High availability options for IBM WebSphere Portal 6.1 search

Andreas Prokoph
Software Architect
IBM Development Laboratory
Boeblingen, Germany

Eitan Shapiro
Software Engineer
IBM Haifa Research Laboratory
Haifa, Israel

June 2009

Summary: Increasingly, users of IBM® WebSphere® Portal have been requesting information on how to provide high availability for the Portal Search service. Out of the box, Portal Search service does not support high availability and thus presents a potential for a single point-of-failure for this service. This white paper explains what options are available to provide certain degrees of high availability for Portal Search service.

Table of Contents

1 Introduction

Providing built-in support for high availability search is dependent on true enterprise scale and class search engines, such as IBM OmniFind® Enterprise Edition. Other search technologies require manual setup for such a service, including the setup of multiple instances of that service paired with the use of a load balancer to distribute the search requests among the available search services.

To understand one of the fundamental issues with search services, it is important to note that the search engine's search index (or search collection) is in essence a "database" that is queried to resolve the search request and return the search result.

However, search index data is not stored in relational database tables; thus is is not possible to draw benefits from the relational database's ability to support clustered environments and thus profit from its service reliability.

Instead, the search engine stores its search index in the file system in a proprietary format. The nature of an inverted search index (which is what most search technologies is based on) does not easily allow for replication of newly added information, as is the case with relational databases (technically speaking, and using the term "easily" purely in the sense of comparison and not technical implementation).

This paper explains what options are available to provide certain degrees of "high availability" for a Portal Search service. We begin with the theoretical aspects of how to achieve this goal, and then we propose solutions that are actually feasible and that will deliver a highly available search service.

The procedures described here, including the references to the product documentation, refer to WebSphere Portal V6.1, but they will also work in a WebSphere Portal V6.0 environment.

NOTE: The solutions discussed here are not formally supported by IBM. This information is meant to be shared with interested parties, as is, and cannot be guaranteed to work as documented in all customer environments. However, if a problem does arise that can be reproduced in a standard environment that has been set up in accordance with the product documentation, then of course support is available.

2 High availability and search engines: The big picture

There are a couple of key aspects for providing high availability of the Portal Search service; specifically, failover support in cases where:

- the file system (hard disks) would fail
- user-initiated search requests are not processed—either because the search index is not accessible or because the server itself hosting the search service does not respond
- both end-user search as well as search administration (for example, search index updates) must be available at all times

In theory, at least, several technical solutions for each of the above scenarios are available; however, only a subset of these scenarios can be implemented without significant administrative and development effort.

The following three subsections describe possible technical solutions and show how to set them up, including explanations of the respective drawbacks or limitations, where applicable.

The options are as follows:

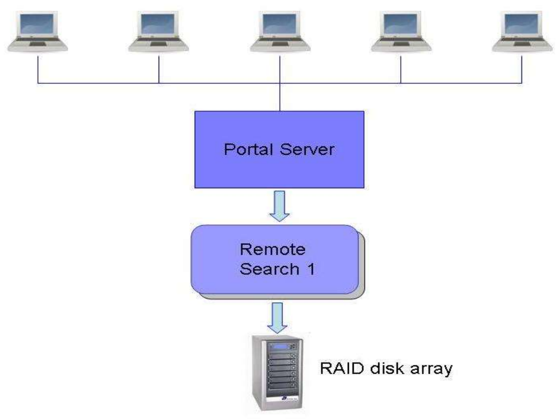
- High availability of the search collections using a redundant array of independent disks (RAID) disk device
- Setting up a cluster of search services using a network dispatcher
- Setting up a main search service and complementing it with at least one search service for failover situations

2.1 High availability using RAID disk device

This solution targets the high availability of the search index itself (in WebSphere Portal we talk about "search collections"). More specifically, the solution is geared toward ensuring that the search index is available through the file system at all times (see figure 1).

NOTE: Remember that a search index is not stored in a relational database; rather, it is a database of its own, managed solely by the search engine and persisted in the file system.

Figure 1. Portal search service setup using a RAID disk device



By introducing a RAID disk device to this hardware we ensure that single-disk failures problems are kept to a minimum.

In general terms, this solution dramatically improves the availability of the search index but not the failures of the search service itself, which processes the search requests and requires access to the search indexes on the file system. Such failures could be a result of the hosting application server going down for one reason or the other.

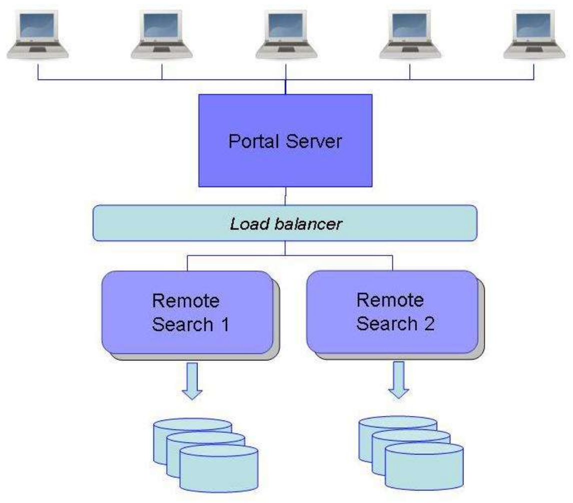
This setup does not require any special search administration tasks, other than making sure the location specified in the mandatory entry field "Location of Collection" refers to a directory on that RAID disk device. For details see the WebSphere Portal 6.1 information center topic, "Managing search services".

