This documentation describes the Health News Harvesting System developed for use within the North Carolina AHEC Digital Library and University of North Carolina Chapel Hill Health Sciences Library web sites.

The Health News Harvesting System gathers, parses, and stores RSS feeds from several of the top health science news sites. Once the information has been retrieved, this system will then extract, parse, and store the associated data, where it can then be searched, sorted, retrieved, and analyzed by UNC Health Sciences Library staff, patrons, and constituencies. This documentation provides a brief summary of concepts concerning Rich Site Summary (RSS) and RSS Aggregation, and explains the application of these concepts within the structures of these information systems.

The final section provides the system code for each component of the Health News Harvesting System in its entirety, as well as in-line annotations to describe individual processes within the components.
Health News Harvesting System: A Rich Site Summary (RSS) Storage and Retrieval System For Use in The University of North Carolina Chapel Hill Health Sciences Library and North Carolina AHEC Digital Library Web Sites

by

David A Dodd

A Master's paper submitted to the faculty of the School of Information and Library Science of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Master of Science in Information Science.

Chapel Hill, North Carolina

April, 2006

Approved by:

Jeffrey Pomerantz
Advisor
Document Purpose

The purpose of this documentation is to describe the RSS Harvesting System developed for use within the North Carolina AHEC Digital Library and University of North Carolina Chapel Hill Health Sciences Library web sites. This documentation provides a brief summary of concepts concerning Rich Site Summary (RSS) and RSS Aggregation, and explains the application of these concepts within the structures of these information systems. It is also important to note that the system, in its current iteration, has been designed with a focus on system functionality rather than content. Existing content was chosen to illustrate the spectrum of potential system applications.

Future Work

From this point forward, a task force comprised of HSL and AHEC Digital Library staff will research RSS sources and evaluate their inclusion as system content. Once those determinations have been made, additional RSS sources will be entered into the database for retrieval purposes. Furthermore, the search functionality will be expanded to allow users to search by field type such as a certain time period, or a specific RSS source. Another future system enhancement will provide the functionality of information filtering and allow users to customize their own, predetermined criteria for search results. In this way, users will be able to create their own customized “news pages.” The system can also be further enhanced to categorize feeds based on subject type, and display RSS feed items outside of the current, supported websites (UNCHSL and AHEC). Therefore, the system content need not be limited to health-news related RSS feeds, and the system may be available on other websites in the entire UNC System and among its partners.

Rich Site Summary

RSS is a structured XML-based metadata schema used to distribute news items, primarily through a web medium. The contents of the news items are organized into a set of standardized xml elements: title, link (URL), pubDate (date of publication), description, language, webmaster, copyright, and several others. For the purposes of this system, I am capturing the data contained within the title, link, pubDate, description elements. These elements are being captured for two reasons: a) they are commonly included in RSS feed items, and provide an impression of item content (ie. what, when, where, and how), and b) these items will allow users to evaluate information relevance, based on the above criteria.

RSS Aggregators

RSS aggregators, also referred to as news aggregators, are software programs that retrieve RSS feeds and present them through a single interface. RSS aggregators are commonly stand-alone client applications, but can also work within an internet browser, as a form of a plug-in or extension of the browser itself. These applications are similar to
current generation email clients, in that they retrieve and store individual RSS feed items in a single list, and provide limited search and organization capabilities. Current RSS aggregators function as independent web sites, or as independent clients existing outside of an internet browser or email application. The Health News Harvesting System seeks to expand on the capabilities of existing RSS aggregators in several ways: a) including site-embedding capabilities; the individual components comprising the system (management tools, search utility, presentation interface, and database structure) are modular, and can be customized for inclusion into other free-standing web sites; b) the system incorporates RSS elements into faceted organizational schema, for the purposes of information presentation and retrieval within websites.

