Voyager® 7.0
WebVoyáge Architecture Overview and Configuration Models

December 2008

ExLibris The bridge to knowledge

CONFIDENTIAL INFORMATION

The information herein is the property of Ex Libris Ltd. or its affiliates and any misuse or abuse will result in economic loss. DO NOT COPY UNLESS YOU HAVE BEEN GIVEN SPECIFIC WRITTEN AUTHORIZATION FROM EX LIBRIS LTD.

This document is provided for limited and restricted purposes in accordance with a binding contract with Ex Libris Ltd. or an affiliate. The information herein includes trade secrets and is confidential.

DISCLAIMER

The information in this document will be subject to periodic change and updating. Please confirm that you have the most current documentation. There are no warranties of any kind, express or implied, provided in this documentation, other than those expressly agreed upon in the applicable Ex Libris contract. This information is provided AS IS. Unless otherwise agreed, Ex Libris shall not be liable for any damages for use of this document, including, without limitation, consequential, punitive, indirect or direct damages.

Any references in this document to third-party material (including third-party Web sites) are provided for convenience only and do not in any manner serve as an endorsement of that third-party material or those Web sites. The third-party materials are not part of the materials for this Ex Libris product and Ex Libris has no liability for such materials.

TRADEMARKS

"Ex Libris," the Ex Libris bridge , Primo, Aleph, Alephino, Voyager, SFX, MetaLib, Verde, DigiTool, Preservation, URM, Voyager, ENCompass, Endeavor eZConnect, WebVoyage, Citation Server, LinkFinder and LinkFinder Plus, and other marks are trademarks or registered trademarks of Ex Libris Ltd. or its affiliates.

The absence of a name or logo in this list does not constitute a waiver of any and all intellectual property rights that Ex Libris Ltd. or its affiliates have established in any of its products, features, or service names or logos.

Trademarks of various third-party products, which may include the following, are referenced in this documentation. Ex Libris does not claim any rights in these trademarks. Use of these marks does not imply endorsement by Ex Libris of these third-party products, or endorsement by these third parties of Ex Libris products.

Oracle is a registered trademark of Oracle Corporation.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

Microsoft, the Microsoft logo, MS, MS-DOS, Microsoft PowerPoint, Visual Basic, Visual C++, Win32, Microsoft Windows, the Windows logo, Microsoft Notepad, Microsoft Windows Explorer, Microsoft Internet Explorer, and Windows NT are registered trademarks and ActiveX is a trademark of the Microsoft Corporation in the United States and/or other countries.

Unicode and the Unicode logo are registered trademarks of Unicode, Inc.

Google is a registered trademark of Google, Inc.

Document released: December 2008

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

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Figures

Figures

Figures

Figures

Figures

21 How Do I Remove the Course Reserve Tab?

22 How Do I Add A New Search Tab?

23 How Do I Add A New Header Tab?

24 How Do I Create Additional Record Views?

Figures

Figures

Figures

Procedures

Procedures

Procedures

Procedures

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. Long lines of code that wrap within the left/right edges of the illustration may contain line breaks resulting from the PDF build that need to be removed for successful processing.

Intended Audience

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

Reason for Reissue

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

- Corrections to Table 3-3 on page 3-4.
- New chapter for implementing DOI/URN handling. See How Do I Implement DOI and URN Handling? on page 25-1.
- New chapter for implementing hook to holdings. See How Do I Implement Hook to Holdings (Citation Server)? on page 26-1.
- New chapter for implementing HTTP post to link resolver. See How Do I Implement HTTP Post to Link Resolver? on page 27-1.
- New chapter for implementing media booking. See How Do I Display Media Bookings in MyAccount? on page 28-1.
- New chapter for implementing ImageServer. See How Do I Implement ImageServer in WebVoyáge? on page 29-1.
- Corrections to chapter table of contents page numbering for Chapters 20 and 21.
- Correction to Chapter 14 references of `displayRecord.css` file to `displayCommon.css` file.
- Updated Procedure 8-1, Create External Search From Bibliographic Record Display, on page 8-2 to include Step 4 regarding the trimData template.

Document Summary

Chapter 1	"Getting Started" Chapter 1 describes the prerequisites for working with and customizing Voyager WebVoyáge 7.0.
Chapter 2	"Architecture" Chapter 2 provides an overview of the WebVoyáge architecture.
Chapter 3	"Display Codes" Chapter 3 describes display codes used in Voyager WebVoyáge 7.0.

