



Questions and Answers  
Webinar: Verity Ultraseek XPA Overview  
March 2, 2005

Q: Does the logging information show more detail when `IndexerAdmin.parse()` fails and throws an exception?

A: The usual way `IndexerAdmin.parse()` fails is when the remote Ultraseek server has a problem parsing the document. The XPA diagnostic log will not show any additional information – however, you can look in the Ultraseek server's `error.log` to see the server's problem report.

Q: Are the samples hooks, or how is the adapter via Inheritance visible?

A: The sample files are full source files. `SearchServlet`, for example, is a fully functional search page which gathers search results from an Ultraseek server. Additional sample files (`MyServlet`, `LoginServlet`, `SSOSearchServlet`, etc.) demonstrate how to customize the behavior of `SearchServlet` using Inheritance. We recommend this approach so your application (or customizations) will have an easier migration path for future versions of XPA.

Q: If we need to treat [a `SearchResultList`] as [a] cursor, does that also mean it is as resource intensive?

A: Yes, `SearchResultList` should be considered a resource intensive object. Background threads will try to keep more `SearchResults` in the list so your application isn't blocked waiting for results from the remote search server. However, the resources required should be considered roughly the same as using a cursor on a SQL request to a database server.

Q: Can certain URLs be configured to go to different backend Ultraseeks for load balancing

A: The recommended approach to load balancing is to use a `MirrorGroup` to balance all your Web containers (`SearchServlet`) among your Ultraseek servers. `MirrorGroup` allocates load based on the search query – which results in better performance as the



Ultraseek server is more likely to have the result already cached.

Q: We are new to Ultraseek. Everything we see seems to be focused towards documents within our site. Can we use a set of web services as sources to Ultraseek?

A: You are correct that Ultraseek's focus is directing users towards documents that match their queries. Ultraseek finds these documents either via an HTTP Spider, a disk scan, an Exchange Server, a NetNews server, or a Database. If you have a different repository for your documents, you must write an XPA application that fetches the documents and feeds them into Ultraseek. See the sample file FileScanner.java for an example of how to do this.

Q: Kerberos?

A: XPA has no specific support of Kerberos. The XPA security approach is done via a Proxy of an HTTP request. This approach is compatible with Kerberos, LiveLink, Netegrity, and many other security providers. But we do not make sure of any specific Kerberos security features.

Q: How do you track click-throughs using XPA and view those reports in the Admin Server Interface?

A: Well, something had to get kicked out of XPA 2.2 – and that was the feature that didn't make it. The next XPA release will have an interface similar to the Query logging feature for tracking click through using the Ultraseek admin interface.

Q: What about the 500 search results limit with HttpAuthentication?

A: The 500 search result limit is still present. The Ultraseek server will return the top 500 relevant results – and your XPA application will filter the unauthorized results. This approach works fine if 90% of the results are “authorized” or “public”. But if only 10% of the results are “authorized” you find the ability to search is severely constrained (as well as having poor performance).

For secure setups where “most results” are “unauthorized” – you need to integrate the search engine more tightly with your security infrastructure. Ways you can do this include:



- Put secure results in different collections (Marketing, Legal, Development) – and constrain the search to the collections the user is authorized to view.
- When indexing documents, also index a tag to indicate the security class of the document. (For instance: security:marketing, security:development). When performing a search, include a SubsetSearchable which specifies the keywords the current user is authorized to view.
- Both of these approaches involve fairly intimate knowledge of your site's security environment. Contract Professional Services for additional information on customizing a solution.

Q: Can XPA support a guided browsing UI (like K2)?

A: No.

Q: Does XPA handle revisit queuing? Specifically, if you're doing some custom indexing like the last use case mentioned, will revisit be automagically handled, or do we have to code that?

A: There are two basic approaches to collecting documents with Ultraseek.

- 1) Use a Spider, Scanner, Database, Exchange, or NetNews collection. Ultraseek determines the revisit interval.
- 2) Use an XPA Direct Indexer collection. Your application determines the revisit interval.

Either approach can also use the ContentAssistant to add additional information to a document when it is being indexed.

If you want Ultraseek to manage revisiting – the better approach is to use an Spider (eg.) collection with a ContentAssistant application to perform custom processing on the document during indexing.

Q: We used XPA2.2 with Ultraseek 5.3 and experienced intermittence problems with result sets not coming back. Has anyone else experienced this?

