Oracle® Ultra Search

Administrator's Guide

10*g* Release 2 (10.1.2)

Part No. B14041-01

November 2004

ORACLE®

Oracle Ultra Search Administrator's Guide 10*g* Release 2 (10.1.2)

Part No. B14041-01

Primary Author: Michele Cyran

Contributors: Sandeepan Banerjee, Stefan Buchta, Chung-Ho Chen, Will Chin, Jack Chung, Ray Hachem, Cindy Hsin, Caroline Johnston, Hassan Karraby, Yasuhiro Matsuda, Colin McGregor, Valarie Moore, Visar Nimani, Steve Yang

Contents

2 Getting Started with Oracle Ultra Search

3 Installing Oracle Ultra Search

4 Using Oracle Ultra Search with Oracle Application Server

5 Oracle Ultra Search Postinstallation Information

6 Security in Oracle Ultra Search

7 Understanding the Oracle Ultra Search Crawler and Data Sources

8 Understanding the Oracle Ultra Search Administration Tool

9 Oracle Ultra Search Developer's Guide and API Reference

10 Tuning and Performance in Oracle Ultra Search

11 Oracle Ultra Search Administration PL/SQL APIs

Send Us Your Comments

Oracle Ultra Search Administrator's Guide 10*g* Release 2 (10.1.2)

Part No. B14041-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com
- FAX: (650) 506-7375 Attn: Oracle Application Server Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle Application Server Documentation
  500 Oracle Parkway, M/S 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This Preface contains these topics:

Audience

Oracle Ultra Search Administrator's Guide is intended for database administrators and application developers who perform the following tasks:

- Install and configure Oracle Ultra Search
- Administer Oracle Ultra Search instances
- Develop Oracle Ultra Search applications

To use this document, you should have experience with the Oracle database management system, SQL, SQL*Plus, and PL/SQL.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

This document contains:

"What's New in Oracle Ultra Search?"

This section describes new features and provides pointers to additional information.

Chapter 1, "Introduction to Oracle Ultra Search"

This chapter provides an overview of Oracle Ultra Search and describes the system configuration.

Chapter 2, "Getting Started with Oracle Ultra Search"

This chapter provides an example scenario that shows installation and use of Oracle Ultra Search.

Chapter 3, "Installing Oracle Ultra Search"

This chapter describes how to install Oracle Ultra Search. Some information in this chapter is generic to all types of Oracle Ultra Search installations. Other information in this chapter is specific to installing and configuring Oracle Ultra Search with the Oracle Database release.

If you are installing Oracle Ultra Search with the Oracle Application Server release, then you should also read Chapter 4, "Using Oracle Ultra Search with Oracle Application Server".

Chapter 4, "Using Oracle Ultra Search with Oracle Application Server"

This chapter contains information specific to installing and configuring Oracle Ultra Search with the OracleAS release.

Chapter 5, "Oracle Ultra Search Postinstallation Information"

This chapter provides postinstallation information, such as how to configure the Oracle Database server for Oracle Ultra Search and how to manage stoplists. It also describes how to upgrade to the most recent Oracle Ultra Search release.

Chapter 6, "Security in Oracle Ultra Search"

This chapter describes the architecture and configuration of security for Oracle Ultra Search.

Chapter 7, "Understanding the Oracle Ultra Search Crawler and Data Sources"

This chapter explains how the crawler works. It also describes crawler settings, data sources, document attributes, data synchronization, and the remote crawler.

Chapter 8, "Understanding the Oracle Ultra Search Administration Tool"

This chapter describes how to use the Oracle Ultra Search administration tool to configure and schedule the Oracle Ultra Search crawler.

Chapter 9, "Oracle Ultra Search Developer's Guide and API Reference"

This chapter explains the following Oracle Ultra Search APIs: query API, crawler agent API, e-mail API, URL rewriter API, and the document service API. It also provides related details about the query applications, the query tag library, and query syntax expansion customization.

Chapter 10, "Tuning and Performance in Oracle Ultra Search"

This chapter describes various ways to tune Oracle Ultra Search and improve performance. These include tuning the Web crawling process, tuning query performance, using the remote crawler, using Oracle Ultra Search on Real Application Clusters, and table data source synchronization.

Chapter 11, "Oracle Ultra Search Administration PL/SQL APIs"

This chapter details some of Oracle Ultra Search's PL/SQL APIs for administration, including those for crawler configuration, crawler scheduling, and instance administration.

Appendix A, "Loading Metadata into Oracle Ultra Search"

This appendix describes the command-line tool for loading metadata into an Oracle Ultra Search database.

Appendix B, "Altering the Crawler Java Classpath"

This appendix explains why and how to alter the crawler Java classpath.

Appendix C, "Oracle Ultra Search Views"

This appendix shows the various views available with Oracle Ultra Search.

Appendix D, "URL Crawler Status Codes"

This appendix lists the codes that the crawler uses to indicate the result of the crawled URL.

Related Documentation

For more information, see these Oracle resources:

- *Oracle Database Concepts*
- *Oracle Database Administrator's Guide*
- *Oracle Database Performance Tuning Guide*
- *Oracle Enterprise Manager Concepts*

Many books in the documentation set use the sample schemas of the seed database, which is installed by default when you install Oracle Database. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network. You must register online before using this; registration is free and can be done at

`http://www.oracle.com/technology/membership/`

If you already have a user name and password, then you can go directly to the documentation section at

http://www.oracle.com/technology/documentation/index.htm

To access the database documentation search engine directly, visit

http://tahiti.oracle.com/

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names and connect identifiers, user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. *Note:* Some programmatic elements use a mixture of `UPPERCASE` and `lowercase`. Enter these elements as shown.	Enter `sqlplus` to start SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.

Convention	Meaning	Example
lowercase italic monospace (fixed-width) font	Lowercase italic monospace font represents placeholders or variables.	You can specify the *parallel_clause*. Run *old_release*.SQL where *old_release* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Anything enclosed in brackets is optional.	DECIMAL (*digits* [, *precision*])
{ }	Braces are used for grouping items.	{ENABLE | DISABLE}
|	A vertical bar represents a choice of two options.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	Ellipsis points mean repetition in syntax descriptions.	CREATE TABLE ... AS *subquery*;
	In addition, ellipsis points can mean an omission in code examples or text.	SELECT *col1*, *col2*, ... , *coln* FROM employees;
Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
lowercase	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose **Start** > ***menu item***	How to start a program.	To start the Database Configuration Assistant, choose **Start** > **Programs** > **Oracle** - ***HOME_NAME*** > **Configuration and Migration Tools** > **Database Configuration Assistant**.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the filename begins with \ \, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32` is the same as `C:\WINNT\SYSTEM32`
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp HR/HR TABLES=employees QUERY=\"WHERE job_id='SA_REP' and salary<8000\"`
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle database version 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle\product\10.1.0`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\product\10.1.0\db_`*n*, where *n* is the latest Oracle home number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Installation Guide for 32-Bit Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

What's New in Oracle Ultra Search?

This section describes Oracle Ultra Search new features, with pointers to additional information. It also explains the Oracle Ultra Search release history.

Document Service API

The Document Service crawler agent API allows generation of attribute data based on document contents. It accepts robot metatag instructions from the agent for the target document, and it transforms the original document contents for indexing control.

See Also: "Oracle Ultra Search Document Service API" on page 9-26

Keyword in Context

The query application has been designed to showcase keyword in context and highlighting features. Keyword in context shows a section of the original document that contains the search terms. Highlighting shows the entire document with the search terms in a different color.

See Also: "Oracle Ultra Search Query Applications" on page 9-29

Secure Crawling

Oracle Ultra Search provides secure crawling using the following types of authentication:

Digest Authentication Oracle Ultra Search supports HTTP digest authentication, and the Oracle Ultra Search crawler can authenticate itself to Web servers employing HTTP digest authentication scheme. This is based on a simple challenge-response paradigm; however, the password is encrypted.

HTML Form Authentication HTML form-based authentication is the most commonly used authentication scheme on the Web. Oracle Ultra Search lets you register HTML forms that you want the Oracle Ultra Search crawler to automatically fill in during Web crawling. HTML form authentication requires that HTTP cookie functionality is enabled, which is the default.

See Also: "Creating Web Sources" on page 8-18

Indexing Control of Dynamically Generated Web Pages

The crawler can be configured to not index Web pages that are dynamically generated (for example, if a URL contains a question mark).

See Also: "Creating Web Sources" on page 8-18

HTTPS

Oracle Ultra Search supports secure socket layer (SSL). This means that in addition to HTTP-based URLs, Oracle Ultra Search can also access HTTPS-based URLs (that is, HTTP over SSL).

> **See Also:** "Oracle Ultra Search with Secure Socket Layer and HTTPS" on page 6-2 and "Creating Web Sources" on page 8-18

Secure Search

Secure searches return only documents that the search user is allowed to view. Each indexed document can be protected by an access control list (ACL). During searches, the ACL is evaluated. If the user performing the search has permission to read the protected document, then the document is returned by the query API. Otherwise, it is not returned.

Oracle Ultra Search stores ACLs in the Oracle XML DB repository. Oracle Ultra Search also uses Oracle XML DB functionality to evaluate ACLs.

> **See Also:** "Secure Search" on page 1-7

Remote Crawler JDBC Caching Support

It is now possible to use the remote crawler without mounting the remote cache directory to the server machine. Instead, the cache files are sent over the crawler's JDBC connection to the server cache directory.

> **See Also:** "JDBC-Based Remote Crawling" on page 10-5 and "Remote Crawler Profiles" on page 8-12

Manual Launch Scheduling

A schedule can be created with no scheduled launch time, so that it can only be started on demand.

> **See Also:** "Data Synchronization" on page 8-27

Crawler Log File Versioning

For each data source, the crawler preserves the latest three log files. This avoids erasing the previous crawling log file on recrawl.

> **See Also:**
>
> "Crawler Logging" on page 11

New PL/SQL Administration APIs

Oracle Ultra Search includes APIs for various administration tasks, such as crawler, schedule, and instance administration.

> **See Also:** Chapter 11, "Oracle Ultra Search Administration PL/SQL APIs"

Integration with Oracle Internet Directory

Oracle Internet Directory is Oracle's native LDAP v3-compliant directory service, built as an application on top of the Oracle Database. Oracle Ultra Search integrates with Oracle Internet Directory in the following areas:

- Oracle Ultra Search administration groups and group membership are stored in Oracle Internet Directory.
- Users are authenticated through Oracle Application Server Single Sign-On and Oracle Internet Directory.
- Oracle Internet Directory performs authorization on Oracle Ultra Search users' administration privileges.

See Also: "Integration with Oracle Internet Directory" on page 1-10

Cookie Support

Cookies remember context between HTTP requests. For example, the server can send a cookie such that it knows if a user has already logged on and does not need to log on again. Cookie support is enabled by default.

Crawler Cache Deletion Control

During crawling, documents are stored in the cache directory. Every time the preset size is reached, crawling stops and indexing starts. In previous releases, the cache file was always deleted when indexing was done. You can now specify *not* to delete the cache file when indexing is done. This option applies to all data sources. The default is to delete the cache file after indexing.

See Also: "Crawler Page" on page 8-9

URL Boundary Rules Include Port Number Inclusion or Exclusion

You can set URL boundary rules to refine the crawling space. You can now include or exclude Web sites with a specific port. For example, you can include `www.oracle.com` but not `www.oracle.com:8080`. By default, all ports are crawled.

See Also: "Creating Web Sources" on page 8-18

Hostname Prefix Allowed in Web Data Source URL Boundary Specification

In previous releases, you could only specify suffix inclusion rules. For example, crawl only URLs ending with `oracle.com`. You can now also specify prefix rules. For example, crawl `oracle.com` but not `stores.oracle.com`.

See Also: "Creating Web Sources" on page 8-18

Default Oracle Ultra Search Instance and Schema

Oracle Ultra Search automatically creates a default Oracle Ultra Search instance based on the default Oracle Ultra Search test user. So, you can test Oracle Ultra Search functionality based on the default instance after installation.

See Also: "Unlock WK_TEST" on page 3-3

Monitoring Oracle Ultra Search Components with Oracle Enterprise Manager

You can use Enterprise Manager's Grid Control to monitor Oracle Ultra Search components. Using Grid Control, you can set up notification rules to send out e-mail notification automatically whenever a schedule status reaches certain severity states.

See Also: *Oracle Enterprise Manager Concepts* guide for more information on the using Grid Control to monitor Oracle Ultra Search components

Crawler Recrawl Policy

You can update the recrawl policy to process documents that have changed or to process all documents.

In previous releases, processing all documents did not help when the crawling scope had been narrowed. For example, if crawling depth was reduced from seven to five, the PDF mimetype was deleted, or a host inclusion rule was removed, then the affected documents would have to be removed manually in a SQL*Plus session.

With this release, all crawled URLs are subject to crawler setting enforcement, not just newly crawled URLs.

See Also: "Editing Synchronization Schedules" on page 8-28

Federated Search

Traditionally, Oracle Ultra Search used centralized search to gather data on a regular basis and update one index that cataloged all searchable data. This provided fast searching, but it required the data source be made crawlable before it could be searched. Oracle Ultra Search now also provides federated search, which allows multiple indexes to perform a single search. Each index can be maintained separately. By querying the data source at search-time, search results are always the latest results. User credentials can be passed to the data source and authenticated by the data source itself. Queries can be processed efficiently using the data's native format.

To use federated search, you must deploy an Oracle Ultra Search search adapter, or searchlet, and create an Oracle Database source. A searchlet is a Java module deployed in the middle tier (inside OC4J) that searches the data in an enterprise information system on behalf of a user. Every searchlet is a JCA 1.0 compliant resource adapter.

See Also: "Federated Sources" on page 8-24

Oracle Ultra Search Release Information

Oracle Ultra Search is released with the Oracle Database, Oracle Application Server, and Oracle Collaboration Suite. Previously, release numbers varied by Oracle product, and Oracle Ultra Search took its version number from the vehicle in which it is packaged. Therefore, later version numbers may actually be earlier versions of the product. For example, Oracle Ultra Search 9.2.0 is an older release than Oracle Ultra Search 9.0.4.

The following table shows the Oracle Ultra Search versions in increasing order:

Oracle Ultra Search Version	Release Vehicle
Oracle Ultra Search release 8.1.7	Oracle database version 8.1.7
Oracle Ultra Search release 9.0.1	Oracle9*i* release 1 (9.0.1)
Oracle Ultra Search release 9.0.2	Oracle9*i*AS release 2 (9.0.2)
Oracle Ultra Search release 9.2	Oracle9*i* release 9.2
Oracle Ultra Search release 9.0.3	Oracle Collaboration Suite 9.0.3
Oracle Ultra Search release 9.0.4	Oracle Application Server release 10*g* (9.0.4)
Oracle Ultra Search release 10.1	Oracle Database 10*g* and Oracle Application Server 10*g*

Note: Beginning with release 10*g*, the Oracle Ultra Search backend version is always the same as the database version.

For OracleAS releases, the Oracle Ultra Search version number is the same as the version of the database used in the Metadata Repository. However, there is an exception to this: If your Metadata Repository is Oracle9*i* release 9.2, then you will have Oracle Ultra Search release 9.0.4.

1

Introduction to Oracle Ultra Search

This chapter contains the following topics:

Overview of Oracle Ultra Search

Oracle Ultra Search is built on the Oracle Database and Oracle Text technology that provides uniform search-and-locate capabilities over multiple repositories: Oracle databases, other ODBC compliant databases, IMAP mail servers, HTML documents served up by a Web server, files on disk, and more.

Oracle Ultra Search uses a 'crawler' to collect documents. You can schedule the crawler to suit the Web sites that you want to search. The documents stay in their own repositories, and the collected information is used to build an index that stays within your firewall in a designated Oracle database. Oracle Ultra Search also provides APIs for building content management solutions.

In addition, Oracle Ultra Search offers the following:

- A complete text query language for text search inside the database
- Full integration with the Oracle Database and the SQL query language
- Advanced features like concept searching and theme analysis
- Attribute mapping to facilitate attribute search across disparate repositories
- Indexing of more than 150 file formats
- Full globalization, including support for Chinese, Japanese, and Korean (CJK), and Unicode

Oracle Ultra Search Components

Oracle Ultra Search is made up of the following components:

Oracle Ultra Search Crawler

The Oracle Ultra Search crawler is a Java process activated by your Oracle server according to a set schedule. When activated, the crawler spawns a configurable number of processor threads that fetch documents from various data sources and index them using Oracle Text. This index is used for querying. Data sources can be Web sites, database tables, files, mailing lists, Oracle Application Server Portal page groups, or user-defined data sources.

The crawler maps links and analyzes relationships. The crawler schedule is integrated with and driven by the `DBMS_JOB` queue mechanism. Whenever the crawler encounters embedded, non-HTML documents during the crawling, it uses Oracle Text filters to automatically detect the document type and to filter and index the document.

> **See Also:** Chapter 7, "Understanding the Oracle Ultra Search Crawler and Data Sources"

Oracle Ultra Search Backend

The Oracle Ultra Search backend consists of an Oracle Ultra Search repository and Oracle Text. Oracle Text provides the text indexing and search capabilities required to index and query data retrieved from your data sources. The backend is not visible to users; it indexes information from the crawler and serves up the query results.

> **See Also:** "Installing the Oracle Ultra Search Backend" on page 3-2 or "Oracle Ultra Search Backend with Oracle Application Server" on page 4-1

Oracle Ultra Search Middle Tier

The Oracle Ultra Search middle tier components are Web applications. The middle tier includes the Oracle Ultra Search administration tool, the APIs and the query applications.

In the Oracle Database release, the Oracle Ultra Search middle tier and backend can reside in the same Oracle home. However, in the OracleAS and Oracle Collaboration Suite releases, the middle tier is located in a different Oracle home than the backend.

Oracle Ultra Search Administration Tool

The administration tool is a J2EE-compliant Web application. You can use it to manage Oracle Ultra Search instances, and you can access it from any browser in your intranet. The administration tool is independent from the Oracle Ultra Search query application. Therefore, the administration tool and query application can be hosted on different computers to enhance security and scalability.

> **See Also:** Chapter 8, "Understanding the Oracle Ultra Search Administration Tool"

Oracle Ultra Search APIs and Query Applications

Oracle Ultra Search provides the following APIs:

- The query API works with indexed data. The Java API does not impose any HTML rendering elements. The application can completely customize the HTML interface.
- The crawler agent API crawls and indexes proprietary document repositories.

- The e-mail Java API accesses archived e-mails and is used by the query application to display e-mails. It can also be used when building your own custom query application.
- The URL rewriter API is used by the crawler to filter and rewrite extracted URL links before they are inserted into the URL queue.
- The Document Service crawler agent API allows generation of attribute data based on the document contents. It accepts robot metatag instructions from the agent for the target document, and it transforms the original document contents for indexing control.

Oracle Ultra Search includes highly functional query applications to query and display search results. The query applications are based on JSP and work with any JSP1.1 compliant engine.

See Also:

- Chapter 9, "Oracle Ultra Search Developer's Guide and API Reference"
- *Oracle Ultra Search API Reference*

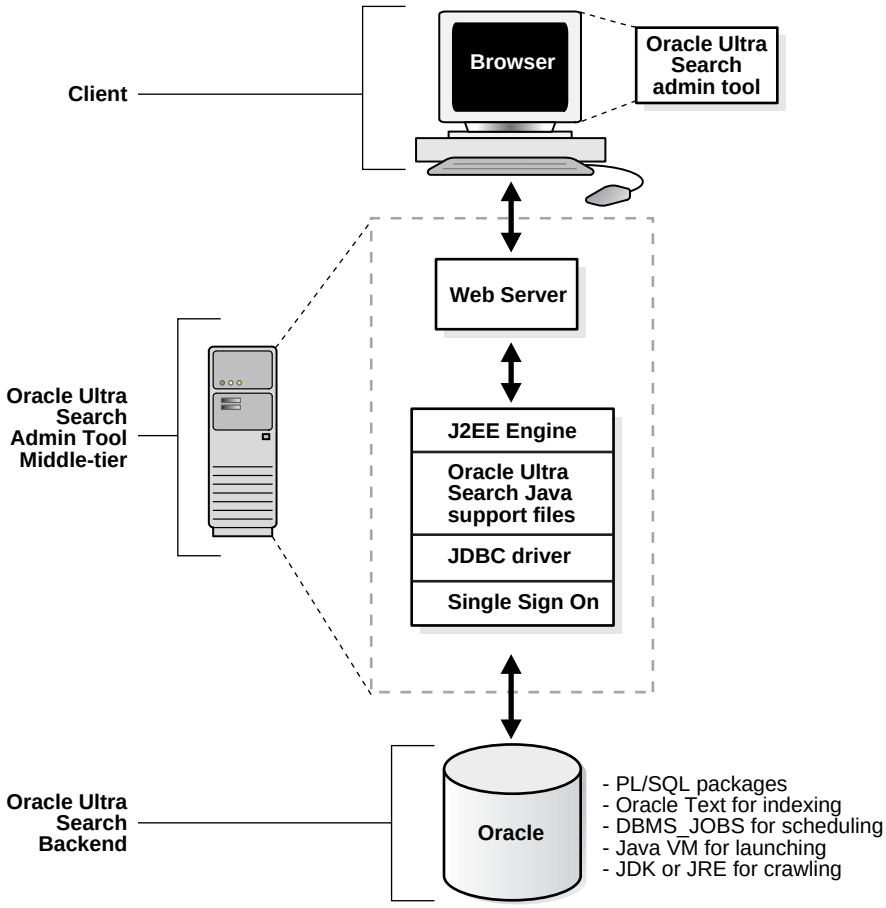
Web Application Concepts

The Oracle Ultra Search administration tool and the Oracle Ultra Search query applications are J2EE-compliant Web applications. These are three tier architecture applications. Figure 1–1 shows the relationship between the browser (the first tier), the Web server and the servlet engine (the middle tier), and the Oracle Database (the third tier).

The Web server accepts requests from the browser and forwards the requests to the servlet engine for processing. The Oracle Ultra Search middle tier then communicates with the Oracle database through the JDBC, as in Figure 1–1.

You can use any browser to access the Oracle Ultra Search administration tool or Oracle Ultra Search query application. The URLs are described in the following section.

Figure 1–1 Oracle Ultra Search Architecture



Oracle Ultra Search Features

This section explains some features in Oracle Ultra Search. It includes the following topics:

- Oracle Ultra Search Instance
- Document and Search Attributes
- Metadata Loader
- Internationalization in Oracle Ultra Search
- Oracle Ultra Search Crawler Features
- Query API
- Secure Search
- Document Relevancy Boosting
- Query Syntax Expansion
- Display URL Support
- Federated Search
- Single Sign-On Authentication

- Integration with Oracle Internet Directory
- Query Applications
- Integration with Oracle Application Server

Oracle Ultra Search Instance

An Ultra Search instance can be created to provide the isolation for the data collections that have been crawled.

You can create a read-only snapshot of a master Oracle Ultra Search instance. This is useful for query processing or for a backup. You can also make a snapshot instance updatable. This is useful when the master instance is corrupted and you want to use a snapshot as a new master instance.

See Also: "Instances Page" on page 8-5

Document and Search Attributes

Document attributes, or metadata, describe the properties of a document. Each data source has its own set of document attributes. The value is retrieved during the crawling process and then mapped to one of the search attributes and stored and indexed in the database. This lets you query documents based on their attributes. Document attributes in different data sources can be mapped to the same search attribute. Therefore, you can query documents from multiple data sources based on the same search attribute.

Oracle Ultra Search has the following default search attributes: Title, Author, Description, Subject, Mimetype, Language, Host, and LastModifedDate. They can be incorporated in search applications for a more detailed search and richer presentation. The list of values (LOV) for a search attribute can help you specify a search query. If attribute LOV is available, then the crawler registers the LOV definition, which includes attribute value, attribute value display name, and its translation.

See Also: "Synchronizing Data Sources" on page 7-2

Metadata Loader

Oracle Ultra Search provides a command-line tool to load metadata into an Oracle Ultra Search database. If you have a large amount of data, this is probably faster than using the HTML-based administration tool.

The metadata loader is a Java application. To use the tool, you must put the metadata in an XML file. It supports the following types of metadata:

- Search attribute list of values (LOVs) and display names
- Document relevance boosting and document loading

See Also: Appendix A, "Loading Metadata into Oracle Ultra Search"

Internationalization in Oracle Ultra Search

Translators can enter the following translation strings:

- Search attribute names
- Attribute LOVs
- Data group names
- Federated data source names

During query time, they can be displayed according to the language preference.

Oracle Ultra Search Crawler Features

You can define, edit, or delete your own data sources and types in addition to the ones provided. You might implement your own crawler agent to crawl and index a proprietary document repository, such as Lotus Notes or Documentum, which contain their own databases and interfaces. The proprietary repository is called a user-defined data source. The module that enables the crawler to access the data source is called a crawler agent.

See Also:

- "Oracle Ultra Search Crawler Agent API" on page 9-14
- *Oracle Ultra Search API Reference*

Robots

You can control which parts of your sites can be visited by robots. If robots exclusion is enabled (default), then the Web crawler traverses the pages based on the access policy specified in the Web server `robots.txt` file. For example, when a robot visits `http://www.oracle.com/`, it checks for `http://www.oracle.com/robots.txt`. If it finds it, the crawler analyzes its contents to see if it is allowed to retrieve the document. If you own the Web sites, then you can disable robots exclusions. However, when crawling other Web sites, you should always comply with `robots.txt` by enabling robots exclusion.

See Also: "Web Sources" on page 8-18

Data Harvesting

For initial planning purposes, you might want the crawler to collect URLs without indexing. After crawling is done, you can examine document URLs and status, remove unwanted documents, and start indexing. You can update the crawling mode to the following:

- Automatically accept all URLs for indexing
- Examine URLs before indexing
- Index only

See Also: "Schedules Page" on page 8-27

URL Rewrite

The URL rewriter is a user-supplied Java module for implementing the Oracle Ultra Search UrlRewriter interface. It is used by the crawler to filter or rewrite extracted URL links before they are put into the URL queue. URL filtering removes unwanted links, and ULR rewriting transforms the URL link. This transformation is necessary when access URLs are used.

See Also:

- "Web Sources" on page 8-18
- "Oracle Ultra Search URL Rewriter API" on page 9-23
- *Oracle Ultra Search API Reference*

Query API

Oracle Ultra Search offers a flexible query API to incorporate search functionality to your sites. The query API includes the following functionality:

- Three attribute types: string, number, and date
- Multivalued attributes
- Display name support for attributes, attribute list of values (LOV), and data groups
- Document relevancy boosting
- Arbitrary grouping of attribute query operator using operators (`AND`, `OR`), with control over attribute operator evaluation order
- Selection of metadata returned in query result

See Also:

- "Oracle Ultra Search Query API" on page 9-2
- *Oracle Ultra Search API Reference*

Secure Search

Oracle Ultra Search supports secure searches, which return only documents satisfying the search criteria that the search user is allowed to view. For secure searches, each indexed document should be protected by an access control list (ACL). During searches, the ACL is evaluated. If the user performing the search has permission to read the protected document, then the document is returned by the query API. Otherwise, it is not returned.

There are two ways to secure a data source:

- Specify a single ACL for protecting all documents of a data source.

  The administrator specifies the permissions of the single ACL in the Oracle Ultra Search administration tool. The resulting ACL is used to protect all documents belonging to that data source.

- Crawl ACLs from the data source.

  The data source is expected to provide the ACL together with the document. This lets each document be protected by its own unique ACL.

  Oracle Ultra Search only supports this mode for user-defined data source types where the crawler agent retrieves the ACL from the data source along with other document attributes. You cannot get an ACL from a data source if it is a Web, table, portal, e-mail, or file type. With agent APIs, the URL property "UrlData.ACL" lets the agent to set the ACL of the URL submitted. Also, the `AclHelper` class in the Agent APIs generates the ACL string to make sure that the ACL string format is correct. Only Distinguished Name (DN) and Global User Id (GUID) can be used as the principal of an ACL.

Oracle Ultra Search performs ACL duplicate detection. This means that if a crawled document's ACL already exists in the Oracle Ultra Search system, then the existing ACL is used to protect the document, instead of creating a new ACL within Oracle Ultra Search. This policy reduces storage space and increases performance.

Oracle Ultra Search supports only a single LDAP domain. The LDAP users and groups specified in the ACL must belong to the same LDAP domain.

Caution: If ACLs are crawled from data sources, then it is the responsibility of the administrator to ensure that the data sources being crawled belong to the same LDAP domain. Otherwise, it is possible that search users can inadvertently be granted permissions to documents that they should not be able to access.

Searches run against a secure-search enabled Oracle Ultra Search instance are slower than those run against a non-secure-search enabled instance. This is because each candidate result could require an ACL evaluation. ACLs are evaluated natively by the Oracle server for optimum performance. Nevertheless, the time taken to return hits in a secure search varies depending on the number of ACL evaluations that must be made.

Dependency on Oracle XML DB

Oracle Ultra Search stores ACLs in the Oracle XML DB repository. Oracle Ultra Search also uses Oracle XML DB functionality to evaluate ACLs. (This dependency *only* exists for those users who are making use of secure searching.)

The ACLs are managed by Oracle Ultra Search. ACLs are uniquely referenced by documents from a single Oracle Ultra Search instance. ACLs are not shared by multiple Oracle Ultra Search instances. For acceptable performance, the ACL cache size must be large enough to contain all ACLs evaluated at run time.

ACLs in the XML DB repository are protected by other ACLs (known as *protector ACLs*). Oracle Ultra Search ensures that the protector ACLs grant appropriate privileges for Oracle Ultra Search to invoke the XML DB ACL evaluation mechanism. The evaluation performance is primarily affected by the total number of ACLs used by all XML DB client applications that also utilize its ACL evaluation mechanism. This set of applications includes Oracle Ultra Search.

Security Considerations When Using Restricting Access to a Data Source An Oracle Ultra Search data source can be protected by a single administrator-specified ACL. This ACL specifies which users and groups are allowed to view the documents belonging to that data source.

Oracle Ultra Search uses the Oracle Server's ACL evaluation engine to evaluate permissions when queries are performed by search users. This ACL evaluation engine is a feature of Oracle XML DB. If an Oracle Ultra Search query attempts to retrieve a document that is protected by an administrator-specified ACL, the ACL is evaluated and subsequently cached.

How long an ACL is cached is controlled by an XML DB configuration parameter. The `acl-max-age` parameter must be modified. The value is a number in seconds that determines how long ACLs are cached.

Because ACLs are cached, it is important to remember that changes to an administrator-specified ACL may not propagate immediately. This only applies to database sessions that existed before the change was made.

See Also: *Oracle XML DB Developer's Guide* and "Enabling Secure Search" on page 6-4

Document Relevancy Boosting

You can override the search results and influence the order that documents are ranked in the query result list with document relevancy boosting. This can promote important documents to higher scores and make them easier to find.

Relevancy boosting assigns a score to a document for specific queries entered by the search user.

Note: The document still has a score computed by Oracle Text if you enter a query that is not one of the boosted queries.

Relevancy boosting has the following limitations:

- Comparison of the user's query against the boosted queries uses exact string matching. This means that the comparison is case-sensitive and space-aware. Therefore, a document with a boosted score for "Ultra Search" is not boosted when you enter "ultrasearch".
- Relevancy boosting requires that the query application pass in the search term using the query API `getResult` method call. The applications are designed to pass the basic search terms as the boost term. Advanced search criteria based on search attributes are ignored.