Chapter 4	"How Do I Build A Separate Display For Serials?" Chapter 4 describes how to build a separate display for serials.
Chapter 5	"How Do I Add Static Links To The Header Or Footer?" Chapter 5 describes how to add static links to the header/footer area.
Chapter 6	"How Do I Remove Information From A Page?" Chapter 6 describes how to remove information from a page.
Chapter 7	"How Do I Add A Map Or Other Information To A Location?" Chapter 7 describes how to add a map or other information to a location.
Chapter 8	"How Do I Create An External Search From A Bibliographic Record Display?" Chapter 8 describes how to create an external search.
Chapter 9	"How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?" Chapter 9 describes how to dynamically disable limits and change search tips based on the selected search index.
Chapter 10	"How Do I Disable AutoComplete?" Chapter 10 describes how to disable AutoComplete.
Chapter 11	"How Do I Display A Favicon?" Chapter 11 describes how to create an icon for a browser tab or title bar display.
Chapter 12	"How Do I Hide Limits On The Advanced Search Page?" Chapter 12 describes how to hide the limit options on an advanced search while, optionally, a user may click to see them.
Chapter 13	"How Do I Build And Display A Persistent Link To A Bibliographic Record?" Chapter 13 describes how to dynamically build and display a persistent link to a bibliographic record.
Chapter 14	"How Do I Change The Format Of The Record Display Page?" Chapter 14 describes how to add class attributes to the Detailed Display Page for improved formatting control.
Chapter 15	"How Do I Add Tracking Codes?" Chapter 15 describes how to add tracking codes.
Chapter 16	"How Do I Implement Google Book Search?" Chapter 16 provides instructions regarding the implementation of Google Book Search.
Chapter 17	"How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?" Chapter 17 provides instructions for implementing Syndetics enhancements.

Chapter 18	"How Do I Implement Geospatial Search?" Chapter 18 provides instructions for implementing geospatial search.
Chapter 19	"How Do I Enable External Authentication?" Chapter 19 provides instructions for enabling external authentication.
Chapter 20	"How Do I Modify Page Messages?" Chapter 20 provides instructions for modifying page messages.
Chapter 21	"How Do I Remove the Course Reserve Tab?" Chapter 21 provides instructions for removing the Course Reserves tab.
Chapter 22	"How Do I Add A New Search Tab?" Chapter 22 provides instructions for adding a new search tab.
Chapter 23	"How Do I Add A New Header Tab?" Chapter 23 provides instructions for creating a new header tab.
Chapter 24	"How Do I Create Additional Record Views?" Chapter 24 provides instructions for creating additional record views such as brief and full.
Chapter 25	"How Do I Implement DOI and URN Handling?" Chapter 25 provides instructions for DOI/URN handling.
Chapter 26	"How Do I Implement Hook to Holdings (Citation Server)?" Chapter 26 provides instructions for implementing hook to holdings.
Chapter 27	"How Do I Implement HTTP Post to Link Resolver?" Chapter 27 provides instructions for implementing HTTP POST to link resolver.
Chapter 28	"How Do I Display Media Bookings in MyAccount?" Chapter 28 provides instructions for implementing media bookings.
Chapter 29	"How Do I Implement ImageServer in WebVoyáge?" Chapter 29 provides instructions for implementing ImageServer.
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

Please contact Customer Support to provide us with your feedback. For Customer Support contact information, see SupportWeb at:

`http://www.exlibrisgroup.com/support_center.htm`.

To Submit Comments by E-mail

To submit comments by e-mail, please send your message to:

docmanager@exlibrisgroup.com

Getting Started

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 7.0 and corresponding Voyager 7.0 integrated library system software installed.
- Have access to an internet browser on your PC.
- Obtain the URL and/or the IP and port address for accessing your instance of Voyager WebVoyáge 7.0.
- Obtain your user ID and password for logging in to Voyager WebVoyáge 7.0. For login 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

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.

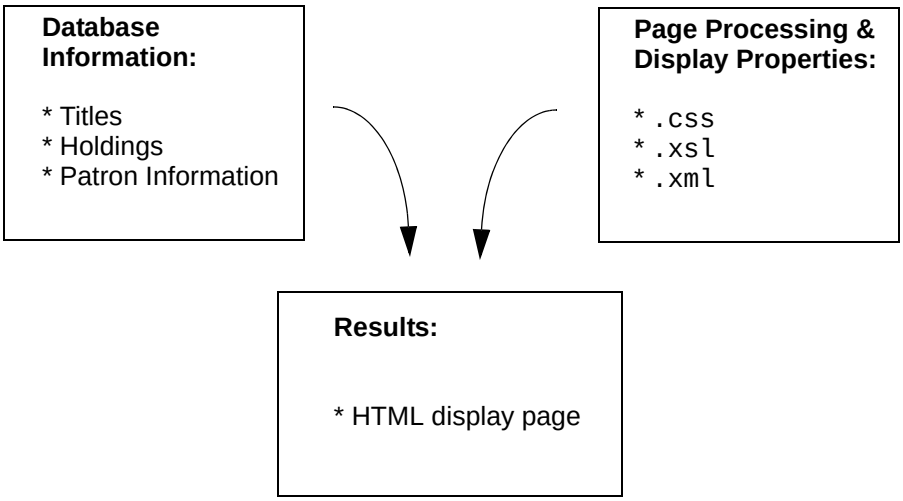

Figure 2-1. Display page build overview

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.