2.2 Clustered search services using a network dispatcher

This solution discusses a setup of multiple search services in which each of the search services represent a duplicate of the other. Once these services have been set up, a network dispatcher can be configured to distribute the search requests among the available search services.

Figure 2 shows a high-level view of such a setup.

Figure 2. Clustered search services



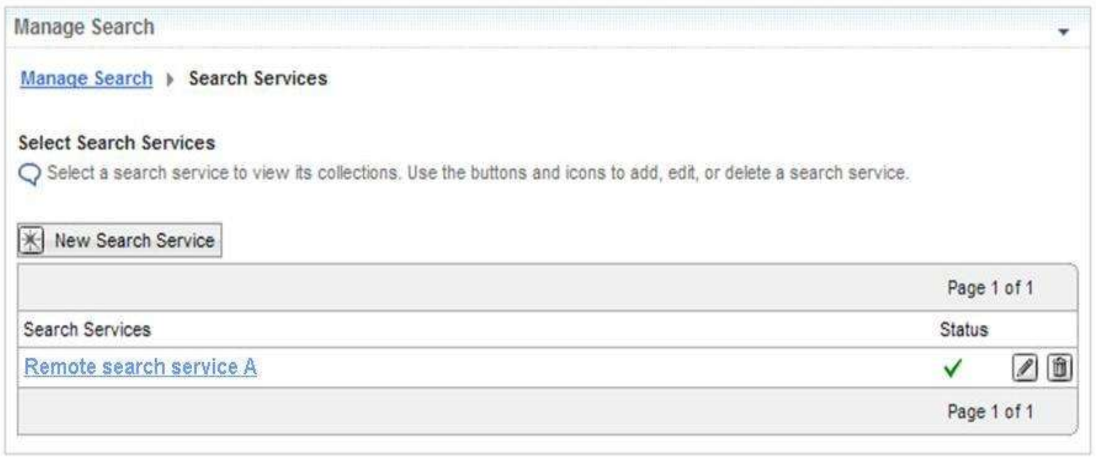
You can build such a clustered search environment by performing these steps:

1. Install two or more remote search services, per the InfoCenter topic Using remote search service. Note that each search service must be installed on a hardware box different than that on which the initial search service is deployed, to ensure high availability of the search service.

2. Once the search services have been installed, instantiate and configure an "initial' (master) search service using the Manage Search administration function located in the Administration pages of the WebSphere Portal server. Note that you must configure the search services as “remote” search services (see the InfoCenter topic, “Configuring a remote search service”).

   Figure 3 shows the resultant Search Services view of the Manage Search portlet.

Figure 3. Manage Search: List of available search services



Manage Search

Manage Search ▸ Search Services

Select Search Services

Select a search service to view its collections. Use the buttons and icons to add, edit, or delete a search service.

New Search Service

Page 1 of 1

Search Services	Status
Remote search service A	✓

Page 1 of 1

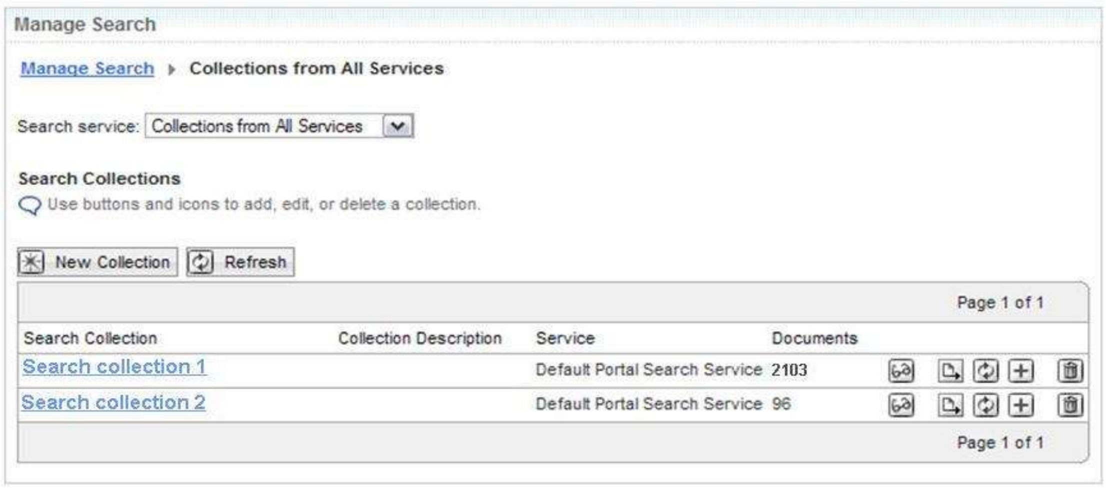
3. The next step is to create a search collection. Note that it is not possible to so by providing a storage location on a shared file system that can be accessed by the search services. Every search service must maintain and administer its own (set of) search collection(s).

4. Once the search collection has been created, you must configure one or more crawlers (content sources) that will later fetch content, and then feed them into the indexer so that they are available for searching.

5. Manually start the configured crawler(s) to start the search collection build process (indexing). When you've confirmed the crawlers have been configured correctly, it is recommended to then define schedules for the crawlers, to automatically update the search collection at the specified days and time.

   In the example shown in figure 4 we have created and built two search collections.

Figure 4. List of configured and active search collections



Manage Search

Manage Search ▸ Collections from All Services

Search service: Collections from All Services

Search Collections

Use buttons and icons to add, edit, or delete a collection.