Health News Harvesting System

The Health News Harvesting System gathers informational feeds from several of the top health science news sites, such as BBC News Health, New York Times Health, The Washington Post health, American Council on Science and Health, MedicineNet, and MedScape. Once the information has been retrieved, this system will then extract, parse, and store the associated data into a SQL database, where it can then be searched, sorted, retrieved, and analyzed by UNC Health Sciences Library staff, patrons, and constituencies. This project requires the use of several software applications and coding schemes, including (but not limited to):

- Cold Fusion middleware coding for data gathering and extraction
- SQL (utilizing Microsoft SQL server) for data storage, retrieval, and analysis
- HTML, CSS, and JavaScript (for interface presentation)

This system performs the task of harvesting RSS content through the following processes:

1. user directs system to the URL of RSS feed
2. system reads and parses contents of feed
3. system determines whether duplicate records exist in the database
4. system stores new RSS items within the database

For more information on this process, see “RSS Retrieval Script” under Code Annotation.

At the time of RSS item insertion, the system also stores several other pieces of relevant information, including the source of the content (ex. BBC News Health). This subsequent information allows RSS feed items to be “filtered” during the information retrieval process.

It is the intention of this project to facilitate the UNC Health Sciences community by providing a single access point to current health news sources, and to augment current
health news discovery efforts by presenting interested parties with a means of analyzing current health news items for trends, relevance, etc. Through my discussions of the system with Health Sciences Library staff, I have identified several potential uses. The Resource Management Services department is interested in utilizing this system as a means of tracking updates and changes among online vendors and providers. The system could be easily modified to locate, parse, and store RSS feeds provided by content providers, such as UpToDate, Facts and Comparisons, PubMed, etc. This use of this system would streamline the processing of “keeping abreast” of important changes within content provided by these sources. Another potential use of this system has been identified by the HSL Application Development Services department. The system could be used within the Health Sciences Library intranet to customize and display items of interest specific to departments within the library. As an example, the Information Technology Operations department could use this system to track important announcements from campus or vendor technology support groups.

The RSS harvesting system has many potential uses for external constituents of the library as well. The system can be incorporated into the Health Sciences Library digital portal, the AHEC Digital Library, and the North Carolina Health Information portal. Patrons using these systems could find up to date information on current health concerns, search RSS items for topics of interest, and customize search setting to receive filtered news feeds specific to source, date range, or subject. Through discussions with several Health Sciences Library and AHEC Digital Library patrons, I have identified needs among specific users groups, such as faculty and doctors-in-residence, and have began to identify other possible uses as well. For example, faculty could use this system in a classroom setting to introduce current topics of interest. These topics could create the opportunity for class discussion based on newly introduced medical practices or health issues.

There are several augmentation and improvements that will need to be incorporated into the RSS harvesting system before its production launch. Currently the RSS harvesting system searches and extracts RSS feeds based on a static list of incorporated RSS feed URLs. In a production capacity, users will be able to add their own RSS feeds into the system, so that these feeds may be dynamically added to the harvesting schedule, and their contents will in turn be included within database storage. Also, there is not a web search utility for the system in its current capacity. For production utilization, I will expand the RSS Harvesting interface to allow RSS item searches based on RSS element types captured within the database. Examples of search fields include date range, title or description keywords or phrases, or RSS feed source (i.e. World Health Organization, NPR Health News, etc.).

My contribution to this system thus far has been in the role of sole designer and developer. I have created the system logic that allows for automatic location and extraction of contents, database storage, etc. It is my intention to expand this project into a dynamic tool for all Health Sciences Library constituents, internal and external, to facilitate the discovery, processing, and dissemination of health-related news and issues. For this system to grow in terms of user acceptance and integration into work processes, continued adoption of RSS standards, and their subsequent use among content providers, vendors, and news sources, is essential. At the present time, most online news sources have incorporated RSS feeds into their information delivery practice. The list of online
providers and vendors incorporating these practices is to date somewhat limited, but certainly shows signs of growth. The code underlying the Health News Harvesting System is shown under Code Annotation below, and is arranged by system component.