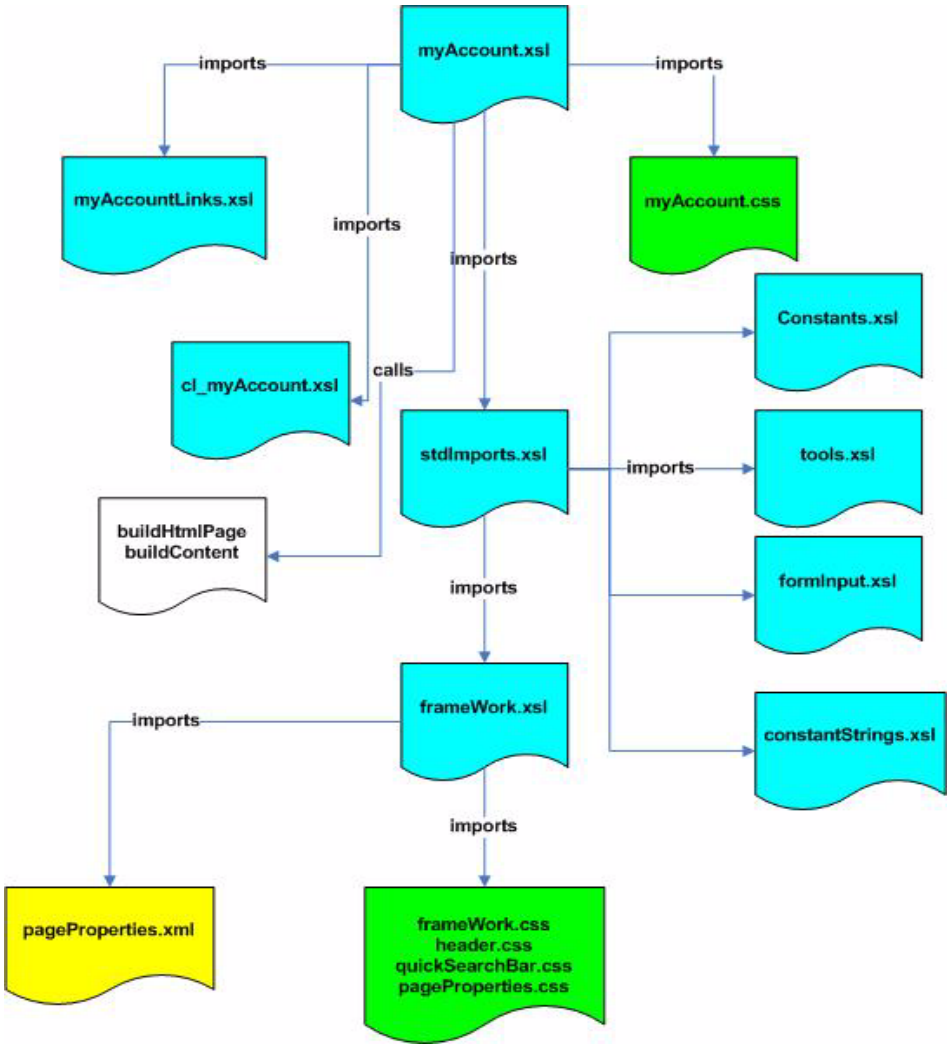

Figure 2-2. Flowchart example for myAccount page

.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.

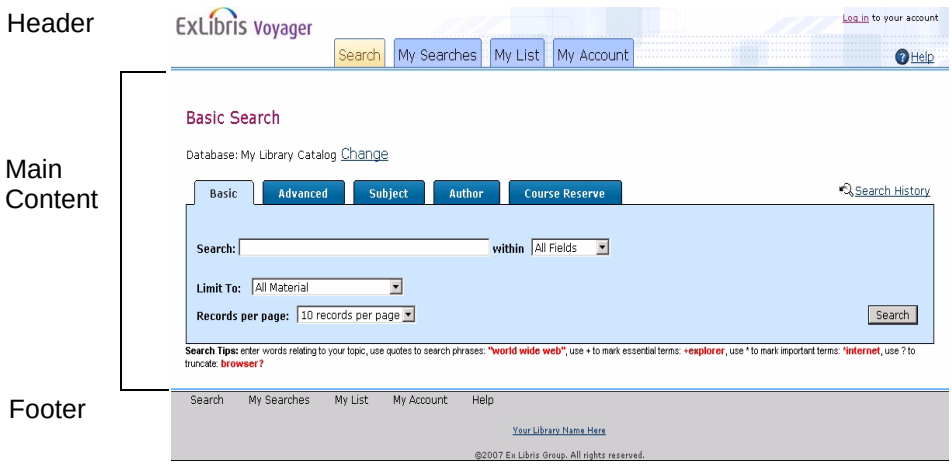

Figure 2-3. Page components

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 independently 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.