New Collection | Refresh

Page 1 of 1

Search Collection	Collection Description	Service	Documents
Search collection 1		Default Portal Search Service	2103
Search collection 2		Default Portal Search Service	96

Page 1 of 1

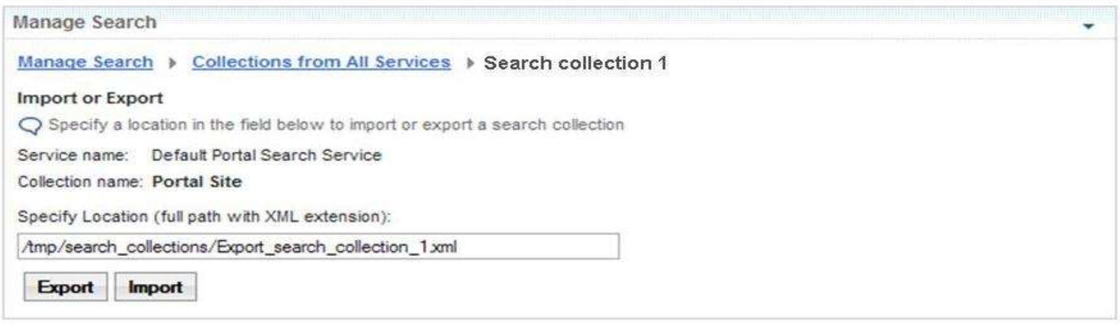
6. After verifying that the search collections are configured correctly, copy the search collection configuration information over to another search service. To do this:

   a. Export the search collection(s) configuration information, which is captured in an XML document containing all information related to the collection's configuration; the content crawlers (including the schedules); as well as the content and documents that have been collected by the crawlers.

   b. Access the search collection's export/import function by clicking the Export/import icon on the right-hand side of the search collection entry in the list of search collections. For more information, refer to the topic, "Exporting and importing search collections".

      The user interface in figure 5 is presented to the administrator, who then specifies a location in the file system to store the output XML file. Note that this file must later be made accessible to the second search service as well.

Figure 5. Exporting a search collection configuration information


Manage Search

Manage Search ▸ Collections from All Services ▸ Search collection 1

Import or Export

Specify a location in the field below to import or export a search collection

Service name: Default Portal Search Service

Collection name: **Portal Site**

Specify Location (full path with XML extension):

/tmp/search_collections/Export_search_collection_1.xml

7. The next step is to create the search collection(s) for the second search service. To do this, you must create an identical set of search collections, where “identical” refers to specifying the same location of the search collection, meaning the same storage location (folder) by name. This is not to be confused with pointing to the same “shared” location in the file system.

   Note that here we don't need to create the content sources (crawlers); these will be created as part of the “Importing the search collection” procedure.

8. For every search collection, load the respective configuration data by importing the configuration information, using the XML file that was generated via the export from the first search service.

   NOTE: In WebSphere Portal V6.1 the notion of “backup and recovery” was introduced for Portal Search. If enabled, you can find the search collection configuration information in the specified storage location for the backup files. Details can be found in the topic, “Backup and recovery of search collections configuration”.

9. After a successful import of the search collection, they are ready to be built:

   - If the XML data was imported via the file generated by manual export, the search collection build procedure will start automatically. This is because the XML export file contains not only the raw configuration data, but also the references (for example, URLs) to the content stored in the search collection. Thus the crawler(s) is fed with that explicit list of content and initially fetches those items in that list to perform a batch load of the search collection.

   - If, however, the XML import information was taken from the backup and recovery file, then the crawling of the content either must be started manually once, or you must wait for the next scheduled crawl(s) to occur.

   NOTE: It is recommended to reconfigure the schedules of the content sources so as to prevent multiple crawlers—from each of the individual search services—from hitting the target servers or content repositories at the same time and generating unnecessary load on those servers.

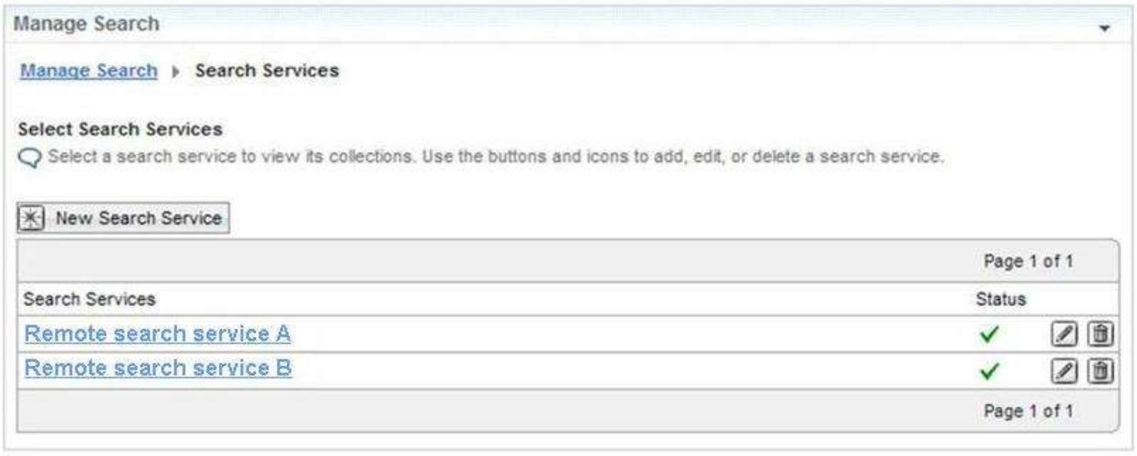
10. Once the required number of search services has been set up, you must then configure a load balancer to take charge of distributing the search requests among the available search services:

- The simplest solution is to configure a Web server to distribute the requests to the search services by setting up the required plug-ins. Of course, if you do this, it makes the Web server a single point of failure (SPOF).