**Annotated System Code**

The following section provides the system code for each component of the Health News Harvesting System code in its entirety, as well as in-line annotations to describe individual processes within the components. Bolded text should be removed before the code is embedded to prevent errors. Each bolded section represents a separate system component, and should be placed within a separate file.

**RSS Retrieval Script**

The RSS retrieval script performs the functions or location and extraction of RSS feeds. This code is set as a scheduled task within the Cold Fusion administration interface to pull new RSS news items once per hour. In the current iteration, the code retrieves the list of URLs for desired RSS feeds from the connected database table, but this process could be modified to pull a comma-delimited list of URLs from within the script itself.

**Retrieve the list of source URLs from the database that are currently marked for retrieval. (Stored RSS sources can be enabled or disable to temporarily halt their relative extraction):**

```cfquery name="RSSList" datasource="developers_Adam">
SELECT rssSourceID, rssSourceURL
FROM rssSource
Where retrieveFlag = 1
</cfquery>

**Extract the title, link, pubDate, and description elements for the last five items contained within the RSS feed for each RSS source. (The code is limited to pull the last five items, to limit system resource allocation to this process):**

```cforeach query="RSSList">

<!--- Example rssSource URL
<cfscript>
</cfscript> --->
```
Assign the contents of the RSS feed to a text variable:

<cfscript>
XMLContent = trim(cfhttp.filecontent);
XMLContent = XMLParse(XMLContent);
</cfscript>

Display of RSS feed items at the time of processing. This allows for error checking at the point of extraction::

<!---- Feed provided by:<BR>
<!---  Feed provided by:<BR>
<a href="#XMLContent.rss.channel.image.link.xmlText#">
<!---  Feed provided by:<BR>
<a href="#XMLContent.rss.channel.image.link.xmlText#">
<!---  Feed provided by:<BR>
<a href="#XMLContent.rss.channel.image.link.xmlText#"> #XMLContent.rss.channel.title.xmlText#<BR>
#XMLContent.rss.channel.description.xmlText#<BR><BR>
</cfoutput>
<cfloop from="1" to="5" index="i"> #XMLContent.rss.channel.title.xmlText#<BR>
#XMLContent.rss.channel.description.xmlText#<BR><BR>
</cfloop>
</cfoutput>
</cfloop>

Assign item elements to system variables. The contents of the date element is transformed from international date format to a standard date format so that it may be properly read by the database as a date/time value:

<cfif LSIsDate(XMLContent.rss.channel.item[i].pubDate.xmlText)>
<cfset rssDate = #LSDateFormat(XMLContent.rss.channel.item[i].pubDate.xmlText, "mm/dd/yyyy")#>
<cfelse>

<cfhttp url="#rssSourceURL#"
   method="GET"
   timeout="15">
</cfhttp>
Remove HTML markup from the description element, and assign to variable:

<cfset rssDesc = ParagraphFormat(ReplaceNoCase(XMLContent.rss.channel.item[i].description.xmlText , "&lt;", "&","\"","ALL"))#>

Check the database table for duplicate records. Duplicate records are recognized by an exact match of the news item title:

<cfquery name="SelectRSSDup" datasource="developers_Adam">
SELECT rssID
FROM rssHarvest
Where rssTitle = '#XMLContent.rss.channel.item[i].title.xmlText#'
</cfquery>

If the record is not a duplicate, the contents of each element are stored within the database:

<cfif SelectRSSDup.recordcount EQ 0>
<cfquery name="InsertRSS" datasource="developers_Adam">
    INSERT INTO rssHarvest (rssTitle, rssDate, rssDesc, rssURL, rssSourceID)
    VALUES
    ('#XMLContent.rss.channel.item[i].title.xmlText#', '#rssDate#', '#rssDesc#', ' #XMLContent.rss.channel.item[i].link.xmlText#', '#rssSourceID#')
</cfquery>
</cfif>
</cfoutput>
</cfloop>

RSS Quick Display

This component allows the user to pull all news feeds items from each RSS source on demand, without limitation to date, time, or category. This particular example is used to display all available news items currently available from the Washington Post Health News feed. Since the contents of the news feed are simply “read” by the process,
no database insertion or preprocessing are necessary. This script duplicates the functionality of a summary “front page” for a chosen RSS source.

**Assign the contents of the RSS feed to a text variable:**

<!--- Get RSS Feed --->

<!-- read XML datasource file into string variable called xmlCode-->
<cfset rss = XMLParse(cfhttp.filecontent)>
<!--- get an array of items --->

<!--- Feed provided by:<BR>
<a href="#XMLContent.rss.channel.image.link.xmlText#">
  <img src="#XMLContent.rss.channel.image.url.xmlText#" alt="#XMLContent.rss.channel.image.title.xmlText#" border="0"><br>
</a><br>

#XMLContent.rss.channel.title.xmlText#<BR>
#XMLContent.rss.channel.description.xmlText#<BR><BR>
</cfoutput>
<hr>
<cfoutput>
<a href="#XMLContent.rss.channel.item[i].link.xmlText#">#XMLContent.rss.channel.item[i].title.xmlText#</a><BR>
#ParagraphFormat(ReplaceNoCase(XMLContent.rss.channel.item[i].description.xmlText ,"&lt;","","ALL"))#
</cfoutput>

**RSS Administration Utility**

The RSS Administration Utility allows system content administrators to manage RSS source records, and to determine which RSS sources are retrieved for a given period of time. For both the North Carolina AHEC Digital Library and University of North Carolina Chapel Hill Health Sciences Library sites, access to this utility is limited to system administrators.

**Query to determine RSS sources currently included within the database.**

<cfquery name="lookupRSSSource" dbtype="ODBC" datasource="Developers_Adam"> SELECT * FROM RSSSource
Set default value of RSS administration form to “none”. If a RSS record has been added, edited, or cloned, the form dynamically sets to that mode instead.