See Also: "Queries Page" on page 8-31

Query Syntax Expansion

Oracle Ultra Search translates each user query into a database query. This process is called query syntax expansion. The Oracle Ultra Search default expansion logic boosts the relevancy of those documents that match the user's query.

The query syntax expansion can be customized with the query API.

See Also: "Customizing the Query Syntax Expansion" on page 9-3

Display URL Support

When gathering information from a database Web application, Oracle Ultra Search lets you specify a URL to display the retrieved data on a browser. The URL points to a screen in the Web application corresponding to the data in the database. This is available for table data sources, file data sources, and user-defined data sources.

See Also: "Using Crawler Agents" on page 7-2

Federated Search

Traditionally, Oracle Ultra Search has used centralized search to gather data on a regular basis and to update one index that cataloged all searchable data. This provided fast searching, but it required that the data source to be crawlable before it could be searched. Oracle Ultra Search now also provides federated search, which allows multiple indexes to perform a single search. Each index can be maintained separately. By querying the data source at search time, search results are always the latest results.

User credentials can be passed to the data source and authenticated by the data source itself. Queries can be processed efficiently using the data's native format.

To use federated search, you must deploy an Oracle Ultra Search search adapter, or searchlet, and create an Oracle source. A searchlet is a Java module deployed in the middle tier (inside OC4J) that searches the data in an enterprise information system. When a user's query is delegated to the searchlet, the searchlet runs the query on behalf of the user. Every searchlet is a JCA 1.0 compliant resource adapter.

See Also: "Federated Sources" on page 8-24

Single Sign-On Authentication

The Oracle Ultra Search administration tool supports the following modes of logging on, depending on the type of user. You can log on as:

- A single sign-on user managed in the Oracle Internet Directory and authenticated with Oracle Application Server Single Sign-On
- A local database schema user in the Oracle Ultra Search database (that it, not using single sign-on)
- A Portal user
- An Enterprise Manager user

Note: Single sign-on is available only with the Oracle Identity Management infrastructure.

See Also: "Logging On to Oracle Ultra Search" on page 8-3

Integration with Oracle Internet Directory

Oracle Internet Directory is Oracle's native LDAP v3-compliant directory service, built as an application on top of the Oracle Database. Oracle Internet Directory hosts the Oracle common identity. All Oracle Web-based products integrate with Oracle Application Server Single Sign-On.

Oracle Ultra Search Administration Groups in Oracle Internet Directory

An Oracle Ultra Search administration group contains a set of users. Each user can belong to one or multiple groups. All groups are created using the `groupOfUniqueNames` and `orclGroup` object classes.

The only way to grant a user administration privileges is to assign them to an administration group. Oracle Ultra Search authorizes the user administration privileges based on the administration groups to which the user belongs. The following groups are created for each Oracle Ultra Search instance:

- Super-users: Users in this group can create or drop Oracle Ultra Search instances and can administer Oracle Ultra Search instances within the installation. Super-users must obey the rules for document relevancy boosting and ACLs defined for each of the documents associated with the Oracle Ultra Search instance. For example, if a document ACL does not grant access to the super-user or group, then the super-user cannot search and browse the document.
- Instance administrators: Users in this group can administer the Oracle Ultra Search instance. Only the instance database schema user and members in the super-users group can drop the instance.

Authorization of the Administration Privileges

The authorization of the administration user is performed in the following steps:

1. After the administration user is successfully authenticated by Oracle Application Server Single Sign-On or the Oracle Ultra Search database, the Oracle Ultra Search GUI brings up the first screen for the user to choose an Oracle Ultra Search instance.
2. The Oracle Ultra Search GUI looks up the Oracle Internet Directory server or Oracle Ultra Search repository to find all Oracle Ultra Search instances that the administration user has privileges to administer.
3. The administration user chooses the Oracle Ultra Search instance from the list.

See Also: *Oracle Internet Directory Application Developer's Guide*

Query Applications

Oracle Ultra Search includes fully functional query applications to query and display search results. The query applications include a search portlet.

The Oracle Ultra Search portlet demonstrates how to write a search portlet for use in Oracle Application Server Portal. It is implemented as a JavaServer Page application.This same portlet is installed as a feature of the Oracle Application Server Portal product.

See Also:

- "Oracle Ultra Search Query API" on page 9-2
- The Oracle Application Server Portal documentation for more information about portlets
- Oracle Ultra Search Query Applications Readme for more information about the query API application

Integration with Oracle Application Server

Although Oracle Ultra Search in the Oracle Application Server is the same product as Oracle Ultra Search in Oracle Collaboration Suite and Oracle Ultra Search in the Oracle Database, there are a few functional differences:

- The Oracle Database is not integrated with OracleAS Portal. However, with OracleAS and Oracle Collaboration Suite installations, Oracle Ultra Search lets Portal users add powerful multi-repository search to their Portal pages. OracleAS and Oracle Collaboration Suite also have the capability to crawl and make searchable Portal's own repository. The Portal crawler recognizes Portal page groups as data sources.
- OracleAS Single Sign-On users can log on once for all components of the Oracle Application Server product, and the Oracle Ultra Search administrative interface allows user management operations on either database users or single sign-on users. Authenticated single sign-on users never see the Oracle Ultra Search logon screen. Instead, they can immediately choose an instance. If the single sign-on user does not have permissions to manage Oracle Ultra Search (set in the **Users Page**), then the single sign-on user receives an error. Single sign-on is available only with the Oracle Identity Management infrastructure.

See Also: `http://portalstudio.oracle.com`

Oracle Ultra Search System Configuration

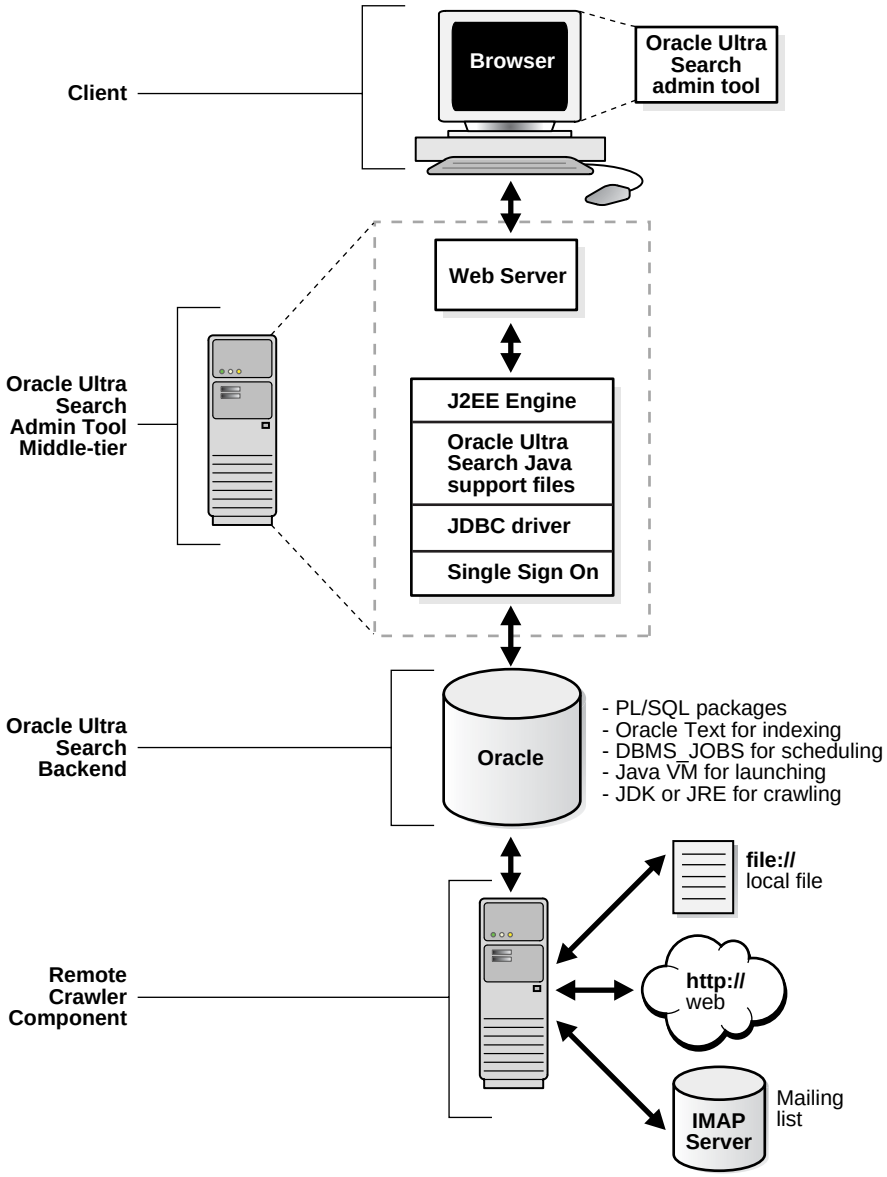
Oracle Ultra Search runs as a client program to the Oracle server. It can be deployed in the backend or in the middle tier of a server configuration.

The Oracle Ultra Search query interface and the administration tool can be accessed from any HTML browser client. The administration tool relies on certain Java classes in the middle tier. This logical middle tier can be the same physical computer as the one that runs the database server, or on a different one running Oracle Application Server. The Oracle Ultra Search database backend consists of the Oracle Ultra Search data dictionary that stores metadata on all the different repositories, as well as the schedules and Java classes needed to drive the crawler. The crawler itself can run either on the database server computer or remotely on another computer.

> **See Also:** Chapter 3, "Installing Oracle Ultra Search" for more information about the components

Figure 1–2 illustrates the Oracle Ultra Search system configuration.

Figure 1–2 Oracle Ultra Search System Configuration



2

Getting Started with Oracle Ultra Search

This chapter contains the following topics:

Overview

This chapter provides information about getting started with Oracle Ultra Search. It features a case study that describes installation and use of Oracle Ultra Search. It enables you to create a browser-based search application to query data sources.

The example uses a fictional customer service call center named Ultra Appliance, Inc. Ultra Appliance is a retail company that sells and supports hundreds of different appliances from dozens of manufacturers nationwide. Customers contact the customer service call center every day to receive technical help for, and assistance with, an appliance.

This chapter describes how you, acting as the Ultra Appliance search administrator, can set up a browser-based search application that enables call center agents to find information for appliances that they support. It also describes how you, acting as an Ultra Appliance call center agent, can use Oracle Ultra Search to query the company intranet and database.

Ultra Appliance call center agents must access a variety of online resources to provide the information needed by the customer. In this example, the Ultra Appliance call center agents have access to two types of information:

- An intranet that contains documents on Web pages with detailed information about appliances sold and serviced by Ultra Appliance.
- A database that contains information about previous maintenance issues and solutions for appliances sold and serviced by Ultra Appliance.

For this example, the Ultra Appliance call center agents search the company intranet and the problem database for information on, and any issues associated with, the Springmaster 2000 refrigerator.

Installation

The instructions in this section are meant to briefly cover installation. For further installation information, see Chapter 3, "Installing Oracle Ultra Search" in this book. You may also want to see the *Oracle Universal Installer Concepts Guide* for detailed installation instructions. Oracle Ultra Search is installed as part of the Oracle Database Server installation, which uses Oracle Universal Installer.

Using the Oracle Universal Installer

Insert the Oracle Installation CD and start the Oracle Universal Installer. Follow the installation wizard instructions to perform an Oracle Database Server install. For more information, see Chapter 3, "Installing Oracle Ultra Search".

Along with the Oracle Database Server, the Oracle Universal Installer also installs the Oracle Ultra Search backend and the Oracle Ultra Search middle tier.

If you have not already done so, unlock the Oracle Ultra Search schema and user:

1. Log in to the database as a DBA user (for example, as SYS).
2. Unlock the Oracle Ultra Search schema, WKSYS, and set its password:

   ```
   ALTER USER wksys account unlock identified by password
   ```

3. Unlock the Oracle Ultra Search WK_TEST schema. Its password is wk_test.

   ```
   ALTER USER wk_test account unlock identified by wk_test
   ```

Accessing the Oracle Ultra Search Administration Application

You must have an Oracle Ultra Search instance. During installation, an instance was created for you, so you need only configure it by following the directions in "Unlock WK_TEST" on page 3-3.

Upon installation, the default Oracle Ultra Search instance, WK_INST, is available; it is built on the default WK_TEST schema.

To access the administration application, start up the Oracle Ultra Search middle tier:

```
$ORACLE_HOME/bin/searchctl start
```

After the middle tier has been started, you can access the administration interface with an HTML browser pointed to:

```
http://your_computer.com:http_port/ultrasearch/admin/index.jsp
```

At the end of the installation, find *http_port* on the Oracle Universal Installer screen. You can find out the value of *http_port* by looking at:

```
$ORACLE_HOME/oc4j/j2ee/OC4J_SEARCH/config/http-web-site.xml
```

Setting up the Query Application

Oracle Ultra Search provides APIs with which you can build customized, J2EE query applications. Oracle Ultra Search also includes an application of this sort.

To set up the query application, do the following steps, using your own values for the host name, the port, and the SID:

1. Add the following entry to the ORACLE_HOME/oc4j/j2ee/OC4J_SEARCH/config/data-sources.xml file:

```
<data-source
class="oracle.jdbc.pool.OracleConnectionCacheImpl"
name="UltraSearchDS"
location="jdbc/UltraSearchPooledDS"
username="wk_test"
password="wk_test"
url="jdbc:oracle:thin:@database_host:oracle_port:oracle_sid"
/>
```

2. If you started the Oracle Ultra Search middle tier to access the Oracle Ultra Search administration application, then you must stop and restart it, with the query application now configured to use the query application:

```
$ORACLE_HOME/bin/searchctl stop
$ORACLE_HOME/bin/searchctl start
```

After the middle tier has been started, you can access the query application with an HTML browser pointed to:

`http://your_computer.com:http_port/ultrasearch/query/search.jsp`.

At the end of the installation, find *http_port* on the Oracle Universal Installer screen. You can find out the value of *http_port* by looking at:

`$ORACLE_HOME/oc4j/j2ee/OC4J_SEARCH/config/http-web-site.xml`

Setting up the Ultra Appliance Demo

To access the Ultra Appliance intranet site, go to:

http://www.oracle.com/technology/products/ultrasearch/gettingstarted/

To set up the Ultra Appliance company database, perform the following steps:

1. Copy the appliances.sql script shown in Example 2–1 into a text editor. Save this file as appliances.sql to your database server computer
2. Upload appliance.sql to your database schema WK_TEST with the following statements:

```
prompt > sqlpls WK_TEST/WK_TEST
SQLPLUS > @appliance.sql;
SQLPLUS > commit;
SQLPLUS > exit
```

Example 2–1 appliances.sql script

```
DROP TABLE product;
CREATE TABLE product (id NUMBER PRIMARY KEY, Description VARCHAR2(200),
     Parts VARCHAR2(80));
INSERT INTO product VALUES(1, 'Springmaster 2000', 'Cantaloupe Tray');
INSERT INTO product VALUES(2, 'TipNClear 2000', 'TipNVac Tray');
INSERT INTO product VALUES(3, 'Spew 2000', 'Extra Dirt' );
INSERT INTO product VALUES(4, 'Hold Em 2000', 'Spare Magnets');
INSERT INTO product VALUES(5, 'Pizza Legend 2000', 'No. 7 Pizza Tube');
INSERT INTO product VALUES(6, 'SnoozePower 2000', 'Lint Screen');
DROP TABLE problems;
CREATE TABLE problems (Problem_ID NUMBER PRIMARY KEY, Customer_Name VARCHAR2(40),
     Product_ID NUMBER, Date_ID DATE, Problem_Description VARCHAR2(200),
     Resolution_Text VARCHAR2(200));
INSERT INTO problems VALUES(1, 'Jones', 4, '10-Aug-03', 'Magnets pointed wrong way',
     'Solved by reversing pet');
INSERT INTO problems VALUES(2, 'Smith', 1, '01-Oct-02', 'Cantaloupe wrong color',
     'Solved by icing down melons');
INSERT INTO problems VALUES(3, 'Chan', 3, '10-Apr-03', 'Clogged by cat hair',
     'Solved by getting new cat');
INSERT INTO problems VALUES(4, 'Ali', 5, '29-May-03', 'Will not work with anchovies',
     'Cannot solve');
INSERT INTO problems VALUES(5, 'Johnson', 2, '28-Feb-03', 'Husband on couch',
     'Solved by removing husband');
INSERT INTO problems VALUES(6, 'Kawamoto', 6, '11-Nov-02', 'Pillow too loud',
     'Solved by turning pillow over');
INSERT INTO problems VALUES(7, 'Weiss', 3, '15-May-03', 'Dirt coming out wrong color',
     'Solved by bleaching dirt');
INSERT INTO problems VALUES(8, 'Claire', 5, '20-Jun-03', 'Overheats cheese',
     'Solved by using better-quality cheese');
INSERT INTO problems VALUES(9, 'Dontenmann', 2, '08-Jan-03', 'Gum stuck, will not shake out',
     'Solved by increasing power');
INSERT INTO problems VALUES(10, 'Glass', 4, '22-Aug-03', 'Ferret allergic to magnets',
     'Solved by washing magnets');
INSERT INTO problems VALUES(11, 'Heyboll', 1, '03-Sep-02', 'Flying cantaloupes injuring family',
     'Solved by ducking');
```

Crawl and Index Ultra Appliance's Intranet Documents

This section describes the steps you take as the Ultra Appliance search administrator to set up Oracle Ultra Search to crawl and index the company intranet. After you perform this setup, the call center agents can use Oracle Ultra Search to obtain information about the Springmaster 2000 refrigerator.

To crawl and index the Ultra Appliance intranet:

1. Log on to Oracle Ultra Search using the Oracle Ultra Search Administration Tool screen.

   In your Web browser, enter the domain and port of the computer where you have Oracle Ultra Search installed, followed by `/ultrasearch/admin/index.jsp`:

   `http://`*`your_computer.domain:http_port`*`/ultrasearch/admin/index.jsp`

   For example:

   `http://comp1.ultrasupply.com:7778/ultrasearch/admin/index.jsp`

2. Enter the user name and password to log in to Oracle Ultra Search. For example:

   Login: `WK_TEST`

   Password: `WK_TEST`

   The login screen is displayed in Figure 2–1.

Figure 2–1 Oracle Ultra Search Login Screen

3. Select an Oracle Ultra Search Instance screen.

   Select the **Instances** tab on the browser view.

   Select the `WK_INST` instance from the pull down menu and click **Apply**.

4. Configure the Oracle Ultra Search crawler settings.

   Select the **Crawler** tab on the browser view.

   Use the following values for crawler settings:

   - Crawler Threads: **20**
   - Number of Processors: **1**
   - Automatic Language Detection: **No**
   - Default Language: **English**
   - Crawling Depth: **No Limit**
   - Crawler Timeout Threshold: **30**
   - Default Character Set: **ISO Latin-1**
   - Cache Directory Location and Size: **/tmp, 5**
   - Crawler Logging: **/tmp/, English**
   - Database Connect String: **Leave this field unchanged.**

5. Create a Web data source.

   Select the **Sources** tab on the browser view.

   Under **Web Sources**, click **Create Web Source**.

   a. Create Web Source: Step 1

      Enter **Ultra Appliance** as the Source name.

      Under the Starting Addresses heading, enter the complete URL of the location of the Ultra Appliance intranet Web site demo:

```
http://www.oracle.com/technology/products/ultrasearch/gettingstarted/
```

Click **Add** to add the Ultra Appliance data source to the list of Web addresses.

Click **Next**.

b. Create Web Source: Step 2 (URL Boundary Rules)

Accept the default values and click **Next**.

c. Create Web Source: Step 3 (Document Types)

Specify the types of document you would like Oracle Ultra Search crawler to crawl.

Under the Document Types header, select HTML, Microsoft Word document, and PDF document. After you make each selection, click >> to add the document types to the list of document types for crawling.

Click **Next.**

d. Create Web Source: Step 4

Accept the default values and click **Finish**.

The Ultra Appliance Web site demo is added to the Web source list.

6. Schedule the Oracle Ultra Search crawler.

Select the Schedule tab and click **Create New Schedule**.

In the Name field, enter **Ultra Appliance**. Click the **Proceed to Step 2**.

Select "**Every 1 week(s) on Monday starting at 0100 hours.**" Under the Indexing option heading, select "**Automatically accept all URLS for indexing.**" Under the Remote Crawler Profiles, select database host from the drop down list. Click **Proceed to step 3**.

Select **Web** from the drop down menu. Select Ultra Appliance from the Available Sources menu and transfer it to the Assigned Sources menu by clicking >>. Click **Finish**.

7. Start crawling and indexing the documents.

On the **Synchronization Schedules** page, locate **Ultra Appliance** in the **Schedules** column.

a. In the **Status** column for the Ultra Appliance row, verify that the status is in the **Scheduled** condition.

b. Click **Scheduled**. The Synchronization Schedule Status page is displayed.

c. Click **Execute Immediately** so that Oracle Ultra Search can crawl and index the Ultra Appliance intranet site.

d. Click **Refresh status** to see schedule status changes. When the schedule status displays **Scheduled,** the crawling is complete.

Crawl and Index Ultra Appliance's Database Documents

This section describes how, acting as the Ultra Appliance search administrator, you set up Oracle Ultra Search to search for the Ultra Appliance company database.

You can configure the Oracle Ultra Search crawler to crawl the database you set up in "Setting up the Ultra Appliance Demo".

To crawl and index the Ultra Appliance database:

1. Follow steps 1 through 4 in "Crawl and Index Ultra Appliance's Intranet Documents".
2. Create a table data source.

   Select the **Sources** tab and then the **Table** subtab on the browser view.

   Under the **Table Sources** header, click **Create New Table Source**.

   a. Create Table Source: Step 1

      Enter a name, such as **Ultra Appliance DB**, in the Name field for the table source name. The name should not be the same as the one you entered when you crawled and indexed Intranet documents.

      Under the **Database Table** heading do not use a database link.

      Enter the schema name of `WK_TEST` in the Schema field.

      Enter the name of **Problems** in the Table name field.

      Click **Locate Table**.

      The following note is displayed if a table is present in your database:

      Note: Successfully located table problems.

      Click **Proceed to Step 2**.

   b. Create Table Source: Step 2

      Accept the default values for Language.

      Under the **Complex Primary Key** heading, add the `PROBLEM_ID` table column by clicking **Add column**.

      Under the **Content Column and Type** heading, select `PROBLEM_DESCRIPTION` for Column and `Plaintext` for the Type.

      Click **Proceed to Step 3**.

   c. Create Table Source: Step 3

      Under the Verify Table Source Details heading, confirm the settings and values that will be displayed in the Oracle Ultra Search output.

      Click **Create Table Source.**

   d. Table Source Logging

      Select **Disable logging mechanism** and click **Apply**.

   e. Create Table Source: Step 4

      Specify the table columns and search attribute mappings you want Oracle Ultra Search crawler to crawl on the database.

      For this example, select:

      * `RESOLUTION_TEXT` for table column with search attribute of `Description [String]`
      * `CUSTOMER_NAME` for table column with search attribute of `Author [String]`

      After you make each selection, click **Map** to add the document types to the list of document types for crawling.

Click **Proceed to step 5**.

f. Create Table Source: Step 5

Select **No display URL** and click **Finish**.

3. Schedule the Oracle Ultra Search Crawler.

See Step 6 on page 2-6 in "Crawl and Index Ultra Appliance's Intranet Documents" for procedures on how to set up a crawler schedule.

Select **Table** from the drop-down list seen on the **Create Schedule: Step 3 of 3** screen.

4. Start crawling and indexing, as shown in Step 7 on page 2-6 in "Crawl and Index Ultra Appliance's Intranet Documents".

Issuing a Query

This section describes the steps you take, acting as the Ultra Appliance call center agent, using Oracle Ultra Search to query the company intranet and database. You can query the Ultra Appliance information sources after you have allowed Oracle Ultra Search to crawl and index the Ultra Appliance intranet and database.

To query the Ultra Appliance intranet:

1. Enter the URL for the Oracle Ultra Search location. For example:

```
http://your_computer.com:http_port/ultrasearch/query/search.jsp
```

2. Enter Springmaster 2000 in the **Search For** field.

You should see the output displayed in Figure 2–2.

Figure 2–2 Oracle Ultra Search Search Output

To query the Ultra Appliance database:

- Enter **Cantaloupe Wrong Color** in the **Search For** field.

You should output similar to the display in Figure 2–2 with links to the Springmaster 2000 Cantaloupe Tray Problem table.

3

Installing Oracle Ultra Search

This chapter contains the following topics:

> **Note:** Some information in this chapter is generic to all types of Oracle Ultra Search installations. Other information in this chapter is specific to installing and configuring Oracle Ultra Search with an Oracle Database release.
>
> If you are installing Oracle Ultra Search with the Oracle Application Server release, then you should also read Chapter 4, "Using Oracle Ultra Search with Oracle Application Server".

Oracle Ultra Search Requirements

This section describes the Oracle Ultra Search system requirements.

Hardware Requirements for Oracle Ultra Search

Oracle Ultra Search hardware requirements are based on the quantity of data that you plan to process using Oracle Ultra Search. Oracle Ultra Search uses Oracle Text as its indexing engine and the Oracle Database as its repository.

> **See Also:** *Oracle Text Application Developer's Guide* and *Oracle Database Performance Tuning Guide*

Sufficient RAM Along with the resource requirements for the database and the Text indexing engine, also consider the memory requirements of the Oracle Ultra Search crawler. The Oracle Ultra Search crawler is a pure Java program. When the crawler is launched, the Java Virtual Machine (JVM) is configured to start with 25MB and grow

to 256MB. When crawling very large amounts of data, these values might need to be adjusted.

The Oracle Ultra Search administration tool is a J2EE 1.2 standard Web application. It can be installed and run on a separate host from the Oracle Ultra Search backend. You might want to install and run this on the same host as the Oracle Ultra Search backend. Regardless of your choice, allocate enough memory for the J2EE engine. Oracle recommends using the Oracle HTTP Server with the Oracle Application Server Containers for J2EE (OC4J). Allocate enough memory for the HTTP Server as well as for the Java Development Kit (JDK) that runs the J2EE engine.

Sufficient Disk Space Because customer requirements vary widely, Oracle cannot recommend a specific amount of disk space. As a general guideline, the minimum requirements are as follows:

- Approximately 3GB of disk space for the Oracle Application Server Infrastructure or database and the Oracle Ultra Search backend.
- 15MB of disk space for the Oracle Ultra Search middle tier on top of the Web server's disk requirements.
- For each remote crawler host, the same amount of disk space as needed to install the Oracle Ultra Search backend.
- Disk space for a large `TEMPORARY` tablespace. Create a `TEMPORARY` tablespace as large as possible, depending on the amount of RAM on the host.
- Disk space for the Oracle Ultra Search instance user's tablespace.
  - The Oracle Ultra Search instance user is a database user that you must explicitly create. All data that is collected and processed as part of crawling and indexing is stored in this user's schema.
  - You should create the tablespace as large as the total amount of data that you want to index. For example, if you estimate that the total amount of data to be crawled and indexed is 10GB, then create a tablespace that is at least 10GB for the Oracle Ultra Search instance user. Make sure to assign that tablespace as the default tablespace of the Oracle Ultra Search instance user.

Software Requirements for Oracle Ultra Search

The Oracle Ultra Search middle tier components are Web applications. Therefore, they require a Web server to run. Oracle recommends using Oracle Application Server.

Installing the Oracle Ultra Search Backend

The Oracle Ultra Search backend consists of the following:

- Oracle Ultra Search database schema: Data dictionary and PL/SQL packages.
- Oracle Ultra Search crawler: Java program plus supporting files, libraries, and so on.
- Oracle Ultra Search remote crawler: Crawler residing on a remote Oracle home.

The Oracle Ultra Search backend is installed as part of the Oracle Database Server installation, which is accomplished using the Oracle Universal Installer.

See Also: *Oracle Universal Installer Concepts Guide*.

Installing the Oracle Ultra Search Middle Tier

The Oracle Ultra Search middle tier includes the following:

- Oracle Ultra Search administration tool
- Oracle Ultra Search Java query API
- Oracle Ultra Search query applications

Installing the Middle Tier with the Oracle Database

The Oracle Ultra Search middle tier is installed as part of the Oracle Database Server installation, which is accomplished using the Oracle Universal Installer.

See Also: "Oracle Ultra Search Middle Tier with Oracle Application Server" on page 4-2

Postinstallation Tasks

This section covers Oracle Ultra Search postinstallation tasks. There are five steps to the postinstallation:

1. Start the Oracle Ultra Search Middle Tier.
2. Unlock WK_TEST.
3. Enable the Oracle Ultra Search Query Applications.
4. Restart the Oracle Ultra Search Middle Tier.
5. Reconfigure the Oracle Ultra Search Backend for the Database Character Set.

This section explains these five steps in more detail.

Start the Oracle Ultra Search Middle Tier

For the Oracle Database release, use the following command to start the Oracle Ultra Search middle tier. You must run this command manually to bring up the Oracle Ultra Search middle tier after installation.

```
$ORACLE_HOME/bin/searchctl start
```

For the Oracle Application Server release, use Oracle Process Manager and Notification Server (OPMN) to start the OC4J_Portal instance. For example:

```
$ORACLE_HOME/opmn/bin/opmnctl startproc instancename=OC4J_Portal
```

Unlock WK_TEST

The Oracle Ultra Search installer creates a default Oracle Ultra Search instance based on the default Oracle Ultra Search test user. You can test Oracle Ultra Search functionality based on the default instance after installation.

The default instance name is `WK_INST`. It is created based on the database user `WK_TEST`. The default user password is `WK_TEST`.

For security purposes, `WK_TEST` is locked after the installation. The administrator should log on to the database as DBA role, unlock the `WK_TEST` user account, and set the password to be `WK_TEST`. The password expires after the installation. If the password is changed to anything other than `WK_TEST`, then you must also update the

cached schema password using administration tool Edit Instance page after you change the password in the database.

The default instance is also used by the Oracle Ultra Search query application. Make sure to update the `data-sources.xml` file.

See Also:

- "Schema Password" on page 8-9
- "Enable the Oracle Ultra Search Query Applications" on page 3-4

Enable the Oracle Ultra Search Query Applications

Caution: Storing clear text passwords in `data-sources.xml` poses a security risk. Avoid this by using password indirection to specify the password. This lets you enter the password in `jazn-data.xml`, which automatically gets encrypted, and point to it from `data-sources.xml`. For more information, see "Creating An Indirect Password" in *Oracle Application Server Containers for J2EE Security Guide*.

The `data-sources.xml` file is located in the `$ORACLE_HOME/oc4j/j2ee/OC4J_SEARCH/config` directory. Under tag `<data-sources>` add the following:

```
<data-source
   class="oracle.jdbc.pool.OracleConnectionCacheImpl"
   name="UltraSearchDS"
   location="jdbc/UltraSearchPooledDS"
   username="username"
   password="password"
   url="jdbc:oracle:thin:@database_host:oracle_port:oracle_sid"
/>
```

In the preceding syntax, the following variables were used:

- *username* and *password* are the Oracle Ultra Search instance owner's database user name and password.
- *database_host* is the host name of the back end database computer.
- *oracle_port* is the port to the user's Oracle Database.
- *oracle_sid* is the SID of the user's Oracle Database.

In addition to user name, password, and JDBC URL, `data-sources.xml` allows configuration of the connection cache size, and the cache scheme.