A: I'm not aware of any problems in this regard. Please contact technical support with details.

Q: Hi. For hit level authentication, can you explain how cookies are used/passed to XPA, onto Ultraseek, and used against the web/app server for yes/no return.



A: First off, your search application (eg: SSOSearchServlet) must be deployed at a URL within your domain so that the user's browser will automatically send the cookie as part of the HTTP search request.

SSOSearchServlet then ignores the cookie, and makes the search request to Ultraseek.

For each result that comes back from Ultraseek, SSOSearchServlet/XPA then tries to fetch the document using the Cookie that came from the user's browser. From this we get an "authorized / unauthorized" decision on each URL.

Q: Does the SSO Java Authorization support NTLM?

A: The implementation is very general – so we make no use of NTLM specific features. It will work under NTML, but is not optimized for NTLM.

Q: What information is logged besides Search term when using User Query Logging

A: Timestamp, number of results, collections being searched, and the search query.

Q: Can I create customized searches of XML documents?

A: Ultraseek is not a general XML query engine. You can search for specific terms within an XML document (example: Oak Street), but not for a specific term within a specific XML context (example: Oak Street in the Address element for a Customer Element).

Q: If a record is not returned in the first 500 results that a user can see, are no results returned?

A: That is correct. See earlier answer on making a tight integration between Ultraseek and your site's security infrastructure.

Q: Is there sample/example for SSO integration?

A: Yes, see the sample file SSOSearchServlet.java



Q: Can we use Site Minder on top of LDAP for Secure Search?

A: For XPA, the use of LDAP is independent of any solution. I assume you want to use LDAP as your authentication warehouse. XPA assume the user has already been authenticated – and just passes their credentials around to see what documents are viewable.

Q: We are using IBM WebSphere Commerce for our shopping sites. If we build a custom java app to get at and spider our commerce data do we provide URLs to Ultraseek or raw data (products, categories, etc)

A: You can use either approach. What Ultraseek wants to do is provide (in response to a search) A URL, a Title for the document at that URL. A description of the Document. Size, etc.

So, you can either give Ultraseek a URL and have it fetch the document and figure out what data to index – or you can give Ultraseek a URL, and the data to index. The second approach is often better if you have “unindexable” data which is served with the URL when it is fetched on your site. (Example: Ads, navigation, copyright boilerplate, review comments from people, etc.). Ultraseek filters out much of this text – but if you’ve already separated it then indexing can be faster and better.

Q: Can you talk in more detail about the nested search example? We filter our result list based on the region and locale that is in the url of the search page. Can we pass that info to the query and have Ultraseek do the filtering for us?

A: Sure! Let’s suppose you have two Search URLs  
<http://www.company.us/Search> and <http://www.company.fr/Search>  
You can automatically add a search filter to the query in order to subset the Documents being searched. For instance,  
Language:en || user’s query  
For the first URL, and  
Language:fr || user’s query  
For the second.

See the Sample file LocaleSearch for an example.



Q: What versions of ZIP files does Ultraseek support via IndexerAdmin.parse()?

A: General ZIP format. We haven't tested files generated from any specific programs.

Q: Could you talk more about your custom indexing as well as custom data sources, specifically SQL databases?

A: I think I've covered several approaches to this in answers to other questions.

Q: I need to develop a web based catalog system. i have some data in a database (id and title) and then related information in PDF documents (objectives and prerequisites). Would it be sensible/possible to write a Java app that searches the database and parses the PDFs to create a searchable collection?

A: Sure! Roughly speaking, the approach is as follows.  
Create a SQL query to access the data you want to index.  
(id, title, PDF\_content). Feed that data into Ultraseek using IndexerAdmin.insert().  
For this example, title becomes the title parameter, PDF\_content becomes  
The content parameter, "application/pdf" is the Mime type, and id  
Is likely used to generate the URL for the document.

Q: When verifying that a user has permission to access a document, where does the access control list need to be defined? We use Netegrity's SiteMinder to protect our pages and it stores authorization information in an LDAP. Can XPA interact with Netegrity

A: XPA isn't optimized for Netegrity – is does a simple Proxy request to determine accessibility.

Q: Does hit level authentication work with https: sites?

A: Yes., provided you have an HTTPS client installed on your application server.