```
<cfparam name="form.submit" default="none">
<cfswitch expression="#form.submit#">
  <cfcase value="Edit,Clone">
  </cfcase>
</cfcase>
</cfdefaultcase>
```

Begin RSS administration form. This is the preliminary form for record selection. Record manipulation (add, edit, clone, delete) is handled in the subsequent form. The form diverges here based on mode selection – add or manage existing feed.

```
<cfform name="showRSS" action="RSSAction.cfm" method="post" scriptsrc="cfform.js">
  <div class="utilityH2" id="NoleftMargin">
    <h2 id="darkBrown">Manage RSS Feeds</h2>
  </div>
  <div class="lightTanBack" id="utilityFormData">
    <div id="QuickActionButton">
      <span class="dataItem">
        <span class="labelLong">Add an RSS Feed</span>
        Add option submit.
        <input type="submit" name="submit" value="Add RSS Feed" id="UserQuickActionButtonLong" />
      </span>
    </div>
  </div>
</cfform>
```

Add option submit.

```
<input type="submit" name="submit" value="Add RSS Feed" id="UserQuickActionButtonLong" />
</span>
</div>
```

List RSS feeds stored within the database, and select mode.

```
<cfif lookupRSSSource.RecordCount eq 0>
  <cfset introMessage = introMessage & "There are no RSS feeds to edit.">
</cfif>
```

