.xsl

The `local_googleBooksAvail.xsl` file in `/m1/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 21-2.

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 21-1. 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 21-2.

```
## mdp add the google book template ##
```

Figure 21-2. Line of code to locate

b. Comment out the lines of code as shown in Figure 21-3.

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

Figure 21-3. 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`.

How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?

22

Description For "How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?"

WebVoyáge 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`.

- `display.xsl`.
- `displayRecord.xsl`.

Syndetic Solutions Implementation

This section describes, as an example, the WebVoyáge 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:

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

pageProperties.xml

The `pageProperties.xml` file located in `/m1/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 22-1.

```
<!-- each library is responsible for creating a relationship with a provider of cover images
          -->
<!-- these options should not be enabled by default -->


<!-- <resultsCoverTag nameIdMatch="page.searchResults.item.type.isbn" linkPRE_TEXT="http:/
          /images.sample.com/images/" linkPOST_TEXT=".gif" altText="Cover Image"/> -->


<!--resultsCoverTag nameIdMatch="page.searchResults.item.type.isbn" linkPRE_TEXT="http://
          www.syndetics.com/hw7.pl?isbn=" linkPOST_TEXT="/SC.gif" altText="Cover Image"/ -
          ->
```

Figure 22-1. Example of pageProperties.xml

The `/SC.gif` in Figure 22-1 is the Syndetics Solutions syntax for small cover (SC). If you prefer a medium or large cover image, use MC or LC, respectively.

resultsFacets.xsl

The `resultsFacets.xsl` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/` defines a template named `buildResultsCoverImage`. See Figure 22-2.

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

```
<!-- ############################ -->
<!-- ## buildResultsCoverImage ## -->
<!--
          ##########################################################################################
          ##################################### -->
<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="{@linkPRE_TEXT}{$tag}{@linkPOST_TEXT}"
          class="resultListCoverImage" alt="{@altText}" style="display:none"
          onload="checkImage(this)"/>
                    </div>
             </xsl:for-each>
</xsl:template>
```

Figure 22-2. 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 22-3.

```
<!-- ## cover image ## --
<xsl:choose>
               <xsl:when test="string-length(page:option/
          page:element[@nameId='page.searchResults.item.type.isbn']/page:value)">
               <!-- ## cover image from isbn ## -->
                   <xsl:call-template name="buildResultsCoverImage">
                       <xsl:with-param name="tag">
                           <xsl:call-template name="trimData">
                               <xsl:with-param name="sData" select="page:option/
          page:element[@nameId='page.searchResults.item.type.isbn']/page:value"/>
                           </xsl:call-template>
                       </xsl:with-param>
                       <xsl:with-param name="tagType"
          select="'page.searchResults.item.type.isbn'"/>
                   </xsl:call-template>
               </xsl:when>
               <xsl:when test="string-length(page:option/
          page:element[@nameId='page.searchResults.item.type.issn']/page:value)">
               <!-- ## cover image from issn ## -->
                   <xsl:call-template name="buildResultsCoverImage">
                       <xsl:with-param name="tag">
                           <xsl:call-template name="trimData">
                               <xsl:with-param name="sData" select="page:option/
          page:element[@nameId='page.searchResults.item.type.issn']/page:value"/>
                           </xsl:call-template>
                       </xsl:with-param>
                       <xsl:with-param name="tagType"
          select="'page.searchResults.item.type.isbn'"/>
                   </xsl:call-template>
               </xsl:when>
</xsl:choose>
```

Figure 22-3. Example of trimData template call in resultsFacets.xsl

resultsTitles.xsl

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`.

imageUtils.js

The `imageUtils.js` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/jscripts/` performs the functions shown in Figure 22-4.

```
   var setDefaultErrImg=""; // Default image to be displayed on error
   var setDefaultErrTxt=""; // Default text to be displayed on error

////////////////////////////////////////////////////////////////////

function checkImage(obj)
{

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

////////////////////////////////////////////////////////////////////
```

Figure 22-4. 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.

displaycfg.xml

The `displaycfg.xml` file located in `/m1/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 22-5.

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.

```
<!-- 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="
           linkPOST_TEXT="/MC.jpg" infoPRE_TEXT="http://syndetics.com/index.aspx?isbn="
           infoPOST_TEXT="/index.html&amp;client=ENTER_YOUR_CLIENT_ID&amp;type=rn12"
           singleInstance="true">
      <displayTag field="020" indicator1="X" indicator2="X" subfield="a"/>
    </coverTags>
-->
```

Figure 22-5. 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 `/m1/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 22-3.

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:

```
http://www.syndetics.com/
index.aspx?isbn=xxx&amp;client=code&amp;showCap
```

What is a Query String?

Consider a Syndetics URL:

```
http://www.syndetics.com/index.aspx?isbn=0002154129/
SC.GIF&client=clientcode&showCaptionBelow=t&caption=Click+fo
r+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 "&":

`isbn=0002154129/SC.GIF`

`client=clientcode`

`showCaptionBelow=t`

`caption=Click+for+more+info` (this is a custom caption)

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:

```
<http://www.syndetics.com/index.aspx?isbn=0002154129/
SC.GIF&client=clientcode&showCaptionBelow=t>
```

Superimposed caption:

```
<http://www.syndetics.com/index.aspx?isbn=0002154129/
SC.GIF&client=clientcode&showCaptionSuperimposed=t>
```

Caption with Medium size Cover:

```
<http://www.syndetics.com/index.aspx?isbn=0002154129/
MC.GIF&client=clientcode&showCaptionBelow=t>
```

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

```
<http://www.syndetics.com/index.aspx?isbn=0002154129/
MC.GIF&client=clientcode&showCaptionBelow=t&caption=Custom+m
essage+goes+here&bgColor=red>
```

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

```
<http://www.syndetics.com/index.aspx?isbn=0002154129/
SC.GIF&client=clientcode&showCaptionBelow=t&bgColor=238,238,
238>
```

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:

```
<http://www.syndetics.com/index.aspx?isbn=0002154129/
SC.GIF&client=clientcode&showCaptionBelow=t&caption=Syndetic
s+Truly+Rocks&bgColor=pink>
```

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

```
<http://www.syndetics.com/index.aspx?isbn=0002154129/
MC.GIF&client=clientcode&showCaptionBelow=t&caption=Syndetic
s+Really+and+Truly+Totally+Awesomelly+Rocks&bgColor=pink>
```

SC.GIF (**S**mall **C**over)

MC.GIF (**M**edium **C**over)

How Do I Implement Geospatial Search?

23

Description For "How Do I Implement Geospatial Search?"

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

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

WebVoyáge 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 23-1 on page 23-3 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:

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

See Procedure 23-1, Geospatial Search Implementation for the steps to provide access to Geospatial Search.

Procedure 23-1. Geospatial Search Implementation

Use the following steps to implement Geospatial Search.

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 `option.geospatialSearch=`.

3. Set this option to ***Y*** to have WebVoyáge display a **Geospatial Search** tab. See Figure 23-1 for a display example.

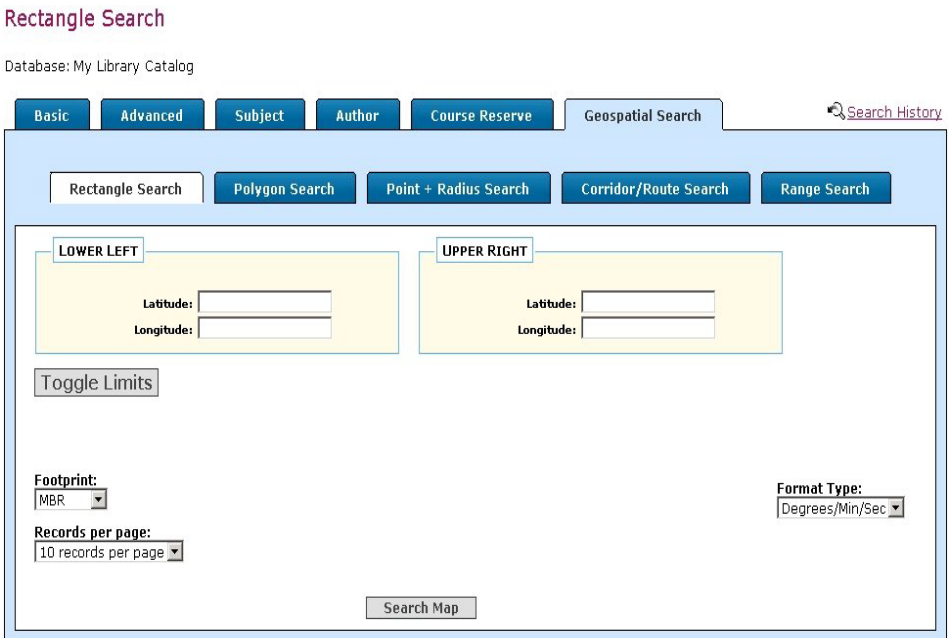

Figure 23-1. Geospatial Search tab

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

5. Open the `pageProperties.xml` file in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/xsl/userTextConfigs/` and locate `<!-- enable these lines if site has geospatial searching`.

   See Figure 23-2.