The following tag specifies the minimum and maximum limits of the cache size, the inactivity time out interval, and the cache scheme.

```
<data-source
   class="oracle.jdbc.pool.OracleConnectionCacheImpl"
   name="UltraSearchDS"
   location="jdbc/UltraSearchPooledDS"
   username="wk_test"
   password="wk_test"
   url="jdbc:oracle:thin:@localhost:1521:isearch"
   min-connections="3"
   max-connections="30"
   inactivity-timeout="30">
 <property name="cacheScheme"  value="1"/>
</data-source>
```

If you are adding the data source for the default Oracle Ultra Search instance user WK_TEST, then make sure to unlock WK_TEST first.

Note: The URL of the JDBC data source can be provided in the form of jdbc:oracle:thin:@*hostname*:*port*:*sid* or in the form of a TNS keyword-value syntax, such as "jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=yes)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=cls02a)(PORT=3999))(ADDRESS=(PROTOCOL=TCP)(HOST=cls02b)(PORT=3999)))(CONNECT_DATA= (SERVICE_NAME=acme.us.com)))"

See Also: "Unlock WK_TEST" on page 3-3

There are three values for the caching schemes:

- 1 = DYNAMIC_SCHEME
- 2 = FIXED_WAIT_SCHEME
- 3 = FIXED_RETURN_NULL_SCHEME

See Also: *Oracle Application Server Containers for J2EE Security Guide*

Restart the Oracle Ultra Search Middle Tier

For the database release, stop and restart the Oracle Ultra Search middle tier. For example:

```
$ORACLE_HOME/bin/searchctl stop
$ORACLE_HOME/bin/searchctl start
```

For the Oracle Application Server release, use Oracle Process Manager and Notification Server (OPMN) to start the OC4J_Portal instance. For example:

```
$ORACLE_HOME/opmn/bin/opmnctl startproc instancename=OC4J_Portal
```

Reconfigure the Oracle Ultra Search Backend for the Database Character Set

You must reconfigure the Oracle Ultra Search backend to adapt to the new character set if one of the following has occurred:

- You choose a database character set (for example, UTF8) other than the default WE8ISO8859P1 character set during the Oracle Ultra Search installation.
- The database character set is changed after installation.

Two SQL scripts (`wk0prefcheck.sql` and `wk0idxcheck.sql`), located in `$ORACLE_HOME/ultrasearch/admin/`, are used for reconfiguration:

- `wk0prefcheck.sql` is run by the `wksys` user to reconfigure default cache character set and index preferences.
- `wk0idxcheck.sql` is needed for reconfiguring instances created before the database character set change (for example, the default instance). This script must be run by the instance owner, and `wk0prefcheck.sql` must be run first because it depends on reconfigured default settings generated by `wk0prefcheck.sql`.

  Running `wk0idxcheck.sql` also drops and recreates the Oracle Text index used by Oracle Ultra Search. If there are already data sources indexed, then you must force a recrawl of all of the data sources.

  `wk0idxcheck.sql` must be run once for each instance. For example, if there are two instances, inst1 and inst2, owned by owner1 and owner2, respectively, then `wk0idxcheck.sql` should be run twice, once by owner1 and once by owner2.

Note: Oracle Ultra Search only supports database character sets supported by Oracle Text. For example, the AL32UTF8 character set is not supported. For Unicode support, use UTF8. For the complete list of supported database character sets, see the *Oracle Text Reference* for lexer types.

Checking Your Oracle Ultra Search Installation

This section describes how to check that your installation was successful.

Testing the Oracle Ultra Search Administration Tool

If you log on to the Oracle Ultra Search administration tool successfully, then you have completed the Oracle Ultra Search administration tool configuration process. Do the following to check the Oracle Ultra Search Administration Tool:

1. Check that the Web Server is running.
2. Attempt to log on to the administration tool:
   a. Visit the following URL

      `http://`*`hostname.domainname:port`*`/ultrasearch/admin/index.jsp`

      In the preceding URL, *hostname.domainname* is the full name of the host where you have installed the Oracle Ultra Search middle tier, and *port* is the default Web server port.

   b. Log on to the Oracle Ultra Search administration tool by entering the Oracle Ultra Search instance owner's database user name and password.During the installation of the Oracle Ultra Search backend, a new Ultra Search instance owner, `WK_TEST` is created.

   The first time any JSP page is accessed, it takes a few seconds to compile. Subsequent accesses are much faster.

Testing the Oracle Ultra Search Query Applications

After you verify that the Oracle Ultra Search administration tool is working, you should be able to run the Oracle Ultra Search query applications.

To test the Oracle Ultra Search query applications, do one of the following:

- Visit the following URL:

  `http://hostname.domainname:port/ultrasearch/query/search.jsp`

- Follow the links in the Oracle Ultra Search welcome page:
  `http://hostname.domainname:port/ultrasearch/index.html`

Locations for query applications are listed in the following section. Access the query source code by going to the directories list. You can also see a working demonstration of each query JSP page with the URL root, and you can append the correct JSP file name at the end of the URL root.

The query application is shipped as `$ORACLE_HOME/ultrasearch/ultrasearch_query.ear`.

Portlet is shipped as `$ORACLE_HOME/ultrasearch/webapp/ultrasearch_portlet.ear`.

Troubleshooting Oracle Ultra Search

This section describes how to troubleshoot Oracle Ultra Search.

Query finds no results

See "Reconfigure the Oracle Ultra Search Backend for the Database Character Set" on page 3-5.

Error when processing binary files

The Oracle Ultra Search crawler uses the Oracle Text INSO filter, `ctxhx`, for processing of binary files. These are non-text, non-HTML files such as PDF files, Microsoft Word files, and so on. For Oracle Ultra Search to be able to use the INSO filter, the shared library path environment variable must contain the `$ORACLE_HOME/ctx/lib` path.

At installation, the Oracle Universal Installer automatically sets the variable to include `$ORACLE_HOME/ctx/lib`. If you restart the database after the installation, then you must manually set your shared library path environment variable to include `$ORACLE_HOME/ctx/lib` before starting the Oracle process. You must restart the database to pick up the new value for filtering to work.

- On UNIX set the `$LD_LIBRARY_PATH` environment variable to include `$ORACLE_HOME/ctx/lib`.
- On Windows set the `$PATH` environment variable to include `$ORACLE_HOME/bin`.

Error when crawling a file data source

If the globalization setting for the environment that starts the Oracle Database is not compatible with the target files' locale, then a file not found error occurs or files or directories with names containing the CJK character. This error occurs in a multibyte language environment like Chinese, Japanese, or Korean. This is because the crawler relies on the correct locale setting to read operating system files.

To correct this, set the correct locale, restart the Oracle Database, and force Oracle Ultra Search to re-crawl the data source. For example:

1. Shutdown the Oracle Database instance:

```
SQL> shutdown immediate
```

2. Set the locale to `'ja'` with the following:

```
> setenv LANG ja
> setenv LC_ALL ja
```

3. Restart the Oracle Database instance:

```
SQL> startup
```

4. Restart the Oracle Ultra Search schedule with a forced re-crawl.

Cannot log on to the Oracle Ultra Search administration tool

The `ultrasearch.properties` file contains configuration information used by Oracle Ultra Search middle tier. You do not need to edit this file, because it is automatically configured by the Oracle Universal Installer.

However, with a software-only or an advanced database installation, you must manually configure the Oracle Ultra Search administration tool by editing it. You must replace `%THIN_JDBC_CONN_STR%` with a JDBC string to the database, and replace `%DOMAIN%` with the domain name.

Here is an example of the `ultrasearch.properties` file:

```
connection.driver=oracle.jdbc.driver.OracleDriver
#If set, The JDBC connection URL specified here will override the dynamically
#acquired one from Oracle Internet Directory.
#This setting is also used by the query sample (gsearch.jsp)
#Example: connection.url=jdbc:oracle:thin:@<host>:<port>:<sid>
connection.url=%JDBC_CONN_STR%
oracle.net.encryption_client=REQUESTED
oracle.net.encryption_types_client=(RC4_56,DES56C,RC4_40,DES40C)
oracle.net.crypto_checksum_client=REQUESTED
oracle.net.crypto_checksum_types_client=(MD5)
oid.app_entity_cn=m16bi.sgtcnsun03.cn.oracle.com
domain=us.oracle.com
```

In the preceding example, the following variables were used:

- *connection.driver* specifies the JDBC driver you are using.
- *connection.url* specifies the database to which the middle tier connects. Oracle Ultra Search supports following formats:
  - *host*:*port*:*SID* (where *host* is the full host name of the Oracle Database instance running Oracle Ultra Search, *port* is the listener port number for the Oracle Database instance, and *SID* is the Oracle Database instance ID)
  - HA-aware string (for example, TNS keyword-value syntax)
- *oracle.net.encryption_client*, *oracle.net.encryption_types_client*, *oracle.net.crypto_checksum_client*, and *oracle.net.crypto_checksum_types_client* control the properties of the secure JDBC connection made to the database. See *Oracle Database JDBC Developer's Guide and Reference* for more information.
- `oid.app_entity_cn` specifies the Oracle Ultra Search middle tier application entity name.
- `domain` specifies the common domain for the Identity Management machine and the Oracle Ultra Search middle tier machine. This enables delegated

administrative service (DAS) list of values to work with Internet Explorer. For example, if the Oracle Ultra Search middle tier in us.oracle.com and the Identity Management machine is uk.oracle.com, then the common domain is oracle.com.

Note: You no longer need to configure the JDBC connect string in the `ultrasearch.properties` file. The database connect information is taken from Oracle Internet Directory.

Installing the Backend on Remote Crawler Hosts

The Oracle Ultra Search remote crawler allows multiple crawlers to run in parallel on different hosts. However, all remote crawler hosts must share common resources, such as common directories and a common Oracle Ultra Search database.

Installing the Backend on Remote Crawler Hosts

The Oracle Ultra Search remote crawler is part of the Oracle Ultra Search backend. The crawler installation procedure is the same as installing the Oracle Ultra Search backend.

On each remote crawler host, the Oracle Ultra Search backend is installed under a common directory known as `ORACLE_HOME`. You should have been prompted by the Oracle Universal Installer to enter this directory. The remote `ORACLE_HOME` directory is referred to as `$REMOTE_ORACLE_HOME`.

If you choose not to install the Oracle HTTP Server during the Oracle Application Server installation, then you must perform the following steps manually for remote crawling:

- Locate the file that defines the environment.
  - On UNIX, `$REMOTE_ORACLE_HOME/ultrasearch/tools/remotecrawler/scripts/unix/define_env`
  - On Windows, `$REMOTE_ORACLE_HOME/ultrasearch/tools/remotecrawler/scripts/winnt/define_env.bat`
- Replace `%ORACLE_HOME%` with the value of the `REMOTE_ORACLE_HOME` environment variable.
- Replace `%s_jreLocation%` with the directory path of a Java runtime environment (JRE) version 1.2.2 and higher. You should specify the root directory of the JRE.
- Replace `%s_jreJDBCclassfile%` with the full path and file name of the Oracle JDBC Thin driver version 12.

Configuring the Remote Crawler

The remote crawler requires a communication channel between the backend database and the remote crawler host.

The mechanisms of communication are RMI and JDBC. Configuration of the remote crawler differs depending on which mechanism you use. The JDBC-based mechanism requires you to supply a database user (or role) during the registration process.

> **See Also:** "Using the Remote Crawler" on page 10-4 more information on the RMI and JDBC mechanisms

The registration process is done by running a SQL script on the Oracle Ultra Search remote crawler host. The SQL script connects over SQL*Plus to the Oracle backend database and registers the remote crawler host.

1. Locate the correct ORACLE_HOME.

   The Oracle Ultra Search middle tier is installed under a common directory known as ORACLE_HOME. If you have installed other Oracle products prior to the Oracle Ultra Search middle tier, then you could have multiple ORACLE_HOME directories on your host. The registration script requires that you enter the ORACLE_HOME directory in which the Oracle Ultra Search middle tier is installed.

2. Locate the WKSYS super-user password.

   You must run the registration script as the WKSYS super-user or as a database user that has been granted super-user privileges.

3. Start SQL*Plus.

   Be sure to run the correct version of SQL*Plus, because multiple versions can reside on the same host if you have previously installed some Oracle products. On UNIX platforms, make sure that the correct values for PATH, ORACLE_HOME and TNS_ADMIN variables are set. On Windows, choose the correct menu item from the Start menu.

   After you have identified how to run the correct SQL*Plus client, you must log on to the Oracle Ultra Search database. To do this, you might need to configure an Oracle Net service setting for the Oracle Ultra Search database.

4. After SQL*Plus is running, log on to the database using the schema and password that you located in Step 2.

5. Run the registration script.

   Start up SQL*Plus as the WKSYS super-user and enter the following:

   ```
   @full_path_of_registration_script
   ```

   - The registration script for RMI-based remote crawling is the following:

     ```
     $REMOTE_ORACLE_
     HOME/ultrasearch/tools/remotecrawler/scripts/<platform>/register.sql
     ```

     For example, if the value for $REMOTE_ORACLE_HOME on a UNIX host is /home/oracle10g, then enter the following at the SQL*Plus prompt to register an RMI-based remote crawler:

     ```
     @/home/oracle10g/ultrasearch/tools/remotecrawler/scripts/unix/register.sql
     ```

     The RMI-based registration script prompts you for three variables:

     – RMI_HOSTNAME: The remote hostname. This is where the RMI registry / daemon will run.

     – RMI_REGISTRY_PORT: The port that the RMI registry is listening on.

     – ORACLE_HOME: The Oracle home located in Step 1.

       For example, /u01/oracle10g on a UNIX host or d:/u01/oracle10g on a Windows host. Remember to use forward slashes for Windows hosts.

- The registration script for JDBC-based remote crawling is the following:

  ```
  $REMOTE_ORACLE_
  HOME/ultrasearch/tools/remotecrawler/scripts/<platform>/register_jdbc.sql
  ```

  For example, if you are running SQL*Plus on Windows, and `$REMOTE_ORACLE_HOME` is in `d:\Oracle\Oracle10g`, then enter the following at the SQL*Plus prompt to register a JDBC-based remote crawler:

  ```
  @d:\Oracle\Oracle10g\ultrasearch\tools\remotecrawler\scripts\winnt\register
  _jdbc.sql
  ```

  The JDBC-based registration script prompts for three variables:

  – `LAUNCHER_NAME`: An arbitrary string used to identify a JDBC-based remote crawler launcher, which is needed when you start up the JDBC-based remote crawler launcher.

  – `CONNECTUSER`: The database user (or role) that the JDBC-based remote crawler launcher will use to establish a database connection and listen for launch events.

  – `ORACLE_HOME`: The Oracle home located in Step 1.

  The registration script invokes the `wk_crw.register_remote_crawler` PL/SQL API. The `REMOTE_CRAWLER_HOSTNAME` and `ORACLE_HOME` variables are used to compose arguments for the `wk_crw.register_remote_crawler` API. You may optionally choose to call this API, especially if you need to register multiple remote crawlers programatically.

6. Verify and complete the remote crawler profile configuration. Be sure to enter the correct values for both variables. To verify that the registration has completed correctly, do the following:

   a. Log on to the Oracle Ultra Search administration tool.

   b. Click the **Remote Crawler Profiles** tab in the **Crawler** tab. You should see the remote crawler launcher you registered in the remote crawler profile list.

      – For RMI-based remote crawlers, you will see the *host*:*port* combination that uniquely identifies the RMI-subsystem.

      – For JDBC-based remote crawlers, you will see the Launcher name.

   c. Click **Edit** to complete the configuration process for the remote crawler profile.

   See Also: *Oracle Net Services Administrator's Guide* for information on how to configure a service setting

Unregistering a Remote Crawler

If you enter any wrong values for the `register.sql` script, then you should unregister the remote crawler using the `unregister.sql` script. Run the unregister script the same way as you ran the registration script. The `unregister.sql` script calls the `wk_crw.unregister_remote_crawler` PL/SQL API. After you have successfully unregistered the remote crawler, you can rerun the `register.sql` script.

Upgrading Oracle Ultra Search Shipped with Oracle Database

Before you upgrade, log on to the Oracle Ultra Search administration tool. Stop and disable all crawler synchronization schedules in every Oracle Ultra Search instance. You can enable all crawler synchronization schedules after the upgrade.

> **See Also:** "Schedules Page" on page 8-27 for details on how to stop and disable the synchronization schedule

To upgrade Oracle Ultra Search shipped with the Oracle Database release, do the following:

1. Run the Oracle Ultra Search backend upgrade. This includes upgrading the Oracle Ultra Search database schemas and server files. Install the new Oracle software, and run Oracle Database Upgrade Assistant to upgrade the database and Oracle Ultra Search component to the new release. See the *Oracle Database Upgrade Guide* for details.
2. Follow the steps in "Installing the Oracle Ultra Search Middle Tier" on page 3-3 to install the new Oracle Ultra Search middle tier.

Post-Upgrade Configuration Steps

After upgrading to the current release, follow these post-upgrade configuration steps:

1. Set the `ORACLE_HOME` and `ORACLE_SID` environment variables to Oracle Database 10*g*.
2. Change directories to `ORACLE_HOME/ultrasearch/admin/`.
3. Run the following statement:

   ```
   sqlplus "sys/password as sysdba"
   ```

4. Run the following statement:

   ```
   @wk0config.sql WKSYSPW JDBC_CONNSTR LAUNCH_ANYWHERE NET_SERVICE_NAME
   ```

   In the preceding statement, the following parameters were used:

   - `WKSYSPW` is the password for the `WKSYS` schema.
   - `JDBC_CONNSTR` is the JDBC connection string. Use the format `hostname:port:sid`, such as `machine1:1521:iasdb`, if the database is not in the Oracle Real Application Clusters environment.

   If the database is in a Oracle Real Application Cluster environment, then use the TNS keyword-value format instead, because it allows connection to any node of the system:

   ```
   (DESCRIPTION=(LOAD_BALANCE=yes)
    (ADDRESS_LIST=
       (ADDRESS=(PROTOCOL=TCP)(HOST=cls02a)(PORT=3001))
       (ADDRESS=(PROTOCOL=TCP)(HOST=cls02b)(PORT=3001)))
    (CONNECT_DATA=(SERVICE_NAME=sales.us.acme.com)))
   ```

   In the preceding syntax, the following parameters were used:

   - `LAUNCH_ANYWHERE` is the mode of the database. Setting it to `TRUE` indicates that the database is in Oracle Real Application Cluster mode; `FALSE` indicates that the database is not in Oracle Real Application Cluster mode.

- NET_SERVICE_NAME is the network service name used by `wk0config.sql` to establish the database connection. Setting it to ""(empty string) while running `wk0config.sql` from the database host eliminates the need to specify the network service name.

Post-Upgrade Example in Non-Oracle Real Application Clusters Environment

The following is an example of the post-upgrade script for a non-Oracle Real Application Cluster environment:

```
@wk0config.sql welcome1 machine:1521:iasdb FALSE""
```

Post-Upgrade Example in Oracle Real Application Clusters Environment

The following is an example of the post-upgrade script for an Oracle Real Application Clusters environment:

```
@wk0config.sql welcome1
"(DESCRIPTION=(LOAD_BALANCE=yes)
   (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=TCP)(HOST=cls02a)(PORT=3001))
      (ADDRESS=(PROTOCOL=TCP)(HOST=cls02b)(PORT=3001)))
   (CONNECT_DATA=(SERVICE_NAME=sales.us.acme.com)))" FALSE ""
```

4

Using Oracle Ultra Search with Oracle Application Server

Oracle Ultra Search is included with Oracle Database as well as Oracle Application Server. This chapter contains information on using Oracle Ultra Search with Oracle Application Server.

Oracle Ultra Search allows Oracle Application Server Portal (OracleAS Portal) users to search multiple repositories, such as Web pages, files on disk, and public pages on other OracleAS Portal instances. It provides a portlet that can be placed on user portal pages. This portlet can be used to issue a search query and to obtain a list of results.

Oracle Ultra Search resides in the Oracle Application Server Metadata Repository (OracleAS Metadata Repository), a centralized Oracle database on the Oracle Application Server middle tier. The OracleAS Metadata Repository contains all the Oracle Ultra Search database objects, including the actual text index information created during crawling. The Oracle Ultra Search backend is automatically installed on the machine that hosts the OracleAS Metadata Repository.

This chapter contains the following topics:

> **Note:** This chapter contains information specific to using Oracle Ultra Search with Oracle Application Server. For information about all Oracle Ultra Search installations, and for information specific to Oracle Ultra Search with Oracle Database, see Chapter 3, "Installing Oracle Ultra Search".

Oracle Ultra Search Backend with Oracle Application Server

Oracle Ultra Search backend is installed as part of an Oracle Database Server installation. Oracle Application Server middle tier uses the Oracle Ultra Search backend. The Oracle Application Server Metadata Repository resides on the Oracle Database.

The Oracle Ultra Search release depends on the Oracle Database Server release. If the OracleAS Metadata Repository resides on Oracle Database 10*g* release 1 (10.1), then you have Oracle Ultra Search 10*g* release 1 (10.1). If the OracleAS Metadata Repository resides on Oracle9*i* Database Server release 2 (9.2), the you have the Oracle Ultra Search release that shipped with the database.

If Oracle Ultra Search was not installed during the database installation, then an error will occur when creating OracleAS Metadata Repository on top of the database. Install Oracle Ultra Search using the Oracle Universal Installer.

> **See Also:** Oracle Database Server documentation for information about installing Oracle Ultra Search at
>
> `http://www.oracle.com/technology/documentation/`

Oracle Ultra Search Middle Tier with Oracle Application Server

Oracle Ultra Search middle tier is part of the Oracle Application Server installation. You must choose the OracleAS Portal and Wireless option from the Oracle Universal Installer menu to install and configure the Oracle Ultra Search middle tier during the Oracle Application Server installation.

> **See Also:** *Oracle Application Server Administrator's Guide* for information about changing the OracleAS Infrastructure Services, such as specifying a different Oracle Internet Directory or OracleAS Metadata Repository used by an Oracle Ultra Search middle tier

Installing the Middle Tier with Oracle Application Server

Start the Oracle Universal Installer on the relevant host. Choose the destination `ORACLE_HOME` name and full path, and complete the following steps:

1. Select **Oracle Application Server**, and click **Next**.
2. Select **Portal and Wireless,** and click **Next**.
3. On the Configuration Options screen, ensure OracleAS Portal is selected. This option allows the OracleAS Portal Configuration Assistant (OPCA) to configure Oracle HTTP Server and OC4J with Oracle Ultra Search.

   If you deselect this option, then you can use Oracle Enterprise Manager 10*g* to configure OracleAS Portal and OracleAS Wireless.

   Continue with the installation until Oracle Application Server is successfully installed.
4. The Oracle Ultra Search Oracle Application Server query API uses the data source functionality of the J2EE container. Under directory `$ORACLE_HOME/j2ee/OC4J_Portal/config`, edit the file `data-sources.xml`.
5. Restart OC4J_Portal instance using the Oracle Process Manager and Notification Server.

> **See Also:** "Enable the Oracle Ultra Search Query Applications" on page 3-4 for information on editing `data-sources.xml`

Configuring Oracle Ultra Search in a Hosted Environment (Optional)

Oracle Ultra Search is configured to be non-hosted during the default installation. To change to a hosted environment, perform the following steps:

In a hosted environment, one enterprise, such as an application service provider, makes Oracle Ultra Search available to other enterprises and stores information for them. The enterprise performing the hosting is called the **default subscriber**, and the enterprises that are hosted are called **subscribers**.

1. Make sure the hosting mode is enabled.
2. Make sure the subscriber is created in the Oracle Internet Directory server.

   See Also: *OracleAS Portal Configuration Guide* section "Enabling Hosting on an Out-of-Box Portal" for instructions about enabling the hosting mode, and section "Adding Subscribers" for instructions about adding a subscriber to Oracle Application Server Single Sign-On and Oracle Internet Directory

3. Make sure you have execute permission on the `usca.sh` script, and setup the `ORACLE_HOME` environment variable. The Oracle Internet Directory user must have the iASAdmins privilege.
4. For each subscriber, run the `usca` script to configure Oracle Ultra Search in the Oracle Internet Directory subscriber context.

   - For UNIX:

     ```
     ORACLE_HOME/ultrasearch/setup/usca.sh -action add_subscriber \
     -user OID_user_DN -password orcladmin_password -subscriber subscriber_DN
     ```

   - For Windows:

     ```
     ORACLE_HOME\ultrasearch\setup\usca.bat -action add_subscriber \
     -user OID_user_DN -password orcladmin_password -subscriber subscriber_DN
     ```

   The script does the following:

   – Creates the reference objects in the subscriber context.

   – Creates default privilege group entry in the subscriber context.

   – Updates the subscriber information in the Oracle Ultra Search metadata repository.

   The following example illustrates how to configure Oracle Ultra Search in the subscriber dc=us, dc=oracle, dc=com:

   ```
   ORACLE_HOME/ultrasearch/setup/usca.sh -action add_subscriber -user
     'cn=orcladmin' -password welcome1 -subscriber 'dc=us,dc=oracle,dc=com'
   ```

   Note: To drop a subscriber, run the `usca` script to remove Oracle Ultra Search entries from the Oracle Internet Directory subscriber context. For example, the following is an example of the UNIX command:

   ```
   ORACLE_HOME/ultrasearch/setup/usca.sh -action remove_subscriber \
       -user OID_user_DN -password orcladmin_password -subscriber \
       subscriber_DN
   ```

Oracle Ultra Search Administrator Privilege Model in the Hosted Environment

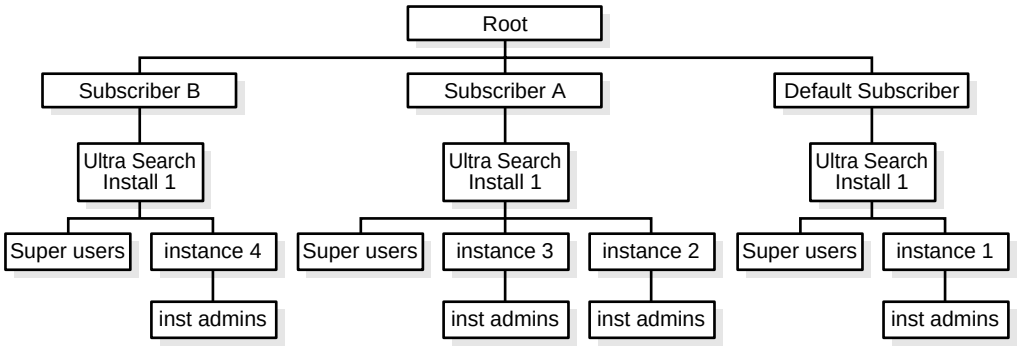
The default subscriber and its search base are specified in the attributes of the Oracle Internet Directory `cn=Common,cn=Products,cn=OracleContext` entry:

- orclDefaultSubscriber
- orclSubscriberSearchBase

In a non-hosted environment, in which there are no subscribers, the enterprise installing Oracle Ultra Search is the default subscriber. All Oracle Ultra Search administration groups (super-user and instance administrator groups) are created under Default Subscriber Oracle Context, such as `cn=OracleContext, dc=us,dc=oracle,dc=com` in the Oracle Internet Directory Information Tree.

Figure 4–1 shows an example of the Oracle Internet Directory topology of a hosted environment. There are two subscribers (A and B) and the default subscriber. Each subscriber has its own super-user privilege group associated with it. There are four Oracle Ultra Search instances created in the Oracle Ultra Search backend Install 1. Instance 1 is associated with the default subscriber. Instance 2 and Instance 3 are associated with Subscriber A. Instance 4 is associated with Subscriber B. Each Oracle Ultra Search instance has its instance administration group associated with it.

Figure 4–1 Oracle Internet Directory Topology of a Hosted Environment



Administration Privilege Model

This section describes the administrator privilege model of Oracle Ultra Search administration tool in the hosted environment. The model applies to both the Oracle Application Server Single Sign-On login mode and the non-single sign-on login mode.

In single sign-on mode, only single sign-on users can log in to the administration tool. Single sign-on users have different privileges depending on the user belonging to the default subscriber or another subscriber. If the single sign-on user belongs to the default subscriber, then the following apply:

- If the single sign-on user has the super-user privilege, then the user can administer all Oracle Ultra Search instances across the default subscriber and any other subscribers (for example, Instances 1, 2, 3, and 4).
- If the single sign-on user has the admin privilege on a particular Oracle Ultra Search instance (for example, Instance 1) within the default subscriber, then the user can administer the instance (Instance 1) that is associated with the default subscriber.

If the single sign-on user belongs to a particular subscriber, then the following apply:

- If the single sign-on user has the super-user privilege, then the user can only administer only Oracle Ultra Search instances within the subscriber. For example, if the user from Subscriber A has the super-user privilege, then the user can only administer Instance 2 and 3.
- If the single sign-on user has the admin privilege on a particular Oracle Ultra Search instance (for example, Instance 2), then the user can administer the instance (Instance 2) that is associated with the subscriber (Subscriber A).

In non-single sign-on mode, only database users can log in to the administration tool.

- If the login database user has the super-user privilege, then the user can administer all Oracle Ultra Search instances across the default subscriber and any other subscribers.
- If the login database user only has the admin privilege on a particular Oracle Ultra Search instance, then the user can administer the instance regardless of whether the instance is associated with the default subscriber or any other subscribers.

Privileges to Create and Drop an Oracle Ultra Search Instance To create or drop an Oracle Ultra Search instance, the user must have the super-user privilege.

In single sign-on mode, the following apply:

- If the login single sign-on user belongs to the default subscriber, then the user can create or drop an instance and associate the instance with any subscriber, including the default subscriber.
- If the login single sign-on user belongs to particular subscriber, then an instance created by the user is associated with the subscriber. Because the user might not have access to create the instance schema in the database, the user must inform the hosting company (default subscriber) to create the database schema for the instance.

In non-single sign-on mode, the database user can create or drop an instance and associate the instance with any subscriber, including the default subscriber.

Privileges to Grant or Revoke a Super-user To grant or revoke a super-user, log in to the administration tool as a super-user.

In single sign-on mode, the following apply:

- If the login single sign-on user belongs to the default subscriber, then the user can do the following:
  - Grant or revoke the super-user privilege to single sign-on users in the default subscriber.
  - Grant or revoke the super-user privilege to single sign-on users in another subscriber.
- If the login single sign-on user belongs to a particular subscriber, then the user can grant or revoke the super-user privilege to users with the same subscriber.

In non-single sign-on mode, only super-users can grant or revoke the super-user privilege to and from other database users.

Privileges to Grant or Revoke an Instance Administrator To grant or revoke an instance administrator, log in to the admin tool as a super-user or an instance administrator.

In single sign-on mode, the following apply:

- The login single sign-on user can grant or revoke only the instance admin privilege to single sign-on users within the same subscriber. For example, the user can grant the admin privilege on Instance 2 or Instance 3 to a single sign-on user in Subscriber A.
- The login single sign-on user cannot grant or revoke the instance admin privilege to single sign-on users within a different subscriber. For example, the user cannot grant the admin privilege on Instance 2 or Instance 3 to a single sign-on user in Subscriber B.

In non-single sign-on mode, only super-users or instance administrators can grant or revoke the instance admin privilege to and from other database users.

5

Oracle Ultra Search Postinstallation Information

This chapter contains the following topics:

Changing Oracle Ultra Search Schema Passwords

There are two Oracle Ultra Search system schemas created during installation: WKSYS and WKPROXY. You can update the schema password in the following way:

With the Oracle Database release, after the database is installed, all user schema accounts are locked. To log on as user WKSYS (or WKPROXY), unlock WKSYS (or WKPROXY) by running the following statement as the SYSTEM or SYS database user:

```
ALTER USER WKSYS ACCOUNT UNLOCK IDENTIFIED BY password;
```

Configuring the Oracle Server for Oracle Ultra Search

The operations described in this section are database administration operations. They can be performed using Oracle Enterprise Manager or SQL*Plus.