Display Codes

3

Display Codes

3

displaycfg.xml

The codes described in Table 3-1 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 3-1. 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 3-2 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 3-2. displaycfg.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.
1022	Supplemental issues from serials.
1024	Indexes from serials.
1030	Order status as shown in the line items of purchase orders.
1040, 1042, 1044	Compressed serials information.
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.

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 3-1 to view this information.

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

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

Figure 3-1. Example display.xsl code

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

Quick Limits

You can define quick limits in the `webvoyage.properties` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`. See Table 3-3 for a list of limit types.

Table 3-3. 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. 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.

NOTE:
The `limits.xml` file is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/userTextConfigs/`.

How Do I Build A Separate Display For Serials?

4

How Do I Build A Separate Display For Serials?

4

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 4-1. 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 4-1 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>
                <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" />
```

Figure 4-1. Sample line of code for sdisplaycfg.xml

```
                <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 4-1. 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 4-2.

```
<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 4-2. New namespace declarations example

   c. Locate `<!-- ## Our Document Holders ## -->`. See Figure 4-3.

   d. Change the variable name `Config` to `MConfig`. See Figure 4-3.

   e. Add a document holders declaration for the serial configuration display for `SConfig` and `recType` as shown in Figure 4-3.

```
<!-- ## 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 4-3. Our Document Holders example

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

```
<!-- ## 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 4-4. 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 4-5. Replacement lines of code

g. Locate `<!-- ## Holdings Data ## -->`. See Figure 4-6.

h. Change the variable `$Config` to `$MConfig` as shown in Figure 4-6.

```
<!-- ## Holdings Data ## -->
<xsl:for-each select="$MConfig">
        <div class="holdingsData">
                        <xsl:call-template name="buildMarcDisplay">
                             <xsl:with-param name="recordType" select="'mfhd'"/>
                        </xsl:call-template>
        </div>
</xsl:for-each>
```

Figure 4-6. Holdings Data example

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?

5

How Do I Add Static Links To The Header Or Footer?

5

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 5-1. 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 5-1) 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 5-1. Header links template

b. Call the template that you created in Step a by adding the instruction (see Figure 5-2, 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 5-2. 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 5-2. 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 5-3, 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 5-3. 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?

6

How Do I Remove Information From A Page?

6

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 6-1. 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 6-1.

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

Figure 6-1. Section comment to locate

   b. Comment out the table heading and table cell lines of code relevant to item type. See Figure 6-2 and Figure 6-3.

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

Figure 6-2. Table heading example

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

Figure 6-3. 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?

7

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

7

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 7-1. 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 7-1 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 7-1. Example code for building hyperlink

```
</xsl:template>

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

</xsl:stylesheet>
```

Figure 7-1. 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 7-2.

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

Figure 7-2. Example code for adding path statement

    b. Call the `locMapLink` template from within the `BMD100` template. See Figure 7-3.

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


<xsl:template name="BMD1000">
<xsl:param name="mfhdID"/>


    <xsl:for-each select="$HoldXML/hol:holdingsRecord/hol:mfhdCollection/
           mfhd:mfhdRecord[@mfhdId = $mfhdID]/mfhd:mfhdData[@name='locationDisplayName']">
        <xsl:if test="string-length(.)">
            <xsl:value-of select="."/>
                  <!-- ## add a map link ## -->
                  <xsl:call-template name="locMapLink" >
                  <xsl:with-param name="mfhd" select="$mfhdID"/>
                  </xsl:call-template>
                  <br/>
        </xsl:if>
    </xsl:for-each>
</xsl:template>


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

Figure 7-3. Example of call for locMapLink

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?

8

How Do I Create An External Search From A Bibliographic Record Display?

8

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 8-1. 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 8-1 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 :
-->
<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>
```

Figure 8-1. Sample code to extract ISBN from bibliographic record

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 8-2.

```
<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 8-2. 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 8-3.

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

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


</xsl:template>

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

</xsl:stylesheet>
```

Figure 8-3. Call for isbnSearch.xsl in displayFacets.xsl

4. (Optional) Set up trimData XSL code. See Figure 17-3 on page 17-4.

   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?