```
Line#
  1     <!-- ## Search Tab Display Order ## -->
  2     <searchTabDisplayOrder>
  3        <tab name="page.search.buttons.basic.button"/>
  4        <tab name="page.search.buttons.advanced.button"/>
  5        <tab name="page.search.buttons.subjectHeading.button"/>
  6        <tab name="page.search.buttons.author.button"/>
  7        <tab name="page.search.buttons.courseReserve.button"/>
  8        <!-- enable these lines if site has geospatial searching
  9        <tab name="page.search.buttons.geospatial.button"/>
 10        <tab name="page.search.geospatial.button"/>
 11        -->
 12     </searchTabDisplayOrder>
 13
 14     <!-- ## GeoSearch Tab Display Order ## -->
 15     <geosearchTabDisplayOrder>
 16        <tab name="map.search.buttons.rectangleSearch.button"/>
 17        <tab name="map.search.buttons.polygonSearch.button"/>
 18        <tab name="map.search.buttons.pointRadiusSearch.button"/>
 19        <tab name="map.search.buttons.corridorSearch.button"/>
 20        <tab name="map.search.buttons.rangeSearch.button"/>
 21     </geosearchTabDisplayOrder>
```

Figure 23-2. Enable Geospatial Search in the pageProperties.xml file

6. Remove the comment lines to enable Geospatial Search.

   In Figure 23-2, the comment lines are lines 8 and 11.

7. Save your changes.

How Do I Enable External Authentication?

24

Description For "How Do I Enable External Authentication?"

External authentication is an optional setting in WebVoyáge to enable patron authetication 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 24-1, External Authentication Implementation for the steps to enable external authentication.

Procedure 24-1. 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 `option.extAuthSystemEnabled=N`. See Figure 24-1.

```
#==============================
# 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

                  ========== truncated ==========

#============================
```

Figure 24-1. 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
#
#########################################################################
```

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 WebVoyáge using the following format:

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

   This URL indicates to WebVoyáge 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?

25

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 25-3 on page 25-4.

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 25-1 for an example of an `errorCode` identified for a customized message.

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

Figure 25-1. errorCode example

See Figure 25-2 for a `blockCode` example used when an SSN login error occurs.

```
<pageMsg blockCode="PATRONSOCMSG">
```

Figure 25-2. blockCode example

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:

```
/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/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:

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

See Procedure 25-1, Modify Page Messages for the steps to modify page messages.

Procedure 25-1. Modify Page Messages

Use the following steps to modify page messages.

1. Make a backup copy of `pageProperties.xml` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/userTestConfigs/`.

2. Open the `pageProperties.xml` file and locate `<!-- ## Override [blockCode]`. See Figure 25-3.

```
<!-- ## 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> eith 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">
```

Figure 25-3. Modifying page messages example

```
         <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">
```

```
        <p class="blockMessage">
        You may not have entered your institution id and name correctly.
        <br/>Retry your request or ask for help at the Circulation or Reference Desk.
        </p>
    </pageMsg>
    <pageMsg blockCode="PATRONBRIEFMSG">
        <p class="blockMessage">
        The system could not identify you from your ID number alone.
       <br/>Please choose your home library and ID number type on this form and try again,
        <br/>or ask for help at the Circulation or Reference Desk.
        </p>
    </pageMsg>
    <pageMsg errorCode="searchResults.noHits">Search resulted in no hits.</pageMsg>
</pageMessages>
```

Figure 25-3. Modifying page messages example (Continued)

3. Define/modify the page message text to match your preference.
4. Save the updated `pageProperties.xml` file.
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?

26

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 26-1. 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:

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

1. Make a backup copy of `pageProperties.xml` that is located in `/m1/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 26-1.

```
<!-- ## Search Tab Display Order ## -->
<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 26-1. Search Tab Display Order section

   b. Comment out the `courseReserve` line in this section. See Figure 26-2.

```
<!-- ## Search Tab Display Order ## -->
<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"/>
   <!-- Tturn off Course Reserve tab
   <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 26-2. Comment out course reserve example

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 `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/htdocs/`.
5. Edit `index.html`.
    a. Find the `More choices` paragraph. See Figure 26-3.

```
<p>More choices:</p>
<ul>
    <li><a href="/vwebv/searchBasic?sk=en_US">Basic search</a></li>
    <li><a href="/vwebv/searchAdvanced?sk=en_US">Advanced search</a></li>
    <li><a href="/vwebv/enterCourseReserve.do?sk=en_US">Course reserve
materials</a></li>
    <li><a href="/vwebv/login?sk=en_US">Log in to use your saved preferences</
a></li>
    <li><a href="/vwebv/myAccount?sk=en_US">Review your account</a></li>
    <li><a href="/vwebv/ui/en_US/htdocs/help/index.html">Read help for
WebVoyÃ¡ge</a></li>
```

Figure 26-3. More choices paragraph example

b. Comment out the `courseReserve` line in this section. See Figure 26-4.

```
<p>More choices:</p>
<ul>
    <li><a href="/vwebv/searchBasic?sk=en_US">Basic search</a></li>
    <li><a href="/vwebv/searchAdvanced?sk=en_US">Advanced search</a></li>
    <!-- Removing course reserve reference
    <li><a href="/vwebv/enterCourseReserve.do?sk=en_US">Course reserve
materials</a></li>
    -->
    <li><a href="/vwebv/login?sk=en_US">Log in to use your saved preferences</
a></li>
    <li><a href="/vwebv/myAccount?sk=en_US">Review your account</a></li>
    <li><a href="/vwebv/ui/en_US/htdocs/help/index.html">Read help for
WebVoyÃ¡ge</a></li>
```

Figure 26-4. 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?

27

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 27-1. Create New Search Tab

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

This procedures 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:

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

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

2. Add labels for the new tab in the Search Pages section. See Figure 27-1.

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

Figure 27-1. Labels for new search tab example

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

4. Bind the new tab to an action. Add the code shown in Figure 27-2 to the Search section of `internal.properties`.

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

Figure 27-2. Bind new tab in Search section example

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

6. Add the XML to create the new tab. Find the `searchTabDisplayOrder` element and add the code shown in Figure 27-3 to `pageProperties.xml`.

```
<tab name="page.search.buttons.newBooks.button"/>
```

Figure 27-3. Example XML for new search tab

Place the code where you would like the new search tab to display. See Figure 27-4 for an example.

```
<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 27-4. 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?

28

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 28-1. Create New Header Tab

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

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. 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/xxxdb/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 28-1.

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

Figure 28-1. 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/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/`.

5. Set the new header tab label in the Header section of `webvoyage.properties` as in Figure 28-2.

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

Figure 28-2. 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 28-3.

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

Figure 28-3. internal.properties file example changes

The `.action` value is a URL. This can be an existing WebVoyáge 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?