```
<p>
  <span class="confirmation">#introMessage#</span>
</p>
```
<span class="dataItem">
<span class="labelLong">Manage an Existing RSS Feed</span>

<!--- Print the table headers --->
<table border="0">
<tr class="shade">
<th>RSS ID</th>
<th>RSS Source Name</th>
<th>Currently Retrieved?</th>
<th>Edit?</th>
<th>Delete?</th>
</tr>
<!--- Start outputting query results to the remaining table cells --->
<cfoutput query="lookupRSSSource">
<!--- Shade every other table row ---->
<cfif (counter mod 2 eq 0)>
<tr class="shade">
<cfelse>
<tr>
</cfif>
<td>&nbsp;&nbsp;#RSSSourceID#&nbsp;&nbsp; </td>
<td> &nbsp;&nbsp; #RSSSourceName# &nbsp;&nbsp; </td>
<td><center><cfif lookupRSSSource.retrieveFlag EQ 1>Yes<cfelse>No</cfif></center></td>
<td>&nbsp;<a href="RSSAction.cfm?rssIDedit=#rssSourceID#">Edit</a>&nbsp;</td>
<td>&nbsp;<a href="RSSAction.cfm?rssIDdelete=#rssSourceID#">Delete</a>&nbsp;</td>
</tr>
<cfset counter=counter+1>
</cfoutput>
</table>
</div>
</cfif>
</cfform>
</cfdefaultcase>
</cfswitch>
</div><!--- END viewDataTab --->
</div><!--- END UtilityWrapper --->

RSS Administration Utility Processing Script

Once the user has determined the proper mode in the above form, the variables are past to the RSS Administration Utility Processing Script.
Initialize default values for form variables.

<cfset counter=1>
<cfset errorMessage = "">
<cfparam name="introMessage" default="">

<cfset link = "">
<cfset header = "">
<cfset rssSourceIDchosen = URL.rssIDdelete>
<cfset form.submit = "DeleteID">
</cif>

<cfswitch expression="#form.submit#">
<cfcase value="Add RSS Feed">
<cfset link = "<a href=\"rssForm.cfm\">Return to the RSS Feed Harvesting Utility</a>">
<cfset header = "Add an RSS Feed">
<cfparam name="rssSourceID" default="">
<cfparam name="rssSourceName" default="">
<cfparam name="rssSourceURL" default="">
<cfparam name="retrieveFlag" default="">
</cfcase>
</cfswitch>

Determine form mode. Processing diverges at this point to two main paths – record entry or manipulation, and database insertion or update.

Based on mode selection, separate form validation methods occur. Upon record add or edit, the script checks for valid entries for each form field. Delete selections are checked for a valid rssSourceID, and present delete confirmation.

<cif IsDefined('URL.rssIDedit')>
    <cfset rssSourceIDchosen = URL.rssIDedit>
    <cfset form.submit = "Edit">
</cif>

<cif IsDefined('URL.rssIDdelete')>
<cfset rssSourceName = lookupRSSSource.rssSourceName>
</cifcase>
<cfcase value="Add,New" delimiters=",">
!---- <cfcase value="Add"> ---->
<cfif form.submit IS "Add">
Present add/edit RSS record entry form. The RSS source name (authoring web site), source URL (persistent domain address), and yes/no retrieve selection (to determine whether or not the RSS source should enter the retrieval process established in the scheduled task), are established here. The rssSourceID is automatically assigned by the database to ensure unique identification.
WHERE rssSourceID = #rssSourceIDchosen#
</cfquery>
<cfset rssSourceID = lookupRSSSource.rssSourceID>
<cfset rssSourceName = lookupRSSSource.rssSourceName>
<cfset rssSourceURL = lookupRSSSource.rssSourceURL>