- Another option is to introduce a hardware-based solution that uses a network dispatcher to handle distribution of the requests between the configured search services.

With the current setup any search requests issued through the Search Center portlet would automatically send the queries to every registered search service. Figure 6 shows two search services with identical search collections.

Figure 6. Two identical search services configured and operational


Manage Search

Manage Search ▸ Search Services

Select Search Services

Select a search service to view its collections. Use the buttons and icons to add, edit, or delete a search service.

New Search Service

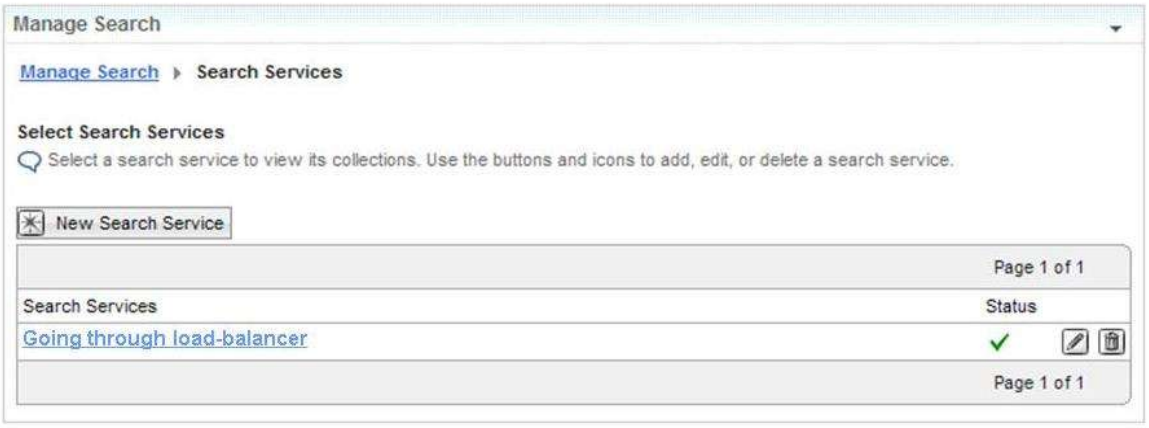
Search Services	Status
Remote search service A	✓
Remote search service B	✓

Page 1 of 1

11. So, to activate the load balancer for the search services, the final step is to replace the configuration information (URLs) of the remote search service(s) with a URL that references the search service through the load balancer.

    Thus, from now on all requests will go through the load-balancer, which decides to what physical search service the request is then sent for processing (see figure 7).

Figure 7. Search services going through load balancer


Manage Search

Manage Search ▸ Search Services

Select Search Services

Select a search service to view its collections. Use the buttons and icons to add, edit, or delete a search service.

New Search Service

Search Services	Status
Going through load-balancer	✓

Page 1 of 1

Since we now have replaced the original search services with a "virtual" search service, from this point on any administrative actions resulting from Manage Search will be handled by whatever search service to which the load balancer has chosen to send the requests.

For example, suppose the search collection "XYZ" must be updated to introduce another content source. When the administrator selects that search service and its search collection, he has no control over what physical search service to which he is applying the changes. So even if such an environment could be set up initially, it is not easy to administer once it's in production.

One option to enable administration of the individual search services is to:

1. Use the administrative features of the network dispatcher to deactivate or disable all search services except the one to be maintained.
2. Perform the required administration steps for that specific search service.
3. Then move on to the next search service to perform the same steps over again.

NOTE: Another big hurdle would be the case in which multiple search services are required for one reason or the other, for example, for the purpose of having remote search services each dealing with specific subdomains in an intranet. However, that would introduce an additional layer of complexity to the overall design of the environment with respect to high availability and is beyond the scope of this paper.

2.3 Main search service coupled with fallback search service

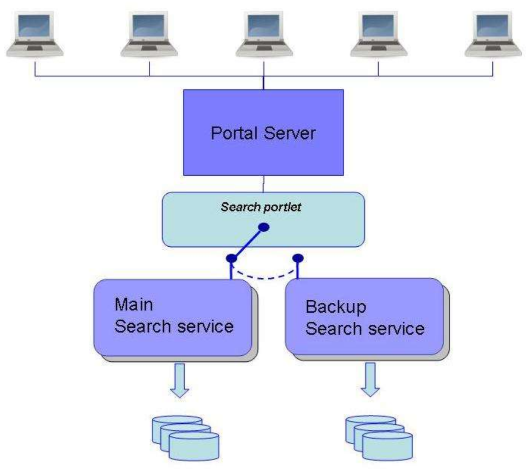
Here we describe a somewhat similar approach to that described in Section 2.2, except that this one allows for more transparency to the search administrator.

The first steps are identical to those above; that is, two or more remote search services are installed, and the search administrator creates and configures the search collections and content sources in one instance of the remote search service, which would be the "main" search service.

Then, once the search collections have been built, that information can be manually replicated to the other search services via the export/import utilities as described in Section 2.2.

The result is multiple search services running with identical sets of search collections and content sources (see figure 8).

Figure 8. Main search service with backup search service in place



However, instead of using a load balancer or network dispatcher to manage sending search requests to the search service, this logic will be implemented in a custom search portlet (for which ready-to-run sample code is provided in the Downloads section).