29

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 29-1. 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:

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

1. Copy `displayRecord.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/` to `displayBriefRecord.xsl`.

2. Change the template call from `buildRecordForm` to `buildBriefRecordForm` as shown in Figure 29-1.

```
<xsl:template name="buildContent">
     <xsl:call-template name="buildBriefRecordForm"/>
</xsl:template>
```

Figure 29-1. Change template call

3. Modify existing code (see Figure 29-2) in `displayBriefRecord.xsl` with the code shown in Figure 29-3.

```
<xsl:include href="./contentLayout/cl_displayRecord.xsl"/>
```

Figure 29-2. Existing displayRecord code

```
<xsl:include href="./contentLayout/cl_displayBriefRecord.xsl"/>
```

Figure 29-3. Example modification for displayBriefRecord code

4. Copy `cl_displayRecord.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/` to `cl_displayBriefRecord.xsl`.

5. Change the template call from `buildRecordForm` to `buildBriefRecordForm` as shown in Figure 29-4.

```
<!-- ################## -->
<!-- ## buildBriefRecordForm ## -->
<!-- ############################################################ -->

<xsl:template name="buildBriefRecordForm">
```

Figure 29-4. Change template call

6. Modify existing code (see Figure 29-5) in `cl_displayBriefRecord.xsl` with the code shown in Figure 29-6.

```
<xsl:variable name="Config" select="document('./configs/displaycfg.xml')"/>
```

Figure 29-5. Existing cl_displayRecord code

```
<xsl:variable name="Config" select="document('./configs/
          displaybriefcfg.xml')"/>
```

Figure 29-6. Example modification for cl_displayBriefRecord code

7. Modify the existing Action Box code (see Figure 29-7) in `cl_displayBriefRecord.xsl` with the code shown in Figure 29-8.

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

            <xsl:call-template name="buildActionBox">

                <xsl:with-param name="pageRecordType"
            select="'actionBox.recordView.link'"/>

            </xsl:call-template>
```

Figure 29-7. Existing Action Box code

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

          <xsl:variable name="moreActions">

              <li><span class="recordPointer"> </span><label>Brief
           Record</label></li>

          </xsl:variable>

          <xsl:call-template name="buildActionBox">

              <xsl:with-param name="pageRecordType"
           select="'actionBox.briefRecordView.link'"/>

              <xsl:with-param name="moreActions" select="$moreActions"/>

          </xsl:call-template>
```

Figure 29-8. Example modification of Action Box code

8. 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/`.

9. Modify the existing Action Box code (see Figure 29-9) in `cl_displayRecord.xsl` with the code shown in Figure 29-10.

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


            <xsl:call-template name="buildActionBox">


                <xsl:with-param name="pageRecordType"
            select="'actionBox.recordView.link'"/>


            </xsl:call-template>
```

Figure 29-9. Existing Action Box code in cl_displayRecord

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


          <xsl:variable name="briefRecordURL">briefHoldingsInfo?<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 29-10. Example modification to Action Box code in cl_displayRecord

10. Modify the existing Action Box code (see Figure 29-11) in `cl_displayStaff.xsl` with the code shown in Figure 29-12.

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


            <xsl:call-template name="buildActionBox">


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


            </xsl:call-template>
```

Figure 29-11. Existing Action Box code in cl_displayStaff

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


          <xsl:variable name="briefRecordURL">briefHoldingsInfo?<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.staffView.link'"/>


              <xsl:with-param name="moreActions" select="$moreActions"/>


          </xsl:call-template>
```

Figure 29-12. Example modification to Action Box code in cl_displayStaff

11. Copy `displaycfg.xml` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/` to `displaybriefcfg.xml`

12. Modify `displaybriefcfg.xml` with your preferences.

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

14. Modify existing code (see Figure 29-13) in `displayFacets.xsl` with the code shown in Figure 29-14.

```
        <xsl:template name="buildActionBox">

        <xsl:param name="pageRecordType"/>
```

Figure 29-13. Existing displayFacets code

```
        <xsl:template name="buildActionBox">

        <xsl:param name="pageRecordType"/>

        <xsl:param name="moreActions"/>
```

Figure 29-14. Modification example for displayFacets code

15. Add the code shown in Figure 29-15 after the code shown in Figure 29-16 in the `displayFacets.xsl` file.

```
<xsl:copy-of select="$moreActions"/>
```

Figure 29-15. Example code to add to displayFacets.xsl

```
<xsl:for-each select="page:element">

    <xsl:choose>

        <xsl:when test="@nameId=$pageRecordType">

            <li><span class="recordPointer"> </
  span><label><xsl:value-of select="page:linkText"/></label></li>

        </xsl:when>

        <xsl:otherwise>

            <li><span class="recordLinkBullet">·</span><a
  href="{page:URL}"><span><xsl:value-of select="page:linkText"/></
  span></a></li>

        </xsl:otherwise>

    </xsl:choose>

</xsl:for-each>
```

Figure 29-16. Existing code in displayFacets.xsl

16. Make a backup copy of the `web.xml` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/WEB-INF/`.

17. Add the code shown in Figure 29-17 to `web.xml` after the last `<filter-mapping>` stanza that references `SelectSkin Filter` but before any other type of stanza.

```
<filter-mapping>

    <filter-name>SelectSkin Filter</filter-name>

    <url-pattern>/briefHoldingsView/*</url-pattern>

</filter-mapping>
```

Figure 29-17. filter-mapping code example

18. Add the code shown in Figure 29-18 to `web.xml` after the last `<servlet-mapping>` stanza but before any other type of stanza.

```
<servlet-mapping>

    <servlet-name>BriefHoldingsInfoServlet</servlet-name>

    <url-pattern>/briefHoldingsInfo/*</url-pattern>

</servlet-mapping>
```

Figure 29-18. servlet-mapping code example

19. Add the code shown in Figure 29-19 to `web.xml` after the last `<servlet>` stanza but before any other type of stanza.

```
<servlet>

    <servlet-name>BriefHoldingsInfoServlet</servlet-name>

    <display-name>BriefHoldingsInfoServlet</display-name>
```

Figure 29-19. Example modification code for web.xml

```
        <servlet-class>

            com.endinfosys.voyager.webvoyage.servlet.PageServlet

        </servlet-class>

        <init-param>

            <param-name>PageCode</param-name>

            <param-value>briefHoldingsInfo</param-value>

        </init-param>

        <init-param>

            <param-name>PageClass</param-name>

            <param-
           value>com.endinfosys.voyager.webvoyage.pages.HoldingsInfoPage</
           param-value>

        </init-param>

        <init-param>

            <param-name>XSLTemplateName</param-name>

            <param-value>displayBriefRecord.xsl</param-value>

        </init-param>

        <init-param>

            <param-name>Properties-Custom</param-name>

            <param-value>webvoyage</param-value>
```

Figure 29-19. Example modification code for web.xml (Continued)

```
        </init-param>

        <init-param>

            <param-name>Properties-Internal</param-name>

            <param-value>internal</param-value>

        </init-param>

        <init-param>

            <param-name>HelpPage</param-name>

            <param-value>holdingsInfo.html</param-value>

        </init-param>

    </servlet>
```

Figure 29-19. Example modification code for web.xml (Continued)

20. Add the code shown in Figure 29-20 at the end of the following file:

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