9

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

9

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 9-1. 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 9-1 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
** Last Modified By : ASP
*/

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


              // save the defautl 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+':
```

Figure 9-1. Example code for searchBasic.js

```
                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 9-1. 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 9-1. Example code for searchBasic.js (Continued)

```
    }
  }
}

addLoadEvent(addSearchCodeChanged);
```

Figure 9-1. 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 9-2.

```
    <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 9-2. 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 9-3), 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>

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

</xsl:template>

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

Figure 9-3. Locate buildBasicSearch template

7. Save and test your changes.

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?

10

How Do I Disable AutoComplete?

10

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 10-1. 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 10-1.

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

Figure 10-1. 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 10-2.

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

Figure 10-2. 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?

11

How Do I Display A Favicon?

11

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

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

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

NOTE:
There are multiple methods for installing a favicon. The method described in this chapter is specific to 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 11-1. 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 11-1 after the `<head>` element.

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

Figure 11-1. 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/`.

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?

12

How Do I Hide Limits On The Advanced Search Page?

12

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 12-1. 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 12-1 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 12-1. 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 12-1. 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 12-2.

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

Figure 12-2. 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 12-3.

```
<!-- 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 12-3. 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 12-4.

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


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


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

</xsl:stylesheet>
```

Figure 12-4. 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?

13

How Do I Build And Display A Persistent Link To A Bibliographic Record?

13

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`.

Instructions

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

Procedure 13-1. 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 13-1 for this example.

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

<!--
**                Note: create persistent link based on bib ID
**                Version : 1.0
**                Created : 16-Nov-2007
**                Created By :
**                Last Modified : 14-Apr-2008
**                Last Modified 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="persistentLink">
<xsl:param name="bibID"/>

    <!-- :doc: you must set this to match your public URL -->
       <xsl:variable name="baseURL">http://xxx.xxx.xxx.xxx:7008/vwebv/holdingsInfo?bibId=</
           xsl:variable>


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

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

</xsl:stylesheet>
```

Figure 13-1. Sample code for local_PersistentLink.xsl

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

3. Edit `cl_displayRecord.xsl`.

   a. Locate the namespace declarations at the top of the file and include a reference to the `local_PersistentLink.xsl` file after the `<xsl:stylesheet>` element. See Figure 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="../pageFacets/local_PersistentLink.xsl"/>
```

Figure 13-2. Include file reference example

   b. Call the persistent link template from the `Bibliographic Data` section. See Figure 13-3.

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


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

Figure 13-3. Persistent link template call in Bibliographic Data section

4. Save and test your changes.

OPTIONAL:

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

How Do I Change The Format Of The Record Display Page?

14

How Do I Change The Format Of The Record Display Page?

14

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 14-1, Add Class Attributes For Formatting, on page 14-2 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 14-1. 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 14-1 for examples.

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

Figure 14-1. 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 14-2 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 14-2. Example code to add to buildMarcDisplay

```
            </xsl:for-each>


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

Figure 14-2. 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 14-3.

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

Figure 14-3. 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?

15

How Do I Add Tracking Codes?

15

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 15-1. 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 15-1.

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

<xsl:template name="buildFooter">

   <xsl:for-each select="/page:page/page:pageFooter">
      <div id="pageFooter">
         <xsl:for-each select="page:tabs[@nameId='page.footer.buttons']">

            <div id="footerTabs" title="{$footerText/footerTabs}">
               <a name="navFooter"></a>
               <h2 class="navFooter"><xsl:value-of select="$footerText/footerTabs"/></h2>
               <ul class="navbar">
                   <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>
```

Figure 15-1. Example of script snippet add to buildFooter template

```
         <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 15-1. 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?

16

How Do I Implement Google Book Search?

16

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

The Voyager 7.0 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 16-1, Disable Google Book Search Feature, on page 16-3.

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 7.0.

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 16-1 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 16-1. 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 16-2.

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 16-1. 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 16-2.

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

Figure 16-2. Line of code to locate

b. Comment out the lines of code as shown in Figure 16-3.

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

Figure 16-3. 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?

17

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

17

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 7.0.

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 17-1.

```
<!-- 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 17-1. Example of pageProperties.xml

The `/SC.gif` in Figure 17-1 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 17-2.

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 17-2. 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 17-3.