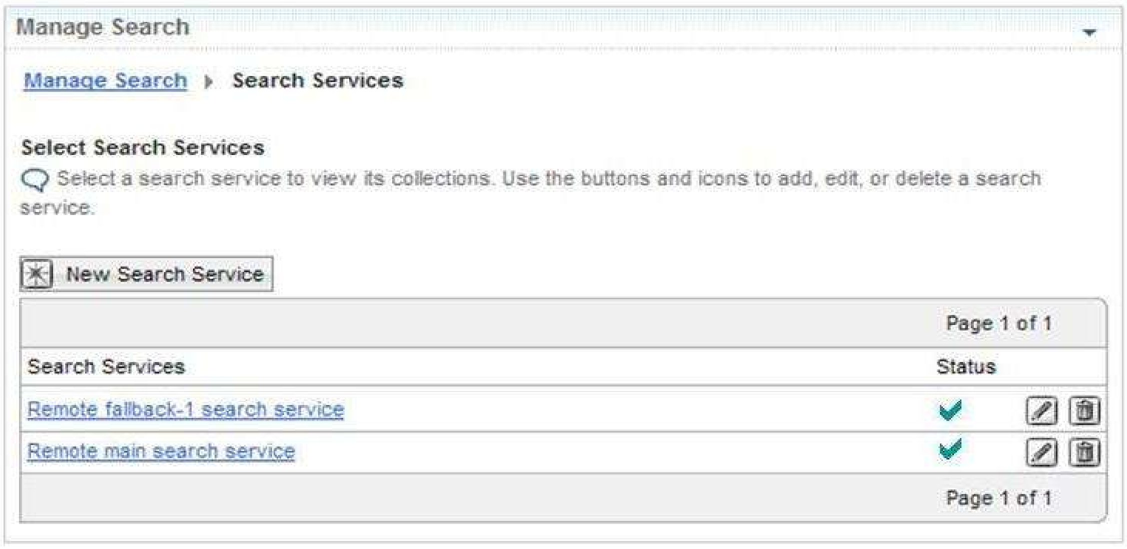
Of course, this nullifies the use of the out-of-the-box search portlets in this environment.

Here are the steps from above, with updates specific to this proposed setup:

1. First create and configure the two search services, in which one is the main search service, and the other is required only when the main service does not respond to search requests; in other words, the fallback service.

   Note that the two search services must be configured as “remote” search services (see “Configuring a remote search service”). Figure 9 shows the resultant Search Services view of the Manage Search portlet.

Figure 9. List of available search services



Manage Search

Manage Search ▸ Search Services

Select Search Services

Select a search service to view its collections. Use the buttons and icons to add, edit, or delete a search service.

New Search Service

Search Services	Status
Remote fallback-1 search service	✓
Remote main search service	✓

Page 1 of 1

2. Create the search collection and configure one or more crawlers (content sources) that fetch the content and feed them into the indexer:

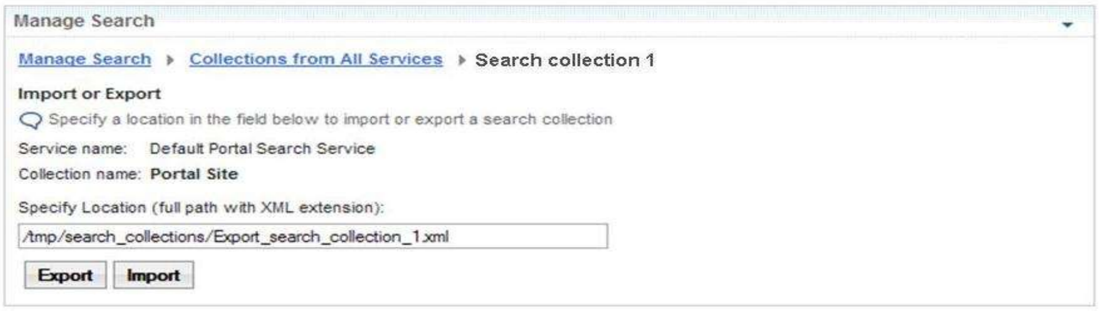
   - The crawlers can then be started manually to build the search collection(s) and run verification tests.

   - It's recommended to define crawler schedules, once the search collections have been successfully built.

3. Now copy the configuration information over to a second search service, that is, the “fallback” search service:

   a. First, export the search collection(s) as described in the topic, “Exporting and importing search collections” (see figure 10).

Figure 10. Exporting a search collection configuration


Manage Search

Manage Search ▸ Collections from All Services ▸ Search collection 1

Import or Export

Specify a location in the field below to import or export a search collection

Service name: Default Portal Search Service

Collection name: Portal Site

Specify Location (full path with XML extension):

/tmp/search_collections/Export_search_collection_1.xml

b. Select the second search service to create a search collection. Unlike the scenario described in the previous section, here the location and name of the search collection need not be the same. Also, it's not necessary to create the content sources (crawlers); these will be created as part of the “Importing the search collection” step.

c. Import the configuration information using the corresponding XML file created from the search collection of the first search service.

d. Finally, wait for the crawlers to be started at the next scheduled time, or invoke the crawler manually.

Again, it is advised to reconfigure the schedules of the content sources, to prevent multiple crawlers—from each of the individual search services—from hitting the servers or content repositories at the same time and causing unnecessary load on those servers.

The remaining step is to enable high availability of “search” using the infrastructure now set up. As stated in the introduction of this section, the logic must be delivered through the search portlet itself. We cannot, however, exploit this environment by using the out-of-the-box search portlets.

For instance, if you use the Search Center portlet, the search requests will return a list of duplicate entries, one entry from each of the search results from the two available search services and their respective search collection(s).

