WebVoyáge in Voyager 7
A Primer by Support

26 May 2009

Purpose

This document serves to introduce customers to the new WebVoyáge interface for Voyager 7.0. It includes our testing experience, instructions for basic changes, and links to other documentation & tools. Please take some time to review this document and the updated Support Policy. We hope they will make your implementation of WebVoyáge in Voyager 7 a little easier. The **WebVoyáge User's Guide**, the **WebVoyáge Architecture Overview and Configuration Modules**, and the **Knowledge Base** pick up on the basics that this document provides.

Table of Contents

Overview

Redesign

The WebVoyáge interface was completely redesigned for Voyager 7 using modern web technologies to better meet our customers' needs and resolve enhancement requests. This new interface is dramatically more customizable, but with that power comes increased complexity and changes to all of the old customization procedures, directory structures, and configuration files. However, the core functionality of WebVoyáge has not changed; search types, search indexes, patron authentication, requests, etc, are all still broadly the same. Further, the WebVoyáge Classic interface will still be available, unchanged, and can be configured as the default interface, so that you can keep using it and your customizations until you're ready to roll out the new one.

New Technology

If you are familiar with the underlying technology of the WebVoyáge Classic interface, you may be interested in the new one. WebVoyáge is now run as Java services in Tomcat servlets, instead of a CGI script wrapping an OS native executable. The front-end that you see is provided by the **vwebv** service, which connects to the back-end service, **vxws**. The vxws service connects to the unchanged opacsvr, just like Pwebrecon.cgi of WebVoyáge Classic does.

XSL is now used to generate the HTML on every page and CSS is used to style it. That means that, with the proper expertise, much more advanced customizations are possible by editing the markup that generates the page. You can also take advantage of the flexibility of CSS to style any element of any page. Institutions will have much greater control over the look and feel of their OPAC.

New Directory

Instead of the familiar `xxxdb/webvoyage/` and `xxxdb/etc/webvoyage/` directories, WebVoyáge now runs from the `xxxdb/tomcat/` directory. Almost all configuration that you'll do will be in a specific skin directory: `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/skin/`. There will be a different directory under `ui/` for each skin you choose to maintain.

New Files

WebVoyáge configuration is no longer done via the `opac.ini` file; most textual changes are now done within `webvoyage.properties`, which has a different, but similar, format. The new configuration files are self-documenting with plenty of comments. The most often used files (all under `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/skin/`) are:

- **`webvoyage.properties`** - Many textual and basic configuration settings (including labels and localization).
- **`css/frameWork.css`** - The primary CSS file for site-wide styling.
- **`/xsl/userTextConfigs/pageProperties.xml`** - Other labels, error message, and search tips. Contains the remaining messages not included in `webvoyage.properties`, including those in which HTML markup might appear.
- **`/xsl/contentLayout/configs/displaycfg.xml`** and **`displayHoldings.xml`** - Configuration files that determine what bibliographic and holdings information displays on the detailed record page. These are XML versions of what used to be `displayN.cfg` files; their format is different because of the XML, but the basic logic is the same.

Things to Consider When Upgrading

From Voyager 6 or Below to Voyager 7

There are many things to consider if you're upgrading to Voyager 7 from any earlier major release. The new interface (Tomcat) means that all OPAC customizations are now made through a whole different set of configuration files. The first thing you should do before making any customizations is take the time to familiarize yourself with the OPAC as it appears under the default configuration. Here are a few of the things that have changed compared to the Classic interface:

- The Builder Search has been renamed to the Advanced Search, though it still functions in the same way as in previous versions of Voyager
- Limits for the Advanced Search are set before you run your search, rather than afterwards with a "Post Limit"
- Request forms cannot be accessed from just anywhere in the system; they are only accessible while viewing a titles list or holdings information

Once you become familiar with the new Tomcat interface and you begin making changes, we think you'll be pleased with what it can offer you. To help get you started with customizations, here is a short list of file equivalents between Voyager 6 and Voyager 7 (all file paths are relative to the root of the skin directory):

- Configurations set in `opac.ini` are now spread out in `webvoyage.properties` as well as the files in the CSS directory of your skin
- `limits.ini` is now `limits.xml`, which can be found in `/xsl/userTextConfigs/`
- `displayn.cfg` has been split between two files to configure the display of bib and holding information. For the bibliographic record, look in `displaycfg.xml`, and for the holdings information, look in `displayHoldings.xml`