```
<!-- ## 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 17-3. 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 17-4.

```
   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 17-4. 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 17-5.

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 17-5. 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 17-3.

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?

18

How Do I Implement Geospatial Search?

18

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 18-1 on page 18-3 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 7.0.

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 18-1, Geospatial Search Implementation for the steps to provide access to Geospatial Search.

Procedure 18-1. 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 18-1 for a display example.

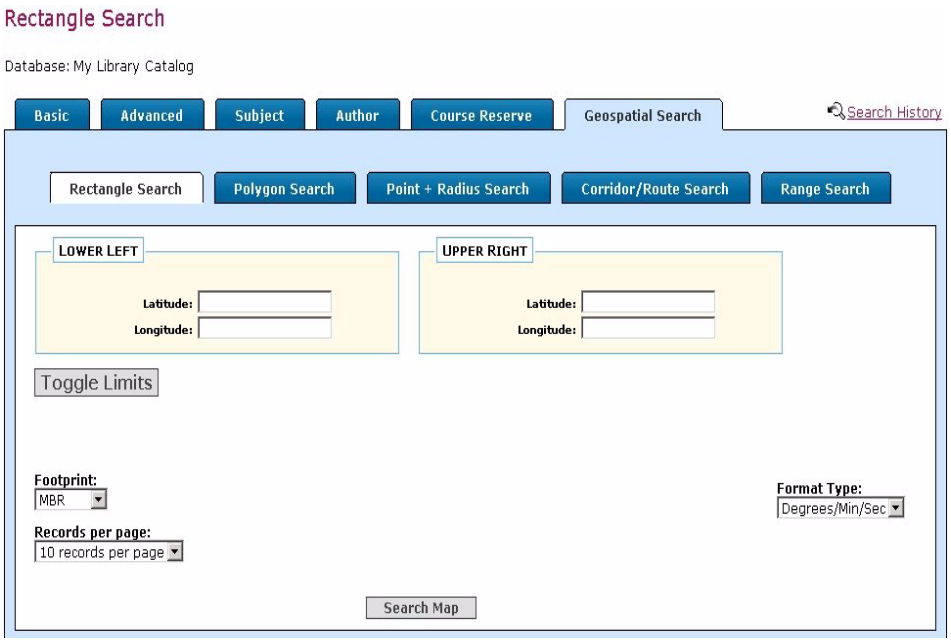

Figure 18-1. Geospatial Search tab

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 18-2.

```
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 18-2. Enable Geospatial Search in the pageProperties.xml file

6. Remove the comment lines to enable Geospatial Search.

   In Figure 18-2, the comment lines are lines 8 and 11.

7. Save your changes.

How Do I Enable External Authentication?

19

How Do I Enable External Authentication?

19

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 7.0.

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 19-1, External Authentication Implementation for the steps to enable external authentication.

Procedure 19-1. 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 19-1.

```
#==============================
# 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 19-1. 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?

20

How Do I Modify Page Messages?

20

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 20-3 on page 20-4.

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 20-1 for an example of an `errorCode` identified for a customized message.

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

Figure 20-1. errorCode example

See Figure 20-2 for a `blockCode` example used when an SSN login error occurs.

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

Figure 20-2. 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 7.0.

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 20-1, Modify Page Messages for the steps to modify page messages.

Procedure 20-1. 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 20-3.

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

Figure 20-3. Modifying page messages example

```
        <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 20-3. 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?

21

How Do I Remove the Course Reserve Tab?

21

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 21-1. 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 21-1.

```
<!-- ## 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 21-1. Search Tab Display Order section

   b. Comment out the `courseReserve` line in this section. See Figure 21-2.

```
<!-- ## 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 21-2. 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 21-3.

```
<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 21-3. More choices paragraph example

b. Comment out the `courseReserve` line in this section. See Figure 21-4.

```
<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 21-4. 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?

22

How Do I Add A New Search Tab?

22

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 22-1. 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 22-1.

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

Figure 22-1. 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 22-2 to the Search section of `internal.properties`.

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

Figure 22-2. 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 22-3 to `pageProperties.xml`.

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

Figure 22-3. Example XML for new search tab

Place the code where you would like the new search tab to display. See Figure 22-4 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 22-4. 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?

23

How Do I Add A New Header Tab?

23

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 23-1. 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 23-1.

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

Figure 23-1. 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 23-2.

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

Figure 23-2. 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 23-3.

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

Figure 23-3. 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?

24

How Do I Create Additional Record Views?

24

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 24-1. 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. Modify existing code (see Figure 24-1) in `displayBriefRecord.xsl` with the code shown in Figure 24-2.

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

Figure 24-1. Existing displayRecord code

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

Figure 24-2. Example modification for displayBriefRecord code

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

4. Modify existing code (see Figure 24-3) in `cl_displayBriefRecord.xsl` with the code shown in Figure 24-4.

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

Figure 24-3. Existing cl_displayRecord code

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

Figure 24-4. Example modification for cl_displayBriefRecord code

5. Modify the existing Action Box code (see Figure 24-5) in `cl_displayBriefRecord.xsl` with the code shown in Figure 24-6.