<cfset retrieveFlag = lookupRSSSource.retrieveFlag>
</cfcase>
<cfcase value="DeleteID">
<cfset link = "<a href="rssForm.cfm">Return to the RSS Feed Harvesting Utility</a>">
<cfset header = "Delete RSS Feed">
<cfquery name="lookupRSSSource" dbtype="ODBC" datasource="Developers_Adam">
SELECT rssSourceID, rssSourceName
FROM rssSource
WHERE rssSourceID = #rssSourceIDchosen#
</cfquery>
<cfset rssSourceID = lookupRSSSource.rssSourceID>

Handle database insertion or record update statements.

<cfset form.submit = "Readd">
<cfset header = "Add RSS Feed">
</cfif>
<cffif form.submit IS "Update">
<cfset header = "Edit RSS Feed">
<cfset form.submit = "Edit">
</cffif>
</cffif>
<cffif NOT errorMessage IS "">
<cfinclude template="../Utilities/Includes/errorMessageOutput.cfm">
<cfelse>
<cffif form.submit IS "Add">
<cfquery name="addRSSSource" dbtype="ODBC" datasource="Developers_Adam">
INSERT INTO rssSource
(rssSourceName, rssSourceURL, retrieveFlag)
VALUES ('#form.rssSourceName#', '#form.rssSourceURL#', #form.retrieveFlag#)
</cfquery>
<cfelseif form.submit IS "Update">
<cfquery name="updateRSSSource" dbtype="ODBC" datasource="Developers_Adam">
UPDATE rssSource
SET rssSourceName = '#form.rssSourceName#', rssSourceURL = '#form.rssSourceURL#', retrieveFlag = #form.retrieveFlag#
WHERE rssSourceID = #form.rssSourceID#
</cfquery>
</cffif>
</cffif>
</cffif>
<cfcase value="Delete">
<cfset link = "<a href="rssForm.cfm">Return to the RSS Feed Harvesting Utility</a>">
<cfset header = "Deletion Successful">
<cfquery name="deleteRSSSource" dbtype="ODBC" datasource="Developers_Adam">
DELETE FROM rssSource
WHERE rssSourceID = #rssSourceID#
</cfquery>
</cfcase>
</cfswitch>

Return confirmation message to the user. If their has been a validation error, present the error to the user, and return the user to the proper form for validity check.

<cfoutput>
<cfif NOT introMessage IS "">
<p><span class="confirmation">#introMessage#</span></p>
</cfif>
</cfoutput>

<div id="linkBack">
<cfoutput>#link#</cfoutput>
</div>

<div id="UtilityWrapper" class="blueBack">
<div class="viewDataTab">
<cfform name="rssForm" action="rssAction.cfm" method="post" scriptsrc="cfform.js">
<div class="utilityH2" id="NoleftMargin">
<cfoutput><h2 id="darkBrown">#header#</h2></cfoutput>
</div>
<div class="lightTanBack" id="utilityFormData">
<cfoutput>
Present record elements for existing rssSource entry from the database, and allow user to manipulate existing element fields.

<!--- #form.submit# --->
<cfif form.submit IS "Edit">
<span class="dataItem">
<span class="label">ID:</span>
<input type="hidden" name="rssSourceID" value="#rssSourceID#">
</span>
</cfif>
</cfoutput>
</div>
</cfform>
</div>
</div>
Add record form.

<cfif (form.submit IS "Edit") OR (form.submit IS "Add RSS Feed") OR (form.submit IS "Readd")>

<cfif (form.submit IS "Edit")>

<RSS Source Name:
<input type="text" name="rssSourceName" value="#rssSourceName#" size="65">
</cfif>

<RSS Source URL:
<input type="text" name="rssSourceURL" value="#rssSourceURL#" size="65">

(REtrieve RSS Feed?:
<input type="radio" name="retrieveFlag" value="1" checked>
Yes