From Voyager 7 to a Newer Release

If you're simply upgrading to a newer minor release or service pack of Voyager 7, you'll find that not much is different. That's not to say that all you have to do is apply the patch and walk away. There are a couple of important details that you need to keep in mind:

- First, and most important, is that Service Pack and Minor Release patches only apply skin updates to the `exl_default` skin. This is to give you a chance to back up any of the changes you've made before you update your skin and re-apply your configurations
- If you've made changes to the XSL code that renders the OPAC, please take some time to test these changes in the `sandbox` skin. Between any two versions we may make modifications to the default XSL code that can break your customization

Common Changes

This is a list of where to find color and branding settings that we believe most sites will want to change immediately. For the file examples, only the relevant settings and variables are listed. There may, for instance, be many other CSS settings that style the same elements—this document will only show the main ones you're interested in. This list is certainly not exhaustive; with CSS styling and XSL templating, the possible changes are nearly infinite. Making the changes listed in this section will quickly get you through the basic customizations for your institution.

Skins

Create a skin

You can easily create a new skin by making a copy of an existing skin directory, such as `exl_default` . Keep in mind that the `exl_default` should not be edited so that you have a backup default config to test against.

```
cd /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/
cp -pr exl_default newskin
```

Switch skins on the fly

You can explicitly tell WebVoyáge which skin to use via the URL, either in a link, or in the address bar. Add `?sk=`*`newskin`* to the end any URL. If there is already a question mark and parameters, use `&sk=`*`newskin`* as follows:

```
http://hostname:7008/vwebv/searchBasic?sk=newskin
http://hostname:7008/vwebv/search?searchArg=blue&searchCode=GKEY
     %5E*&limitTo=none&recCount=10&searchType=1&sk=newskin
```

Set the default skin

If no skin is explicitly set, WebVoyáge will display `en_US`, but that can be changed.

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/WEB-INF/web.xml

```
<init-param>
   <param-name>DefaultSkin</param-name>
   <param-value>en_US</param-value>
</init-param>
```

Colors and images

Logo image

This variable is in the "Header" section. Make sure you include the path from `/vwebv` onward (as shown).

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/webvoyage.properties

```
page.header.logo.image=/vwebv/ui/skin/images/webVoyageLogo.jpg
```

Background color

The main background color of all pages is set using CSS. It's white by default. Note that the light blue and white image across the top is not transparent: it will continue to have a white background even if you change this setting.

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/css/frameWork.css

```
body {
background:white none repeat scroll 0%;
}
```

Search Box

The color of the main search box is set through CSS. It's light blue by default. This color appears elsewhere; you may wish to do a global search-n-replace for color value **`#C9E5FF`** in the CSS files.

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/css/searchPages.css

```
#searchForm {
background-color:#C9E5FF;
}
```

Search Tabs

You will probably want to match the active tab color to the search box color. Tabs in WebVoyáge are made up of two separate images: a thin one on the left and a wide one on the right. Both of these come in two colors: light blue for active and hover, and dark blue for inactive. These images are called by CSS; you can edit the CSS to call new images, but the easiest way to change the color is to take the existing images and edit them to the desired colors. Make a copy of the original image (such as `search_nav_right_off.gif.ORIGINAL`) and then edit the image that's called by WebVoyáge.

Files: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/images/

- search_nav_right_off.gif
- search_nav_left_off.gif
- search_nav_right_on.gif
- search_nav_left_on.gif

Header Tabs

The Header Tabs are built exactly like the Search Tabs, but with different images. See above.

Files: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/images/

- tabrB2.png
- tablB2.png
- tableftB.png
- tabrightB.png

Searching and Record Display

Basic Search types

These variables are in the "Search Basic" section of `webvoyage.properties`. The comments in the beginning of the section give tips about creating new search types. Each search type should have four lines: the first declares a new search type (choose a unique and arbitrary variable name, then use that for the next three lines), the second gives the search code from SysAdmin, the third gives the display order, and the fourth sets the label that displays in WebVoyáge. For the display order, it's okay to skip numbers (e.g. have 1, 3, 4, and 6 for your display orders), but make sure no two search types have the same number, or some search types won't display. The default `webvoyage.properties` file has a number of common search types commented out with a "#" at the beginning of the line. You can enable those by uncommenting the four lines for that search type.