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


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


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


            </xsl:call-template>
```

Figure 24-5. Existing Action Box code

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


          <xsl:variable name="moreActions">


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


          </xsl:variable>
```

Figure 24-6. Example modification of Action Box code

```
        <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 24-6. Example modification of Action Box code (Continued)

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

7. Modify the existing Action Box code (see Figure 24-7) in `cl_displayRecord.xsl` with the code shown in Figure 24-8.

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

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

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

          </xsl:call-template>
```

Figure 24-7. 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 24-8. Example modification to Action Box code in cl_displayRecord

8. Modify the existing Action Box code (see Figure 24-9) in `cl_displayStaff.xsl` with the code shown in Figure 24-10.

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


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


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


            </xsl:call-template>
```

Figure 24-9. 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 24-10. Example modification to Action Box code in cl_displayStaff

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

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

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

12. Modify existing code (see Figure 24-11) in `displayFacets.xsl` with the code shown in Figure 24-12.

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

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

Figure 24-11. Existing displayFacets code

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

        <xsl:param name="pageRecordType"/>

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

Figure 24-12. Modification example for displayFacets code

13. Add the code shown in Figure 24-13 after the code shown in Figure 24-14 in the `displayFacets.xsl` file.

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

Figure 24-13. 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>
```

Figure 24-14. Existing code in displayFacets.xsl

```
                </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 24-14. Existing code in displayFacets.xsl (Continued)

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

15. Add the code shown in Figure 24-17 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 24-15. filter-mapping code example

16. Add the code shown in Figure 24-16 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 24-16. servlet-mapping code example

17. Add the code shown in Figure 24-17 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>

    <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>
```

Figure 24-17. Example modification code for web.xml

```
        <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>

    </init-param>

    <init-param>

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

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

    </init-param>

    <init-param>

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

Figure 24-17. Example modification code for web.xml (Continued)

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

        </init-param>

    </servlet>
```

Figure 24-17. Example modification code for web.xml (Continued)

18. Save and test your changes.

OPTIONAL:

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

How Do I Implement DOI and URN Handling?

25

How Do I Implement DOI and URN Handling?

25

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 25-1 for the default code/configuration provided at installation.

Table 25-1. 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)?

26

How Do I Implement Hook to Holdings (Citation Server)?

26

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 26-1 for a holdings display example. In Figure 26-1, 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 26-1. Hook to Holdings display example

How Do I Implement HTTP Post to Link Resolver?

27

How Do I Implement HTTP Post to Link Resolver?

27

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 27-1 for an example.

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

Figure 27-1. 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 27-2.

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.