Step 1: Tune the Oracle Database

Increase the Size of the Oracle Redo Logs, if necessary

Every instance of an Oracle database has an associated online redo log, which is a set of two or more online log files that record all committed changes made to the database. Online redo logs protect the database in the event of an instance failure. The size of redo log files determines the frequency of redo log file switches. This, in turn, significantly impacts text indexing speed. To reduce the frequency of log file switches, ensure that the redo log files are each 100MB or more.

The following section lists some tips on how to increase the redo log file sizes, if necessary. Enter the statements in the following section with the appropriate Oracle administrator privileges.

See Also:

- *Oracle Database Performance Tuning Guide*
- *Oracle Database Administrator's Guide*

1. Locate redo log files and determine their sizes:

```
SELECT v$logfile.member, v$logfile.group#, v$log.status, v$log.bytes
FROM v$log, v$logfile
WHERE v$log.group# = v$logfile.group#;
```

2. Add larger redo log files:

```
ALTER DATABASE ADD LOGFILE 'redo_log_directory/newredo1.log' size 100m;
ALTER DATABASE ADD LOGFILE 'redo_log_directory/newredo2.log' size 100m;
ALTER DATABASE ADD LOGFILE 'redo_log_directory/newredo3.log' size 100m;
```

   A production database should have more log members for each log group, and different storage devices should be used to increase performance and reliability.

3. Drop the old log files. For each old redo log file, enter the `ALTER SYSTEM SWITCH LOGFILE` statement until that log file's status is `INACTIVE`. This is necessary to ensure that Oracle is not using that log file when you try to drop it.

   Then, drop the old redo log file with the following statement:

```
ALTER DATABASE DROP LOGFILE 'redo_log_directory/redo01.log';
ALTER DATABASE DROP LOGFILE 'redo_log_directory/redo02.log';
ALTER DATABASE DROP LOGFILE 'redo_log_directory/redo03.log';
```

4. Manually delete the old log files from the file system. For each old redo log file, use the appropriate operating system statement to delete the unwanted log file from the file system.

Increase the Size of the Undo Space

Every Oracle database must have a method of maintaining information that is used to roll back, or undo, changes to the database. Such information consists of records of the transactions, primarily before they are committed. Oracle refers to these records collectively as undo. The undo space created by the Oracle Installer may be too small. Oracle recommends that you use automatic undo management and increase the undo space.

See Also: *Oracle Database Administrator's Guide* for details on using automatic undo management

Tune Oracle Initialization Parameters

Set the following values in the initialization file:

- `PROCESSES`: Set this to 50 or more.
- `SORT_AREA_SIZE`: Set this to 5MB or more.
- `SORT_AREA_RETAINED_SIZE`: Set this to 5MB or more.
- `JOB_QUEUE_PROCESSES`: Set this to three or higher. (Set it to at least one.) This is needed because the Oracle Ultra Search crawler is launched by scheduling a database job. If this is zero, then no database jobs are run. As a result, any attempts to launch the Oracle Ultra Search crawler fail. Also consider other requirements for job queue processes when you set this value.

See Also: Oracle Ultra Search Readme for the latest information on initialization parameters relating to Oracle Ultra Search

Step 2: Create and Assign the Temporary Tablespace to the CTXSYS User

The starter database created by the Oracle Installer may create a temporary tablespace that is too small. Oracle Ultra Search uses the Oracle Text engine intensively. Therefore, a large temporary tablespace must be created for the Oracle Text system user CTXSYS. If you want greater read and write performance, create the tablespace on raw devices.

When you have created the temporary tablespace, assign it as the temporary tablespace for the CTXSYS user. To do so, you must log on as the SYSTEM or SYS user. Assign the temporary tablespace to the CTXSYS user with the following statement:

```
ALTER USER CTXSYS TEMPORARY TABLESPACE new_temporary_tablespace;
```

See Also: *Oracle Database Administrator's Guide* for information on how to create a temporary tablespace

Step 3: Create a Large Tablespace for Each Oracle Ultra Search Instance User

For each Oracle Ultra Search instance, you must create a tablespace large enough to contain all data obtained during the crawling and indexing processes. This amount is subject to the amount of data you intend to crawl and index. However, it is often not possible to know in advance how much data you intend to collect. Try to obtain an estimate of the cumulative size of all data you want to crawl.

If you cannot estimate the size, then try to allocate as much space as possible. If you run out of disk space, then Oracle Ultra Search is able to resume crawling after you add more datafiles to the instance tablespace.

Here is an example of how to create a new tablespace:

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 150M;
```

Pay attention to the STORAGE clause in your CREATE TABLESPACE statement. The amount of data to be stored in the tablespace can be very large. This can cause the Oracle server to progressively allocate many new extents when more storage space is needed. If the extent management clause specifies that each new extent is to be larger than the previous extent (that is, the PCTINCREASE setting is nonzero), then you could encounter the situation where the next extent that the Oracle server wants to allocate is larger than what is available. In such a situation, indexing halts until new extents can be added to the tablespace.

To mitigate this problem, certain instance-specific tables have explicit storage parameter settings. The initial extent size, next extent size, and PCTINCREASE setting are defined for these tables. These tables are created when a new instance is created. The tables and their storage clause settings are as follows:

```
DR$WK$DOC_PATH_IDX$I
               (initial extent size 5M, next extent size 50M, PCTINCEASE 1)
DR$WK$DOC_PATH_IDX$K
               (initial extent size 5M, next extent size 50M, PCTINCEASE 1)
```

If you want greater read and write performance, create the tablespace on raw devices.

Be sure to create a new large tablespace for each Oracle Ultra Search instance user.

See Also:

- *Oracle Database SQL Reference* for more information on creating tablespaces and managing storage settings
- *Oracle Database Administrator's Guide* for information on how to create a tablespace

Step 4: Create and Configure New Users for Oracle Ultra Search Instances

Oracle Ultra Search uses Oracle's fine grained access control feature to support multiple Oracle Ultra Search instances within one physical database. This is especially useful for large organizations or application service providers (ASPs) that want to host multiple disjoint search indexes within one installation of Oracle.

Note: Oracle Ultra Search requires that each Oracle Ultra Search virtual instance belong to a unique database user. Therefore, as part of the installation process, you must create one or more new database users to own all data for your Oracle Ultra Search instance.

If you intend to create more than one database instance, you should also create multiple user tablespaces: one for each user.

You must grant the WKUSER role to database users hosting new Oracle Ultra Search instances.

See Also: "Users Page" on page 8-34

Enter the following statements to create and configure a new user. Run these statements as the WKSYS, SYSTEM, or SYS database user.

```
CREATE USER username
             IDENTIFIED BY password DEFAULT TABLESPACE default_tbs
             TEMPORARY TABLESPACE temporary_tbs QUOTA UNLIMITED
             ON default_tbs;
```

where *username* = name of the Oracle Ultra Search instance owner

and *password* = password of the Oracle Ultra Search instance owner

and *default_tbs* = default tablespace for the Oracle Ultra Search instance created in Step 3

and *temporary_tbs* = temporary tablespace created in Step 2

```
GRANT WKUSER TO username;
```

After these steps are completed, WKSYS or an Oracle Ultra Search super-user can create an Oracle Ultra Search instance on this user schema.

If you want this user to have the general administrative privilege or the super-user privilege, then log on as an Oracle Ultra Search super-user or WKSYS and go to the **Users** page in the administration tool to grant the appropriate privilege.

Step 5: Alter the Index Preferences

This step is optional.

An empty index is created when an Oracle Ultra Search instance is created. The existing index preferences, such as language-specific parameters, are defined in the `$ORACLE_HOME/ultrasearch/admin/wk0pref.sql` file.

You can modify these preferences so that all new Oracle Ultra Search instances use the modified preferences, or you can alter the index using your own preferences immediately after an instance is created. Alter the index using SQL.

Note: The crawler transforms all documents into HTML files with binary document filtering before indexing begins.

See Also:

- *Oracle Text Application Developer's Guide*
- *Oracle Text Reference*

Configuring Oracle Ultra Search for SSL

Starting with Oracle Database 10g, Oracle Ultra Search supports secure socket layer (SSL). This means that in addition to HTTP-based URLs, Oracle Ultra Search can also access HTTPS-based URLs (that is, HTTP over SSL).

Ultra Search treats SSL as a service of the JVM; thus, any customization, tuning, or configuration of the SSL should be done on the JRE level.

Oracle Ultra Search uses the SSL service provided by Sun Microsystem's JDK. Sun's SSL implementation lets you easily replace the default Trust Manager, Key Manager, and truststore with the ones you choose.

Truststore is a list of public SSL certificates identifying entities with which a peer is allowed to communicate over SSL. A truststore needs to be maintained. You may need to update expired certificates with new issues, or you may want to add new certificates to the truststore, effectively adding new entities to the list of ones you are willing to communicate with over SSL.

For example, say you want to service a secure Oracle Portal installation using Oracle Ultra Search. And say your Oracle Portal installation does not have a certificate issued by one of the well known certificate authorities like Verisign; rather, it uses a self-signed certificate issued by you. That means that your Oracle Portal's certificate will not be in the truststore(s) used by Oracle Ultra Search. Consequently, you will need to add to the truststore the certificate identifying your Oracle Portal installation.

Besides third party tools, Sun provides its own truststore management tool called *keytool*. You can use keytool to add, update, remove, and import certificates to and from Sun's truststore. Keytool also lets you create your own self-signed certificates.

Keep in mind that there might be more then one truststore that needs updating. The Oracle Ultra Search crawler and the midtier can use different JDK installations (for example, when the two are on separate tiers, as with Oracle Application Server or Oracle Collaboration Suite deployments). Each has its own separate truststore, which would need to be updated.

See Also: Sun Microsystems documentation, including the JSSE User's Guide, for more information about using Sun's keytool key and certificate management utility, for information on customization of the SSL service, and for information on truststore management

Managing Stoplists

Every Oracle Ultra Search instance has a stoplist associated with it. A stoplist is a list of words that are ignored during the indexing process. These words are known as stopwords. Stopwords are not indexed because they are deemed not useful, or even disruptive, to the performance and accuracy of indexing.

Default Oracle Ultra Search Stoplist

During the installation process, a default stoplist is created for the Oracle Ultra Search product. Subsequently, when an Oracle Ultra Search instance is created, a copy of the default stoplist is created for the Oracle Ultra Search instance.

The default stoplist is created under the `WKSYS` schema. The default stoplist name is `wk_stoplist`. (This list is defined in the file `$ORACLE_HOME/ultrasearch/admin/wk0pref.sql`, which is run at installation).

Modifying Instance Stoplists

Modify the default stoplist by adding or removing stopwords from it. However, remember that these modifications do not affect existing Oracle Ultra Search instances. They only affect Oracle Ultra Search instances that are created after the modifications are made.

Modifying instance stoplists should be done as a last resort. Use one of the following methods:

- Modify the default stoplist before creating the instance.
- Replace the instance stoplist immediately after creating the instance.

Replacing the instance stoplist immediately after creating the instance affects only that instance. You must first create a user-defined stoplist.

In both cases, the result is that the Oracle Ultra Search instance stoplist is modified and defined before initial crawling. This means that all documents collected by the Oracle Ultra Search crawler are evaluated against the correct stoplist. It is important to modify the stoplist before initial crawling to avoid having to recrawl all documents.

Modifying Instance Stoplists Before Initial Crawling

1. Modify the default stoplist before creating the instance.

   For example, to add the stopword "web" to the default stoplist, log on as user `WKSYS` in SQL*Plus, and run the following statement:

   ```
   EXEC ctx_ddl.add_stopword('wk_stoplist','web');
   ```

   To remove the stopword "web" from the default stoplist, log on as user `WKSYS` in SQL*Plus, and run the following statement:

   ```
   EXEC ctx_ddl.remove_stopword('wk_stoplist','web');
   ```

   Subsequently, the stoplists of all new instances reflect the modifications made to the default stoplist.

2. Replace the instance stoplist immediately after creating the instance:

   You must create a new user-defined stoplist. Log on as the owner of the instance in SQL*Plus, and run the following statements:

```
BEGIN   ctx_ddl.create_stoplist('example_stoplist');
        ctx_ddl.add_stopword('example_stoplist','example_stopword');
        ... (add more stopwords by repeated the previous
        line with new stopwords) ...
END;
/
```

To replace an instance stoplist with this new stoplist, log on as the owner of the instance in SQL*Plus, and run the following statement:

```
ALTER INDEX wk$doc_path_idx rebuild parameters('replace stoplist example_
stoplist');
```

> **See Also:** "Changing Oracle Ultra Search Schema Passwords" on page 5-1 for information about changing the WKSYS password

Modifying Instance Stoplists After Initial Crawling

If necessary, alter an instance stoplist after initial crawling with one of the following methods:

1. Add stopwords to the instance stoplist:

   Choosing to add stopwords to the instance stoplist does not affect any documents already crawled or indexed. This is not an expensive operation.

   For example, to add the stopword "web" to the instance stoplist, log on as the owner of the instance in SQL*Plus, and run the following statement:

   ```
   ALTER INDEX wk$doc_path_idx rebuild parameters('add stopword web');
   ```

2. Replace the instance stoplist after initial crawling:

   Defining a new stoplist and replacing the instance stoplist with it invalidates the entire index. If you choose this method, you must force the Oracle Ultra Search crawler to recrawl all documents in the index. To do this, click **Process All Documents** in the **Edit Schedule** page. This is a very expensive operation. Therefore, this option should be the last resort.

Configuring the Query Application

The Oracle Ultra Search query application is deployed automatically with the Oracle Ultra Search installation. However, because Oracle Ultra Search allows multiple instances using different schema users, the query application is not configured for how to connect to the database automatically. Database connection is configured by creating a data source in OC4J (not to be confused with an Oracle Ultra Search data source). This is done by editing the `data-sources.xml` file.

> **See Also:** "Enable the Oracle Ultra Search Query Applications" on page 3-4

Oracle Ultra Search lets multiple instances use different schema users, so multiple query applications can co-exist on the same database. Each query application requires its database connection information to be defined with `data-sources.xml`. They must be defined to have different location values, such as `jdbc/UltraSearchPooledDS1`, `jdbc/UltraSearchPooledDS2`, and so on. Correspondingly, the query application must be deployed multiple times in OC4J.

Each application deployment must be configured to use the correct entry in `data-sources.xml`. This is done by editing the JSP source for query. For the

complete search application, edit `common_customize_instance.jsp` and edit the following line to use the correct location value:

```
String m_datasource_name = "jdbc/UltraSearchPooledDS"
```

6

Security in Oracle Ultra Search

The ability to control user access to Web content is critical. This chapter describes the architecture and configuration of security for Oracle Ultra Search.

This chapter contains the following sections:

- About Oracle Ultra Search Security
- Configuring a Security Framework for Oracle Ultra Search
- Enabling Secure Search

See Also:

- *Oracle Application Server Security Guide* for an overview of Oracle Application Server security and its core functionality
- *Oracle Identity Management Concepts and Deployment Planning Guide* for guidance on OracleAS Infrastructure security

About Oracle Ultra Search Security

This section describes the Oracle Ultra Search security model. It contains the following sections:

- Oracle Ultra Search Security Model
- Oracle Ultra Search with Secure Socket Layer and HTTPS
- Classes of Users and Their Privileges
- Resources Protected by Oracle Ultra Search
- Authorization and Access Enforcement
- How Oracle Ultra Search Leverages Security Services
- How Oracle Ultra Search Leverages the Oracle Identity Management Infrastructure
- Oracle Ultra Search Extensibility and Security

Oracle Ultra Search Security Model

Security problems, such as unauthorized access to information, can lead to loss of productivity. Search engines like Oracle Ultra Search provide access to a vast variety of content repositories in a single gateway. Each one of these repositories has its own security model that determines whether a particular user can access a particular document. Because Oracle Ultra Search provides access to data from multiple

repositories, existing security information in each repository must be carefully supported to avoid unauthorized access.

This section describes the security architecture of Oracle Ultra Search. Security is implemented at the following levels:

- User authentication

  This is the identification of a user, through LDAP and Oracle Internet Directory, at Oracle Ultra Search front-end interfaces.

- User entitlement

  This determines whether a user can access information about a particular item in the results list. It is implemented by access control lists (ACLs). Oracle Ultra Search provides mapped-security to third-party repositories by retrieving the access control list for each document at the time of indexing and storing them in Oracle Ultra Search. Oracle Ultra Search does not need any connection with the repository itself to validate access privileges.

- Security of Oracle Ultra Search

  Actual Oracle Ultra Search security is handled by the dictionary data in the Oracle Ultra Search database, the administrative user, and password data.

Oracle Ultra Search with Secure Socket Layer and HTTPS

Starting with Oracle Database 10g, Oracle Ultra Search supports secure socket layer (SSL). This means that in addition to HTTP-based URLs, Oracle Ultra Search can also access HTTPS -based URLs (that is, HTTP over SSL).

> **See Also:** "Configuring Oracle Ultra Search for SSL" on page 5-5 for detailed information on configuring Oracle Ultra Search with SSL

Classes of Users and Their Privileges

To grant an Oracle Ultra Search user administration privileges, you must assign the user to an administration group. Each user can belong to one or more groups. The following groups are created for each Oracle Ultra Search instance:

1. Instance administrators: Users in this group can only manage instances for which they have privileges.
2. Super-users: Users in this group can manage all instances, including creating instances, dropping instances, and granting privileges.

Oracle Ultra Search also has two classes of users:

1. Single Sign-on users: These users are managed by the Oracle Internet Directory and are authenticated by OracleAS Single Sign-On. The Oracle Ultra Search administration tool identifies all Oracle Ultra Search instances to which the single sign-on user has access. This is available only if you have the Oracle Identity Management infrastructure installed.
2. Database users: These users (not single sign-on users) exist in the database on which Oracle Ultra Search runs.

Oracle Ultra Search Default Users

New Oracle Ultra Search instances contain the following users:

- WK_TEST: This is the instance administrator user that hosts the default instance, called WK_INST. In other words, WK_TEST is the instance administrator for WK_INST. For security purposes, WK_TEST is locked after the installation. The administrator should login to the database as DBA role, unlock the WK_TEST user account, and set the password to be WK_TEST. (The password expires after the installation.) If you change the password to anything other than WK_TEST, then you must also update the cached schema password using the administration tool **Edit Instance** page after you change the password in the database.
- WKSYS: This is a database super-user. WKSYS can grant super-user privileges to other users, such as WK_TEST. All Oracle Ultra Search database objects are installed in the WKSYS schema.

Note: The WKUSER role is required to host instances.

Resources Protected by Oracle Ultra Search

All publicly crawled data is publicly accessible.

The following resources are protected by Oracle Ultra Search:

- Crawled data that uses an access control list (ACL) is protected; in other words, it is private to users named by the ACL.
- All passwords are protected.
- User-defined data source parameters are protected.

Authorization and Access Enforcement

There are three possible entry points to Oracle Ultra Search:

1. The database: This contains all data. All data and metadata is protected with row level security. All passwords are encrypted.
2. The Oracle Ultra Search administration tool: This does not contain crawled data. You must authenticate with OracleAS Single Sign-On or database authentication.
3. The Oracle Ultra Search query tool: This contains crawled data. Unauthenticated users can see only public data. Authenticated users can see public data and ACL-protected information. Users must authenticate themselves to see private information.

How Oracle Ultra Search Leverages Security Services

Oracle Ultra Search uses the following to leverage security services:

- Oracle Ultra Search uses secure socket layers (SSL), the industry standard protocol for managing the security of message transmission on the Internet. This is used for securing RMI connections, HTTPS crawling, and secure JDBC.
- JAZN: Oracle Application Server Containers for J2EE (OC4J) implements a Java authentication and authorization service (JAAS) provider called JAZN. This provides application developers with user authentication, authorization, and delegation services to integrate into their application environments.

See Also: "Enabling Secure Search" on page 6-4

How Oracle Ultra Search Leverages the Oracle Identity Management Infrastructure

Oracle Ultra Search uses OracleAS Single Sign-On and Oracle Internet Directory to leverage the Oracle Identity Management infrastructure.

With OracleAS Single Sign-On, you can log on once for all components, and the Oracle Ultra Search administrative interface allows user management operations on either database users or single sign-on users. Authenticated single sign-on users never see the Oracle Ultra Search logon screen. Instead, they can immediately choose an instance. The Oracle Ultra Search administration tool and the query tool use single sign-on.

Oracle Internet Directory is Oracle's native LDAP v3-compliant directory service, built as an application on top of the Oracle database. Oracle Internet Directory hosts the Oracle common identity. All Oracle Ultra Search instances are registered with Oracle Internet Directory.

> **See Also:** "Integration with Oracle Internet Directory" on page 1-10

Oracle Ultra Search has native identity management; therefore, in the absence of the Oracle Identity Management infrastructure, Oracle Ultra Search uses the native user management available with the Oracle database.

Oracle Ultra Search Extensibility and Security

Oracle Ultra Search is extensible (for example, the crawler agent is extensible), but this poses no extra security considerations.

Configuring a Security Framework for Oracle Ultra Search

This section describes special security configuration steps within Oracle Ultra Search.

Configuring Security Framework Options for Oracle Ultra Search

Storing clear text passwords in `data-sources.xml` poses a security risk. Avoid this by using password indirection to specify the password. This lets you enter the password in `jazn-data.xml`, which is automatically encrypted, and point to it from `data-sources.xml`.

> **See Also:**
>
> - "Enable the Oracle Ultra Search Query Applications" on page 3-4
> - *Oracle Application Server Containers for J2EE Services Guide*

Enabling Secure Search

Oracle Ultra Search supports secure searches, which return only documents satisfying the search criteria that the search user is allowed to view. For secure searches, each indexed document should be protected by an access control list (ACL). During searches, the ACL is evaluated. If the user performing the search has permission to read the protected document, then the document is returned by the query API. Otherwise, it is not returned.

This section describes how to enable secure search.

Step 1: Check the database version requirements and configure Oracle Identity Management.

Before you can set up a secure Oracle Ultra Search installation, you must do the following:

- Install or upgrade the Oracle database to 9.2.0.4 or higher. The middle tier and IM (identity management) version should be 9.0.4 or higher. If you have a 9.2.0.4 database, you can use `RepCA` to convert a 9.2.0.4 database to an Oracle Application Server 9.0.4 metadata repository.
- Install and configure the Oracle Internet Directory

  The middle tier and IM (identity management) version should be 9.0.4 or higher.
- Register the database to Oracle Internet Directory.

  You can use `repCA` to register the database to Oracle Internet Directory. After registration, you need to perform these manual steps:

  – Add the distinguished name of the database to the database server parameter file, as an `RDBMS_SERVER_DN` initialization parameter value.

  – Restart the database, so that the new initialization parameter takes effect.
- Configure the Oracle-Oracle Internet Directory SSL link and to establish a secure connection between the database and Oracle Internet Directory

> **See Also:** *Oracle Internet Directory Application Developer's Guide* for information on configuring Oracle Internet Directory for SSL and *Oracle Advanced Security Administrator's Guide* for information on configuring the database for SSL

Secure search functionality requires that the Oracle Ultra Search database is Oracle version 9.2.0.4 or higher and that the Oracle Ultra Search database is linked to a compatible instance of Oracle Internet Directory. This is necessary because Oracle Ultra Search utilizes XML DB functionality, which requires a certain version of Oracle. Oracle XML DB, in turn, requires a live link to Oracle Internet Directory, through which it retrieves all LDAP principal information. The connection between Oracle and Oracle Internet Directory must be running at all times for secure search to work. To set up this link, configure the Oracle Database to use Oracle Identity Management.

Step 2: Restart the Oracle listener.

In the previous step, you configured the Oracle Database to use Identity Management. That process involved configuring `ORACLE_HOME` for directory usage. You must make sure to restart the Oracle listener to inherit the changes made to `ORACLE_HOME`. Restart the listener, if you have not already done so.

Step 3: Install or upgrade Oracle Ultra Search, if necessary.

After you have configured the Oracle Ultra Search database to work with Oracle Internet Directory, you can install or upgrade the Oracle Ultra Search backend into the Oracle Server, if you have not already done so.

Step 4: Create the /sys/apps/ultrasearch folder.

Immediately after installation or upgrade, you must run a SQL script to create the `/sys/apps/ultrasearch` folder in the XML DB repository. This folder stores all Oracle Ultra Search ACLs in XML DB.

To create the `/sys/apps/ultrasearch` folder, do the following:

1. cd to `$ORACLE_HOME/ultrasearch/admin`
2. Login to the Oracle Ultra Search database using SQL*Plus as user `WKSYS`
3. Invoke the SQL script: `@wk0prepxdb.sql`

> **See Also:** "Changing Oracle Ultra Search Schema Passwords" on page 5-1 for information on changing the `WKSYS` password

Upon termination, the `wk0prepxdb.sql` script lists all Oracle Ultra Search-related XML DB resources by running the following SQL:

```
SELECT any_path FROM resource_view WHERE any_path LIKE '%ultrasearch%';
```

Running that SQL statement must show two rows:

```
/sys/apps/ultrasearch
/sys/apps/ultrasearch_acl.xml
```

If you do not see this confirmation, then this step has failed, and you cannot proceed. Recheck that all previous steps were performed correctly.

Step 5: Turn on secure search functionality in Oracle Ultra Search.

Because there is currently no way to programatically verify a proper Oracle-Oracle Internet Directory installation, the secure search functionality in Oracle Ultra Search is turned off by default. You must explicitly turn on this feature after completing all previous steps.

Step 6: Turn On Secure Search in the Query Application.

To turn on secure search functionality in Oracle Ultra Search:

1. Login to the Oracle Ultra Search database using SQL*Plus as user `WKSYS`
2. Invoke the following PL/SQL API: `exec WK_ADM.SET_SECURE_MODE(1)`

The argument `(1)` indicates that you are turning on secure search.

After you have turned on secure search functionality, you can create Oracle Ultra Search instances that are secure search-enabled.

Note: At any subsequent point in time, you can turn off security by invoking `WK_ADM.SET_SECURE_MODE(0)`. Doing so designates that any instances created after that will not support secure searches. However, existing secure search-enabled instances are not modified. Hence, if the Oracle-Oracle Internet Directory link ceases to function, you cannot perform searches on crawled documents that are secured.

Oracle Ultra Search supports secure searches, which return only documents satisfying the search criteria that the search user is allowed to view.

To turn on secure search in the query application, follow these steps:

1. Deploy Oracle Ultra Search query (`ultrasearch_query.ear`).
2. Edit the OC4J `jazn.xml` file to connect to Oracle Internet Directory. For example:

```
<jazn provider="LDAP" default-realm="us" location="ldap://localhost:3060">
     <property name="ldap.user" value="orcladmin"/>
     <property name="ldap.password" value="!welcome"/>
     </jazn>
```

3. Restart OC4J.
4. Edit `applications/ultrasearch_query/META-INF/orion-application.xml` to turn on JAZN LDAP.
5. Edit `applications/ultrasearch_query/query/WEB-INF/web.xml` to enable login functionality in `usearch.jsp`. For example:

```
<init-param>
<param-name>login enabled</param-name>
<param-value>true</param-value>
</init-param>
```

6. Enable `mod_osso` in Apache.
7. Access `http://`*`hostname`*`:`*`port`*`/ultrasearch/query/usearch.jsp` to see the login function, and test secure search.

See Also: "Secure Search" on page 1-7

7

Understanding the Oracle Ultra Search Crawler and Data Sources

This chapter contains the following topics:

See Also: "Tuning Query Performance" on page 10-2

Overview of the Oracle Ultra Search Crawler

The Oracle Ultra Search crawler is a Java process activated by your Oracle server according to a set schedule. When activated, the crawler spawns processor threads that fetch documents from various data sources. These documents are cached in the local file system. When the cache is full, the crawler indexes the cached files using Oracle Text. This index is used for querying.

Note: An empty index is created when an Oracle Ultra Search instance is created. You can alter the index using SQL. The existing preferences, such as language-specific parameters, are defined in the `$ORACLE_HOME/ultrasearch/admin/wk0pref.sql` file.

Crawler Settings

Before you can use the crawler, you must set its operating parameters, such as the number of crawler threads, the crawler timeout threshold, the database connect string, and the default character set. To do so, use the **Crawler Settings Page** in the administration tool.

At installation, the Oracle Installer automatically sets the variable to include `$ORACLE_HOME/ctx/lib`. However, if you restart the database after the installation,

then you must manually set your shared library path environment variable to include `$ORACLE_HOME/ctx/lib` before starting the Oracle process. You must restart the database to pick up the new value for filtering to work.

For example, on UNIX set the `$LD_LIBRARY_PATH` environment variable to include `$ORACLE_HOME/ctx/lib`, and on Windows set the `$PATH` environment variable to include `$ORACLE_HOME/bin`.

> **See Also:** "Crawler Page" on page 8-9 for more information on crawler settings and Web Sources on page 8-18 for more information on settings such as robots exclusion, the UrlRewriter, indexing dynamic Web pages, and HTTP cookies

Crawler Data Sources

In addition to the Web access parameters, you can define specific data sources on the Sources page in the administration tool. You can define one or more of the following data sources:

- Web sites
- Database tables
- Files
- Mailing lists
- Oracle Application Server Portal page groups
- User-defined data sources (requires crawler agent)

Using Crawler Agents

If you are defining a user-defined data source to crawl and index a proprietary document repository or management system, such as Lotus Notes or Documentum, then you must implement a crawler agent as a Java class. The agent collects document URLs and associated metadata from the proprietary document source and returns the information to the Oracle Ultra Search crawler, which enqueues it for later crawling. For more information on defining a new data source type, see the User-Defined subtab in Sources page in the administration tool.

Synchronizing Data Sources

You can create synchronization schedules with one or more data sources attached to it. Synchronization schedules define the frequency at which the Oracle Ultra Search index is kept up to date with existing information in the associated data sources. To define a synchronization schedule, use the Sources page in the administration tool.

Display URL and Access URL

For some applications, for security reasons, the URL crawled is different from the one seen by the end user. For example, crawling on an internal Web site inside a firewall might be done without security checking, but when queried by the user, a corresponding mirror URL outside the firewall must be used. This mirror URL is called the display URL.

By default, the display URL is treated as the access URL unless a separate access URL is provided. The display URL must be unique in a data source; so two different access URLs cannot have the same display URL.

See Also: "Sources Page" on page 8-17

Document Attributes

Document attributes, or metadata, describe the properties of a document. Each data source has its own set of document attributes. The value is retrieved during the crawling process, then mapped to one of the search attributes, and then stored and indexed in the database. This lets you query documents based on their attributes. Document attributes in different data sources can be mapped to the same search attribute. Therefore, you can query documents from multiple data sources based on the same search attribute.

If the document is a Web page, the attribute can come from the HTTP header or it can be embedded inside the HTML in metatags. Document attributes can be used for many things, including document management, access control, or version control. Different data sources can have different attribute names which are used for the same idea; for example, "version" and "revision". It can also have the same attribute name for different ideas; for example, "language" as in natural language in one data source but as programming language in another.

Oracle Ultra Search has the following default search attributes: Title, Author, Description, Subject, Mimetype, Language, Host, and LastModifedDate. They can be incorporated in search applications for a more detailed search and richer presentation.

Search attributes can also be created in the following ways:

- System-defined search attributes, such as title, author, description, subject, and mimetype
- Search attributes created by the system administrator
- Search attributes created by the crawler. (During crawling, the crawler agent maps the document attribute to a search attribute with the same name and data type. If not found, then the crawler creates a new search attribute with the same name and type as the document attribute defined in the crawler agent.)

The list of values (LOV) for a search attribute can help you specify a search query. If attribute LOV is available, then the crawler registers the LOV definition, which includes attribute value, attribute value display name, and its translation.

Crawling Process for the Schedule

The first time the crawler runs, it must fetch Web pages, table rows, files, and so on based on the data source. It then adds the document to the Oracle Ultra Search index. The crawling process for the schedule is broken into two phases:

1. Queuing and Caching Documents
2. Indexing Documents

Queuing and Caching Documents

Figure 7–1 on page 7-5 and Figure 7–2 on page 7-5 illustrate an instance of the crawling cycle in a sequence of nine steps. The example uses a Web data source, although the crawler can also crawl other data source types.

Figure 7–1 illustrates how the crawler and its crawling threads are activated. It also shows how the crawler queues hypertext links to control its navigation. This figure corresponds to Steps 1 through 5.

Figure 7–2 illustrates how the crawler caches Web pages. This figure correspond to Steps 6 through 8.

The steps are the following:

1. Oracle spawns the crawler according to the schedule you specify with the administration tool. When crawling is initiated for the first time, the URL queue is populated with the seed URLs. See Figure 7–1.
2. The crawler initiates multiple crawling threads.
3. The crawler thread removes the next URL in the queue.
4. The crawler thread fetches the document from the Web. The document is usually an HTML file containing text and hypertext links.
5. The crawler thread scans the HTML file for hypertext links and inserts new links into the URL queue. Duplicate links already in the document table are discarded.
6. The crawler caches the HTML file in the local file system. See Figure 7–2 on page 7-5.
7. The crawler registers URL in the document table.
8. The crawler thread starts over by repeating Step 3.

Fetching a document, as described in Step 4, can be time-consuming because of network traffic or slow Web sites. For maximum throughput, multiple threads fetch pages at any given time.

Note: URLs remain visible until the next crawling run. When the crawler detects that the URL is no longer there, it is removed from the `wk$doc` table where Oracle Text automatically marks this document as deleted, even though the index data still exists. Cleanup is done through index optimization, which can be scheduled separately.

Figure 7–1 Queuing URLs

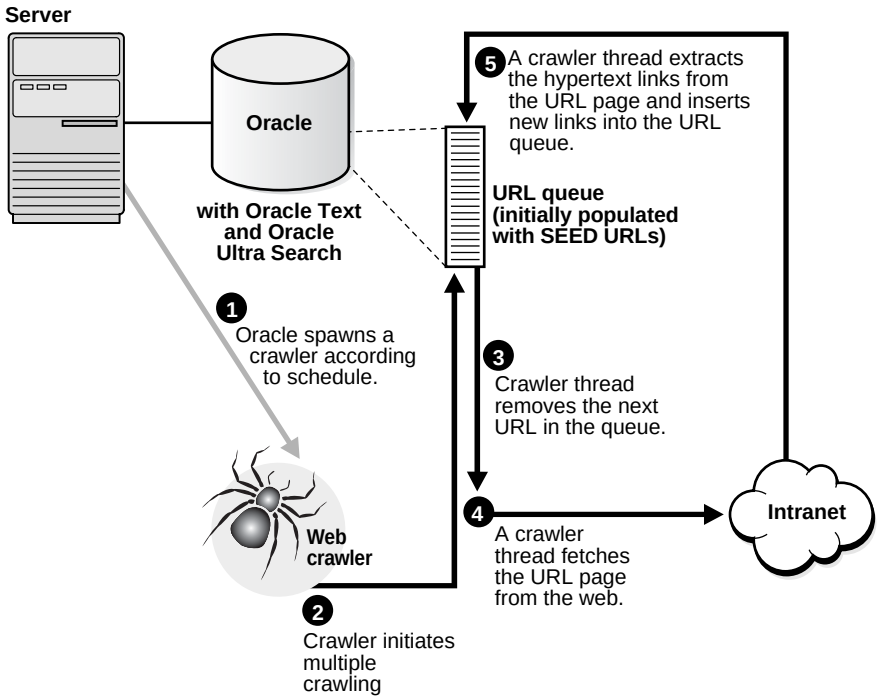
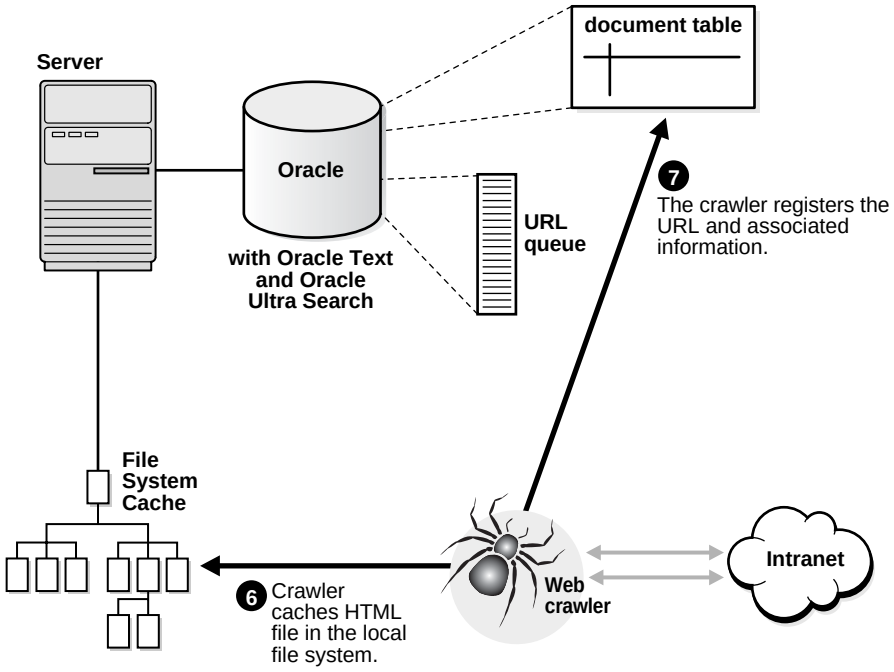


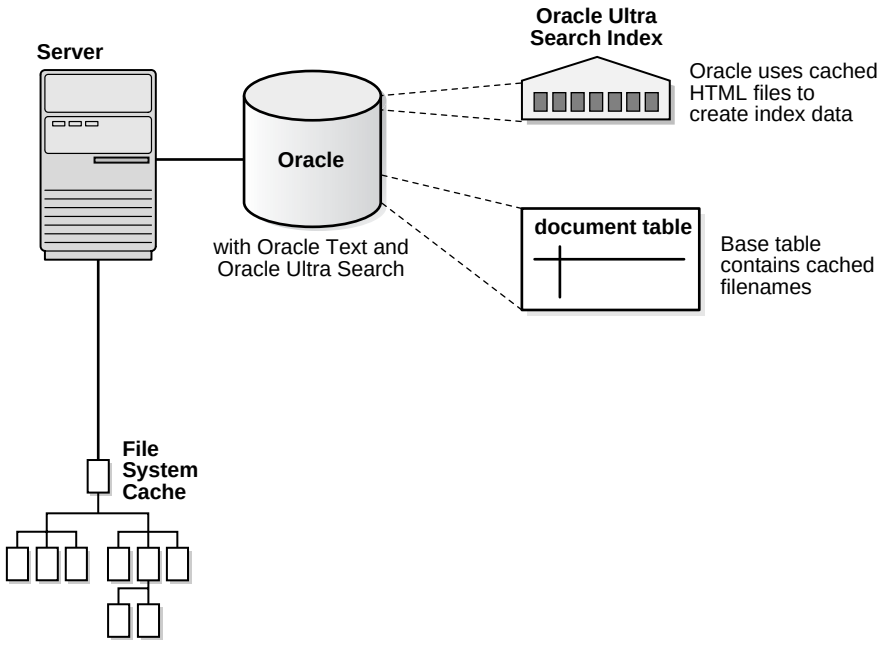
Figure 7–2 Caching URLs



Indexing Documents

When the file system cache is full (default maximum size is 20 MB), document caching stops and indexing begins. In this phase, Oracle Ultra Search augments the Oracle Text index using the cached files referred to by the document table. See Figure 7–3.

Figure 7–3 Indexing Documents



Data Synchronization

After the initial crawl, a URL page is only crawled and indexed if it has changed since the last crawl. The crawler determines if it has changed with the HTTP If-Modified-Since header field or with the checksum of the page. URLs that no longer exist are marked and removed from the index.

To update changed documents, the crawler uses an internal checksum to compare new Web pages with cached Web pages. Changed Web pages are cached and marked for reindexing.

The steps involved in data synchronization are the following:

1. Oracle spawns the crawler according to the synchronization schedule you specify with the administration tool. The URL queue is populated with the data source URLs assigned to the schedule.
2. The crawler initiates multiple crawling threads.
3. Each crawler thread removes the next URL in the queue.
4. Each crawler thread fetches the document from the Web. The page is usually an HTML file containing text and hypertext links.
5. Each crawler thread calculates a checksum for the newly retrieved page and compares it with the checksum of the cached page. If the checksum is the same, then the page is discarded and the crawler goes to Step 3. Otherwise, the crawler moves to the next step.
6. Each crawler thread scans the document for hypertext links and inserts new links into the URL queue. Links that are already in the document table are discarded.
7. The crawler caches the document in the local file system. See Figure 7–2.
8. The crawler registers the URL in the document table.

9. If the file system cache is full or if the URL queue is empty, then Web page caching stops and indexing begins. Otherwise, the crawler thread starts over at Step 3.

Web Crawling Boundary Control

Oracle Ultra Search provides the following mechanisms to control the scope of a Web data source crawling:

- URL boundary rule (domain rule and path rule)
- `Robots.txt` file and robots `META` tag
- Crawling depth
- URL Rewriter API

URL Boundary Rule

The URL boundary rule consists of domain rules and path rules. A domain rule specifies the set of Web sites allowed using a host name prefix or suffix. A path rule specifies the URL file path allowed or disallowed for a particular host. You can specify an inclusion or exclusion rule for both a domain rule and a path rule. Exclusion rules always override inclusion rules. Path rules are always host-specific.

For example, an inclusion domain ending with `oracle.com` limits the Oracle Ultra Search crawler to hosts belonging to Oracle world wide. Anything ending with `oracle.com` is crawled, but `http://www.oracle.com.tw` is not crawled. If you change the inclusion domain to `someurl.com` with a new seed `http://www.someurl.com`, then all `oracle.com` URLs are dropped by the crawler.

An exclusion domain `uk.oracle.com` prevents the crawler from crawling Oracle hosts in the United Kingdom. You can also include or exclude Web sites with a specific port. (By default, all ports are crawled.) You can have port inclusion or port exclusion rules for a specific host, but not both.

All URLs must pass domain rules before being checked for path rules. Path rules let you further restrict the crawling space. Path rules are host-specific, but you can specify more than one path rule for each host. For example, on the same host, you can include the path `/host/doc` and exclude the path `/host/doc/private`. Note that path rules are prefix-based.

Regular expression-based domain and path rules are not supported in the current release.

The following rules restrict the crawler to only crawl `www.oracle.com` and `otn.oracle.com`. Furthermore, only URLs under `/products/database/` and `/products/ias/` but not under `/products/ias/web_cache/` will be crawled.

```
Domain inclusion: www.oracle.com
Domain inclusion: otn.oracle.com
Path inclusion for otn.oracle.com:        /products/database/
                                          /products/ias/
Path exclusion for otn.oracle.com:        /products/ias/web_cache/
```

robots.txt Protocol and robots Metatag

The `robots.txt` protocol is the webmaster's path rule for any spider or crawler that visits his or her Web site.

The following sample `/robots.txt` file specifies that no robots should visit any URL starting with `/cyberworld/map/` or `/tmp/`, or `/foo.html`:

```
# robots.txt for http://www.acme.com/

User-agent: *
Disallow: /cyberworld/map/
Disallow: /tmp/
Disallow: /foo.html
```

By default, the Oracle Ultra Search crawler observes the `robots.txt` protocol, but it also allows the user to override it. If the Web site is under the user's control, then a specific robots rule can be tailored for the crawler by specifying the Oracle Ultra Search crawler agent name "User-agent: Ultra Search." For example:

```
User-agent: Ultra Search

Disallow: /tmp/
```

The robots `metatag` can instruct the crawler to either index a Web page or follow the links within it.

Crawling Depth

Crawling depth controls how deep the crawler follows a link starting from the given seed URL. Because crawling is multi-threaded, this is not a deterministic control, as there may be different routes to a particular page.

The crawling depth limit applies to all Web sites in a given Web data source.

URL Rewriter

You implement the URL rewriter API as a Java class to perform link filtering or rewriting. Extracted links within a crawled Web page are passed to this module for checking. This enables ultimate control over which links extracted from a Web page are allowed and which ones should be discard.

See Also: "Oracle Ultra Search URL Rewriter API" on page 9-23

URL Redirection and Boundary Rule Enforcement

Earlier Oracle Ultra Search releases (9.0.2, 9.0.3, and 9.2.0.4) applied the same boundary checking to a redirected URL. Thus, a redirected URL would be rejected if it was outside the boundary rule. If the redirected URL was to be crawled, you had to make sure it was covered by the boundary rule.

In 9.2.0.5, Oracle Application Server 10*g*, and Oracle Database 10*g* the redirected URL is always allowed if it is a temporary redirection (HTTP status 302, 307). For permanent redirection (status 301), the redirected URL is still subject to boundary rules.

HTTP metatag redirection is always checked against boundary rules.

Oracle Ultra Search Remote Crawler

To increase crawling performance, set up the Oracle Ultra Search crawler to run on one or more computers separate from your database. These computers are called remote crawlers. However, each computer must share log and mail archive directories with the database computer.

To configure a remote crawler, you must first install the Oracle Ultra Search middle tier on a computer other than the database host. During installation, the remote crawler is registered with the Oracle Ultra Search system, and a profile is created for the remote crawler. After installing the Oracle Ultra Search middle tier, you must log on to the Oracle Ultra Search administration tool and edit the remote crawler profile. You can then assign a remote crawler to a crawling schedule. To edit remote crawler profiles, use the Crawler Settings page in the administration tool.

See Also: "Using the Remote Crawler" on page 10-4

Oracle Ultra Search Crawler Status Codes

The crawler uses a set of codes to indicate the crawling result of the crawled URL. Besides the standard HTTP status codes, it uses its own codes for non-HTTP related situations. Only URLs with status 200 will be indexed.

See Also: Appendix D, "URL Crawler Status Codes"

8

Understanding the Oracle Ultra Search Administration Tool

The Oracle Ultra Search administration tool lets you manage Oracle Ultra Search instances. This chapter helps guide you through the screens on the Oracle Ultra Search administration tool. It contains the following topics:

Oracle Ultra Search Administration Tool

The Oracle Ultra Search administration tool is a J2EE-compliant Web application. You can use it to manage Oracle Ultra Search instances. To use the administration tool, log on as either a database user, an Enterprise Manager super-user, a Portal user, or a single sign-on user through any browser.

> **Note:** The Oracle Ultra Search administration tool and the Oracle Ultra Search query applications are part of the Oracle Ultra Search middle tier. However, the Oracle Ultra Search administration tool is independent from the Oracle Ultra Search query application. Therefore, they can be hosted on different computers to enhance security or scalability.

With the administration tool, you can do the following:

- Log on to Oracle Ultra Search
- Create Oracle Ultra Search instances
- Manage administrative users
- Define data sources and assign them to data groups
- Configure and schedule the Oracle Ultra Search crawler
- Set query options
- Translate search attributes and LOV and data group display names to different languages

Setting Crawler Parameters

To configure the Oracle Ultra Search crawler, you must do the following:

- Set crawler parameters, such as the crawler log file directory. To do so, use the Crawler Page.
- Set Web access parameters, such as authentication and the proxy server. To do so, use the Web Access Page.
- Define data sources. Data sources can be Web pages, database tables, files, e-mail mailing lists, Oracle Sources (for example, Oracle Application Server Portals or federated sources), or user-defined data sources. You can assign one or more data sources to a crawler schedule. To define data sources, use the Sources Page. You can also set parameters for the source, such as domain inclusions or exclusions for Web sources or the display URL template or column for table sources.
- Define synchronization schedules. The crawler uses the synchronization schedule to reconcile the Oracle Ultra Search index with current data source content. To define crawling schedules, use the Schedules Page.

Setting Query Options

Use query options to let users limit their searches. Searches can be limited by document attributes and data groups.

Attributes

Search attributes can be mapped to HTML metatags, table columns, document attributes, and e-mail headers. Some attributes, such as author and description, are predefined and need no configuration. However, you can customize your own attributes. To set custom search attributes to expose to the query user, use the Attributes Page.

Data Groups

Data source groups are logical entities exposed to the search engine user. When entering a query, the search engine user is asked to select one or more data groups to search from. A data group consists of one or more data sources. To define data groups, use the Queries Page.

Online Help in Different Languages

Oracle Ultra Search provides context-sensitive online help, which can be viewed in different languages. You can change the language preferences on the Users Page.

Logging On to Oracle Ultra Search

The following users can log on to the Oracle Ultra Search administration tool:

- Single Sign-on users: These users are managed by the Oracle Internet Directory and are authenticated by OracleAS Single Sign-On. The Oracle Ultra Search administration tool identifies all Oracle Ultra Search instances to which the single sign-on user has access. This is available only if you have the Oracle Identity Management infrastructure installed.
- Database users (non-single sign-on): These users exist in the database on which Oracle Ultra Search runs.
- Enterprise Manager users
- Portal single sign-on users

To log on to the administration tool, point your Web browser to one of the following URLs:

- For non-single sign-on mode:

  `http://`*`hostname:port`*`/ultrasearch/admin/index.jsp`

- For single sign-on mode:

  `http://`*`hostname:port`*`/ultrasearch/admin_sso/index.jsp`

Immediately after installation, the only users able to create and manage instances are the following:

- The `WKSYS` database user
- The Enterprise Manager user
- The `PORTAL` single sign-on user belonging to the default company [not supported in the Oracle database release]
- The `ORCLADMIN` single sign-on user belonging to the default company [this is available only if the Oracle Identity Management infrastructure is installed]

After you are logged on as one of these special users, you can grant permission to other users, enabling them to create and manage Oracle Ultra Search instances. Using the Oracle Ultra Search administration tool, you can only grant and revoke Oracle Ultra Search related permissions to and from exiting users. To add or delete users, use the Oracle Internet Directory for single sign-on users or Oracle Enterprise Manager for local database users.

Note: The Oracle Ultra Search product database dictionary is installed in the `WKSYS` schema.

See Also:

- Chapter 3, "Installing Oracle Ultra Search"
- "Changing Oracle Ultra Search Schema Passwords" on page 5-1 for information about changing the WKSYS password
- "Instances Page" on page 8-5 for more information about creating Oracle Ultra Search instances
- "Users Page" on page 8-34 for more information about granting permission to other users
- "Logging On and Managing Instances as Single Sign-On Users" on page 8-4 for more information about how Oracle Ultra Search handles single sign-on users

Logging On and Managing Instances as Single Sign-On Users

Note: Single Sign-On is available only if the Oracle Identity Management infrastructure is installed.

Logging On to Oracle Ultra Search

When a single sign-on user logs on to the Oracle Ultra Search administration tool, the user is first prompted with the single sign-on login screen.

Enter the single sign-on user name and password. After OracleAS Single Sign-On authenticates the user, the user sees a list of Oracle Ultra Search instances that they have the privileges to manage.

There are different URLs for different users. For example:

- Single sign-on users:

  `http://host:http_port/ultrasearch_admin_sso/index.jsp`

- Portal users:

  `http://host:http_port/pls/portal`

- Enterprise Manager users:

  `http://host:em_port/`

Granting Privileges to Single Sign-On Users

You might need to grant super-user privileges, or privileges for managing an Oracle Ultra Search instance, to a single sign-on user. This process is slightly different, depending on whether Oracle Application Server Portal is running in hosted mode or non-hosted mode, as described in the following list:

Note: A single sign-on user is uniquely identified by Oracle Ultra Search with a single sign-on nickname and subscriber nickname combination.

- In non-hosted mode, the subscriber nickname is not required when granting privileges to a single sign-on user. This is because there is exactly one subscriber in Oracle Application Server Portal in non-hosted mode.
- In hosted mode, the subscriber nickname is required when granting privileges to a single sign-on user. This is because there can be more than one subscriber in Oracle Application Server Portal, and two or more users with the same single sign-on nickname (for example, PORTAL) could be distinct single sign-on users distinguished by their subscriber nickname. When running Portal in hosted mode, also note the following:
  - When granting permissions for the default subscriber user, always specify DEFAULT COMPANY for the subscriber nickname, even though the actual nickname could be different; for example, ORACLE. The actual nickname is not recognized by Oracle Ultra Search.
  - When logging in to OracleAS Single Sign-On as the default subscriber user, leave the subscriber nickname blank. Alternatively, enter DEFAULT COMPANY instead of the actual subscriber nickname; for example, ORACLE so that it is recognized by Oracle Ultra Search.

Note: At any point after installation, you can run an Oracle Application Server Portal script to alter the running mode from non-hosted to hosted. Whenever this is done, the Oracle Application Server Portal script invokes an Oracle Ultra Search script to inform Oracle Ultra Search of the change from non-hosted to hosted modes.

See Also: *Hosting Developer's Guide* at http://www.oracle.com/technology

Instances Page

After successfully logging on to the Oracle Ultra Search administration tool, you find yourself on the **Instances Page**. This page manages all Oracle Ultra Search instances in the local database. In the top left corner of the page, there are tabs for creating, selecting, editing, and deleting instances.

Before you can use the administration tool to configure crawling and indexing, you must create an Oracle Ultra Search instance. An Oracle Ultra Search instance is identified with a name and has its own crawling schedules and index. Only users granted super-user privileges can create Oracle Ultra Search instances.

Creating an Instance

To create an instance, click **Create**. You can create a regular instance or a read-only snapshot instance. Only users with super-user privileges can create new instances.

Note: If you define the same data source within different instances Oracle Ultra Search, then there could be crawling conflicts for table data sources with logging enabled, e-mail data sources, and some user-defined data sources.

Creating a Regular Instance

To create an instance, do the following:

1. Prepare the database user.

   Every Oracle Ultra Search instance is based on a database user and schema with the WKUSER role.

   The database user you create to house the Oracle Ultra Search instance should be assigned a dedicated self-contained tablespace. This is important if you plan to ever create snapshot instances of this instance. To do this, create a new tablespace. Then, create a new database user whose default tablespace is the one you just created.

   See Also:

   - "Configuring the Oracle Server for Oracle Ultra Search" on page 5-1 for information and instructions on configuring database users for Oracle Ultra Search
   - "Creating a Snapshot Instance" on page 8-6

2. Follow the instance creation steps in the Oracle Ultra Search administration tool.

   From the main instance creation page, click **Create Instance**, and provide the following information:

   - Instance name
   - Database schema: this is the user name from Step 1.
   - Schema password

   You can also enter the following optional index preferences:

   - Lexer

     Specify the name of the lexer you want to use for indexing. The lexer breaks text into tokens according to your language. These tokens are usually words. The default lexer is `wksys.wk_lexer`, as defined in the `wk0pref.sql` file. After the instance is created, the lexer can no longer be changed.

   - Stoplist

     Specify the name of a stoplist you want to use during indexing. The default stoplist is `wksys.wk_stoplist`, as defined in the `wk0pref.sql` file. Avoid modifying the stoplist after the instance has been created.

   - Storage

     Specify the name of the storage preference for the index of your instance. The default storage preference is `wksys.wk_storage`, as defined in the `wk0pref.sql` file. After the instance is created, the storage preference cannot be changed.

     See Also:

     - *Oracle Text Reference* for more information on these creating and modifying lexers, stoplists, and storage
     - "Managing Stoplists" on page 5-6

Creating a Snapshot Instance

A snapshot instance is a copy of another instance. Unlike a regular instance, a snapshot instance is read only; it does not synchronize its index to the search domain. After the master instance re-synchronizes to the search domain, the snapshot instance

becomes out of date. At that point, you should delete the snapshot and create a new one.

> **Note:** The snapshot and its master instance cannot reside on the same database.

A snapshot instance is useful for the following purposes:

- Query Processing

  Two Oracle Ultra Search instances can answer queries about the same search domain. Therefore, in a set amount of time, two instances can answer more queries about that domain than one instance. Because snapshot instances do not involve crawling and indexing, snapshot instance creation is fast and inexpensive. Thus, snapshot instances can improve scalability.

- Backups

  If the master instance becomes corrupted, its snapshot can be transformed into a regular instance by editing the instance mode to updatable. Because the snapshot and its master instance cannot reside on the same database, a snapshot instance should be made updatable only to replace a corrupted master instance.

A snapshot instance does not inherit authentication from the master instance. Therefore, if you make a snapshot instance updatable, you must re-enter any authentication information needed to crawl the search domain.

To create a snapshot instance, do the following:

1. Prepare the database user.

   As with regular instances, snapshot instances require a database user. This user must have been granted the `WKUSER` role.

2. Copy the data from the master instance.

   This is done with the transportable tablespace mechanism, which does not allow renaming of tablespaces. Therefore, a snapshot instance cannot be created on the same database as its master.

   Identify the tablespace or the set of tablespaces that contain all the master instance data. Then, copy it, and plug it into the database user from Step 1.

3. Follow snapshot instance creation in the Oracle Ultra Search administration tool.

   From the main instance creation page, click **Create Read-Only Snapshot Instance**, and provide the following information:

   - Snapshot instance name
   - Snapshot schema name: this is the database user from Step 1.
   - Snapshot schema password
   - Database link: this is the name of the database link to the database where the master instance lives.
   - Master instance name

4. Enable the snapshot for secure searches.

   If the master instance for the snapshot of is secure-search enabled and if the destination database that you are making a snapshot in supports secure-search

enabled instances, then you must also run a PL/SQL procedure in the destination database where you are creating the snapshot.

Running this procedure translates the IDs of the access control lists (ACLs) in the destination database, rendering them usable. Log on to the database as the WKSYS user. Invoke the procedure as follows:

```
exec WK_ADM.USE_INSTANCE('instance_name');
exec WK_ADM.TRANSLATE_ACL_IDS();
```

where *instance_name* is the name of the snapshot instance

Make sure that this statement completes successfully without error.

See Also:

- Chapter 5, "Oracle Ultra Search Postinstallation Information" for information on changing the WKSYS password and for instructions on configuring database users for Oracle Ultra Search
- *Oracle Database Administrator's Guide* for details on using transportable tablespaces

Selecting an Instance

You can have multiple Oracle Ultra Search instances. For example, an organization can have separate Oracle Ultra Search instances for its marketing, human resources, and development portals. The administration tool requires you to specify an instance before it lets you make any instance-specific changes.

To select an instance, do the following:

1. Click **Select** on the **Instances Page**.
2. Select an instance from the pull-down menu.
3. Click **Apply**.

Note: Instances do not share data. Data sources, schedules, and indexes are specific to each instance.

Deleting an Instance

To delete an instance, do the following:

1. Click **Delete** on the **Instances Page**.
2. Select an instance from the pull-down menu.
3. Click **Apply**.

Note: To delete an Oracle Ultra Search instance, the user must be granted the super-user privileges.

Editing an Instance

To edit an instance, click **Edit** on the **Instances Page**.

You can change the instance mode (make the instance updatable) or change the instance password.

Instance Mode

You can change the instance mode to updatable or read only. Updatable instances synchronize themselves to the search domain on a set schedule, whereas read-only instances (snapshot instances) do not do any synchronization. To set the instance mode, select the box corresponding the to mode you want, and click **Apply**.

Schema Password

An Oracle Ultra Search instance must know the password of the database user in which it resides. The instance cannot get this information directly from the database. During instance creation, Oracle provides the database user password, and the instance caches this information.

If this database user password changes, then the password that the instance has cached must be updated. To do this, enter the new password and click **Apply**. After the new password is verified against the database, it replaces the cached password.

Crawler Page

The Oracle Ultra Search crawler is a Java application that spawns threads to crawl defined data sources, such as Web sites, database tables, or e-mail archives. Crawling occurs at regularly scheduled intervals, as defined in the Schedules Page.

On the Crawler page, you can configure various crawler settings.

Configure the Settings

Crawler Threads

Specify the number of crawler threads to be spawned at run time.

Number of Processors

Specify the number of central processing units (CPUs) that exist on the server where the Oracle Ultra Search crawler will run. This setting determines the optimal number of document conversion threads used by the system. A document conversion thread converts multiformat documents into HTML documents for proper indexing.

Automatic Language Detection

Not all documents retrieved by the Oracle Ultra Search crawler specify the language. For documents with no language specification, the Oracle Ultra Search crawler attempts to automatically detect language. Click **Yes** to turn on this feature.

The language recognizer is trained statistically using trigram data from documents in various languages (for instance, Danish, Dutch, English, French, German, Italian, Portuguese, and Spanish). It starts with the hypothesis that the given document does not belong to any language and ultimately refutes this hypothesis for a particular language where possible. It operates on Latin-1 alphabet and any language with a deterministic Unicode range of characters (like Chinese, Japanese, Korean, and so on).

The crawler determines the language code by checking the HTTP header content-language or the LANGUAGE column, if it is a table data source. If it cannot determine the language, then it takes the following steps:

1. If the language recognizer is not available or if it is unable to determine a language code, then the default language code is used
2. If the language recognizer is available, then the output from the recognizer is used.

This language code is populated in LANG column of the wk$url and wk$doc tables. Multilexer is the only lexer used for Oracle Ultra Search. All document URLs are stored in wk$doc for indexing and wk$url for crawling.

Default Language

If automatic language detection is disabled, or if a Web document does not have a specified language, then the crawler assumes that the Web page is written in this default language. This setting is important, because language directly determines how a document is indexed.

> **Note:** This default language is used only if the crawler cannot determine the document language during crawling. Set language preference in the Users Page.

You can select a default language for the crawler or for data sources. Default language support for indexing and querying is available for the following languages:

- Polish
- Chinese
- Hungarian
- Norwegian
- Romanian
- Finnish
- Japanese
- Spanish
- Slovak
- English
- Turkish
- Danish
- Swedish
- Russian
- German
- Korean
- Dutch
- Italian
- Greek
- Portuguese
- Czech
- Hebrew
- French
- Arabic

Crawling Depth

A Web document can contain links to other Web documents, which can contain more links. This setting lets you specify the maximum number of nested links the crawler will follow.

> **See Also:** "Tuning the Web Crawling Process" on page 10-1 for more information on the importance of the crawling depth

Crawler Timeout Threshold

Specify, in seconds, a crawler timeout threshold. The crawler timeout threshold is used to force a timeout when the crawler cannot access a Web page.

Default Character Set

Specify the default character set. The crawler uses this setting when an HTML document does not have its character set specified.

Cache Directory

Specify the absolute path of the cache directory. During crawling, documents are stored in the cache directory. Every time the preset size is reached, crawling stops and indexing starts.

If you are crawling sensitive information, then make sure that you set the appropriate file system read permission to the cache directory.

You can choose whether or not to have the cache cleared after indexing.

Crawler Logging

Specify the following:

- Level of detail: everything or only a summary
- Crawler logfile directory
- Crawler logfile language

The log file directory stores the crawler log files. The log file records all crawler activity, warnings, and error messages for a particular schedule. It includes messages logged at startup, runtime, and shutdown. Logging everything can create very large log files when crawling a large number of documents. However, in certain situations, it can be beneficial to configure the crawler to print detailed activity to each schedule log file. The crawler logfile language is the language the crawler uses to generate the log file.

The crawler maintains multiple versions of its log file. The format of the log file name is:

`i`*`instance_id`*`ds`*`data_source_id`*`.`*`MMDDhhmm`*`.log`

where *`MM`* is the month, *`DD`* is the date, *`hh`* is the launching hour in 24-hour format, and *`mm`* is the minutes. For example, if a schedule for data source 23 of instance 3 is launched at 10 pm, July 8th, then the log file name is `i3ds23.07082200.log`. Each successive schedule launching will have a unique log file name. If the total number of log files for a data source reaches the system-specified limit, then the oldest log file will be deleted. The number of log files is a scheduler property and applies to all of the data sources assigned to the scheduler.

Database Connect String

The database connect string is a standard JDBC connect string used by the remote crawler when it connects to the database. The connect string can be provided in the form of `[hostname]:[port]:[sid]` or in the form of a TNS keyword-value syntax; for example:

```
"(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=...)(PORT=1521)...))"
```

See Also: *Oracle Database JDBC Developer's Guide and Reference*

You can update the JDBC connect string to a different format; for example, an LDAP format. However, you cannot change the JDBC connect string to point to a different database. The JDBC connect string must be set to the database where the middle tier points; that is, the middle tier and the JDBC should point to the same database.

In a Real Application Clusters environment, the TNS keyword-value syntax should be used, because it allows connection to any node of the system. For example,

```
 "(DESCRIPTION=(LOAD_BALANCE=yes)(ADDRESS=(PROTOCOL=TCP)(HOST=cls02a)(PORT=3001))
(ADDRESS=(PROTOCOL=TCP)(HOST=cls02b)(PORT=3001)))(CONNECT_DATA=(SERVICE_
NAME=sales.us.acme.com)))"
```

Remote Crawler Profiles

Use this page to view and edit remote crawler profiles.

A remote crawler profile consists of all parameters needed to run the Oracle Ultra Search crawler on a remote computer other than the Oracle Ultra Search database. To register a remote crawler, you need to use the PL/SQL API `wk_crw.register_remote_crawler`. You can choose either RMI-based or JDBC-based remote crawling.

To configure the remote crawler, click **Edit**. Here is a list of configuration parameters that you can change for the remote crawler:

- Cache file access mode. You have two options for the remote crawler to handle cache files:
  - Through a JDBC connection.

    In this case, the remote crawler will send cache files over the crawler's JDBC connection to the server's cache directory.
  - Through a mounted file system.

    If you choose this option, the cache file will be saved in the remote crawler cache directory. The remote crawler cache directory must be mounted to the server side crawler cache directory (specified under **Crawler** -> **Settings** tab); otherwise, the documents cannot be indexed.

    See Also: "Using the Remote Crawler" on page 10-4 for more on crawling with JDBC connections
- Cache directory location (absolute path)
- Crawler log file directory
- Mail archive path
- Number of crawler threads
- Number of processors
- Initial Java heap size (in megabytes)

- Maximum Java heap size (in megabytes)
- Java classpath

Crawler Statistics

Use this page to view the following crawler statistics:

Summary of Crawler Activity

This provides a general summary of crawler activity:

- Aggregate crawler statistics
- Total number of documents indexed
- Crawler statistics by data source type

Detailed Crawler Statistics

This includes the following:

- List of hosts crawled and indexed
- Document distribution by depth
- Document distribution by document type
- Document distribution by data source type

Crawler Progress

This displays crawler progress for the past week. It shows the total number of documents that have been indexed for exactly one week prior to the current time. The Time column rounds the current time to the nearest hour.

Problematic URLs

This lists errors encountered during the crawling process. It also lists the number of URLs that cause each error.

Web Access Page

For the crawler to contact Web pages that reside outside your firewall, you must register a proxy on the Proxies page. If the proxy requires authentication, then you must enter the proxy authentication information on the Authentication page.

Proxies

Specify a proxy server if the search space includes Web pages that reside outside your organization's firewall.

To set the proxy, enter the proxy server name and port. For example, `myproxy.mydomain`, `8080`. Because internal Web sites should not go through the proxy server, specify proxy domain exceptions if the proxy server is set. Enter the host name suffix that should not go through the proxy in the exception field. Use the suffix of the host name without "http". For example, `us.oracle.com`, `oracle.com`, `uk.oracle.com`, and `oraclecorp.com`. The no proxy checking is strictly a suffix matching of the host name. IP address can only be used when the URL crawled is also specified in IP for the host name. In other words, they must be consistent.

If the proxy requires authentication, then specify the proxy login user name and password in the Authentication page.

Authentication

Use this page to enter authentication information that applies to all data sources.

Note: The data source specific authentication takes precedence over this global authentication.

HTTP Authentication

Oracle Ultra Search supports both basic and digest authentication.

Specify the user name and password for the host and realm for which HTTP authentication is required. For example, enter "myproxy.mydomain", "LDAP", "myname", and "mypassword" as the host, realm, user name, and password.

The realm is a name associated with the protected area of a Web site. It is a string that you provide to log on to such a protected page.

For proxy authentication, enter the user name, password, and realm of the proxy server.

HTML Form Authentication

HTML forms are used on the World Wide Web to collect information from a user. One common usage is to collect authentication information. Text boxes for entering your username and password on an HTML page use HTML form authentication.

HTML forms vary in complexity. They can be simple or very complex incorporating a lot of Javascript. The following example shows a simple HTML form with two text fields.

```
<FORM action="http://somesite.com/prog/adduser" method="post" name="MyForm">
  <LABEL for"i1>Username: </LABEL>
  <INPUT name="username" type="text" id="i1"><BR>
  <LABEL for="i2>Password: </LABEL>
  <INPUT name="passwd" type="passwd" id="i2"><BR>
</FORM>
```

In a browser this will look like Figure 8–1.

Figure 8–1 HTML Form

Where:

`name` = form name

`method` = weather an HTTP POST or GET should be used for form data submission

`action` = the URL where the data should be submitted (this is usually a servlet of some sort)

`INPUT` elements = these are called form controls. In this example, there is a "text" box type control, and a "password" type, which is also a text box but it hides the text typed into it. Although there are many different types of controls, when the form is submitted, the controls translate to simple name-value pairs and are sent as such to the action URL. The name is the name of the INPUT element, whereas the value is the value entered by the user (using a text box, a dropdown, and so on).

> **See Also:** HTML form documentation available from W3C (www.w3c.org)

HTML Forms in Ultra Search

Ultra Search lets you register HTML forms. During crawling, if the crawler finds one of the registered forms, it automatically fills out the form using the data you submitted during form's registration.

Registering Forms

You can register HTML forms using the Ultra Search administration too. You can create Ultra Search instance-level form registration entries or data source-level entries that are only visible to the particular data source.

Wizard-Based HTML Form Registration

The preferred way to register a form is using the HTML form registration wizard. The lets you fill out the form as usual while it tries to capture the submitted data. You specify the page with the form; the wizard fetches it and displays it; you fill out and submit the form, as when accessing it directly through a Web browser; the wizard shows the Web site's response to the submitted form; and then you confirm if the form submission was a success.

This whole process is quick and simple. However, the HTML form registration wizard cannot handle forms that use Javascript. It might handle some depending on what is being done with Javascript, but there are no guarantees. The alternative is to do a manual form registration.

Manual HTML Form Registration

To do manual registration you must be familiar with the anatomy of the form you are registering. You can see the form by browsing to it and asking from the Web browser to display the page source.

As with the wizard based registration, you start by specifying the URL of the page containing the HTML form. Next you provide the following bits of information:

- name = Form name
- action URL = URL where the form is submitted (see preceding section on the form anatomy)
- success URL = URL where one gets redirected upon successful submission of the form
- controls = All the form controls and their desired values

When Javascript is used, some of these values might not be obvious, because the Javascript might be manipulating them or adding new controls dynamically. So, you need to understand the role that Javascript is playing.

Note: The Oracle Ultra Search crawler will choose the form to use based on the form's URL and the form name. URL parameters are not included during matching; thus, they are truncated during form registration.

Attributes Page

When your indexed documents contain metadata, such as author and date information, you can let users refine their searches based on this information. For example, users can search for all documents where the author attribute has a certain value.

The list of values (LOV) for a document attribute can help specify a search query. An attribute value can have a display name for it. For example, the attribute country might use country code as the attribute value, but show the name of the country to the user. There can be multiple translations of the attribute display name.

To define a search attribute, use the **Search Attributes** subtab. Oracle Ultra Search provides the following default search attributes: Title, Author, Description, Subject, Mimetype, Language, Host, and LastModifedDate. They can be incorporated in search applications for a more detailed search and richer presentation.You can also define your own.

After defining search attributes, you must map between document attributes and global search attributes for data sources. To do so, use the **Mappings** subtab.

Note: Oracle Ultra Search provides a command-line tool to load metadata, such as search attribute LOVs and display names, into an Oracle Ultra Search database. If you have a large amount of data, this is probably faster than using the HTML-based administration tool. For more information, see Appendix A, "Loading Metadata into Oracle Ultra Search".

Search Attributes

Search attributes are attributes exposed to the query user. Oracle Ultra Search provides system-defined attributes, such as author and description. Oracle Ultra Search maintains a global list of search attributes. You can add, edit, or delete search attributes. You can also click **Manage LOV** to change the list of values (LOV) for the search attribute. There are two categories of attribute LOVs: one is global across all data sources, the other is data source-specific.

To define your own attribute, enter the name of the attribute in the text box; select string, date, or number; and click **Add**.

You can add or delete LOV entry and display name for search attributes. Display name is optional. If display name is absent, then LOV entry is used in the query screen.

Note: LOV is only represented as string type. If LOV is in date format, then you must use "DD-MM-YYYY" to enter the LOV.

To update the policy value, click **Manage LOV** for any attribute.

A data source-specific LOV can be updated in three ways:

1. Update the LOV manually.
2. The crawler agent can automatically update the LOV during the crawling process.
3. New LOV entries can be automatically added by inspecting attribute values of incoming documents.

Caution: If the update policy is agent-controlled, then the LOV and all translated values are erased in the next crawling.

Mappings

This section displays mapping information for all data sources. For user-defined data sources, mapping is done at the agent level, and document attributes are automatically mapped to search attributes with the same name initially. Document attributes and search attributes are mapped one-to-one. For each user-defined data source, you can edit the global search attribute to which the document attribute is mapped.

For Web, file, or table data sources, mappings are created manually when you create the data source. For user-defined data sources, mappings are automatically created on subsequent crawls.

Click **Edit Mappings** to change this mapping.

Editing the existing mapping is costly, because the crawler must recrawl all documents for this data source. Avoid this step, unless necessary.

Note: There are no user-managed mappings for e-mail sources. There are two predefined mappings for e-mails. The "From" field of an e-mail is intrinsically mapped to the Oracle Ultra Search author attribute. Likewise, the "Subject" field of an e-mail is mapped to the Oracle Ultra Search subject attribute. The abstract of the e-mail message is mapped to the description attribute.

Sources Page

A collection of documents is called a source. The data source is characterized by the properties of its location, such as a Web site or an e-mail inbox. The Oracle Ultra Search crawler retrieves data from one or more data sources.

The different types of sources are:

- Web Sources
- Table Sources
- E-mail Sources
- File Sources
- Oracle Sources
- User-Defined Sources (requires a crawler agent)

See Also:

- "Schedules Page" on page 8-27 to assign one or more data sources to a synchronization schedule
- "Queries Page" on page 8-31 to assign data sources to data groups to enable restrictive querying

You can create as many data sources as you want. The following section explains how to create and edit data sources.

Web Sources

A Web source represents the content on a specific Web site. Web sources facilitate maintenance crawling of specific Web sites.

Creating Web Sources

To create a new Web source, do the following:

1. Specify a name for the Web source and a starting address. This is the URL for the crawler to begin crawling. The starting address can be HTTP or HTTPS.
2. Set URL boundary rules to refine the crawling space. You can include or exclude hosts or domains beginning with, ending with, or equal to a specific name.

   For example, an inclusion domain ending with `oracle.com` limits the Oracle Ultra Search crawler to hosts belonging to Oracle worldwide. Anything ending with `oracle.com` is crawled; but, `http://www.oracle.com.tw` is not crawled. If you change the inclusion domain to `yahoo.com` with a new seed `http://www.yahoo.com`, then all oracle.com URLs are dropped by the crawler.

   An exclusion domain uk.oracle.com prevents the crawler from crawling Oracle hosts in the United Kingdom. You can also include or exclude Web sites with a specific port. (By default, all ports are crawled.) You can have port inclusion or port exclusion rules for a specific host, but not both. Exclusion rules always override inclusion rules.
3. Specify the types of documents the Oracle Ultra Search crawler should process for this source. HTML and plain text are default document types that the crawler always processes.
4. Specify the authentication settings. This step is optional. Under **HTTP Authentication**, specify the user name and password for host and realm for which authentication is required. The realm is a name associated with the protected area of a Web site. Under **HTML Forms**, you can register HTML forms that you want the Oracle Ultra Search crawler to automatically fill out during Web crawling. HTML form support requires that HTTP cookie functionality is enabled. Cookies remember context between HTTP requests. For example, the server can send a cookie such that it knows if a user has already logged on and does not need to log on again. Cookie support is enabled by default. Click **Register HTML Form** to register authentication forms protecting the data source. Note: For the form URL to be crawled, you must verify that the URL is not excluded in the `robots.txt` file. If so, then you must disable robot exclusion for this data source. (By default, Oracle Ultra Search enables robot exclusion.)
5. Choose either **No ACL** or **Ultra Search ACL** for the data source. When a user performs a search, the ACL (access control list) controls which documents the user can access. The default is no ACL, with all documents considered searchable and

visible. You can add more than one group and user to the ACL for the data source. The option to choose is only available if the instance is security-enabled.

6. Define, edit, or delete metatag mappings for your Web source. Metatags are descriptive tags in the HTML document header. One metatag can map to only one search attribute.

7. Override the default crawler settings for each Web source. This step is optional. The parameters you can override are the crawling depth, the number of crawler threads, the language, the crawler timeout threshold, the character set, the maximum cookie size, the maximum number of cookies, and the maximum number of cookies for each host. You can also enable or disable robots exclusion, language detection, the UrlRewriter, indexing dynamic Web pages, HTTP cookies, and whether content of the cookie log file is shown. (You can also edit those in **Edit Web Sources**.)

   Robots exclusion lets you control which parts of your sites can be visited by robots. If robots exclusion is enabled (default), then the Web crawler traverses the pages based on the access policy specified in the Web server `robots.txt` file. For example, when a robot visits `http://www.foobar.com/`, it checks for `http://www.foobar.com/robots.txt`. If the robot finds it, the crawler analyzes its contents to see if it is allowed to retrieve the document. If you own the Web sites, then you can disable robots exclusions. However, when crawling other Web sites, you should always comply with `robots.txt` by enabling robots exclusion.

   The URL Rewriter is a user-supplied Java module for implementing the Oracle Ultra Search UrlRewriter interface. It is used by the crawler to filter or rewrite extracted URL links before they are put into the URL queue. URL filtering removes unwanted links, and ULR rewriting transforms the URL link. This transformation is necessary when access URLs are used.

   The UrlRewriter provides the following possible outcomes for links:

   - There is no change to the link. The crawler inserts it as it is.
   - Discard the link. There is no insertion.
   - A new display URL is returned, replacing the URL link for insertion.
   - A display URL and an access URL are returned. The display URL may or may not be identical to the URL link.

   The generated new URL link is subject to all existing host, path, and mimetype inclusion and exclusion rules.

   You must put the implemented rewriter class in a jar file and provide the class name and jar file name here.

   If Index Dynamic Page is set to **Yes**, then dynamic URLs are crawled and indexed. For data sources already crawled with this option, setting Index Dynamic Page to **No** and recrawling the data source removes all dynamic URLs from the index.

   Note: There is a restriction that the crawler cannot crawl and index dynamic Web pages written in JavaScript.

   Some dynamic pages appear as multiple search hits for the same page, and you may not want them all indexed. Other dynamic pages are each different and need to be indexed. You must distinguish between these two kinds of dynamic pages. In

general, dynamic pages that only change in menu expansion without affecting its contents should not be indexed. Consider the following three URLs:

```
http://itweb.oraclecorp.com/aboutit/network/npe/standards/naming_
convention.html

http://itweb.oraclecorp.com/aboutit/network/npe/standards/naming_
convention.html?nsdnv=14z1

http://itweb.oraclecorp.com/aboutit/network/npe/standards/naming_
convention.html?nsdnv=14
```

The question mark ('?') in the URL indicates that the rest of the strings are input parameters. The duplicate hits are essentially the same page with different side menu expansion. Ideally, the query should yield only one result:

```
http://itweb.oraclecorp.com/aboutit/network/npe/standards/naming_
convention.html
```

Dynamic page index control applies to the whole data source. So, if a Web site has both kinds of dynamic pages, you need to define them separately as two data sources in order to control the indexing of those dynamic pages.

See Also:

- "Oracle Ultra Search URL Rewriter API" on page 9-23
- "Using Crawler Agents" on page 7-2
- "Crawler Page" on page 8-9 for information on default languages

Table Sources

A table source represents content in a database table or view. The database table or view can reside in the Oracle Ultra Search database instance or in a remote database. Oracle Ultra Search accesses remote databases using database links.

See Also: "Limitations With Database Links" on page 8-21

Creating Table Sources

To create a table source, click **Create Table Source**, and follow these steps:

1. Specify a table source name, and the name of the database link, schema, and table. (Database links are configured manually using SQL `CREATE DATABASE LINK` against the Oracle Ultra Search instance in question. After you create the database link, it shows up in the drop down list.) Click **Locate Table**.
2. Specify settings for your table source, such as the default language and the primary key column. You can also specify the column where final content should be delivered, and the type of data stored in that column; for example, HTML, plain text, or binary. For information on default languages, see "Crawler Page" on page 8-9.
3. Verify the information about your table source.
4. Decide whether or not to use the Oracle Ultra Search logging mechanism to optimize the crawling of table data sources. When crawling is enabled, only newly updated documents are revisited during the crawling process. You can enable logging for Oracle tables, enable logging for non-Oracle tables, or disable the logging mechanism. If you enable logging, then you are prompted to create a log table and log triggers. Oracle SQL statements are provided for Oracle tables. If you

are using non-Oracle tables, then you must manually create a log table and log triggers. Follow the examples provided to create the log table and log triggers. After you have created the table, enter the table name in **Log Table Name**.

5. Map table columns to search attributes. Each table column can be mapped to exactly one search attribute. This lets the search engine seamlessly search data from the table source.

6. Specify the display URL template or column for the table source. This step is optional. Oracle Ultra Search uses a default text viewer for table data sources. If you specify Display URL, then Oracle Ultra Search uses the Web URL defined to display the table data retrieved. If Display URL column is available, then Oracle Ultra Search uses the column to get the URL to display the table data source content. You can also specify display URL templates in the following format: `http://hostname:port/path?parameter_name=$(key1)` where key1 is the corresponding table's primary key column. For example, assume that you can use the following URL to query the bug number 1234567, and the bug number is the primary key of the table: `http://bug:7777/pls/bug?rptno=1234567`. You can set the table source display URL template to `http://bug:7777/pls/bug?rptno=$(key1)`.

   The **Table Column to Key Mappings** section provides mapping information. Oracle Ultra Search supports table keys in `STRING`, `NUMBER`, or `DATE` type. If key1 is of `NUMBER` or `DATE` type, then you must specify the format model used by the Web site so that Oracle knows how to interpret the string. For example, the date format model for the string '11-Nov-1999' is 'DD-Mon-YYYY'. You can also map other table columns to Oracle Ultra Search attributes. Do not map the text column.

7. Specify the ACL (access control list) policy for the data source. When a user performs a search, the ACL controls which documents the user can access. The default is no ACL, with all documents considered public and visible. Alternatively, you can specify to use Oracle Ultra Search ACL. You can add more than one group and user to the ACL for the data source. The option to choose is available only if the instance is security-enabled.

See Also: *Oracle Database SQL Reference* for more on format models

Editing Table Sources

On the main **Table Sources** page, click **Edit** to change the name of the table source. You can change, add, or delete table column and search attribute mappings; change the display URL template or column; and view values of the table source settings.

Table Sources Comprised of More Than One Table

If a table source has more than one table, then a view joining the relevant tables must be created. Oracle Ultra Search then uses this view as the table source. For example, two tables with a master-detail relationship can be joined through a `SELECT` statement on the master table and a user-implemented PL/SQL function that concatenate the detail table rows.

Limitations With Database Links

The following restrictions apply to base tables or views on a remote database that are accessed over a database link by the crawler.

- If the text column of the base table or view is of type `BLOB` or `CLOB`, then the table must have a `ROWID` column. A table or view might not have a `ROWID` column for various reasons, including the following:

- A view is comprised of a join of one or more tables.
- A view is based on a single table using a GROUP BY clause.

The best way to know if a remote table or view can be safely crawled by Oracle Ultra Search is to check for the existence of the ROWID column. To do so, run the following SQL statement against that table or view using SQL*Plus:

```
SELECT MIN(ROWID) FROM table_name/view_name;
```

- The base table or view cannot have text columns of type BFILE, RAW.

E-mail Sources

An e-mail source derives its content from e-mails sent to a specific e-mail address. When the Oracle Ultra Search crawler searches an e-mail source, it collects all e-mails that have the specific e-mail address in any of the "To:" or "Cc:" e-mail header fields.

The most popular application of an e-mail source is where an e-mail source represents all e-mails sent to a mailing list. In such a scenario, multiple e-mail sources are defined where each e-mail source represents an e-mail list.

To crawl e-mail sources, you need an IMAP account. At present, the Oracle Ultra Search crawler can only crawl one IMAP account. Therefore, all e-mails to be crawled must be found in the inbox of that IMAP account. For example, in the case of mailing lists, the IMAP account should be subscribed to all desired mailing lists. All new postings to the mailing lists are sent to the IMAP e-mail account and subsequently crawled. The Oracle Ultra Search crawler is IMAP4 compliant.

When the Oracle Ultra Search crawler retrieves an e-mail message, it deletes the e-mail message from the IMAP server. Then, it converts the e-mail message content to HTML and temporarily stores that HTML in the cache directory for indexing. Next, the Oracle Ultra Search crawler stores all retrieved messages in a directory known as the archive directory. The e-mail files stored in this directory are displayed to the search end user when referenced by a query result.

To crawl e-mail sources, you must specify the user name and password of the e-mail account on the IMAP server. Also specify the IMAP server host name and the archive directory.

Creating E-mail Sources

To create e-mail sources, you must enter an e-mail address and a description. Optionally, you can specify e-mail aliases and ACL policy. The description can be viewed by all search end users, so you should specify a short but meaningful name. When you create (register) an e-mail source, the name you use is the e-mail of the mailing list. If the e-mails are not sent to one of the registered mailing lists, then those e-mails are not crawled.

You can specify e-mail address aliases for an e-mail source. Specifying an alias for an e-mail source causes all e-mails sent to the main e-mail address, as well as the alias address, to be gathered by the crawler. An alias is useful when two or more e-mail addresses are logically the same. For example, an e-mail source representing the distribution list `list@company.com` can have the alternate address `list@my.company.com`. If `list@my.company.com` is added to the alias list, then e-mail sent to that address are treated as if they were sent to `list@company.com`.

Specify the ACL (access control list) policy for the data source. When a user performs a search, the ACL controls which documents the user can access. The default is no ACL,

with all documents considered searchable and visible. You can add more than one group and user to the ACL for the data source.

File Sources

A file source is the set of documents that can be accessed through the file protocol on the local machine.

To edit the name of a file source, click **Edit**.

Creating File Sources

To create a new file source, do the following:

1. Specify a name for the file source and the default language.
2. Designate files or directories to be crawled. If a URL represents a single file, then the Oracle Ultra Search crawler searches only that file. If a URL represents a directory, then the crawler recursively crawls all files and subdirectories in that directory.
3. Specify inclusion and exclusion paths to modify the crawling space associated with this file source. This step is optional. An inclusion path limits the crawling space. An exclusion path lets you further define the crawling space. If neither path is specified, then crawling is limited to the underlying file system access privileges. Path rules are host-specific, but you can specify more than one path rule for each host. For example, on the same host, you can include path `files://host/doc` and exclude path `files://host/doc/unwanted`.
4. Specify the types of documents the Oracle Ultra Search crawler should process for this file source. HTML and plain text are default document types that the crawler always processes.
5. Oracle Ultra Search displays file data sources in text format. However, if you specify display URL for the file data source, then Oracle Ultra Search uses the URL to display the file data source.

   With display URL for file data sources, the URL uses network protocols, such as HTTP or HTTPS, to access the file data source. To generate display URL for the file data source, specify the prefix of the original file URL and the prefix of the display URL. Oracle Ultra Search replaces the prefix of the file URL with the prefix of the display URL.

   For example, if your file URL is `file:///home/operation/doc/file.doc` and the display URL is `https://webhost/client/doc/file.doc`, then you can specify the file URL prefix to `file:///home/operation` and the display URL prefix to `https://webhost/client.`
6. Specify the ACL (access control list) policy for the data source. When a user performs a search, the ACL controls which documents the user can access. The default is no ACL, with all documents considered searchable and visible. Alternatively, you can specify using the Oracle Ultra Search ACL. You can add more than one group and user to the ACL for the data source. The option to choose is only available if the instance is security-enabled.

Oracle Sources

You can create, edit, or delete Oracle sources. You can choose a federated source or a crawlable source from Oracle Application Server Portal. A federated source is a repository that maintains its own index. Oracle Ultra Search can issue a query, and the

repository can return query results. Oracle Ultra Search also supports the crawling and indexing of Oracle Application Server Portal installations. This enables searching across multiple portal installations.

Note: When Oracle Ultra Search crawls content from Oracle Portal, it gathers all metadata (that is, attribute values) with the actual indexable content. This is then text-indexed. When you search for string "xxx", if that string occurs in either the attributes or in the content, then the document is returned.

This is different from how Oracle Portal behaves alone. With Oracle Portal, when you search for string "xxx", only the content of a document (page/item in Portal terminology) is searched. Attributes are treated separately.

Oracle Portal Sources

Oracle Ultra Search can only crawl public Oracle AS Portal sources. See the *Oracle Application Server Portal Configuration Guide* for how to set up public pages.

To create a Portal source, you must first register your portal with Oracle Ultra Search. To register your portal:

1. Provide a name and portal URL base. The portal name is used to identify this portal entry in the **Oracle Portal List** page. The URL base is the beginning portion of the portal homepage. This include host name, port number, and DAD. After it is created, the portal URL base is not updatable. Click **Register Portal**. Oracle Ultra Search attempts to contact the Oracle Application Server Portal instance and retrieve information about it.
2. Choose one or more page groups for indexing. A portal data source is created for each page group. Click **Delete** to remove existing portal data sources.

You can edit the types of documents the Oracle Ultra Search crawler should process for a portal source. HTML and plain text are default document types that the crawler always processes. To edit document types, click **Edit** for the portal source after it has been created.

See Also: The Oracle Application Server Portal documentation

Federated Sources

To create a federated source, specify the name and JNDI for the new data source. By default, no resource adapter is available.

To create a federated source, you must manually deploy the Oracle Ultra Search resource adapter, or *searchlet*. A searchlet is a Java module deployed in the middle tier (inside OC4J) that searches the data in an enterprise information system on behalf of a user. A searchlet is a Java module deployed in the middle tier (inside OC4J) that searches the data in an enterprise information system on behalf of a user. When a user's query is delegated to the searchlet, the searchlet runs the query on behalf of the user. Every searchlet is a JCA 1.0 compliant resource adapter.

See Also: The JCA 1.0 specification from Javasoft for detailed information on resource adapters and Java Connector Architecture

Deploying and BInding the Oracle Ultra Search Searchlet The Oracle Ultra Search searchlet enables queries against one Oracle Ultra Search instance. The Oracle Ultra Search

searchlet is packaged as ultrasearch_searchlet.rar and is shipped under the $ORACLE_HOME/ultrasearch/adapter/ directory.

To deploy the Oracle Ultra Search searchlet in OC4J standalone mode, use admin.jar as follows:

```
 java -jar admin.jar ormi://<hostname> <admin> <welcome> -deployconnector -file
ultrasearch_searchlet.rar -name UltraSearchSearchlet
```

At this point, ultrasearch_searchlet.rar has been deployed in OC4J. However, it has not been instantiated to connect to any Oracle Ultra Search instance. The Oracle Ultra Search searchlet can be instantiated multiple times, to connect to several Oracle Ultra Search instances, by repeating the following steps.

To instantiate the searchlet, configuration parameters values must be specified, and a JNDI location must be specified where the searchlet instance should be bound to. To do this, you must manually edit oc4j-ra.xml. This file is typically located under the $J2EE_HOME/application-deployments/default/UltraSearchSearchlet/ directory. The Oracle Ultra Search searchlet requires four configuration properties: connectionURL, userName, password, and instanceName. For example, to bind a searchlet under eis/UltraSearch to connect to the default instance WK_TEST on machine dbhost, the following entry can be used:

```
<connector-factory location="eis/UltraSearch" connector-name="Ultra Search
Adapter">
 <config-property name="connectionURL" value="jdbc:oracle:thin:@dbhost:1521:sid"/>
 <config-property name=:userName" value="wk_test"/>
 <config-property name="passwors" value="wk_test"/>
 <config-property name="instanceName" value="wk_test"/>
</connector-factory>
```

After editing oc4j-ra.xml, restart the OC4J instance. If you do not see errors upon restart, then the searchlet has been successfully instantiated and bound to JNDI.

Deploying and Binding the Federator Searchlet The federator searchlet interacts with other searchlets to provide a single point of search against multiple repositories. For example, the federator searchlet can invoke multiple Oracle Ultra Search searchlets to simultaneously query against multiple Oracle Ultra Search instances. In the same manner, the federator searchlet can invoke searchlets for Oracle Files, Email, and so on.

The federator searchlet is configured and managed with the Oracle Ultra Search administration tool, under the **Federated Sources** tab.

The federator searchlet is packaged as federator_searchlet.rar and is shipped under the $ORACLE_HOME/ultrasearch/adapter/ directory.

The deployment procedure for federator_searchlet.rar is similar to the deployment of the Oracle Ultra Search searchlet. To deploy the federator searchlet in OC4J standalone, use admin.jar as follows:

```
  java -jar admin.jar ormi://<hostname> <admin> <welcome> -deployment -file
federator_searchlet.rar -name FederatorSearchlet
```

To instantiate the searchlet, the federator searchlet requires four configuration properties: connectionURL, userName, password, and instanceName in the oc4j-ra.xml file. This file is typically located under the $J2EE_HOME/application-deployments/default/FederatorSearchlet/ directory. For example:

```
<connector-factory location="eis/Federator" connector-name="Federator Adapter">
 <config-property name="connectionURL" value="jdbc:oracle:thin:@dbhost:1521:sid"/>
 <config-property name="userName" value="wk_test"/>
 <config-property name="password" value="wk_test"/>
 <config-property name=InstanceName" value="wk_test"/>
</connector-factory>
```

After editing `oc4j-ra.xml`, restart the OC4J instance. If you do not see errors upon restart, then the searchlet has been successfully instantiated and bound to JNDI.

User-Defined Sources

Oracle Ultra Search lets you define, edit, or delete your own data sources and types in addition to the ones provided. You might implement your own crawler agent to crawl and index a proprietary document repository or management system, such as Lotus Notes or Documentum, which contain their own databases and interfaces.

For each new data source type, you must implement a crawler agent as a Java class. The agent collects document URLs and associated metadata from the proprietary document source and returns the information to the Oracle Ultra Search crawler, which enqueues it for later crawling.

See Also: "Oracle Ultra Search Crawler Agent API" on page 9-14

To define a new data source, you first define a data source type to represent it.

Creating User-Defined Data Source Types

To create, edit, or delete data source types, click **Manage Source Types**. To create a new type, click **Create New Type**.

1. Specify data source type name, description, and crawler agent Java class file or jar file name. The crawler agent Java classpath is predefined at installation time. The agent collects the list of document URLs and associated metadata from the proprietary document source and returns it to the Oracle Ultra Search crawler, which enqueues the information for later crawling. The agent class file or jar file must be located under `$ORACLE_HOME/ultrasearch/lib/agent/`.
2. Specify parameters for this data source type. If you add parameters, you must enter the parameter name and a description. Also, you must decide whether to encrypt the parameter value.

Edit data source type information by changing the data source type name, description, crawler agent Java class/jar file name, or parameters.

Creating User-Defined Sources

To create a user-defined data source, select the type and click **Go**.

1. Specify a name, default language, and parameter values for the data source. For information on default languages, see the Crawler Page.
2. Specify the authentication settings. This step is optional. Under **HTTP Authentication**, specify the user name and password for host and realm for which authentication is required. The realm is a name associated with the protected area of a Web site. Under **HTML Forms**, you can register HTML forms that you want the Oracle Ultra Search crawler to automatically fill out during Web crawling. HTML form support requires that HTTP cookie functionality is enabled. Cookies remember context between HTTP requests. For example, the server can send a cookie such that it knows if a user has already logged on and does not need to log

on again. Cookie support is enabled by default. Click **Register HTML Form** to register authentication forms protecting the data source.

3. Specify the ACL policy for the data source: no ACL, repository-generated ACL, or Oracle Ultra Search ACL. When a user performs a search, the ACL controls which documents the user can access. The default is no ACL, with all documents considered searchable and visible. For the Oracle Ultra Search ACL, you can add more than one group and user to the ACL for the data source.
4. Specify mappings. This step is optional. Document attributes are automatically mapped directly to the search attribute with the same name during crawling. If you want document attributes to map to another search attribute, then you can specify it here. The crawler picks up attributes that have been returned by the crawler agent or specified here.
5. Edit crawling parameters.
6. Specify the document types that the crawler should process for this data source. By default, HTML and plain text are always processed.

You can edit user-defined data sources by changing the name, type, default language, or starting address.

Schedules Page

Use this page to schedule data synchronization and index optimization. Data synchronization means keeping the Oracle Ultra Search index up to date with all data sources. Index optimization means keeping the updated index optimized for best query performance.

See Also: "Synchronizing Data Sources" on page 7-2

Data Synchronization

The tables on this page display information about synchronization schedules. A synchronization schedule has one or more data sources assigned to it. The synchronization schedule frequency specifies when the assigned data sources should be synchronized. Schedules are sorted first by name. Within a synchronization schedule, individual data sources are listed and can be sorted by source name or source type.

Creating Synchronization Schedules

To create a new schedule, click **Create New Schedule** and follow these steps:

1. Name the schedule.
2. Pick a schedule frequency and determine whether the schedule should automatically accept all URLs for indexing or examine URLs before indexing. For initial planning purposes, you might want the crawler to collect URLs without indexing. After crawling is done, you can examine document URLs and status, remove unwanted documents, and start indexing. You can also associate the schedule with a remote crawler profile.

   You can set the frequency to **Manual Launch**. In this case, the interval remains in SCHEDULED status until you explicitly invoke data synchronization by clicking **Execute Immediately** (see "Launching Synchronization Schedules" on page 8-29).
3. Assign data sources to the schedule. After a data source has been assigned to a group, it cannot be assigned to other groups.

Updating Schedules

Update the indexing option in the **Update Schedule** page.

Editing Synchronization Schedules

After a synchronization schedule has been defined, you can do the following in the **Synchronization Schedules List**:

- To assign the schedule to either a crawler that runs on the database host or a remote crawler that runs on a separate host, click **Hostname**.
- To change its frequency, click the schedule interval text.
- To alter its status, click **Status**.
- To delete it, click **Delete**.
- To edit its name, data source assignments, recrawl policy, or crawling mode, click **Edit**. When the crawler retrieves a document, it checks to see if it has changed. By default, if the document has not changed, the crawler does not process it. In certain situations, you might want to force the crawler to reprocess all documents. Click **Edit** to edit schedules in the following ways:
  - Update schedule name. This step is optional. To change the schedule name, specify a name for the schedule, and click **Update Schedule Name**.
  - Assign data sources to schedule. To assign a data source, select one or more available sources and click >>. After a data source has been assigned to a group, it cannot be assigned to any other group. To undo assignments of a data source, select one or more scheduled sources and click <<.
  - Update crawler recrawl policy. You can update the recrawl policy to the following:
    * **Process Documents That Have Changed**: This is maintenance crawling. Only documents that have changed are recrawled and indexed. For Web data sources, if there are new links in the updated document, then they are followed. For file data sources, new files are collected if its parent directory has changed.
    * **Process All Documents**: The crawler recrawls the data source. For example, suppose you want to crawl only text and HTML on a Web site. Later, you also want to crawl Microsoft Word and Adobe PDF documents. You must modify the document types for the source, edit the schedule to select **Process All Documents**, then rerun the schedule so that the crawler picks up PDF and doc document types for this data source. The crawler treats every document as if it has been changed, which means each document is fetched and processed again.

    Upon relaunching the schedule, the following rules determine which URLs will be recrawled:

    * If the previous crawl did not finish (for example, you stopped the crawling or the database tablespace was full), then the crawler only crawls URLs left in the URL queue. URLs already crawled are not touched on recrawl.
    * If the URL queue is empty but there is a new seed added since the last crawl, then the crawler only crawls the new seed.
    * If the URL queue is empty and there is no new seed URL, then the crawler recrawls all crawled URLs.

Therefore, if you stop the crawler and set Index Dynamic Pages to **No**, this only affects the URLs in the queue yet to be crawled. The already crawled dynamic pages are removed from the index on the third recrawl when the queue is empty.

Note: All crawled URLs are subject to crawler setting enforcement, not just newly crawled URLs.

– Update crawling mode. You can update the crawling mode to the following:

  * Automatically accept all URLs for indexing: This mode crawls and indexes.
  * Examine URLs before indexing: This mode crawls only. For initial planning purposes, you might want the crawler to collect URLs without indexing. After crawling is done, you can examine document URLs and status, remove unwanted documents, and start indexing.
  * Index only: This mode indexes only.

  The crawler behaves differently for the documents collected.

Crawling mode and recrawl policy can be combined for six different combinations. For example, **Process All Documents** and **Index Only** forces reindexing existing documents in this data source, while **Process Documents That Have Changed** and **Index Only** re-indexes only changed documents.

Launching Synchronization Schedules

A schedule's synchronization frequency can be identical to another schedule's synchronization frequency. This gives you maximum flexibility in managing data source synchronization.

You can launch a synchronization schedule in the following ways:

- Set a schedule frequency and wait for the predetermined launch time.
- Run it immediately. To do so, click **Status**, then **Execute Immediately**.
- Manually start the schedule.

Note: Launching a synchronization schedule can take a very long time. If a schedule has been launched before, then the next time a schedule is launched, all URLs that belong to the data source to be crawled by the schedule are updated to put into a queue. Depending on the number of URLs associated with that data source, the enqueue operation may take a long time. The administration tool displays the schedule state as 'Launching' the entire time.

The launch of a schedule does not perform any enqueue if the URL queue is not empty or if there is a new seed added since the last crawl. For example, if the user stopped the crawler earlier or if the crawler terminated because of insufficient Oracle table space, then the URL queue is not empty. So, on the next launch the crawler does not try to enqueue; instead it works on the existing URL queue until it is empty. In other words, enqueue is only performed when the queue is empty at launch time.

Synchronization Status and Crawler Progress

Click the link in the status column to see the synchronization schedule status. To see the crawling progress for any data source associated with this schedule, click **Statistics**.

If you decide to examine URLs before indexing for the schedule, then after you run the schedule, the schedule status is shown as "Indexing Pending".

In data harvesting mode, you should begin crawling first. After crawling is done, click **Examine URL** to examine document URLs and status, remove unwanted documents, and start indexing. After you click **Begin Index**, you see schedule status change from launching, running, scheduled, and so on.

You can see the following statistics:

- Data source type
- Data source name
- Start time
- Finish time
- Elapsed time
- Total indexing time
- Total size of document data collected
- Average document size
- Average fetch throughput

It also contains the following statistics:

- Documents to fetch
- Documents fetched: This includes all types of documents fetched.
- Document fetch failures: This can be an Oracle HTTP Server timeout or another HTTP server error.
- Documents rejected: The document is not within the URL boundary rule.
- Documents discovered: This includes all types of documents discovered.
- Documents indexed
- Documents non-indexable: This can be a file directory, a portal page that is a discovery node, or a robot metatag that specifies no index.
- Document conversion failures: The binary file filter failed.

Index Optimization

Index Optimization

To ensure fast query results, the Oracle Ultra Search crawler maintains an active index of all documents crawled over all data sources. This lets you schedule when you would like the index to be optimized. The index should be optimized during hours of low usage.

Note: Increasing the crawler cache directory size can reduce index fragmentation.

Index Optimization Schedule

You can specify the index optimization schedule frequency. Be sure to specify all required data for the option that you select. You can optimize the index immediately, or you can enable the schedule.

Optimization Process Duration

Specify a maximum duration for the index optimization process. The actual time taken for optimization does not exceed this limit, but it could be shorter. Specifying a longer optimization time results in a more optimized index. Alternatively, you can specify that the optimization continue until it is finished.

If your Oracle Ultra Search instance is secure-search enabled, then the index optimization process also triggers garbage collection of unused access control lists (ACLs).

Queries Page

This section lets you specify query-related settings, such as data source groups, URL submission, relevancy boosting, and query statistics.

Data Groups

Data groups are logical entities exposed to the search engine user. When entering a query, the user is asked to select one or more data groups from which to search.

A data group consists of one or more data sources. A data source can be assigned to multiple data groups. Data groups are sorted first by name. Within each data group, individual data sources are listed and can be sorted by source name or source type.

To create a new data source group, do the following:

1. Specify a name for the group.
2. Assign data sources to the group. To assign a Web or table data source to this data group, select one or more available Web sources or table sources and click >>. After a data source has been assigned to a group, it cannot be assigned to any other group. To unassign a Web or table data source, select one or more scheduled sources and click <<.
3. Click **Finish**.

URL Submission

URL Submission Methods

URL submission lets query users submit URLs. These URLs are added to the seed URL list and included in the Oracle Ultra Search crawler search space. You can allow or disallow query users to submit URLs.

URL Boundary Rules Checking

URLs are submitted to a specific Web data source. URL boundary rules checking ensures that submitted URLs comply with the URL boundary rules of the Web data source. You can allow or disallow URL boundary rules checking.

Relevancy Boosting

Relevancy boosting lets administrators override the search results and influence the order that documents are ranked in the query result list. This can be used to promote important documents to higher scores, which makes these documents easier to find.

See Also: "Document Relevancy Boosting" on page 1-9

There are two methods for locating URLs for relevancy boosting: locate by search or manual URL entry.

Locate by Search

To boost a URL, first locate a URL by performing a search. You can specify a host name to narrow the search. After you have located the URL, click **Information** to edit the query string and score for the document.

Manual URL Entry

If a document has not been crawled or indexed, then it cannot be found in a search. However, you can provide a URL and enter the relevancy boosting information with it. To do so, click **Create**, and enter the following:

1. Specify the document URL. You must assign the URL to a data source. This document is indexed the next time it is crawled.
2. Enter scores in the range of 1 to 100 for one or more query strings. When a user performs a search using the exact query string, the score applies for this URL.

The document is searchable after the document is loaded for the term. The document is also indexed the next time the schedule is run.

With manual URL entry, you can only assign URLs for Web data sources. Users get an error message on this page if no Web data source is defined.

Note: Oracle Ultra Search provides a command-line tool to load metadata, such as document relevance boosting, into an Oracle Ultra Search database. If you have a large amount of data, this is probably faster than using the HTML-based administration tool. For more information, see Appendix A, "Loading Metadata into Oracle Ultra Search".

Query Statistics

Enabling Query Statistics

This section lets you enable or disable the collection of query statistics. The logging of query statistics reduces query performance. Therefore, Oracle recommends that you disable the collection of query statistics during regular operation.

Note: After you enable query statistics, the table that stores statistics data is truncated every Sunday at 1:00 A.M.

Viewing Statistics

If query statistics is enabled, you can click one of the following categories:

- Daily Summary of Query Statistics
- Top 50 Queries
- Top 50 Ineffective Queries
- Top 50 Failed Queries

Daily Summary of Query Statistics This summarizes all query activity on a daily basis. The statistics gathered are:

- Average query time: the average time taken over all queries
- Number of queries: the total number of queries made in the day
- Number of hits: the average number of results returned by each query

Top 50 Queries This summarizes the 50 most frequent queries in the past 24 hours.

- Query string: the query string
- Average query time: the average time to return a result
- Number of queries: the total number of queries in the past 24 hours
- Number of hits: the average number of results returned by each query
- Frequency: the number of queries divided by total number of queries over all query strings
- Percentage of ineffective queries: the number of ineffective queries divided by total number of queries over all query strings

Top 50 Ineffective Queries This summarizes the 50 most frequent queries in the past 24 hours. Each row in the table describes statistics for a particular query string.

- Query string: the query string
- Number of queries: the total number of queries made in the past 24 hours
- Percentage of ineffective queries: the number of ineffective queries divided by total number of queries for that string

Top 50 Failed Queries This summarizes the top 50 queries that failed over the past 24 hours. A failed query is one where the search engine user did not locate any query results.

The columns are:

- Query string: the query string
- Number of queries: the total number of queries made in the past 24 hours
- Frequency: the percentage occurrence of a failed query
- Cumulative frequency: the cumulative percentage occurrence of all failed queries

See Also: "Tuning Query Performance" on page 10-2

Configuration

You can configure the query application and the federation engine with several parameters, including the maximum number of hits and whether to enable relevancy boosting.

Note: The Table Display URL, the File Display URL, and the E-mail Display URL are relative URLs. For Oracle Portal to work, you must replace these URLs with full URLs here, including hostname, port, and path.

Users Page

Use this page to manage Oracle Ultra Search administrative users. You can assign a user to manage an Oracle Ultra Search instance. You can also select a language preference.

Preferences

This section lets you set preference options for the Oracle Ultra Search administrator.

You can specify the date and time format. The pull-down menu lists the following languages:

- English
- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Simplified Chinese
- Spanish
- Traditional Chinese

You can also select the number of rows to display on each page.

Super-Users

A user with super-user privileges can perform all administrative functions on all instances, including creating instances, dropping instances, and granting privileges. Only super-users can access this page.

Single sign-on users can use a delegated administrative service (DAS) list of values to add another single sign-on user as a super-user. These users are authenticated by OracleAS Single Sign-On before allowing access. Database users can add another database user as a super-user.

To grant super-user administrative privileges to another user, enter the user name of the user. Specify also whether the user should be allowed to grant super-user privileges to other users. Then click **Add**.

Privileges

Only instance owners, users that have been granted general administrative privileges on this instance, or super-users are allowed to access the Privileges page. Instance owners must have been granted the `WKUSER` role.

Single sign-on users can use a delegated administrative service (DAS) list of values to add privileges to another single sign-on user. These users are authenticated by OracleAS Single Sign-On before allowing access. Database users can add privileges to another database user.

Note: Database users cannot grant privileges to single sign-on users, and single sign-on users cannot grant privileges to database users. The DAS list of values only shows single sign-on users.

Granting general administrative privileges to a user allows that user to modify general settings for this instance. To do this, enter the user name and specify whether the user should be allowed to grant administrative privileges to other users. Then click **Add**.

To remove one ore more users from the list of administrators for this instance, select one or more user names from the list of current administrators and click **Remove**.

Note: General administrative privileges do not include the ability to create or delete an instance. These privileges belong to super-users.

See Also: "Step 4: Create and Configure New Users for Oracle Ultra Search Instances" on page 5-4

Globalization Page

Oracle Ultra Search lets you translate names to different languages. This page lets you enter multiple values for search attributes, list of values (LOV) display names, and data groups.

Search Attribute Name

This section lets you translate attribute display names to different languages.

The pull-down menu lists the following languages:

- English
- Arabic
- Brazilian Portuguese
- Canadian French
- Czech
- Danish
- Dutch
- Finnish
- French
- German
- Greek
- Hebrew
- Hungarian

- Italian
- Japanese
- Korean
- Latin American Spanish
- Norwegian
- Polish
- Portuguese
- Romanian
- Russian
- Simplified Chinese
- Slovak
- Spanish
- Swedish
- Thai
- Traditional Chinese
- Turkish

LOV Display Name

This section lets you translate data group names to different languages. Select a search attribute from the pull-down menu: author, description, mimetype, subject, or title. Select the LOV type, and then select the language from the pull-down menu.

Data Group Name

This section lets you translate data group display names to different languages. The pull-down menu lists the language options.

9

Oracle Ultra Search Developer's Guide and API Reference

This chapter explains the Oracle Ultra Search APIs and related information. This chapter contains the following topics:

See Also: *Oracle Ultra Search API Reference*

Overview of Oracle Ultra Search APIs

Oracle Ultra Search provides the following APIs:

Query API

The query API works with indexed data. The Java API does not impose any HTML rendering elements. The application can completely customize the HTML interface.

Crawler Agent API

The crawler agent API crawls and indexes proprietary document repositories.

E-Mail API

The e-mail API is used by the Oracle Ultra Search query application to display e-mails. It can also be used when building your own custom query application.

URL Rewriter API

The URL rewriter API is used by the crawler to filter and rewrite extracted URL links before they are inserted into the URL queue.

Document Service API

The document service API allows generation of attribute data based on the document contents.

Oracle Ultra Search also includes highly functional query applications to query and display search results. The query applications are J2EE-compliant Web applications.

The following dependencies for `ultrasearch_query.jar` must be included if you are using it outside OC4J:

- `classes12.jar`
- `orai18n.jar`
- `orai18n-mapping.jar`
- `orai18n-translation.jar`
- `$ORACLE_HOME/oc4j/j2ee/home/jazn.jar`
- `$ORACLE_HOME/oc4j/j2ee/home/jazncore.jar`
- `$ORACLE_HOME/jlib/ldapjclnt10.jar`

Oracle Ultra Search Query API

Oracle Ultra Search provides a Java API for querying indexed data. The API methods retrieve and display query results. Because it is written in Java, it is compatible with a large spectrum of Web application servers that support any Java-based technology, such as JSP version 1.1 and higher. The API uses JDBC connection pooling for scalability.

The Java API does not impose any HTML rendering elements. The application can completely customize the HTML interface. For example, you can build the following:

- Basic search form
- Advanced search form
- Query result display
- Help page
- Feedback page
- Register URL

You embed Oracle Ultra Search query functionality in your Web application with the supplied Oracle Ultra Search Java query API. The API supports two methods:

1. Methods that retrieve query result data only.
2. Methods that retrieve HTML code containing query result data.

The data-only methods do not return any HTML and can be used when you require full control over the HTML code to be rendered. The methods that retrieve HTML code support features such as allowing you to embed query input boxes and result lists in your Web application.

With the Oracle Ultra Search Java query API you can:

- Retrieve query results
- Set query properties, such as the total number of hits to return, and so on
- Set the query session language

- Access Oracle Ultra Search tables to retrieve Oracle Ultra Search dictionary data, such as all defined data groups and attributes
- Customize and generate your query interface and search result screen with procedures that return blocks of HTML code that you can embed into your Web application
- Search end user submit URLs to the seed URL list

The Oracle Ultra Search Java query API is encapsulated in the `oracle.ultrasearch.query` package.

See Also: "Tuning Query Performance" on page 10-2

Customizing the Query Syntax Expansion

Oracle Ultra Search uses the Oracle Text engine to index and search documents. When a user specifies a certain query string, Oracle Ultra Search takes that string and transforms it into an Oracle Text query expression. This process is called query syntax expansion.

You can customize Oracle Ultra Search to use your own implementation of the query syntax expansion.

The default query expansion lets you specify a query syntax similar to most internet search engines. The syntax boosts scores for documents that match the user's query in the document title string attribute. The syntax for `Contains` is the same when used on the document content and on string attributes.

The default query syntax expansion is implemented in the `oracle.ultrasearch.query.Contains` class. To customize query expansion, use the `oracle.ultrasearch.query.CtxContains` class.

This section describes the default query expansion rules, and how to customize the query syntax expansion to suit your organization's preferences.

Default Query Syntax Expansion Implementation

The default query syntax expansion implementation directly affects the following:

- End user query syntax: The way the user enters a query string
- Scoring: The way the documents matching the query are scored
- Expansion rules: The way the user's query string is transformed into an Oracle Text query string

The default query syntax expansion is implemented in the `oracle.ultrasearch.query.Contains` class. The query applications makes use of this syntax expansion for content search as well as string attribute search.

End User Query Syntax

The end user query syntax defined by the default query syntax expansion implementation is similar to the standard text query syntax employed by most search engines on the Web.

- Token: A token is a string enclosed in double-quotes ("). It can be a single word or a phrase.
- Operators: The default implementation defines three operators. They are the [+], [-] and [*] operators. These operators are defined by the default implementation.

Change these operators to whatever you prefer in your own custom implementation.

The plus operator [+] specifies that the token immediately following it must appear in all documents included in the search result.

The minus operator [-] specifies that the token immediately following it cannot appear in any document included in the search result.

The asterisk [*] specifies a wildcard search. It matches zero or more characters. A token starting with the asterisk is ignored. The asterisk can only be specified at the end (right side) or middle of a token. For example, "hel*o" and "hell*" use the asterisk correctly, but "*ello" is unacceptable.

The following table summarizes the rules for the Oracle Ultra Search end user query syntax:

Note: All end user query strings are encased in square braces. For example, the end user query string Oracle Applications is notated as [Oracle Applications].

Rule	Description
Single word search	Entering one word finds documents that contain that word. For example, searching for [Oracle] finds all documents that contain the word "Oracle" anywhere in that document. Note: Searching for [Oracle] is not equivalent to [Oracle*].
Multiple word search	Entering more than one word finds documents that each contain any of those words in any order. For example, searching for [Oracle Applications] finds documents that contain "Oracle" or "Applications" or "Oracle Applications."
Compulsory inclusion [+]	Attaching a [+] in front of a word requires that the word be found in all matching documents. For example, searching for [Oracle + Applications] only finds documents that contain the word "Applications." Note: In a multiple word search, you can attach a [+] in front of every token including the very first token.
Compulsory exclusion [-]	Attaching a [-] in front of a word requires that the word must not be found in all matching documents. For example, searching for [Oracle - Applications] only finds documents that do not contain the word "Applications". Note: In a multiple word search, you can attach a [-] in front of every token except the very first token.
Phrase matching ["..."]	Putting quotes around a set of words only finds documents that contain that precise phrase. For example, searching for ["Oracle Applications"] finds only documents that contain the string "Oracle Applications."

Rule	Description
Wildcard matching [*]	Attaching a [*] to the right-hand side of a word returns left side partial matches. For example, searching for the string [Ora*] finds documents that contain all words beginning with "Ora," such as "Oracle" and "Orator." You can also insert an asterisk in the middle of a word. For example, searching for the string [A*e] retrieves documents that contain words such as "Apple", "Ate", "Ape", and so on. Wildcard matching requires more computational processing power and is generally slower than other types of queries.

Scoring Classes

There are three ways documents are matched against an end user query string. These three ways are known as scoring "classes." Documents are scored and ranked higher if they satisfy the requirements for a higher class. Within each class, documents are also ranked differently depending on how well they match the conditions of that scoring class.

Class 1 is the most heavily weighted class. The score is derived from the number of occurrences of a precise phrase in a document. A document that has more instances of the precise phrase have a higher score than another document that has fewer occurrences of the precise phrase.

Class 2 is the next more heavily weighted class. In this class, the closer the tokens appear in a document, the higher the score becomes. For example, an end user query string [Oracle Applications Financials] can result in three documents found. None of the three documents contain the precise phrase "Oracle Applications Financials." However, document X contains the all three tokens "Oracle", "Applications", and "Financials" in the same sentence separated by other words. Document Y contains the individual tokens in the same paragraph but in different sentences. Document Z contains the same three tokens, but each token resides in different paragraphs. In this scenario, document X has the highest score, because the tokens are closest together. Likewise, Y has a higher score than Z.

Class 3 is the least weighted class. A document that has more tokens gets a higher score. For example, an end user query string [Oracle Applications Financials] can result in three documents found. Document X might contain all three tokens. Document Y might contain the tokens "Oracle" and "Applications" only. Document Z might contain only the token "Oracle." In this scenario, document X has a higher score than Y. Likewise, Y has a higher score than Z.

Expansion Rules

As mentioned previously, the end user query is expanded to an Oracle Text query. The expanded query string rules are captured in BNF (Backus Naur Form) notation. Again, these rules are the rules that Oracle Ultra Search uses as a default query syntax expansion implementation.

The rules that define an expanded query:

<expanded query> ::= (<expression> within <title section>)*2, <expression>

<expression> ::= <generic query expression> | <simple query expression>

<generic query expression> ::= (([<plus expression>*100 &]) (<main expression>)) [<minus expression>]

<simple query expression> ::= (<phrase expression>)*2, (<main expression>)

<main expression> ::= (<near expression>)*2, (<accum expression>)

The following list contains some terms and their meanings, which explain some of the terms used in the preceding rules:

A <plus expression> is an AND expression of all plus tokens.

A <minus expression> is a NOT expression of all minus tokens.

A <phrase expression> is a PHRASE formed by all tokens in the <main expression>

A <near expression> is a NEAR expression of all tokens but minus tokens.

An <accum expression> is an ACCUMULATE expression of all tokens but minus tokens.

A <simple query expression> is used only when the end user query has multiple tokens and does not have any operator or a double quote. Otherwise, a <generic query expression> is used.

If there is no token that is neither plus token nor minus token, then the <plus expression> and the <accum expression> are eliminated.

Examples of Applying the Rules

The following table illustrates how the default query syntax expansion implementation converts end user query strings into Oracle Text compatible query strings.

End User Query String	Expanded Query String Understandable by Oracle Text
[Oracle]	`((({Oracle}) within TITLE__31)*2,({Oracle}))`
[Oracle + Applications]	`(((((Applications})*10)*10&(({Oracle};{Applications})*2,({Oracle},{Applications}))) within TITLE__31)*2,((((Applications})*10)*10&(({Oracle};{Applications})*2,({Oracle},{Applications})))))`
[Oracle - Applications]	`(((({Oracle})~{Applications}) within TITLE__31)*2,(({Oracle})~{Applications}))`
["Oracle Applications"]	`((({Oracle Applications}) within TITLE__31)*2,({Oracle Applications}))`
[Ora*]	`((((Ora%)) within TITLE__31)*2,((Ora%)))`
[Oracle Applications]	`(((({Oracle Applications})*2,(({Oracle};{Applications})*2,({Oracle},{Applications}))) within TITLE__31)*2,(({Oracle Applications})*2,(({Oracle};{Applications})*2,({Oracle},{Applications})))))`

Customizing the Rules

Customize this expansion to suit your organization's purposes by defining and implementing your own query syntax expansion. You should have detailed understanding of Oracle Text queries using the `ctxsys.contains` operator. Oracle Text offers a rich set of linguistic features, such as thesaurus, theme, stemming, and soundex as a part of its query language.

See Also:

- *Oracle Text Application Developer's Guide*
- *Oracle Text Reference*

To customize Oracle Ultra Search to use your own implementation of the query syntax expansion, use the `oracle.ultrasearch.CtxContains` class in your query application instead of the `oracle.ultrasearch.query.Contains` class. `CtxContains` lets you use any Oracle Text query as a part of an Oracle Ultra Search query. Use the following steps:

1. Construct a Oracle Text query based on the user's input. For example, if the user's input is "cat", using the stemming feature, you can construct a Text query "$cat", which will find documents with "cat" or "cats". You can use any tool to construct the Text query, as long as it is a string object. Depending on the complexity of user's query syntax, you might want to leverage some existing lexers in Java.
2. Construct a `CtxContains` using the Text query. For example:

   ```
   String textQuery = "$cat";
   oracle.ultrasearch.Query query = new oracle.ultrasearch.CtxContains
   (textQuery);
   ```

   The preceding code constructs a query for documents with "cat" or "cats". You can also limit that query to document titles (not content) as follows:

   ```
   String textQuery = "cat";
   StringAttribute titleAttribute = instanceMetaData.getStringAttribute("TITLE");
   oracle.ultrasearch.Query query = new oracle.ultrasearch.CtxContains (textQuery,
   titleAttribute);
   ```

3. You can optionally combine the `CtxContains` with any other Oracle Ultra Search query by joining them with the And / Or query operators.
4. Run the query by invoking the `getResult` method with the constructed query object.

   See Also: *Oracle Ultra Search Java API Reference* for detailed information on the `oracle.ultrasearch.query.CtxContains` API

Oracle Ultra Search Query Tag Library

On top of the Java query API, Oracle Ultra Search provides a JSP tag library as an alternative for developing search applications. Based on the Sun Microsystems JavaServer Pages specification version 1.1, the Oracle Ultra Search tag library better separates the dynamic / Java development effort from the static / HTML development effort, and enables Web developers who are unfamiliar with Java to incorporate search functionality into their applications.

The Oracle Ultra Search tag library provides a subset of the features in the Java query API. Advanced features, such as custom query expansion and URL submission, are not available as tags. The main features of the tag library are the following: ability to retrieve search attributes, groups, languages, and LOVs for rendering the advance query form; and ability to iterate through the resulting result set, and retrieve document attributes and properties for rendering the result page.

The tag library is summarized in following table:

Tag	Description	Attributes
instance	This tag establishes a connection to an Oracle Ultra Search instance.	instanceId username password URL dataSourceName tablePagePath emailPagePath filePagePath
showAttributes	For an advanced query, use this tag to show the list of attributes available.	instance locale
showGroups	For an advanced query, use this tag to show the list of groups.	instance locale
showLanguages	For an advanced query, use this tag to show the list of languages defined in the instance.	instance
showLOV	Show all values defined for a search attribute.	instance locale attributeName attributeType
getResult	Perform the search.	resultId instance query queryLocale documentLanguage from to boostTerm withCount
fetchAttribute	This is a nested tag within getResult to specify which attributes of each document should be fetched along with the query results. There can be any number of nested fetchAttribute tags.	attributeName attributeType
showHitCount	If withCount="true" in the getResult tag, then the result includes a total number of hits, and you can use showHitCount to display this number.	result
showResults	Renders the results of the search.	result instance
showAttributeValue	Renders a document attribute.	attributeName attributeType

Details of these tags are described in the following subsections. Note the following requirements for using Oracle Ultra Search tags:

- Install the file `ultrasearch_query.jar` and include it in classpath or the `WEB-INF/lib` directory of the Web application. This file is provided with the Oracle Ultra Search installation under the `ultrasearch/lib` directory.

- Make sure that the tag library description file, ultrasearch-taglib.tld, is deployed with the application and is in the location specified in the taglib directives of your JSP pages, such as in the following example: <%@ taglib uri="/WEB-INF/ultrasearch-taglib.tld" prefix="US" %>

The Oracle Ultra Search tag library definition (TLD) file can be found in `$ORACLE_HOME/ultrasearch/sample/query/WEB-INF/ultrasearch-taglib.tld` after `sample.ear` has been deployed. It is also packaged with `ultrasearch_query.jar` under the name `META-INF/taglib.tld`.

Query Tag Descriptions

The following section describes each Oracle Ultra Search tag, its attributes, and action. Examples are shown without any static HTML, which can be inserted to format the output.

<instance> Tag: Connecting to the Oracle Ultra Search Instance

This tag establishes a connection to an Oracle Ultra Search instance. Some basic parameters must be established for this tag to work, such as JDBC connection string, schema user name/password, Oracle Ultra Search instance name, and so on.

Attribute Name	Description
instanceId="name"	Names the instance defined by this tag. This name is then used by other Oracle Ultra Search tags to specify the instance being searched.
username	Creates a database connection.
password	Creates a database connection.
url	Gets the URL used to create a JDBC connection. This attribute is optional if dataSourceName is specified.
dataSourceName	The JNDI name that identifies a JDBC data source. Users should set either the URL or data source name properties. This is optional if URL is specified.
instanceName	The name of the Oracle Ultra Search instance that is owned by the schema user. If the schema user owns only one Oracle Ultra Search instance, then this is optional.
tablePagePath	The URL path of the Web application that renders the contents of a database table.
emailPagePath	The URL path of the Web application that renders the contents of an e-mail.
filePagePath	The URL path of the Web application that renders the contents of a file.

This tag defines a scripting variable of the name set by the instanceId property. All the other tag properties correspond to a property in the `oracle.ultrasearch.query.QueryInstance` class. Either the URL or the `dataSourceName` attribute should be set. They are exclusive of each other.

The following example uses the URL property to connect to the database.