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/webvoyage.properties

```
page.search.basic.search.code.authorKey=
page.search.basic.search.code.authorKey.code=NAME@
page.search.basic.search.code.authorKey.order=16
page.search.basic.search.code.authorKey.display=Author Keyword
```

Basic Search limits/post-search filters

These variables are in the "Limit to" section of `webvoyage.properties`. The syntax is similar to the search syntax above: the first line declares a new limit variable (with a unique variable name) that you'll use for the next three lines, second gives the actual limit, third gives the order in the drop-down, and fourth sets the label the user will see. Keep in mind that these limits are used for both pre- and post-search filtering. After searching, users can add and remove filters at will. As a result, it's useful to include only one limit per drop-down option and let the user stack them after the search.

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/webvoyage.properties

```
page.search.limitTo.Videos=
page.search.limitTo.Videos.limit=MEDI=v
page.search.limitTo.Videos.order=4
page.search.limitTo.Videos.text=Videos
```

Bibliographic/holdings information to display

These XML files contain the same information that used to be in the displayN.cfg files. They also display according to the same logic; the only difference is the formatting.

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/xsl/contentLayout/configs/displaycfg.xml
File:
/m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/xsl/contentLayout/configs/displayHoldings.xml

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

Most fields you may want to display are already in the file, but commented out with XML comments. Remove the start and end comments (see the example below) to display these fields.

```
<!-- In XML, text between these start and end markers will be ignored. -->
```

Functions

Timeout

The timeout is set in two places:

- `timeout.time` in `webvoyage.properties` sets the amount of time before the browser will redirect a user elsewhere (see `exit.do` below) and how much warning they're given before this redirect. This is like the JavaScript warning that some sites have implemented in the Classic interface.
- `<session-timeout>` in `web.xml` gives the amount of time before the session actually times out on the server side. This is analogous to the webrecon `Timeout` value in `webvoyage/cgi-bin/webvoyage.ini`.

You can set the amount of warning a user is given before being redirected with `timeout.grace`: if it's set to "2", the user will be prompted two minutes before being redirected.

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/webvoyage.properties

```
#==============================
# Timeout configuration.
#==============================
# How many minutes before an idle WebVoyáge session times out?
# Must be less than Tomcat's session timeout, defined in web.xml.
timeout.time=10
# How many minutes before timeout should the user be warned? Should be less than
    the timeout.time value.
timeout.grace=2
# Where do we go when a session times out?
timeout.page=exit.do
```

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/WEB-INF/web.xml

```
<session-config>
    <!-- Be sure to coordinate changes in this value
        with the value of timeout.time in all
        webvoyage.properties files.  The value of
        timeout.time must be less than the value
        of session-timeout. -->
    <session-timeout>11</session-timeout>
</session-config>
```

Logout and exit.do

For both of these variables, use a fully-qualified URL (that is, include a domain name or external IP address). When a user logs out (`option.startPage` variable), he or she might not be done using WebVoyáge, so it's useful to redirect to a WebVoyáge page. The `exit.do` action, by default, occurs when a user clicks the logo or allows the session to time out (`option.exitURL` variable); it should be set to a page outside of WebVoyáge. If it's set to an internal page (such as searchBasic), sessions will time out, redirect to a page that creates a new session, time out, create a new session, etc. A web browser left idle will continue creating sessions even if it's not in use.

File: /m1/voyager/*xxx*db/tomcat/vwebv/context/vwebv/ui/*skin*/webvoyage.properties

```
# Which page should the system return to on logout?
option.startPage=http://library.example.edu:7008/vwebv/searchBasic

# What page or URL should the system go to in response to the exit.do action?
# This can be a WebVoyáge page, such as searchBasic, or an external URL
option.exitURL=http://library.example.edu:7008/index.html
```

The Support Team's Experience

The Skin Contest

Support was starting from scratch with the new UI, just as we knew our customers would be. We had to find a way to get everyone up to speed quickly and learn the ins and outs of customizing the new interface. We came up with the "Skin It To Win It" contest as a way to get everyone in Support involved with and learning the new UI.

Support was divided into two teams (Team 1 and Team A, later, the Tikis and the Zombies), each of which was charged with creating the "most awesome" skin for WebVoyáge. Particularly, they focused on usability, new functionality, creativity, and, of course, learning. They had two months to come up with a new skin and to demonstrate that every single person had learned something about customizing this new UI. The results are in and we are proud to say that both teams exceeded these goals. We learned a lot during this process and the information in this guide is a direct result of our experience.