```
JkMount /vwebv/bri* ajp13_lb_XXXdb_vwebv
```

Figure 29-20. Example code to add to XXXdb.jkmounts.conf

The `bri` in Figure 29-20 matches the beginning of the name of the new brief holdings view (briefHoldingsInfo).

NOTE:
This change requires that Apache be restarted.

21. Save and test your changes.

OPTIONAL:

22. Back out your changes, if necessary, by deploying your backup file copies.

How Do I Implement DOI and URN Handling?

30

DOI/URN Overview

WebVoyáge 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

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

DOI/URN Implementation

This section describes the WebVoyáge 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:

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

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:

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

See Table 30-1 for the default code/configuration provided at installation.

Table 30-1. DOI/URN Implementation

`# Print property DOI and URN URL to convert the DOI and URN to the physical location`
`# of the digital item`
`#=========================================================================#`
`property.DOI=http://hdl.handle.net/`
`property.URN=http://hdl.handle.net/`

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)?

31

Hook to Holdings Implementation

The Hook to Holdings functionality enables WebVoyáge to display holdings data from the local database when a remote (zcit or vcit) database is searched. See Figure 31-1 for a holdings display example. In Figure 31-1, 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*.

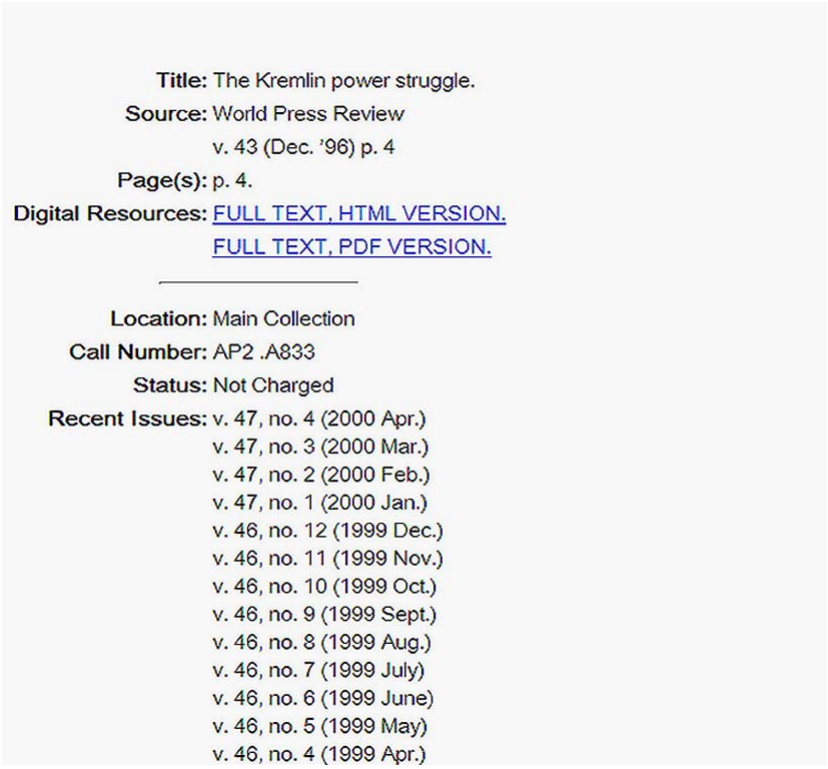

Figure 31-1. Hook to Holdings display example

How Do I Implement HTTP Post to Link Resolver?

32

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 WebVoyáge server and the `voyager.ini` file that resides on the client PC.

voyager.ini

To configure the `voyager.ini` file, edit the `[MARC POSTing]` stanza. See Figure 32-1 for an example.

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

Figure 32-1. voyager.ini configuration example

The `[MARC POSTing]` stanza is designed to display the MARC record in WebVoyáge 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 32-2.

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.

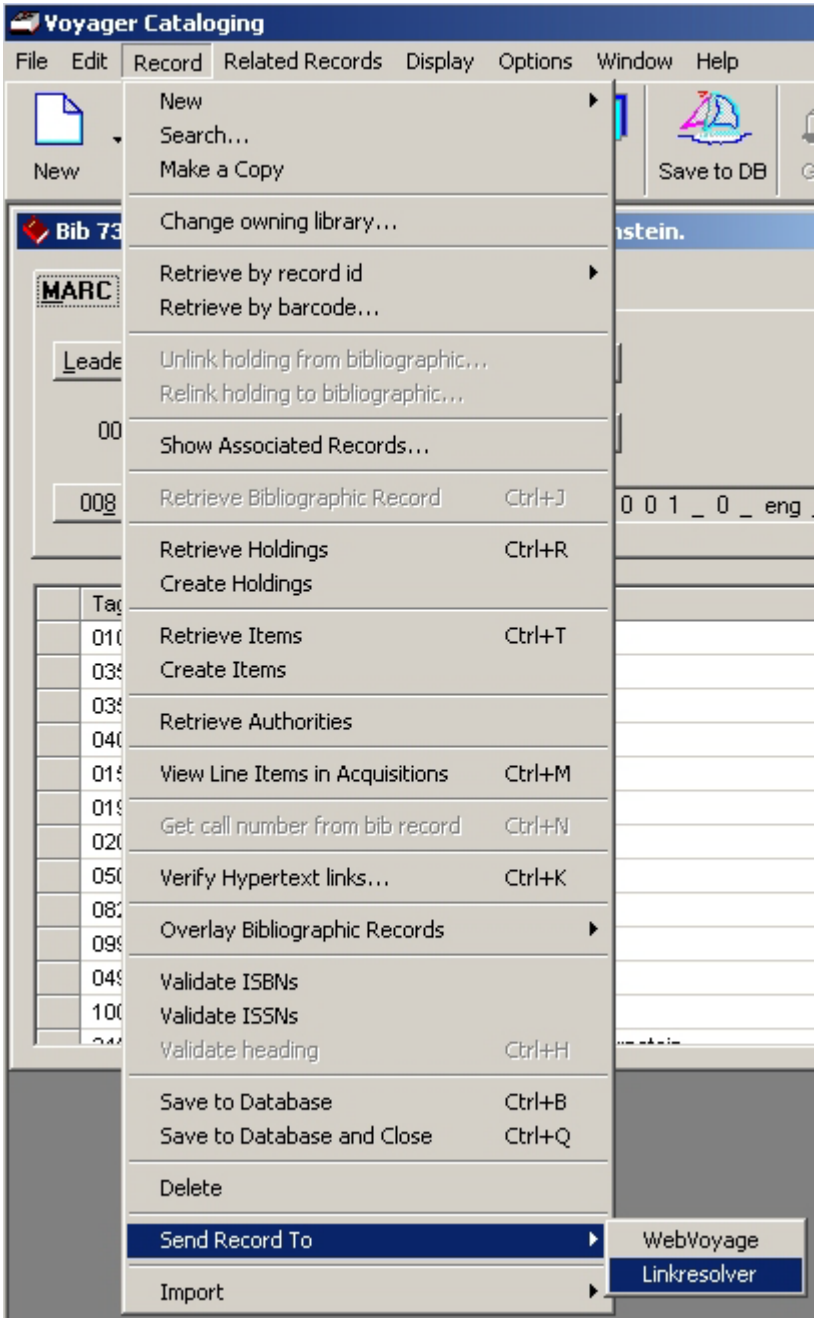

Figure 32-2. Linkresolver option example

linkresolver.properties

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 32-3.
- Identify the fields to be used for the OpenURL construction and how to parse them. See Figure 32-4 for a citation database example.

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

Figure 32-3. Link Resolver URL example

For each field identified (as in Figure 32-4), the following keys need to be specified (except for genre):

- Field name
- Tag/field number
- Subfield

  Multiple subfields may be specified for this parameter formatted in the following manner:

  `openUrl.cfg.LOCAL.key1.subfield=abcpz`

  You may also use an asterisk to indicate all subfields available for the tag as in the example below:

  `openUrl.cfg.LOCAL.key1.subfield=*`
- Length 1 (number of positions)
- Parse start
- Parse end
- Length 2 (number of positions)

For genre, the definition is configured in the following manner:

- A combination of leaders 6 and 7 (in the bibliographic record) is used

  An asterisk may be used for either leader 6 or 7 or both.

  Example:

  `openUrl.cfg.LOCAL.genre1=*m:book`

  `openURL.cfg.LOCAL.genre7=**:book`

  Matching is done sequentially based on the order of the genre definitions in the `linkresolver.properties` configuration.
- When leader 6 and 7 values match the `linkresolver.properties` definitions, the genre tag in the URI is the text string identified after the colon

The possible genres are:

- Journal
- Book
- Conference
- Article
- Preprint
- Proceeding
- Bookitem

Multiple genres may be specified. The default genres are:

```
openUrl.cfg.LOCAL.numGenres=7
openUrl.cfg.LOCAL.genre1=*m:book
openUrl.cfg.LOCAL.genre2=*s:journal
openUrl.cfg.LOCAL.genre3=*a:bookitem
openUrl.cfg.LOCAL.genre4=*d:bookitem
openUrl.cfg.LOCAL.genre5=*b:article
openURL.cfg.LOCAL.genre6 =*i:article
openURL.cfg.LOCAL.genre7=**:book
```

In the Figure 32-4 example, the following fields are defined:

- Genre
- Title
- Author’s last name

- Author's first name
- Date/Year (260)
- ISSN
- ISBN
- CODEN
- Volume
- Issue
- Year (773)
- Spage (first page number/start page)