Thus we must introduce and develop a custom search portlet that (1) implements the necessary logic that selects what search collection to use from the list of available search collections and then (2) issues the search request against it to find the requested information.

As a first, straightforward example we assume that we have two search services, each hosting a single search collection. In a nutshell, that search portlet would need to implement the following:

- query all search services for a list of available search collections
- each of the available search services returns its search collection
- the search portlet by default selects the “main” search collection
- that search collection is then used to execute the search request and return the search result back to the requesting client
- if that search service cannot be reached, or an error is thrown when querying the search collection, then the second search service and its search collection are used to execute the query

This example demonstrates the basic logic of the processing required in the search portlet and how failover will work: If the default remote search service or its search collection is not available, the logic implemented in the search portlet will then select the second search service and its respective search collection to execute the search request and return the search result.

A variation of this setup could be to also allow for some level of load distribution among the available search services, but this is not covered diagrammatically in the sample search portlet included with this paper.

A rough description of the logic flow is:

- query all search services for a list of available search collections
- each of the available search services returns its search collection
- the search portlet then takes a random pick from the list of available search collections
- that search collection is then used to perform the search and return the search result back to the requesting client

Another dimension to this setup is the case when more than one search collection is required. For this to work efficiently we propose to either:

- adapt a naming scheme to the search collections so that similar search collections can be identified using similar names, for example, by adding remote service identifiers as suffix information to the search collection name.

OR

- select from the list of available search services, and then direct a federated search request against the set of its search collections.

2.4 Recommendations

Summing up, here are our recommendations for each of the three options described above:

✗**Clustered search services using load balancing.** This is the least desirable option. The drawbacks with respect to lack of product support for administering the search services and search collections outweighs by far the advantages of not requiring a new customized search portlet.

✓**High availability using RAID disks.** This option is appropriate in cases where failover support for failing hardware (hard disks) is of highest importance.

✓**Main search service plus failover search service(s).** This option is appropriate when more complete coverage for failure of a remote search service is of importance. Of course, combining it with “High availability using RAID disks” is a viable option as well.

- Recall that this option requires the additional effort of developing a custom search portlet, and that it's also helpful to apply appropriate naming schemes to the search collection(s), both for search administration purposes as well as for supporting the logic within the custom-built search portlet.)

3 Sample search portlet to demonstrate failover support

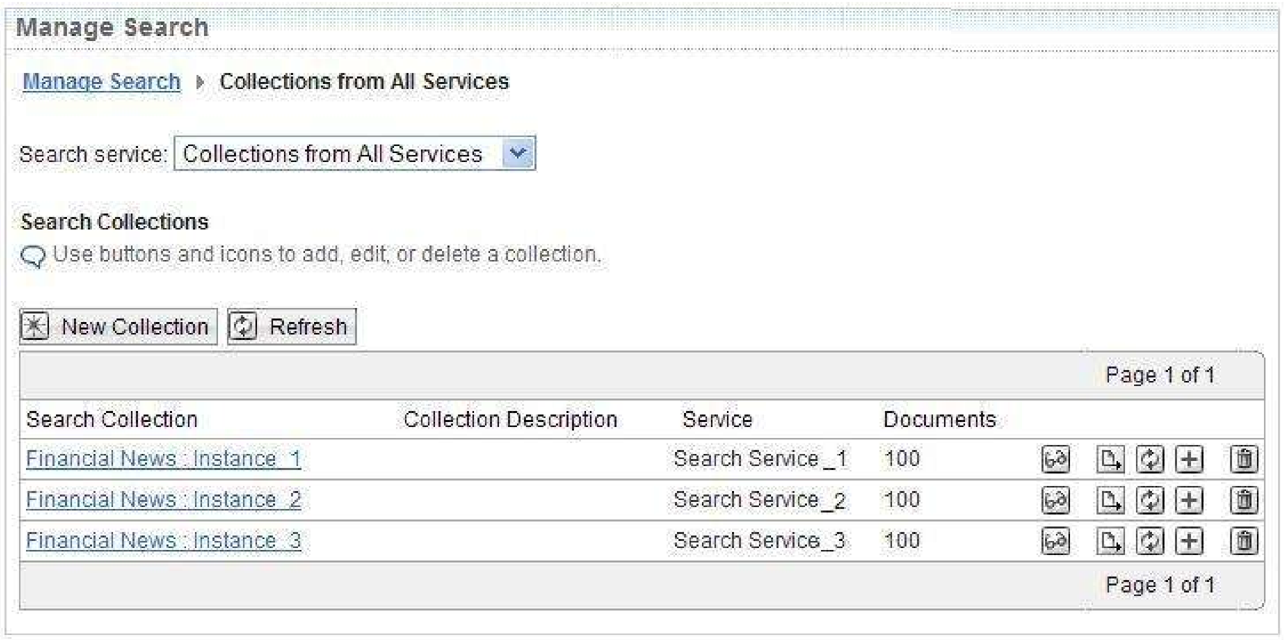
The sample search portlet in the Downloads section of this paper is based on the BasicSiapiPortlet package that was introduced in the developerWorks article, “Introducing the Search and Indexing

API in WebSphere Portal V6.0". This updated version demonstrates a coding example of how to enable failover support in a search portlet.

The original sample search portlet supports searching within a set of search collections and supports search scopes as well. For the purpose of failover support we focus only on searching within one or more search collections only. The basic concept implemented here assumes that search collections with the same name prefix are available as alternative collections when the default search collection is not available.