You can view our contest skins on the Preview Server:

TikiVoyage

http://preview1.hosted.exlibrisgroup.com:8008/vwebv/ui/tiki/htdocs/index.html

ZombieVoyage

http://preview1.hosted.exlibrisgroup.com:8008/vwebv/ui/zombie/htdocs/index.html

(Please note that our skins include some advanced customizations that fall outside the boundaries of our Support Policy. Further, they may contain settings from beta versions that have since been updated for General Release.)

Lessons Learned

- CSS can get complicated fast. Before you start editing, take the time to familiarize yourself with the CSS files. Get an idea of what selectors are being used, and in which files they appear.
- Elements of different pages that look similar may not be sharing CSS. If you change the color of a sidebar in one type of page, other sidebars won't necessarily be affected.
- When in doubt, shift-reload. Some CSS and HTML may be cached by your browser. If you don't see a change you expect, try clicking on the “reload” button while holding the Shift key to fully update the page.
- Firebug is indispensable. This plugin for Firefox makes everything so much easier. See the *Tools* section below.
- The WebVoyáge Architecture Overview and Configuration Models document is a great scaffold for adding features, but its scope is very limited. Some recipes require you to write additional scripts to achieve functionality. Due to typographical limitations, you can't always just copy-n-paste code into files. It's very important to understand what the example code is doing and why before you plug it into your interface.
- Internal Server Errors will give you informative logs. If you are editing an XSL file and make a syntax error (such as forgetting to close a tag), WebVoyáge will display a generic error message. However, it will log a stack trace that usually tells you exactly what went wrong. Just look for the "Root Cause" section in the Tomcat log: `/m1/voyager/xxxdb/tomcat/vwebv/logs/vweb.YYYY-MM-DD.log`.

- It pays to track your changes. Commenting files, maintaining a revision history, and maintaining a list of modifications all help both with ongoing customization and at upgrade time. See *Best Practices* below for more information.

Resources

WebVoyáge Architecture Overview and Configuration Models

The WebVoyáge Architecture Overview and Configuration Models document provides examples of configuration changes that libraries may want to make. Many of the tutorials are excellent bits of added functionality that stand on their own as useful additions to the interface. They're also exceptionally valuable as jumping-off points for additional customizations.

Because the customizations in the WebVoyáge Architecture Overview and Configuration Models tend to be advanced, our support for these projects is limited. We'll be happy to help you implement the tutorials and discuss the implications of further customizations, but we can't write or troubleshoot additional code. If you run into a problem, we'll have a look and see if we can identify problems with your syntax or formatting, but we may advise you to revert to vanilla configurations and recommit your changes one by one.

Tools

Firebug

Firebug is an add-on for Firefox that is absolutely essential for developing HTML and CSS. After installing it, you can enable it from the menu or by hitting F12 and clicking on the **Enable** link. The **Inspect** function (top-left button in the Firebug window) allows you to hover the mouse over any page element and see what CSS has been applied and where it comes from. It also allows you to temporarily add, delete, or change any CSS definition and immediately see the effect. You can get Firebug at:

- https://addons.mozilla.org/en-US/firefox/addon/1843

W3 Schools

The W3 Schools have excellent tutorials on HTML, CSS, XML, and XSLT. The HTML and CSS tutorials feature a "TryIt Editor" that allows you to experiment with HTML and CSS code and see the results interactively.

W3Schools Online Web Tutorials:

- http://www.w3schools.com/

Support Policy

One of the biggest things we learned during the Skin Contest is that it's time to rethink how we support WebVoyáge. With that in mind, we've written up a new Support Policy for the new WebVoyáge UI that makes some allowances for how much more open the software is and the support we think you will need to fully implement it. As always, we're also trying to strike a balance with setting boundaries that allow us to help as many customers as possible.

- http://supportweb.exlibrisgroup.com/cust/voy/downloads/pdf/7.0/WV_CSPolicy.pdf

Best Practices

Test Skin

Voyager 7 ships with three identical skins by default: `en_US`, `sandbox`, and `exl_default`.

- `en_US` is the default skin, and should be configured to your production needs. (You can set up a different default skin—see *Set the default skin* in the *Common Changes* section above.)
- `sandbox` is for you to develop and practice changes on. Test new customizations on the `sandbox` skin and ensure that they're stable before moving those changes into production.
- `exl_default` **must not be changed**. It must contain all vanilla files so that if you encounter errors you can determine whether it's a result of skin customizations or something else.