```
#genre definition
openUrl.cfg.LOCAL.numGenres=7
openUrl.cfg.LOCAL.genre1=*m:book
openUrl.cfg.LOCAL.genre2=*s:journal
openUrl.cfg.LOCAL.genre3=*a:bookitem
openUrl.cfg.LOCAL.genre4=*d:bookitem
openUrl.cfg.LOCAL.genre5=*b:article
openURL.cfg.LOCAL.genre6=*i:article
openURL.cfg.LOCAL.genre7=**:book
#title=245/a/0///0/
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
#aulast=100/a/0//, /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=,
```

Figure 32-4. Example of fields/subfields identified

```
openUrl.cfg.LOCAL.key2.len2=0
#aufirst=100/a/0/, //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
#date-year=260/c/4//./0/
openUrl.cfg.LOCAL.key4.key=date-year
openUrl.cfg.LOCAL.key4.tag=260
openUrl.cfg.LOCAL.key4.subfield=c
openUrl.cfg.LOCAL.key4.len1=4
openUrl.cfg.LOCAL.key4.parseStart=
openUrl.cfg.LOCAL.key4.parseEnd=.
openUrl.cfg.LOCAL.key4.len2=0
#issn=022/a/9///0/
openUrl.cfg.LOCAL.key5.key=issn
openUrl.cfg.LOCAL.key5.tag=022
openUrl.cfg.LOCAL.key5.subfield=a
openUrl.cfg.LOCAL.key5.len1=9
openUrl.cfg.LOCAL.key5.parseStart=
openUrl.cfg.LOCAL.key5.parseEnd=
openUrl.cfg.LOCAL.key5.len2=0
#isbn=020/a/10///0/
openUrl.cfg.LOCAL.key6.key=isbn
openUrl.cfg.LOCAL.key6.tag=020
openUrl.cfg.LOCAL.key6.subfield=a
openUrl.cfg.LOCAL.key6.len1=10
openUrl.cfg.LOCAL.key6.parseStart=
openUrl.cfg.LOCAL.key6.parseEnd=
openUrl.cfg.LOCAL.key6.len2=0
#coden=030/a/6///0/
openUrl.cfg.LOCAL.key7.key=coden
openUrl.cfg.LOCAL.key7.tag=030
openUrl.cfg.LOCAL.key7.subfield=a
```

Figure 32-4. Example of fields/subfields identified (Continued)

```
openUrl.cfg.LOCAL.key7.len1=6
openUrl.cfg.LOCAL.key7.parseStart=
openUrl.cfg.LOCAL.key7.parseEnd=
openUrl.cfg.LOCAL.key7.len2=0
#volume=773/g/0/Volume:/, Issue:/0/
openUrl.cfg.LOCAL.key8.key=volume
openUrl.cfg.LOCAL.key8.tag=773
openUrl.cfg.LOCAL.key8.subfield=g
openUrl.cfg.LOCAL.key8.len1=0
openUrl.cfg.LOCAL.key8.parseStart=Volume:
openUrl.cfg.LOCAL.key8.parseEnd=, Issue:
openUrl.cfg.LOCAL.key8.len2=0
#issue=773/g/0/Issue:/, Date:/0/
openUrl.cfg.LOCAL.key9.key=issue
openUrl.cfg.LOCAL.key9.tag=773
openUrl.cfg.LOCAL.key9.subfield=g
openUrl.cfg.LOCAL.key9.len1=0
openUrl.cfg.LOCAL.key9.parseStart=Issue:
openUrl.cfg.LOCAL.key9.parseEnd=, Date:
openUrl.cfg.LOCAL.key9.len2=0
#Year=773/g/0/Date://4/
openUrl.cfg.LOCAL.key10.key=Year
openUrl.cfg.LOCAL.key10.tag=773
openUrl.cfg.LOCAL.key10.subfield=g
openUrl.cfg.LOCAL.key10.len1=0
openUrl.cfg.LOCAL.key10.parseStart=Date:
openUrl.cfg.LOCAL.key10.parseEnd=
openUrl.cfg.LOCAL.key10.len2=4
#spage=300/a/0///0/
openUrl.cfg.LOCAL.key11.key=spage
openUrl.cfg.LOCAL.key11.tag=300
openUrl.cfg.LOCAL.key11.subfield=a
openUrl.cfg.LOCAL.key11.len1=0
openUrl.cfg.LOCAL.key11.parseStart=
openUrl.cfg.LOCAL.key11.parseEnd=
openUrl.cfg.LOCAL.key11.len2=0
```

Figure 32-4. Example of fields/subfields identified (Continued)

OpenURL Standard

The OpenURL standard identifies the key fields available for use such as title, aulast, aufirst, and so on. WebVoyáge 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.

Voyager supports both the 1.0 and 0.1 standard. Refer to the comments in the `linkresolver.properties` file for additional information.

How Do I Display Media Bookings in MyAccount?

33

Media Bookings Overview

When configured, Media Scheduling bookings activity displays on My Account page in WebVoyáge. 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

WebVoyáge 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 WebVoyáge 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 33-1.

```
option.mediaBookings.patronInfoDisplay=Y
```

Figure 33-1. Media bookings parameter setting in WebVoyáge

- Set the `page.myAccount.mediaBookings.cancelAllowed` parameter to `Y` (Yes). See Figure 33-2.

  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.

```
page.myAccount.mediaBookings.cancelAllowed=Y
```

Figure 33-2. Media bookings cancel allowed parameter example

- Modify label display and/or media bookings display options. See Figure 33-3 for the default settings.

  The `page.myAccount.mediaBookings.itemInfo` parameter may be set to the options in Table 33-1.

```
page.myAccount.mediaBookings.upcomingBookings=Upcoming Bookings
page.myAccount.mediaBookings.chargedBookings=Charged Bookings

page.myAccount.mediaBookings.select.label=Select All:
page.myAccount.mediaBookings.select.option.all=All
page.myAccount.mediaBookings.cancel.all=Y
page.myAccount.mediaBookings.cancel.button=Cancel Selected Bookings
page.myAccount.mediaBookings.cancel.button.message=Cancel Selected Bookings
page.myAccount.mediaBookings.cancelBooking=Cancel Booking
page.myAccount.mediaBookings.startTime=Start Time
page.myAccount.mediaBookings.endTime=End Time
page.myAccount.mediaBookings.confirmationNumber=Confirmation Number
page.myAccount.mediaBookings.bookingType=Booking Type
page.myAccount.mediaBookings.room=Room
page.myAccount.mediaBookings.equipmentCount=Equipment Count
page.myAccount.mediaBookings.itemCount=Item Count
page.myAccount.mediaBookings.bookingType.pickup=Pickup
page.myAccount.mediaBookings.bookingType.delivery=Delivery
page.myAccount.mediaBookings.equipment=Equipment
page.myAccount.mediaBookings.items=Items
page.myAccount.mediaBookings.itemInfo=\\t
```

Figure 33-3. Media bookings display options

Table 33-1. page.myAccount.mediaBookings.itemInfo options

Options	Description
\t	the item's title
\i	the item's enumeration, chronology and year
\n	the item's copy number
\c	the item's call number
\b	the item's barcode
\l	the item's location

Table 33-1. page.myAccount.mediaBookings.itemInfo options

Options	Description
\a	the item's author

How Do I Implement ImageServer in WebVoyáge?

34

WebVoyáge ImageServer Overview

For locations with ImageServer installed, WebVoyáge 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

WebVoyáge uses the following files to implement the ImageServer feature:

- `webvoyage.properties`.
- `display.xsl`.
- `resultsFacets.xsl`.

ImageServer Implementation

To implement the ImageServer function in WebVoyáge, you need to configure the `webvoyage.properties` file that resides on the WebVoyáge 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 can activate and customize this feature for your site. See Figure 34-1 and Figure 34-2.