Figure 11 shows three search collections with the same name prefix "Financial News" hosted by three different search services (suffixed 1, 2, and 3). In the supplied code we use the special character ':' as the delimiter between the common name and the distinguishing suffix identifier 'Instance n'.

Figure 11. Search Collection from All Services



Manage Search

Manage Search ▸ Collections from All Services

Search service: Collections from All Services

Search Collections

Use buttons and icons to add, edit, or delete a collection.

New Collection | Refresh

Page 1 of 1

Search Collection	Collection Description	Service	Documents
Financial News : Instance 1		Search Service_1	100
Financial News : Instance 2		Search Service_2	100
Financial News : Instance 3		Search Service_3	100

Page 1 of 1

The intent of providing the sources for the search portlet is so that the user interface can be modified to suit the needs of users' search. For example, suppose we move the advanced search options to an advanced search form, and then provide the search collection name as a configuration parameter for the portlet, so that the user is not required to perform the selection.

Such a parameter is supported by the sample search portlet and is called "collection_prefix". If this parameter is defined, the portlet will look for the first collection with a matching prefix and will use it as the default collection. The relevant code can be found in Java™ source file BasicSiapiPortletSessionBean.java (see listing 1).

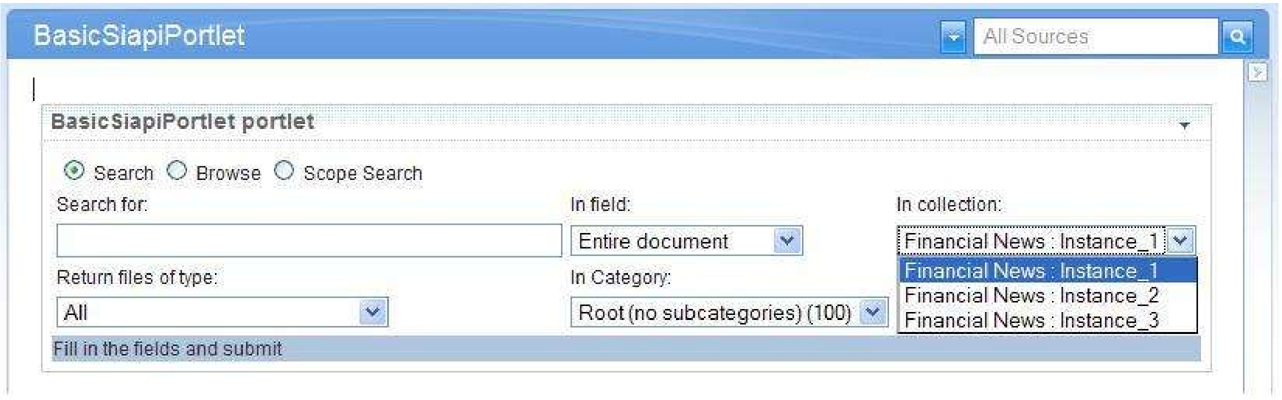
Listing 1. Code for default collection

```
/**
 * Read from the configuration the prefix name for the default collection and look for it.
 * Pick the first collection that matches the prefix.
 * @throws SiapiException if there was an error during the process
 */
public void resolveDefaultCollection(PortletRequest request, BasicSiapiPortletSessionBean sessionBean)
throws SiapiException {
        Searchable[] searchables = getAvailableSearchables();
        String defaultCollectionNamePrefix=request.getPortletSettings().getAttribute("collection_prefix");
        if(defaultCollectionNamePrefix != null) {
                defaultCollectionNamePrefix = defaultCollectionNamePrefix.trim();
                Searchable searchable = sessionBean.getMatching2PrefixCollection(
                        searchables, defaultCollectionNamePrefix);
                if(searchable != null) {
                        setCurrCollection(searchable);
                        return;
                }
        }
        setCurrCollection(searchables[0]);
}

/**
 * Pick the first collection that matches the name prefix
 * @param namePrefix which needs to match to one of the collections name
 * @return Searchable that represents the selected collection
 */
private Searchable getMatching2PrefixCollection(Searchable[] searchables , String namePrefix)
throws SiapiException {
        for (int i = 0; i < searchables.length; i++) {
                if (searchables[i].getCollectionInfo().getLabel().startsWith(namePrefix)) {
                        return searchables[i];
                }
        }
        return null;
}
```

For demonstration purposes the search portlet lists all three search collections from the three active search services (see figure 12). Selecting one of the search collections from the “In collection” drop-down list will direct the search requests to the respective search service and execute the query against that selected search collection.

Figure 12. Sample BasicSiapiPortlet search portlet


BasicSiapiPortlet
All Sources
BasicSiapiPortlet portlet
Search
Browse
Scope Search
Search for:
In field:
Entire document
In collection:
Financial News : Instance_1
Financial News : Instance_1
Financial News : Instance_2
Financial News : Instance_3
Return files of type:
All
In Category:
Root (no subcategories) (100)
Fill in the fields and submit

To confirm whether the implemented logic of the portlet actually supports failover scenarios, do the following:

1. Open a new browser window, log on to WebSphere Portal as an administrator, and go to the Manage Search page.

2. Delete the search collection that had been selected for the previous search in the first browser window.

3. Now go back to the browser window with the still-open search portlet and try the search again. Notice that the identical search result is returned, even though that chosen search collection has been deleted.

This scenario demonstrates the failover logic implemented within the *getFirstFailoverCollection()* method call, which is invoked as the result of the "search" method having thrown an exception because the selected search collection is now unavailable.