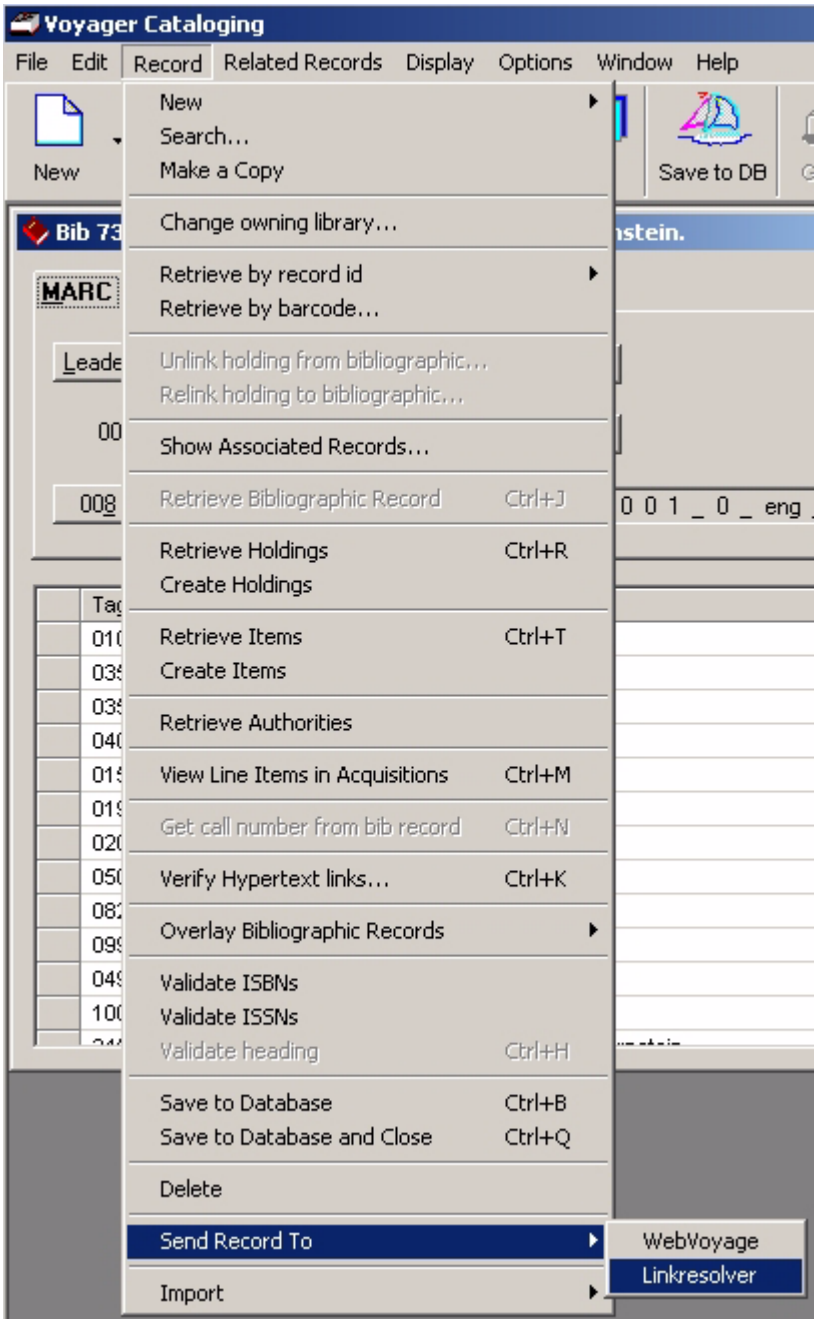

Figure 27-2. Linkresolver option example

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 27-3.
- Identify the fields to be used for the OpenURL construction and how to parse them. See Figure 27-4 for a citation database example.

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

Figure 27-3. Link Resolver URL example

For each field identified (as in Figure 27-4), the following keys need to be specified:

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

In the Figure 27-4 example, the following fields are defined:

- Title.
- Author's last name.
- Author's first name.
- Volume.
- Date/Year.
- Pages.
- ISSN.

```
openUrl.cfg.LOCAL.key1.key=title
openUrl.cfg.LOCAL.key1.tag=245
openUrl.cfg.LOCAL.key1.subfield=a
openUrl.cfg.LOCAL.key1.len1=0
openUrl.cfg.LOCAL.key1.parseStart=
openUrl.cfg.LOCAL.key1.parseEnd=
openUrl.cfg.LOCAL.key1.len2=0

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

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

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

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

Figure 27-4. Example of fields/subfields identified

```
openUrl.cfg.LOCAL.key6.parseStart=
openUrl.cfg.LOCAL.key6.parseEnd=
openUrl.cfg.LOCAL.key6.len2=0

openUrl.cfg.LOCAL.key7.key=pages
openUrl.cfg.LOCAL.key7.tag=773
openUrl.cfg.LOCAL.key7.subfield=g
openUrl.cfg.LOCAL.key7.len1=0
openUrl.cfg.LOCAL.key7.parseStart=p.
openUrl.cfg.LOCAL.key7.parseEnd=
openUrl.cfg.LOCAL.key7.len2=0

openUrl.cfg.LOCAL.key8.key=issn
openUrl.cfg.LOCAL.key8.tag=773
openUrl.cfg.LOCAL.key8.subfield=x
openUrl.cfg.LOCAL.key8.len1=0
openUrl.cfg.LOCAL.key8.parseStart=
openUrl.cfg.LOCAL.key8.parseEnd=
openUrl.cfg.LOCAL.key8.len2=0
```

Figure 27-4. 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 Figure 27-5 for the URL of the documentation that describes the OpenURL standard.

```
http://www.niso.org/kst/reports/standards/
kfile_download?id%3Austring%3Aiso-8859-1=Z39-88-
2004.pdf&pt=RkGKiXzW643YeUaYUqZ1BFwDhIG4-
24RJbcZBWg8uE4vWdpZsJDs4RjLz0t90_d5_ymGsj_IKVa86hjP37r_hJoB5U
Ocx94omusn8lIQY8E%3D
```

Figure 27-5. OpenURL standard details

How Do I Display Media Bookings in MyAccount?

28

How Do I Display Media Bookings in MyAccount?

28

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 28-1.

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

Figure 28-1. Media bookings parameter setting in WebVoyáge

- Set the `page.myAccount.mediaBookings.cancelAllowed` parameter to `Y` (Yes). See Figure 28-2.

  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 28-2. Media bookings cancel allowed parameter example

- Modify label display and/or media bookings display options. See Figure 28-3 for the default settings.

  The `page.myAccount.mediaBookings.itemInfo` parameter may be set to the options in Table 28-1.

```
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 28-3. Media bookings display options

Table 28-1. 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 28-1. page.myAccount.mediaBookings.itemInfo options

Options	Description
\a	the item's author

How Do I Implement ImageServer in WebVoyáge?

29

How Do I Implement ImageServer in WebVoyáge?

29

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 29-1 and Figure 29-2.

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

Figure 29-1. 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 29-2. webvoyage.properties ImageServer configuration example

Procedure 29-1. 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 29-1.

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.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

P

Q

R

S

T

U

W

X