<input type="radio" name="retrieveFlag" value="0" checked>
No
</cfif>

Edit record form.

<cfif form.submit IS "Edit">

<!--- <input type="hidden" --->

<input type="submit" name="submit" value="Update" class="white" id="greenBack" />
</cfif>

<cfelseif form.submit is "DeleteID">

ID:
#rssSourceID#</cfif>

</cfif>
Output confirmation message to user based on data manipulation type.

Add record confirmation, and output of record detail.

<cfelseif (form.submit is "Add") OR (form.submit is "Update")>
<cfif form.submit is "Add">
<cfquery name="lookupRSSSourceEdits" dbtype="ODBC"
datasource="Developers_Adam">
SELECT *
FROM rssSource
WHERE (rssSourceID IN
(SELECT MAX(rssSourceID)
FROM rssSource))
</cfquery>
</cfif>
You added the following new record:
<br/>

Edit record confirmation, and presentation of edited record fields.

<cfelse>
<cfquery name="lookupRSSSourceEdits" dbtype="ODBC"
datasource="Developers_Adam">
SELECT *
FROM rssSource
WHERE rssSourceID = #form.rssSourceID#
</cfquery>
You have made the following updates to [#oldrssSourceID# - #oldrssSourceName#]:
<br/>
</cfelse>
RSS Search Utility

The RSS Search Utility comprised of three sections – search action (item retrieval), in-line search box, and record item display. The search utility allows for Boolean (and/or) functionality for one or more terms. Search results are presented descending date order (most current records display at the top of search results).

Describe purpose of search field to the user.

<cfset intro = "Use this search feature to locate articles from top world health news sources.">

If search is submitted, begin record retrieval.

<cfif isDefined('form.searchSubmitted')>
<!--- Search Form Submitted --->
<cfif form.terms IS "">
<cfset feedback = "-1"
Initialize search variables.

<cfset rssResults="prime">
<cfset newRssResults="prime">
<cfset scRssResults="prime">
<cfset newSCRssResults="prime">
<cfset keyTerm="prime">
<cfset String = form.terms>
<cfset c = "1">

Loop record retrieval over each search term entered into the search field.

<cfset Position = getPosition(String, c, " ")>
<cfloop condition="Len(#Position#) NEQ 0">

Recognition of boolean terms within search field, as a determination of search term relationship.

<cfif (Position EQ "AND") OR (Position EQ "OR")>
<cfset keyTerm= ListDeleteAt(keyTerm, 1, " ")>

Select RSS items that have matching terms in title or description.

<cfquery name="searchRSS" datasource="Developers_Adam">
SELECT DISTINCT rssID
FROM rssHarvest
WHERE (rssTitle LIKE '%#keyTerm#%') OR
(rssDesc LIKE '%#keyTerm#%')
</cfquery>

If this is not the first search term and the Boolean operator is "and", Select the intersection of query results and result set list. Add additional retrieved records to the search set.

<cfif IsDefined("junction") and junction EQ "AND">
<cfloop query="searchRSS">
<cfif ListFind(rssResults, rssid) NEQ 0>
<cfset newRssResults = listAppend(newRssResults, rssid)>
</cfif>
</cfloop>
<cfset rssResults = newRssResults>

Else if this is the first search term or the Boolean operator is "or", select search results meeting either condition.
Select RSS records that have matching terms in title or description.