```
#=========================================
# Option to activate (True) or deactivate (False) thumbnail
# Default is deactivate thumbnail
#=========================================
# option.thumbnail.activate=False
option.thumbnail.activate=True
```

Figure 34-1. webvoyage.properties ImageServer configuration example

```
#=============================================================================
#
# Image Server cofiguration for link, thumbnail alter text
#
#=============================================================================
#imageServer.scanDoc=http://localhost:classicWebVoyagePort/cgi-bin/Pscandoc.cgi?
#imageServer.scandocAltText=Scan Document
#imageServer.thumbnailAltText=Thumbnail
#imageServer.loginRequiredText=**** Login Required ****
```

Figure 34-2. webvoyage.properties ImageServer configuration example

Procedure 34-1. Implement ImageServer Function in WebVoyáge

To implement ImageServer function for WebVoyáge, do the following:

1. Confirm that the `option.thumbnail.activate` variable is active and set to the value of `True`. See Figure 34-1.

2. Remove any comment notation (#) to activate the `imageServer.<xxxxx>=` variables.

3. Customize the `imageServer.scanDoc` variable to reflect your sites's URL or IP address of the server running scandoc and the port of the Classic WebVoyáge.

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 `/m1/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 `/m1/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?

35

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 35-1) and the message to be displayed on the title and holdings pages. See Implement Messages for Status Patron Groups on page 35-2 for the steps to do this.

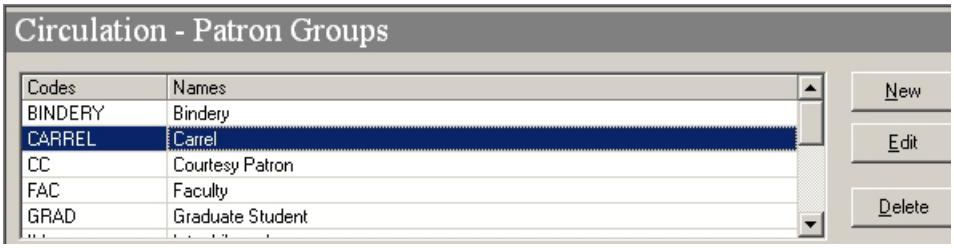

Figure 35-1. Example of Patron Groups in Voyager System Administration

Files

WebVoyáge 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 35-1. 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:

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

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

2. Locate the status patron groups messages section of the `webvoyage.properties` file. See Figure 35-2 for an example.

```
################################################################################
#
#  Status Patron Groups messages for display on title and holdings pages
```

Figure 35-2. Example of status patron groups messages section

```
#
################################################################################
page.searchResults.titles.statusPatronGroup.GROUPCODE=Message for title page
page.myAccount.chargedItem.item.statusPatronGroup.GROUPCODE=Message for holdings charged
          item
page.myAccount.reqPending.item.statusPatronGroup.GROUPCODE=Message for holdings with
          request pending
```

Figure 35-2. Example of status patron groups messages section (Continued)

3. Edit the status patron groups messages section to include entries for each patron group defined in Voyager System Administration. See Figure 35-3 for an example entry for the CARREL patron group code.

```
################################################################################
#
#  Status Patron Groups messages for display on title and holdings pages
#
################################################################################
page.searchResults.titles.statusPatronGroup.CARREL=At the carrel
page.myAccount.chargedItem.item.statusPatronGroup.CARREL=At the carrel
page.myAccount.reqPending.item.statusPatronGroup.CARREL=At the carrel with requests pending
```

Figure 35-3. Example of status patron groups messages customized

TIP:
You may imbed substitution tokens in your messages as in \\i shelved at Carrel \\l until \\d. See Table 35-1 for a list of tokens.

Table 35-1. Substitution Tokens

Token	Description
\\i	Standard item information.
\\c	Outstanding number of requests.
\\d	Date.

Table 35-1. Substitution Tokens

Token	Description
\\e	Date unless date is today.
\\t	Time.
\\u	Time if date is today.
\\l	Location.
\\F	First name field from the patron record. **NOTE:** Only for patron status groups.
\\L	Last name field from the patron record. **NOTE:** Only for patron status groups.
\\b	Number representing the limit for a certain item request/fine.
\\p	Number representing the amount of fines/requests that you have.
\\n	Item loan period.
\\rsloc	Routing Source Location. Routing source is the place where the item was discharged.
\\rslib	Routing Source Library. Routing source is the place where the item was discharged.
\\rtloc	Routing Target Location. Routing target is the destination.
\\rtlib	Routing Target Location. Routing target is the destination.

4. Save your changes to the `webvoyage.properties` file.

How Do I Add/Modify Search Results Page Icons?

36

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

WebVoyáge 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 36-1. 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:

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

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

2. Locate the `<bibFormats>` section of the `pageProperties.xml` file. See Figure 36-1 for an example.

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

Figure 36-1. pageProperties <bibFormats> example