Now you could repeat the test, deleting the second collection as well, and see how the current opened search session is redirected to the third collection.

The required code for the demonstrated failover support is fairly simple and shown in the snippet in listing 2. This code can be found in the Java source file: *BasicSiapiPortletSessionBean.java*.

Listing 2. Code for failover support

```
/**
 * Executes the current query
 * @throws SiapiException if there was a Siapi error
 */
 private void executeQuery() throws SiapiException {
  ........
        try {
                // Execute the query
                resultSet = this.currSearchable.search(currQuery);
        } catch (SiapiException e) {
                //Switch to failover collection
                Searchable failoverSearchable = getFirstFailoverCollection(this.currSearchable);
                if(failoverSearchable != null) {
                        this.currSearchable = failoverSearchable;
                        executeQuery();
                        return;
                } else {
                        throw e;
                }
        }
  ........
 }

/**
 * List of collections that should not be used again for this session
 * because the search failed but they are still being returned as
 * available Searchables.
 */
 private List invalidSearchables = new ArrayList();

/**
 * Find the first failover collection for the current collection based on collections name prefix
 * @return the first failover collection or null if there is no such collection
 */
 private static final char seperator = ':';
 private Searchable getFirstFailoverCollection(Searchable currSearchable)
   throws SiapiException {
        String currSearchableLabel = currSearchable.getCollectionInfo().getLabel();
        String prefix = currSearchableLabel.substring(
                        0,currSearchableLabel.indexOf(seperator));
        Searchable[] searchables = getAvailableSearchables();
        for(int i=0; i<searchables.length; i++) {
                // Skip the collection itself
                String collecionID = searchables[i].getCollectionInfo().getID();
                if (0 == collecionID.compareTo(
                                currSearchable.getCollectionInfo().getID())) {
                        invalidSearchables.add(collecionID);
                        continue;
                }
                if(searchables[i].getCollectionInfo().getLabel().startsWith(prefix)) {
                        if(invalidSearchables.contains(searchables[i].getCollectionInfo().getID())) {
                                continue;
                        }
                        return searchables[i];
                }
        }
        return null;
}
```

The *getAvailableSearchables()* method is responsible for (1) retrieving a list of search collections that are identical in terms of having the same prefix (*getLabel() method*) and (2) looking for the first collection in the list—retrieved using the *getAvailableSearchables()* call—that starts with the same prefix as the current selected search collection.

If the current collection can be found in the available list, then mark it not for use. If the search fails again, continue to the next available search collection with the same matching prefix.

4 Wiring the custom search portlet with the search box in the WebSphere Portal theme

As already stated, if you elect to use the proposed solution to provide failover support of the Portal Search service, you cannot use the out-of-the-box Search Center portlet. By default the Search Center portlet queries all search collections managed by all search services (provided that the user is entitled to do so), so the result would be duplicate entries (one from each duplicate search collection).

For that reason it is necessary to decouple the Search Center from the search box in the theme and tie the search box to the newly created custom search portlet. The custom search portlet must support the "query" parameter, which will be passed from the search box in the theme.

The steps to do this for WebSphere Portal versions 6.0 and 6.1, respectively, are documented in the Lotus Support Technotes, "Directing the search box in the theme to a customized Search portlet in WebSphere Portal V6.0"and "Directing the search box in the theme to a customized Search portlet in WebSphere Portal V6.1".

The code for the search box of the default theme is in the file **banner_searchControl.jspf,** which can be found in <wp_profile>\installedApps\<node_name>\wps.ear\wps.war\themes\html \<theme_name>.

In short, the steps are as follows:

1. Create a WebSphere Portal page with the search portlet in it.

2. Check all occurrences of the <*portal-navigation:urlGeneration*> tag in the above-mentioned file; the value of the **contentNode** attribute should contain the unique name of the new page.

3. Then replace the value of the **layoutNode** attribute with the unique name of your new portlet window.

5 Conclusion

High availability of Portal Search can be achieved in one of two ways. The first is to ensure failover support of the underlying file system, using RAID disk devices, which provide failover support when a hard disk fails but not the server or services themselves.

The second option is to set up identical remote search services on different servers, each hosting the same set of search collections. A custom search portlet (the source provided as a sample with this article) then takes care of sending the search request to an available search service, and that transparent to an end user.

If more sophisticated support is required, then you might consider enterprise search software like IBM OmniFind Enterprise Edition, which has high availability built into the system.

6 Resources

developerWorks WebSphere Portal product page:
http://www.ibm.com/developerworks/websphere/

developerWorks WebSphere Portal zone:
http://www.ibm.com/developerworks/websphere/zones/portal/

WebSphere Portal and Lotus Web Content Management Product documentation:
http://www.ibm.com/developerworks/websphere/zones/portal/proddoc.html

WebSphere Portal Family wiki:
http://www-10.lotus.com/ldd/portalwiki.nsf

WebSphere Portal 6.1 information center:
http://publib.boulder.ibm.com/infocenter/wpdoc/v6r1m0/index.jsp

7 About the authors

Andreas Prokoph is a software architect at the IBM Development Lab in Boeblingen, Germany, working in the field of text search and information retrieval for the past 18 years. He has held various positions as technical lead and architect for many search products and solutions, ranging from intranet search engines to client-side embedded search technologies. You can reach Andreas at pkp@de.ibm.com.

Eitan Shapiro holds a BSc degree in Information Systems Engineering from the Technion, Haifa, Israel. He joined IBM in 2005 and is the Team Lead of the Enterprise Information Discovery team, which develops search solutions for WebSphere Portal and Lotus® Quickr™.