```
<cfquery name="searchRSS" datasource="Developers_Adam">
SELECT DISTINCT rssID
FROM rssHarvest
WHERE (rssTitle LIKE '%#keyTerm#%') OR (rssDesc LIKE '%#keyTerm#%')
</cfquery>
```

Return results for each search term included within the Boolean “and” statement (if applicable).

```
<cfif IsDefined("junction") and junction EQ "AND">
<cfloop query="searchRSS">
<cfif ListFind(rssResults, rssid) NEQ 0>
<cfset newRssResults = listAppend(newRssResults, rssid)>
</cfif>
</cfloop>
<cfset rssResults = newRssResults>
</cfif>
```

Return results for each search term included within the Boolean “or” statement (if applicable).

```
<cfelse>
<cfloop query="searchRSS">
<cfif ListFind(rssResults, rssid) EQ 0>
<cfset rssResults = ListAppend(rssResults, rssid)>
</cfif>
</cfloop>
<cfset rssResults = rssResults>
</cfelse>
```
Retrieve full search result record information and present retrieved rss Records to the user.

```cfquery name="getRSSItems" dbtype="ODBC" datasource="Developers_Adam">
SELECT rssID, rssTitle, rssDate, rssURL, rssSourceName
FROM rssHarvest, rssSource
WHERE rssID in (#rssResults#)
AND rssHarvest.rssSourceID = rssSource.rssSourceID
</cfquery>

Determine number of records meeting condition(s) of search terms.

```cfset hits = getRSSItems.RecordCount>

<cfif hits EQ 0>
<!--- If search returns 0 hits --->
<cfset feedback = "0">
<cfelse>
<cfset feedback = "1">
</cfif>

</cfif>
</cfif>

If no records met the search term or terms, present “no records found” message to user.

```html>
<head>
<cfinclude template="Includes/header.cfm">
<script src="Scripts/setInitialFocus.js" language="JavaScript1.2"
type="text/javascript"></script>
</head>

Output the search box at the top of the page each time for immediate repetition of search entry.

```div id="searchBox">
<cfoutput>
<cfform name="searchForm" method="post" action="RSSsearch.cfm"
scriptsrc="cfform.js">
<input type="hidden" name="searchSubmitted" value="1" />
```
<input type="text" name="terms" size="25" maxlength="255" value="#form.terms#" />
<input type="image" name="search" id="search" value="Search"
src="Images/searchButton.gif" width="68" height="28" alt="Search" />
</cfoutput>
</div>

<cfswitch expression="#feedback#">
<cfcase value="-1">
If no terms were entered into the search field upon submission, return error.

<p class="feedbackMessage" id="searchFeedback">
You did not enter any search terms. Please try again.
</p>
</cfcase>
<cfcase value="0">
If no results were found based on the term set, return error.

<cfoutput>
<p class="feedbackMessage" id="searchFeedback">
Your search: "<strong><span class="darkGreen">#String#</span></strong>" returned 0 hits. Please try again.<br />
<ul>
<li>Check your spelling.</li>
<li>Try other search terms.</li>
<li>Truncate your search terms. No truncation symbol is necessary. <br />ex. Change 'nursing' to 'nurs'</li>
</ul>
</p>
</cfoutput>
</cfcase>
<cfcase value="">
<!--- Form not submitted --->

 &nbsp;
</cfcase>
<cfdefaultcase>
Output retrieval set in-line.

<div id="anchorList">
<cfoutput>
Present user with the number of results that met the term set.
</cfoutput>
</cfoutput>
</div>
</cfswitch>
Database Create Statements

The database structure for the Health News Harvesting System consists of two tables: rssHarvest and rssSource. The rssHarvest table store individual RSS items from the included RSS sources, while the rssSource table stores information specific to the RSS sources themselves, such as the rssSource Name and location. These tables are joined on the rssSourceID, which identifies the one-to-many relationship of source and item relationships. Below are the SQL database create statements, which will allow duplication of the table structure for use in external database models.

CREATE TABLE [dbo].[rssHarvest] (  [rssID] [smallint] IDENTITY (1, 1) NOT NULL ,  [rssTitle] [varchar] (250) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,  [rssDate] [datetime] NOT NULL ,  [rssDesc] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,  [rssURL] [varchar] (250) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
CREATE TABLE [dbo].[rssSource] (  
    [rssSourceID] [smallint] IDENTITY (1, 1) NOT NULL ,  
    [rssSourceName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,  
    [rssSourceURL] [varchar] (250) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,  
    [retrieveFlag] [bit] NOT NULL  
) ON [PRIMARY]
Appendix A: Screenshots of System Components
The RSS Search Utility displays the search field and record item display. The search utility allows for Boolean (and/or) functionality for one or more terms. Search results are presented in descending date order.
Current display interface (last five news items for top five source providers):

This screen allows the user to view all news feeds items form each RSS source on demand, without limitation to date, time, or category. This particular example is used to display the last five news items for the top five source providers.
Administrative Utility:

The RSS Administration Utility allows system content administrators to manage RSS source records, and to determine which RSS sources are retrieved for a given period of time.
Database screen capture:

This is a screen capture of the combination of Health News Harvesting System database tables used to store RSS feed records.