```
<bibFormat type="bd" icon="icon_manuscript.gif">Archival/Manuscript
    Material</bibFormat>
<bibFormat type="bm" icon="icon_manuscript.gif">Archival/Manuscript
    Material</bibFormat>
<bibFormat type="bs" icon="icon_serial.gif">Periodical</bibFormat>
<bibFormat type="ca" icon="icon_music.gif">Music</bibFormat>
<bibFormat type="cb" icon="icon_serial.gif">Periodical</bibFormat>
```

Figure 36-1. pageProperties <bibFormats> example (Continued)

3. Edit the <bibFormats> section of the pageProperties.xml file to specify the following (see Table 36-1):
   - The item type.
   - The icon image file to be used.
   - The label to display.

Table 36-1. <bibFormats> options

Option	Description
type=	Use to specify the item type.
icon=	Use to specify the name of the image. **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?

37

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 37-1 on page 37-2 and Figure 37-2 on page 37-3.

Files

WebVoyáge uses the following file to store renewal messages for display when patrons request renewals:

- `webvoyage.properties`.

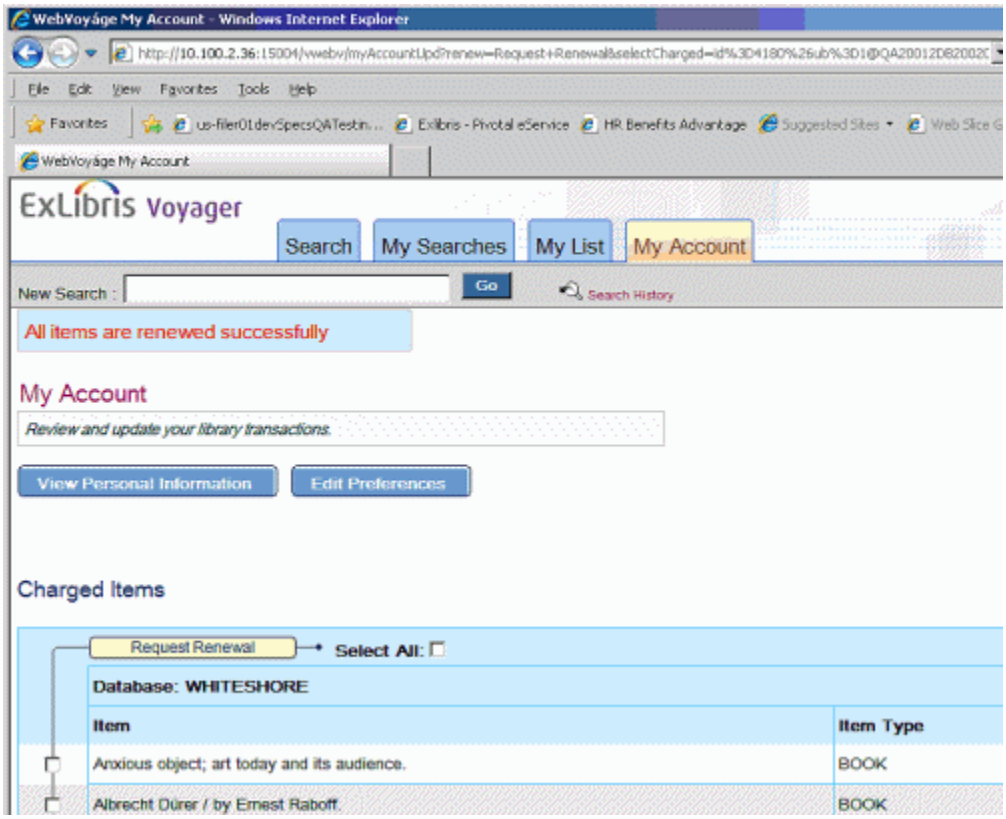

Figure 37-1. Renewal Status Message (Successful)

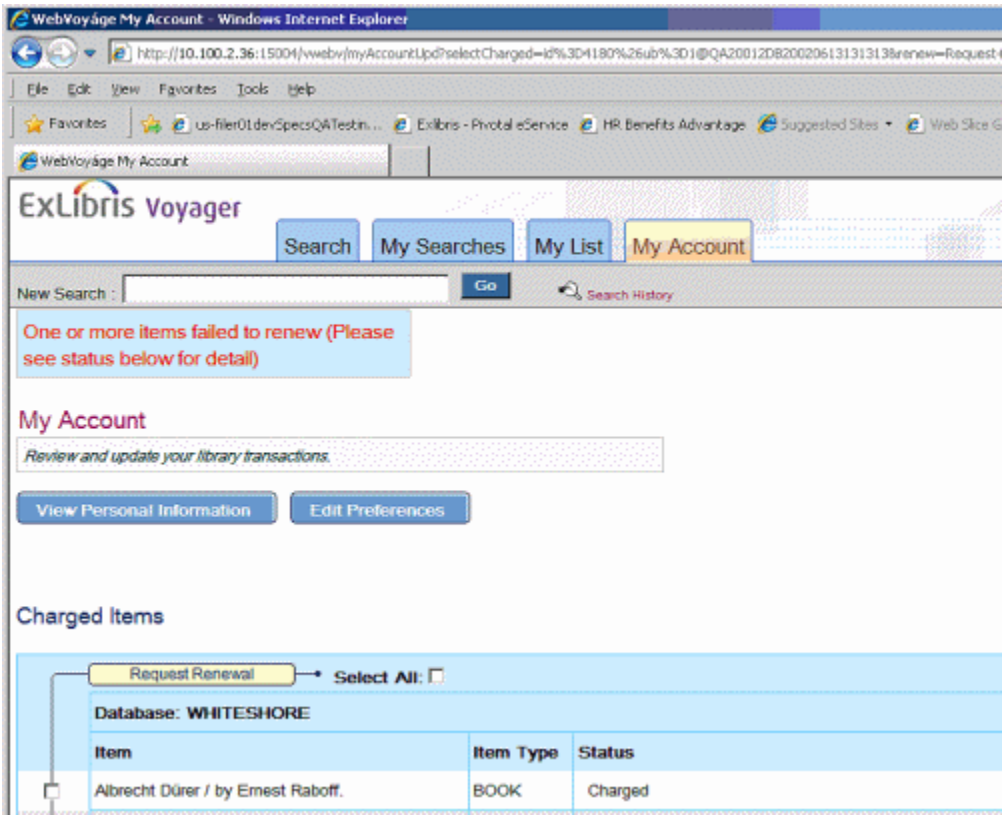

Figure 37-2. Renewal Status Message (Failed)

Instructions

This section provides instructions for how to modify renewal status messages.

Procedure 37-1. 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/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`

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

2. Locate the MyAccount page charged item renewal message section of the `webvoyage.properties` file. See Figure 37-3 for an example.

```
#========================================================================#
# 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 37-3. Example of renewal status messages

3. Edit the renewal status messages to your preference.

4. Save your changes to the `webvoyage.properties` file.

How Do I Add CGI Script Support?

38

Description For "How Do I Add CGI Script Support?"

This chapter describes how to add CGI script support for Voyager 7.x versions and later of WebVoyáge.

Files

This chapter uses the following files:

- `<xxxdb>_vwebv_httpd.conf`
- `apachectl`
- `helloworld.cgi`

Instructions

This section provides the instructions for adding CGI script support by updating the existing WebVoyáge virtual host.

Procedure 38-1. Adding CGI Script Support

Use the following procedure to add CGI script support.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name that can be found in the `/m1/voyager/directory`.

1. Login to the server as the root user.

   ```
   su -
   ```

2. Change the directory to the Apache `ActivatedVirtualHosts` directory.

   ```
   cd /m1/shared/apache2/conf/ActivatedVirtualHosts
   ```

3. Backup up the original file.

   ```
   cp <xxxdb>_vwebv_httpd.conf ../ConfiguredVirtualHosts/
   <xxxdb>_vwebv_httpd.conf.$$
   ```

4. Edit the file.

   ```
   vi <xxxdb>_vwebv_httpd.conf
   ```

5. Add the text shown in Figure 38-1 to the existing virtual host:

   ```
   JkUnMount /vwebv/*.cgi ajp13_lb_<db>_vwebv

   ScriptAlias /vwebv/cgi-bin/ /m1/voyager/<db>/tomcat/vwebv/
   context/vwebv/htdocs/cgi-bin
   ```

```
Listen xxx
<VirtualHost *:xxxx>
...
     JkUnMount /vwebv/*.cgi ajp13_lb_<db>_vwebv
     ScriptAlias /vwebv/cgi-bin/ /m1/voyager/<db>/tomcat/vwebv/context/vwebv/
          htdocs/cgi-bin/
</VirtualHost>
```

Figure 38-1. Text to Add to Existing Virtual Host

6. Check to ensure that the configuration is okay.

   ```
   cd /m1/shared/apache2/bin

   ./apachectl -t
   ```

7. Restart Apache.

   ```
   ./apachectl restart
   ```

8. Log in to the server as the voyager user.

   ```
   su - voyager
   ```

9. Make the cgi-bin directory.

   ```
   cd /m1/voyager/<db>/tomcat/vwebv/context/vwebv/htdocs

   mkdir cgi-bin
   ```

10. Create a sample `.cgi` script to test.

    ```
    cd /m1/voyager/<db>/tomcat/vwebv/context/vwebv/htdocs/cgi-
    bin

    vi helloworld.cgi
    ```

11. Add the code shown in Figure 38-2 to the `helloworld.cgi` file.

```
#!/m1/shared/bin/perl

print "Content-type: text/html\n\n";
print "Hello, world!\n";
```

Figure 38-2. Code to Add to helloworld.cgi

12. Type `:wq!` to quit.
13. Make the script executable.

    `chmod 755 helloworld.cgi`

14. Test the CGI functionality by entering the following URL in your Web browser:

    `http://<ip|name>:xxxx/vwebv/cgi-bin/helloworld.cgi`

    This should open a page that displays Hello World.

    Any errors means that setup was not successfully

OPTIONAL:

15. *Back out your changes, if necessary, by deploying your backup file copies.*

How Do I Suppress Patron Barcode/ ID Prompts on Request Forms?

39

Description For “How Do I Suppress Patron Barcode/ID Prompts on Request Forms?”

This chapter describes how to suppress patron barcode/ID prompts on request forms using the `page.patronRequests.idPrompt` feature in the `webvoyage.properties` file. With this feature, the patron barcode/ID request prompt (see Figure 39-1) can be optional on the patron request page.

When the patron barcode/ID prompt is suppressed, Voyager checks all of the patron’s barcodes to determine if the patron is allowed to place the request for the item.