Accessibility

The default WebVoyáge skin was designed with accessibility as a primary requirement, and has been thoroughly tested to be easy to navigate with a screen reader. (That's why you'll see apparently odd things such as headers that are positioned entirely offscreen—they still serve an important navigation and labeling function.) As you go about planning your customizations, keep this accessibility in mind and strongly consider maintaining the default skin as an accessible alternative to your customized skins. The following are a few tips to keep in mind as you plan accessible customizations.

CSS is designed for presentation, not content

Because CSS is part of a design philosophy that divorces content from presentation, you're reasonably safe modifying CSS styling on any given page without marring the accessibility of the site. The exception is if you begin using CSS to serve content (such as background images that contain labels not otherwise represented in the text of the site). Use CSS to style the look and feel of the site, not to add content necessary for labeling or navigation.

Javascript can easily break accessibility

Javascript allows for powerfully interactive, dynamic web pages. However, javascripting that "helps" usability by limiting user control, requiring a mouse to function, or quietly altering content on the page tends to be inaccessible. Javascript can aid both usability and accessibility, but it should be used with care. A good rule of thumb when planning interactive Javascript is: do not include any Javascript that is required to for the page to function. That is, turning off Javascript should not break your page.

Structural changes to the HTML may render the page less accessible

There are plenty of changes you can safely make to the HTML of WebVoyáge (through the XSL files) that won't affect usability. Keep in mind, however, that the pages in the new interface have been designed to be easily navigated with a screen reader. Keep accessible design in mind as you modify the HTML, particularly if you add new tools or sections to the interface.

For a fuller description of accessibility requirements, have a look at the following resources:

Section 508: The Road to Accessibility:

- http://www.section508.gov/

W3C's Web Content Accessibility Guidelines

- http://www.w3.org/TR/WCAG10/

Commenting Files

We cannot overstate the usefulness of commenting your files as you work. At a minimum, add a comment to clearly demarcate where you've made changes (before and after your alterations). If you make elaborate customizations, it's helpful to document your goals and the method. The following is an example comment from ZombieVoyage's header.xsl:

```
<!-- Begin JL alteration to move the "help" button to a tab, 04-22-2008.  This
required broadening the xPath selectors.  Also note that I've removed the help
     "post"
text.  All of it is link text now and post text will be ignored. -->
<li class="off" title="{$headerText/help}">
   <a href="{/page:page//page:element[@nameId='page.header.help.link']/page:URL}"
    target="_help">
      <span><xsl:value-of
    select="/page:page//page:element[@nameId='page.header.help.link']/page:linkText
    "/></span>
   </a>
</li>
<!-- End JL change to move the "help" button to a tab -->
```

Revision History

As you go about customizing your OPAC, you'll probably make many changes to a few of the same files. In this case, it's helpful to maintain a revision history. Before beginning substantial alterations, back up the stable version of the file with a new name that includes the date you altered it:

```
cp -p webvoyage.properties webvoyage.properties.2008-05-15
```

You may find that this leaves a lot of extra files in a directory if you've altered the files a lot. That's a good thing. Because customizations made to another file may have unexpected interactions down the road, it's good to have a revision history so you can roll back your changes one by one when you encounter a problem. Since advanced customizations have only very limited support, we may advise you to return to default configurations. Rolling back a few changes to the last stable version of your customizations is better than losing all your work and restarting from vanilla configs.

List of changed files and preparing for an upgrade

When a new release of Voyager comes out, the patch or upgrade may include fixes that alter files you've customized. When that happens a new, vanilla version of the file may be put into place. You'll need to recommit your customizations after the upgrade. Because of this, it's handy to maintain a list of files you've altered with a brief description what was changed, such as the following:

```
.../skin/xsl/pageTools/frameWork.xsl:
-added an import for overlay.css
-changed the onLoad statement to load init() script
.../skin/xsl/contentLayout/cl_displayRecord.xsl:
-added separate display of serials from Architecture guide, chapter 3
-added persistent link to records from Architecture guide, chapter 12
```

The extent of detail you go into will depend on your context. If there's a chance that someone else will have to recommit your customizations (which is always possible), it may be worth describing more about how the changes were made. This sort of documentation will pay major dividends when upgrade time rolls around.