NOTE:
When the following methods are used for authentication, the `page.patronRequests.idPrompt` feature is ignored and the barcode/ID prompt is not shown like when `N` (see Procedure 39-1, Suppressing Patron Barcode/ID Prompts on Request Forms, on page 39-2) is selected for `page.patronRequests.idPrompt`:

- Patron Directory Services (PDS)
- WebVoyáge patron authentication adapater
- External site authentication through vxws services

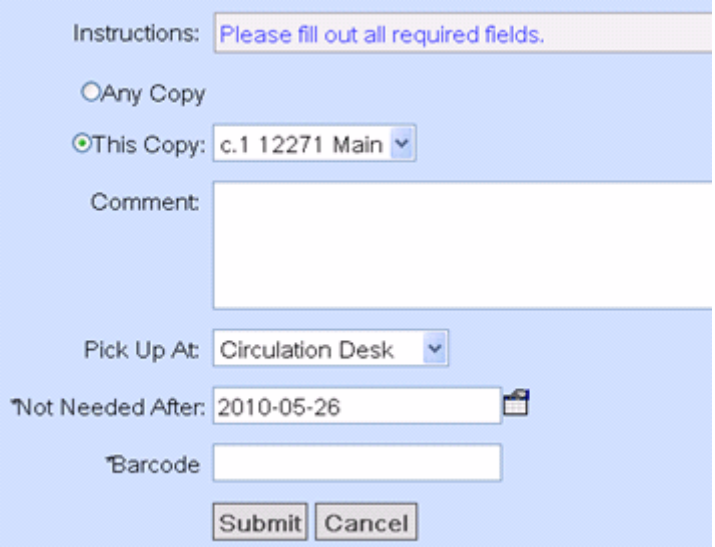

Figure 39-1. Barcode Prompt Example

Files

This chapter uses the following file:

- `webvoyage.properties`.

Instructions

This section provides the instructions for suppressing patron barcode/ID prompts on request forms.

Procedure 39-1. Suppressing Patron Barcode/ID Prompts on Request Forms

Use the following procedure to suppress patron barcode/ID prompts on request forms.

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 the `webvoyage.properties` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/`.

2. Locate the page.patronRequests.idPrompt section of the `webvoyage.properties` file. See Figure 39-2.

```
#############
# Patron Requests  barcode/IID input:
# There are two options for the value page.patronRequests.idPrompt.
# Y = Show the barcode or Institution ID prompt on the request page, and make it mandatory
# N = Do not show the barcode or Institution ID prompt on the request page at all.
#############
page.patronRequests.idPrompt=Y
```

Figure 39-2. page.patronRequests.idPrompt Section in webvoyage.properties

3. Enter `N` to suppress the barcode/ID prompt on the patron request page.

   The default setting is `Y`.

4. Save the `webvoyage.properties` file.

How Do I Customize Time Format for Media Equipment and Media Item Booking Forms?

40

Media Equipment and Media Item Booking Forms Time Format Overview

The media equipment and media item booking forms provide the following options:

- Select Start Date and Time
- Select End Date and Time

See Figure 40-1 and Figure 40-2.

These options can be customized to display either the 12-hour or 24-hour format on the booking forms.

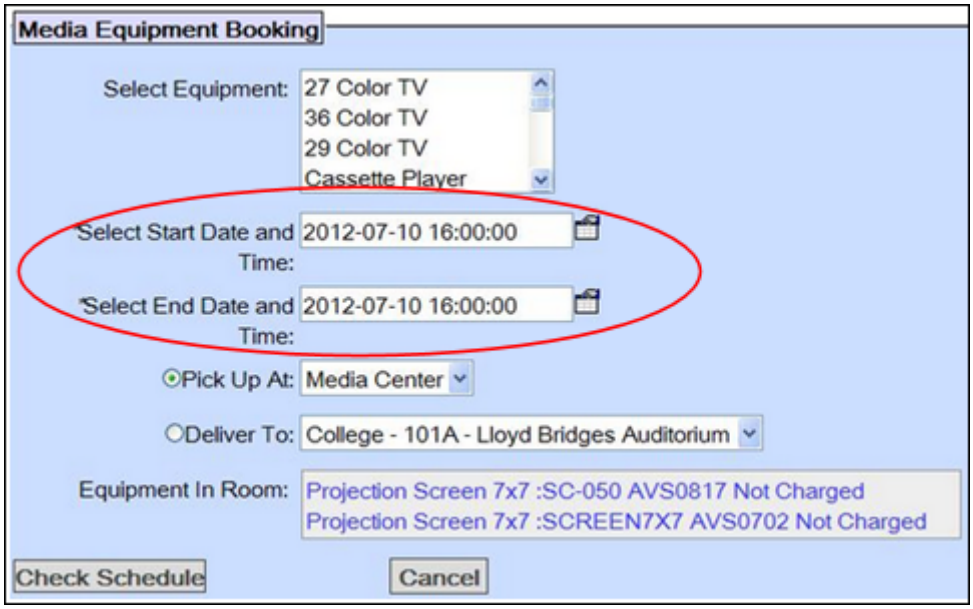

Figure 40-1. Media Equipment Booking Form

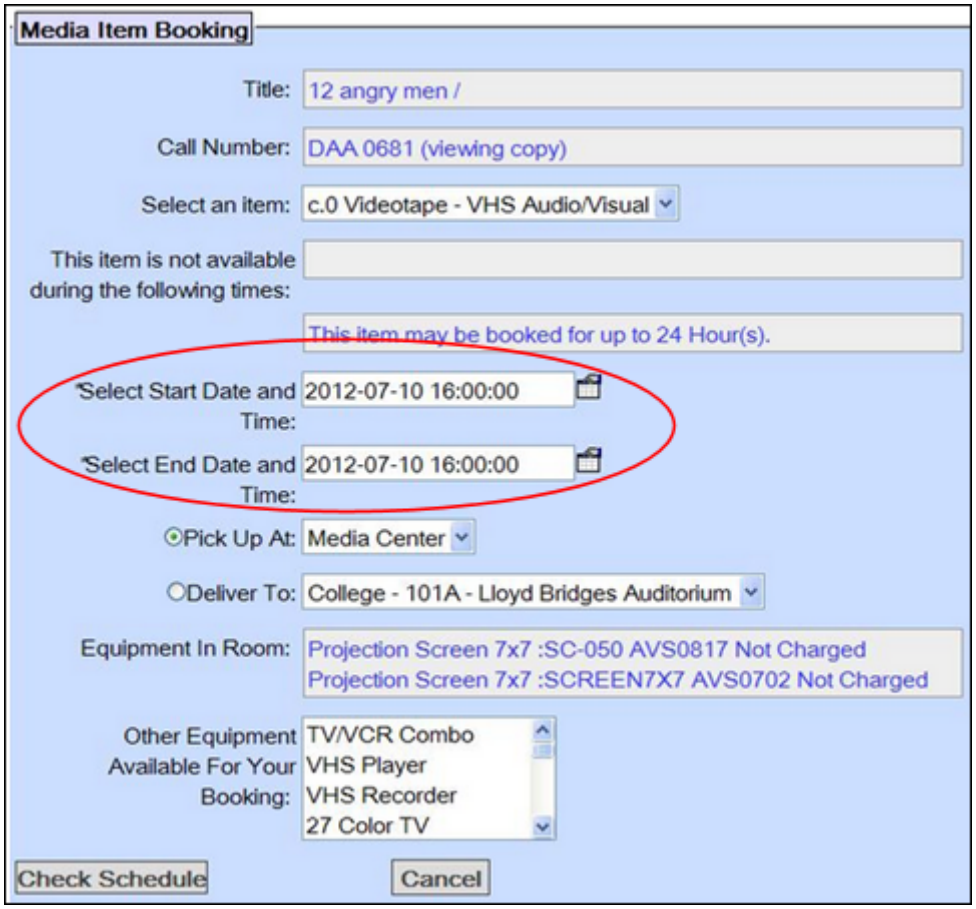

Figure 40-2. Media Item Booking Form

Files

WebVoyáge uses the `constants.xsl` file to implement customizing the time format for the media equipment and media item booking forms.

Media Equipment and Media Item Booking Forms Time Format Customization

To customize the media equipment and media item booking forms time format, you need to configure the `constants.xsl` file that resides on the WebVoyáge server. The `constants.xsl` file is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/xsl/common/` where `xxxdb` is your database name.

In the `constants.xsl` file, you need to set the `timeFormat` variable to 12 or 24. See Figure 40-3.

```
<!-- ## This is where we will configure the time format used on the Media Scheduling
     ## request forms as well as the Media Request Confirmation Page
     ## Set the value to 12 or 24
     ## 12   :  12 Hour format, uses AM / PM
     ## 24   :  24 Hour format
     ## any other value will default to 24 hour format
-->
<xsl:variable name="timeFormat" select="'24'"/>
```

Figure 40-3. Customize Time Format

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X