```
<US:instance
 instanceId="mybookstore"
 url="oracle:jdbc:thin:@dbhost:1521:inst1"
 username="scott"
 password="tiger"
 tablePage="../display.jsp"
 emailPage="../mail.jsp"
 filePage="../display.jsp"
/>
```

<iterAttributes> Tag: Show All Search Attributes

When a user wants to perform an advanced query, the application needs to show the list of attributes that are available, the list of groups, and the list of languages defined in the instance. This can be done using some iteration tags that define script variables for page rendering.

Each attribute in Oracle Ultra Search has a name, a type, and a display name that is translated depending on the locale that is set for the QueryInstance tag. The attribute type should be used to determine which operators can be used on this attribute and how to parse the user's input.

Attribute Name	Description
instance="name"	This is a mandatory attribute to refer to the object defined by the instance tag.
locale="locale"	This determines the display name fetched using this tag.

This tag is an iteration tag. It loops through all the search attributes in the instance referred to by the instance tag attribute. In each loop, it defines a scripting variable named "attribute", which is an `oracle.ultrasearch.query.Attribute` object. It also defines a string variable named "displayname", which is the localized name of the attribute.

The following example shows all the attributes in "mybookstore" instance, using their English display names.

```
<US:iterAttributes instance="mybookstore" locale="<%=Locale.ENGLISH%>" >
<%= attribute %>
<%= displayname %>
</US:iterAttributes>
```

<iterGroups> Tag: Show All Search Groups

Similar to the showAttributes tag, the showGroups tag iterates through all the groups defined in an instance.

Attribute Name	Description
instance="name"	This is a mandatory attribute to refer to the object defined by the instance tag.
locale="locale"	This determines the display name fetched using this tag.

This tag loops through all the search groups in the instance referred to by the instance tag attribute. In each loop, it defines a scripting variable named "group", which is an `oracle.ultrasearch.query.Group` object. It also defines a string variable named "displayname", which is the localized name of the group.

The following example shows all the groups in "mybookstore" instance, using their English display names.

```
<US:iterGroups instance="mybookstore" locale="<%=Locale.ENGLISH%>" >
<%= group %>
<%= displayname %>
</US:iterGroups >
```

<iterLanguages> Tag: Show All Search Languages

Similar to the showAttributes tag, the showLanguages tag iterates through all the languages defined in an instance. Because each language is defined by a `java.util.Locale` object, their display names are not handled by Oracle Ultra Search. Therefore, this tag does not define the displayname scripting variable.

Attribute Name	Description
instance="name"	This is a mandatory attribute to refer to the object defined by the instance tag.

This tag is an iteration tag. It loops through all the search languages in the instance referred to by the instance tag attribute. In each loop, it defines a scripting variable named "language", which is a `java.util.Locale` object. The display name for the language is provided by Java as a property of the object itself (through the `getDisplayName` method).

The following example shows all the languages in "mybookstore" instance, using their English display names.

```
<US:iterLanguages instance="mybookstore">
<%= language %>
<%= language.getDisplayName (Locale.ENGLISH) %>
</US:iterLanguages >
```

<iterLOV> Tag: Show All Values Defined for a Search Attribute

Attribute Name	Description
instance="name"	This is a mandatory attribute to refer to the object defined by the instance tag.
locale="locale"	This determines the display name fetched using this tag.
attributeName="attname"	The name of the attribute whose LOV is being fetched in this LOV.
attributeType="string | number | date"	The type of the attribute whose LOV is being fetched in this LOV. This is needed because attribute name does not uniquely identify an attribute in the instance.

This tag is an iteration tag. It loops through all the values in a search attribute's LOV. In each loop, it defines a scripting variable named "value", which is either a `java.lang.String`, `java.util.Date`, or `java.math.BigDecimal` object, depending on the attribute type. It also defines a string variable named "displayname", which is the localized display name of the value.

The following example shows all the values for a string attribute named "Dept" in "mybookstore" instance, using their English display names.

```
<US:iterLOV instance="mybookstore" attribute_name="Dept" attribute_type="String" >
<%= value %>
<%= displayname %>
</US:iterLOV >
```

Formulating the Query

Oracle Ultra Search supports a set of classes for building queries. Currently these classes do not have any tag equivalents.

<getResult> Tag: Perform Search

This tag performs the search and returns the result by defining a scripting variable of the type `oracle.ultrasearch.query.Result`.

Attribute Name	Description
resultId="name"	This names the result generated by this tag. This name is then used by other tags to render the result on the page.
instance="name"	This is a mandatory attribute to refer to the object defined by the instance tag.
query="<%= expression %>"	This specifies a query object to search with.
queryLocale="locale"	This specifies the locale of the query object.
documentLanguage="locale"	This specifies the language of the documents for which to search. This is optional. If it is not specified, then all languages are included in the search.
from="number"	This specifies the index of the first result.
to="number"	This specifies the index of the last result.
boostTerm="string"	This specifies the search term that is used for relevance boosting. This is optional.
withCount="true | false"	This specifies whether the result has an estimate of the total result count. This is optional. If unspecified, the behavior is same as withCount=false.

The <getResult> tag corresponds to the getResult method on the `oracle.ultrasearch.query.Instance` class. The attributes of tag map to the parameters of the method, with the exception that getResult method can specify the attributes to fetch. The <getResult> tag require the use of the nested <fetchAttribute> tag to accomplish metadata selection.

The following example shows a search for the first 20 documents of a query in English that appears in French documents.

```
<US:getResult
 resultId="searchresult"
 instance="mybookstore"
 query=""
 queryLocale=""
 documentLanguage=""
 from="1" to="20">
</US:getResult>
```

<fetchAttribute> Tag: Metadata Selection

This tag is used as nested tag inside <getResult>. It specifies which attributes of each document should be fetched along with the query result. Each <getResult> can have any number of nested <fetchAttribute> tags.

Attribute Name	Description
attributeName="attname"	The name of the attribute whose LOV is being fetched in this LOV.
attributeType="string | number | date"	The type of the attribute whose LOV is being fetched in this LOV. This is needed because attribute name does not uniquely identify an attribute in the instance.

Each occurrence of the <fetchAttribute> adds to the list of attributes passed to the getResult invoked by the <getResult> tag.

The following example shows the same search in <getResult> tag, but fetching title and publication-date attributes of each book.

```
<US:getResult
 resultId="searchresult"
 instance="mybookstore"
 query=""
 queryLocale=""
 documentLanguage=""
 from="1" to="20">
<US:fetchAttribute
 attributeName="title"
 attributeType="string" />
<US:fetchAttribute
 attributeName="publication-date"
 attributeType="date" />
</US:getResult>
```

<showHitCount> Tag: Show Estimated Hit Count

After the search is performed, the result must be rendered. If withCount=true is in the <US:getResult> tag, then the result contains a count of total hits, and <showHitCount> tag can be used to display it.

Attribute Name	Description
result="name"	This refers to the resultId specified in the <US:getResult> tag.

This tag outputs the result count to the page.

The following shows the result count of the a search result.

```
<US:showHitCount result="searchresult" />
```

<iterResult> Tag: Render the Results

This tag is an iteration tag. It loops through all the documents in a search result.

Attribute Name	Description
result="name"	This refers to the resultId specified in the <US:getResult> tag.
instance="name"	This refers to the instanceId specified in the <US:instance> tag.

The tag loops through all the documents in a search result and defines a scripting variable "doc" that is a `oracle.ultrasearch.query.Document` object. In addition, it can have nested tags of <showAttributeValue>, which helps to render the document's attributes. It is an error if the result specified is not one obtained from search on the instance specified. In other words, the result must come from the instance.

The following example shows the URL of all documents in a search result.

```
<US:iterResult
result="searchresult"
instance="mybookstore">
</US:iterResult>
```

<showAttributeValue> Tag: Render a Document Attribute

This tag shows an attribute of a document within the <US:iterResult> tag.

Attribute Name	Description
attributeName="attname"	The name of the document attribute.
attributeType="string | number | date"	The type of the document attribute. This is needed because attribute name does not uniquely identify an attribute in the instance.
default="default string"	A value to output when the document has no value for this attribute. This is useful when a document has no title. The string "No Title" can be displayed as the default value.

This tag looks up the document attribute value and renders it on the page. If the attribute was not fetched as part of the search result, then nothing is output to the page.

The following example shows the title and publication dates of all documents in a search result.

```
<US:iterResult
result="searchresult"
instance="mybookstore">
<US:showAttributeValue attributeName="title" attributeType="string" default="No
Title" />
<US:showAttributeValue attributeName="publication-date" attributeType="date" />
</US:iterResult>
```

Oracle Ultra Search Crawler Agent API

You can implement a crawler agent to crawl and index a proprietary document repository, such as Lotus Notes or Documentum. In Oracle Ultra Search, the proprietary repository is called a user-defined data source. The module that enables the crawler to access the data source is called a crawler agent.

The agent collects document URLs and associated metadata from the user-defined data source and returns the information to the Oracle Ultra Search crawler, which enqueues it for later crawling. The crawler agent must be implemented in Java using the Oracle Ultra Search crawler agent API.

Oracle Ultra Search provides a sample implementation of user-defined crawler agents using the Oracle Ultra Search agent API. Upon invocation, this sample agent connects to a specified Oracle database and retrieves the contents of a table for the crawler to collect and index.

The sample agents are fully functional and can be customized to adapt to other database-based data sources. These agents performs the following tasks:

- Read data source parameters
- Connect to the database that contains the data source
- Initialize fetching document URL and attributes from the data source
- Fetch document URL and attributes from the data source
- Disconnect from the data source

Crawler Agent Overview

A crawler agent does the following:

- Authenticates the crawler for accessing the data source
- Provides access to the data source document through a HTTP URL (display URL)
- Provides the metadata of the document in the form of document attributes
- Maps each document attribute to a common attribute name used by end users
- Provides a "flattened" view of the data source, such that documents are retrieved one by one in a streaming fashion
- Instructs the crawler to parse the URL document for standard metadata, like author and title, if necessary
- Optionally provides the list of URLs that have changed since a given time stamp
- Optionally provides an access URL in addition to the display URL for the processing of the document

From the crawler's perspective, the agent retrieves the list of URLs from the target data source and saves it in the crawler queue before processing it.

Note: If the crawler is interrupted for any reason, then the agent invocation process is repeated with the original last crawl time stamp. If the crawler finished enqueuing URLs fetched from the agent and is half way done crawling, then the crawler only starts the agent, but does not try to fetch URLs from the agent. Instead, it finishes crawling the URLs already enqueued.

There are two kinds of crawler agents:

- Standard Agent
- Smart Agent

Standard Agent

The standard agent returns the list of URLs currently existing in the data source. It does not know whether any of the URLs had been crawled before, and it relies on the crawler to find any updates to the target data source. The standard agent's interaction with the crawler is the following:

- Crawler marks all existing URLs of this data source for garbage collection, assuming they no longer exist in the target data source.

- Crawler calls the agent to get an updated list of URLs. It marks for crawling every URL that already exists. If it is new, it inserts it into the URL table and queue.
- Crawler deletes the URLs that are still marked for garbage collection.
- Crawler goes through every URL marked for crawling and checks for updates.

Smart Agent

The smart agent uses a modified-since time stamp (provided by the crawler) to return the list of URLs that have been updated, inserted, and deleted. The crawler only crawls URLs returned by the agent and does not recrawl existing ones. For URLs that were deleted, the crawler removes them from the URL table. If the smart agent can only return updated or inserted URLs but not deleted URLs, then deleted URLs are not detected by the crawler. In this case, you must change the schedule crawler recrawl policy to periodically run the schedule in force recrawl mode. Force recrawl mode signals to the agent to return every URL in the data source.

The agent API isDeltaCrawlingCapable tells the crawler whether the agent it invokes is a standard agent or a smart agent. The agent API startCrawling(boolean forceRecrawl, Date lastCrawlTime) lets the crawler tell the agent the last crawl time and whether the crawler is running in force recrawl mode.

Document Attributes and Properties

Document attributes, or metadata, describe document properties. Some attributes can be irrelevant to your application. The crawler agent creator must decide which document attributes should be extracted and saved. The agent also can be created such that the list of collected attributes are configurable. Oracle Ultra Search automatically registers attributes returned by the agent. The agent can decide which attributes to return for a document.

Library Path and Java Class Path

Any other Java class needed by the agent should be included the agent jar file. This is because Oracle Ultra Search automatically adds the agent jar file to the crawler Java class path, and Oracle Ultra Search does not let you add other class paths from the administration interface. To add a new class path, see Appendix B, "Altering the Crawler Java Classpath"

If the agent code also relies on a particular library file (for example, a .ddl file on Windows or a .so file on UNIX), then the library path environment variable (`PATH` on Windows, `LD_LIBRARY_PATH` on UNIX) must contain the path to it. Make sure that Oracle is started from this environment. As the crawler is spawned by the Oracle process, it automatically inherits all environment variables from Oracle, including the library path.

Crawler Agent Functionality

This section describes aspects of the crawler agent.

Data Source Type Registration

A data source type is an abstraction of a data source. You can define new data source types with the following attributes:

- Name of data source type: For example, Lotus Notes. The name cannot be more than 100 bytes.
- ID of data source type: This is automatically assigned.

- Description of the data source type: This limit is 4000 bytes.
- Agent Java class name: For example, WebDbAgent. The location of this class is predefined by Oracle Ultra Search in `$ORACLE_HOME/ultrasearch/lib/agent/` and cannot be changed.
- Agent Java jar file name: The agent class can be stored in a Java jar file. This jar file must be in `$ORACLE_HOME/ultrasearch/lib/agent/`, where `$ORACLE_HOME` is the Oracle home directory where the Oracle Ultra Search backend, *not* the middle tier, is installed.
- Parameters: Parameters are the properties of a data source; for example, seed URL, inclusion pattern, and robots exclusion for a Web data source. Define a parameter by specifying a parameter name (100 bytes maximum) and a description (4000 bytes maximum). By default, a parameter is not encrypted.
- Encryption: Should the value of this parameter be encrypted when stored.

Oracle Ultra Search does not enforce the occurrence of parameters. You cannot specify a particular parameter to have 0 or more, at least 1, or only 1 occurrence.

Agent Class Dependency The crawler agent class has a dependency on `log4j.jar`, Apache's logging facility. The crawler hangs when the log4j logger class is first referenced in the agent Java class. This section describes how to put this classpath into Oracle Ultra Search.

You can bundle log4j classes and the agent class into one jar file and specify this jar file for the custom type in the administration tool.

Or, you can package your own `manifest.mf` file in the `agent.jar`. You can then specify a dependent library path, which will be loaded by JVM. For example, assume that you are in: /home/pdevulap/make_jar directory, and you have the following files and directories:

```
META-INF/MANIFEST.MF
oracle/marketing/search/crawleragent/SearchCrawlerAgent.java + other class files
log4j-1.2.8.jar
```

Example of the contents of `MANIFEST.MF`:

```
Manifest-Version: 1.0
Created-By: Praveen Devulapalli
Main-Class: oracle.marketing.search.crawleragent.SearchCrawlerAgent
Class-Path: log4j-1.2.8.jar
```

If there are other jars, simply separate them with spaces

1. Create the agent jar file:

   ```
   jar cvfm SearchCrawlerAgent.jar META-INF/MANIFEST.MF
   oracle/marketing/search/crawleragent/*
   ```

   This creates `SearchCrawlerAgent.jar`, along with the manifest file specified. In creating the jar file, use lowercase 'm' to tell the jar utility to use your manifest file.

   Note: log4j is NOT included in this jar file.

2. Move the jar files to the appropriate location of Oracle Ultra Search; that is, `$ORACLE_HOME/ultrasearch/lib/agent/SearchCrawlerAgent.jar` and `$ORACLE_HOME/ultrasearch/lib/agent/log4j-1.2.8.jar`

To update the existing agent jar file with the new class path:

1. Check if the jar already has a `MANIFEST.MF` file:

```
% jar tf my.jar

META-INF/MANIFEST.MF

a/b/c/d/xyz.class

...
```

   If the jar does not have the `MANIFEST.MF`, then skip step 2.

2. Extract the manifest file:

```
% jar xf my.jar META-INF/MANIFEST.MF  --you must specify the complete path as
it appears in the jar

META-INF/MANIFEST.MF
```

3. Change the contents of `MANIFEST.MF`, and update the jar file with the new version:

```
%jar umf META-INF/MANIFEST.MF my.jar
```

Data Source Registration

After a data source type is defined, any instance of that data source type can be defined:

- Data source name
- Description of the data source; limit to 4000 bytes
- Data source type ID
- Default language; default is 'en' (English)
- Parameter values; for example, seed - `http://www.oracle.com depth - 8`

Data Source Attribute Registration

You can add new attributes to Oracle Ultra Search by providing the attribute name and the attribute data type. The data type can be string, number, or date. Attributes with the same name but different data type can be added. Attributes returned by an agent are automatically registered if they have not been defined.

User-Implemented Crawler Agent

The crawler agent has the following requirements:

- The agent must be implemented in Java.
- The agent must support the Java agent APIs defined by Oracle Ultra Search.
- The agent must return the URL attributes and properties.
- The agent optionally can authenticate the crawler's access to the data source.
- The agent must "flatten" the data source such that each document is retrieved one by one in a streaming fashion. This is to encapsulate the crawling logic of a specific data source into the agent.
- The agent must decide which document attributes Oracle Ultra Search should keep. Any attribute not defined in Oracle Ultra Search is registered automatically.

- The agent can map attributes to data source properties. For example, if an attribute "ID" is the unique ID of a document, then the agent should return (document_key, 4) where "ID" has been mapped to the property "document_key" and its value is 4 for this particular document.
- If the attribute LOV is available, then the agent returns them upon request.

Interaction Between the Crawler and the Crawler Agent

The crawler crawls data sources defined by the user through the invocation of the user-supplied crawler agent. The crawler can do the following:

- Invoke the crawler agent of the defined data source
- Supply data source parameter information to the agent
- Authenticate itself with the agent if needed
- Retrieve a list of URLs and associate attributes / properties that must be crawled
- Use the URL provided by the agent to retrieve the document
- Detect insert, update, and delete to the data source
- Retrieve attribute LOV data if available

Crawler Agent APIs and Classes

The crawler agent API is a collection of methods used to implement a crawler agent. A sample implementation of a crawler agent `SampleAgent.java` is provided under `$ORACLE_HOME/ultrasearch/extension/`.

UrlData: The crawler agent uses this interface to populate document properties and attribute values. Oracle Ultra Search provides a basic implementation of this interface that the agent can use directly or extend if necessary. The class is DocAttributes with a constructor that has no argument. The agent might decide to create a pool of UrlData objects and cycle through them during crawling. In the most simple implementation, the agent creates one DocAttributes object, repeatedly resets and populates the data, and returns this object.

LovInfo: The crawler agent uses this interface to submit attribute LOV definitions.

DataSourceParams: The crawler agent uses this interface to read and write data source parameters.

AgentException: The crawler agent uses this exception class when an error occurs.

CrawlerAgent: This interface lets the crawler communicate with the user-defined data source. The crawler agent must implement this interface.

Sample Agent Files

The sample agent files are located in the `$ORACLE_HOME/ultrasearch/extension` directory. You can view the agent source code using your preferred text editor.

There is a `SampleAgent_readme.htm` file and a `SampleAgent.java` file. These are for the sample crawler agent implementation using agent APIs.

Setting up the Sample Crawler Agent

This section describes how to set up the sample crawler agent.

Compiling and Building the Agent Jar File

The Java source code for the sample agent first must be compiled into class files and put into a jar file in the `$ORACLE_HOME/ultrasearch/lib/agent/` directory, where `$ORACLE_HOME` is the Oracle home directory where the Oracle Ultra Search backend, *not* the middle tier, is installed.

The classes needed for compilation are the JDK class (`classes.zip`), Oracle JDBC Thin Driver (`classes12.zip`), and `ultrasearch.jar`. For example:

```
$ORACLE_HOME/jdk/bin/javac -J-ms16m -J-mx96m -O -classpath $ORACLE_
HOME/dbjava/lib/classes12.zip:
$ORACLE_HOME/ultrasearch/lib/ultrasearch.jar SampleAgent.java
```

To build the `SampleAgent.jar` file, enter the following:

```
$ORACLE_HOME/jdk/bin/jar cv0f  $ORACLE_HOME/ultrasearch/lib/agent/SampleAgent.jar
SampleAgent.class 'SampleAgent$DocNode.class'
```

Creating a Data Source Type

A data source type that uses the sample agent must be created first.

- Name: URL table type
- Description: Table with rows of URLs
- Agent Name: SampleAgent
- Agent Jar File: sampleagent

Defining Data Source Parameters

Define parameters for a data source type:

- Database Connect String (DB connection)
- User Name (schema owner of the URL table)
- Password (schema owner password, encrypted)
- Table Name (URL table name)
- URL Column (Column holding doc URLs)
- Ignore Flag Column (1 for ignoring, 0 otherwise)
- Language Column (Document Language)
- Attribute List (List of column for attributes)
- It is in the following format: [column name / attribute name] <data type> [column name / attribute name] <data type> ... where <data type> 0 is number, 1 is string, and 2 is date. For example, if the document has 4 attributes: Company Name, Category, Revenue, S&P Rating, then it is specified as: [Company Name / Company / 1][Category / Classification / 1][Revenue / Revenue / 0][Rating / Analyst Rating / 1]
- Log File Name (log file)
- Log Directory (Location of log file)

Defining a Data Source of this Type

A data source is defined, which initializes the data source parameters. For example, the value specified accesses a table whose schema is the following:

```
TABLE NEWS (
ARTICLE_NO    NUMBER,
NEWS_URL      VARCHAR2(740),
TITLE         VARCHAR2(200),
AUTHOR        VARCHAR2(100),
PUB_DATE      DATE default SYSDATE,
PUBLISHER     VARCHAR2(100),
PRICE         NUMBER,
LANG          VARCHAR2(10),
IGNORE        NUMBER DEFAULT 0,
PRIMARY KEY (NEWS_URL)
);
```

- Database Connect String: dlsun1710:5521:search
- User Name: SCOTT
- Password: TIGER
- Table Name: NEWS
- URL Column: NEWS_URL
- Ignore Flag Column: IGNORE
- Language Column: LANG
- Attribute List: [ARTICLE_NO / Article Number / 0][TITLE / Article Title / 1][AUTHOR / Author / 1][PUB_DATE / Report Date / 2][PUBLISHER / Newspaper / 1][PRICE / Download Cost / 0]
- Log File Name: `testagent.log`
- Log Directory: `/tmp/ultrasearch/`

Oracle Ultra Search Java E-mail API

Oracle Ultra Search provides a Java API for accessing archived e-mails. The Oracle Ultra Search query application uses the API to display e-mails addressed to mailing lists that have been indexed by the Oracle Ultra Search system. The API can also be used to build your own custom query application.

The application user-interface logic is entirely controlled in the JSP. Therefore, you can customize the look-and-feel to your needs.

E-mail documents contain valuable information, but they are not structured to find specific relevant information easily. Oracle Ultra Search lets you retrieve and index e-mails on a server that supports the IMAP4 protocol.

An e-mail source is a data source that derives its content from e-mails sent to a specific e-mail address. When the Oracle Ultra Search crawler searches an e-mail source, the crawler collects all e-mails that have the specific e-mail address in any of the "To:" or "Cc:" e-mail header fields.

> **Note:** Oracle Ultra Search stores copies of all retrieved e-mails in the local file system of the Oracle Ultra Search server installation.

A possible application of an e-mail source is where an e-mail source represents all e-mails sent to a mailing list. In such a scenario, multiple e-mail sources are defined where each e-mail source represents an e-mail list.

Oracle Ultra Search e-mail crawling and rendering is built on top of the JavaMail API using Sun Microsystems' reference implementation of JavaMail. This enables Oracle Ultra Search to provide a Java API for accessing indexed e-mails. The API is known as the Oracle Ultra Search Java e-mail API. This API lets you retrieve information such as e-mail header information, e-mail body content, and attachments of an e-mail.

Use this API to embed Oracle Ultra Search e-mail browsing functionality into JavaServer Page (JSP) or servlet-based Web applications. Oracle Ultra Search ships a fully functional JSP Web application that directly uses this API to render indexed e-mails. Because the source code is viewable, you can use it as an example for building your own customized e-mail browser.

JavaMail Implementation

Oracle Ultra Search requires a JavaMail 1.1 compliant implementation. The reference implementation by Sun Microsystems is JavaMail version 1.2. This reference implementation is shipped with Oracle Ultra Search.

Java E-mail API

The Oracle Ultra Search Java e-mail API is encapsulated in the `oracle.ultrasearch.query` package.

Mailing List Browser Application Files

The mailing list browser applications files are located in the `$ORACLE_HOME/ultrasearch/sample/query` directory. You can directly view the mailing list browser application source code using your preferred text editor.

The following tables describe all mailing list browser application files, README file, and stylesheets:

File	Description
`SampleAgent_readme.html`	Readme
`mail.css`	Style sheet for the e-mail Web application

JavaServer Page Mailing List Browser Applications Files:

File	Description
`mail.jsp`	Mailing list browser applications that selectively include HTML code returned by other JSP files, depending on what the end user wants to view
`mailindex.jsp`	JSP page that displays all e-mail sources (mailing lists) of an Oracle Ultra Search instance
`mailmsgs.jsp`	JSP page that displays all e-mails for an e-mail source (mailing list)
`mailreader.jsp`	JSP page that displays an e-mail
`mailutil.jsp`	JSP page that defines various functions that are used by `mailreader.jsp`

Graphics Files for All Applications:

File	Description
`images/ultra_mediumbanner.gif`	Oracle Ultra Search banner
`images/wsd.gif`	Background image used in query application

Setting up the Mailing List Browser Application

For detailed instructions on setting up the JSP mailing list browser application, see "Installing the Oracle Ultra Search Middle Tier" on page 3-3.

Oracle Ultra Search URL Rewriter API

A URL rewriter is a user supplied Java module that implements the Oracle Ultra Search UrlRewriter Java interface. When activated, it is used by the crawler to filter and rewrite extracted URL links before they are inserted into the URL queue.

Web crawling generally consists of the following steps:

1. Get the next URL from the URL queue. (Web crawling stops when the queue is empty.)
2. Fetch the contents of the URL.
3. Extract URL links from the contents.
4. Insert the links into the URL queue.

The generated new URL link is subject to all existing host, path, and mimetype inclusion and exclusion rules.

There are two possible operations that can be done on the extracted URL link:

- Filtering: removes the unwanted URL link
- Rewriting: transforms the URL link

URL Link Filtering

Users control what type of URL links are allowed to be inserted into the queue with the following mechanisms supported by the Oracle Ultra Search crawler:

- `robots.txt` file on the target Web site; for example, disallow URLs from the `/cgi` directory
- Hosts inclusion and exclusion rules; for example, only allow URLs from www.acme.com
- File path inclusion and exclusion rules; for example, only allow URLs under the `/archive` directory
- Mimetype inclusion rules; for example, only allow HTML and PDF files
- Robots metatag `NOFOLLOW`; for example, do not extract any link from that page
- Black list URL; for example, URL explicitly singled out not to be crawled

Note: All URLs must pass domain rules before being checked for path rules. Path rules let you further restrict the crawling space. Path rules are host-specific, but you can specify more than one path rule for each host. For example, on the same host, you can include path `files://host/doc` and exclude path `files://host/doc/unwanted`.

With these mechanisms, only URL links that meet the filtering criteria are processed. However, there are other criteria that users might want to use to filter URL links. For example:

- Allow URLs with certain file name extensions
- Allow URLs only from a particular port number
- Disallow any PDF file if it is from a particular directory

The possible criteria could be very large, which is why it is delegated to a user-implemented module that can be used by the crawler when evaluating an extracted URL link.

URL Link Rewriting

For some applications, due to security reasons, the URL crawled is different from the one seen by the end user. For example, crawling is done on an internal Web site behind a firewall without security checking, but when queried by an end user, a corresponding mirror URL outside the firewall must be used.

A *display URL* is a URL string used for search result display. This is the URL used when users click the search result link. An *access URL* is a URL string used by the crawler for crawling and indexing. An access URL is optional. If it does not exist, then the crawler uses the display URL for crawling and indexing. If it does exist, then it is used by the crawler instead of the display URL for crawling.

For regular Web crawling, there are only display URLs available. But in some situations, the crawler needs an access URL for crawling the internal site while keeping a display URL for the external use. For every internal URL, there is an external mirrored one.

For example:

```
http://www.acme-qa.us.com:9393/index.html
http://www.acme.com/index.html
```

When the URL link `http://www.acme-qa.us.com:9393/index.html` is extracted and before it is inserted into the queue, the crawler generates a new display URL and a new access URL for it:

Access URL:

`http://www.acme-qa.us.com:9393/index.html`

Display URL:

`http://www.acme.com/index.html`

The extracted URL link is rewritten, and the crawler crawls the internal Web site without exposing it to the end user.

Another example is when the links that the crawler picks up are generated dynamically and can be different (depending on referencing page or other factor) even though they all point to the same page. For example:

```
http://compete3.acme.com/rt/rt.wwv_media.show?p_type=text&p_id=4424&p_
currcornerid=281&p_textid=4423&p_language=us

http://compete3.acme.com/rt/rt.wwv_media.show?p_type=text&p_id=4424&p_
currcornerid=498&p_textid=4423&p_language=us
```

Because the crawler detects different URLs with the same contents only when there is sufficient number of duplication, the URL queue could grow to a huge number of URLs, causing excessive URL link generation. In this situation, allow "normalization" of the extracted links so that URLs pointing to the same page have the same URL. The algorithm for rewriting these URLs is application dependent and cannot be handled by the crawler in a generic way.

When a URL link goes through a rewriter, there are the following possible outcomes:

- The link is inserted with no changes made to it.
- The link is discarded; it is not inserted.
- A new display URL is returned, replacing the URL link for insertion.
- A display URL and an access URL are returned. The display URL may or may not be identical to the URL link.

Creating and Using a URL Rewriter

Follow these steps to create and use a URL rewriter:

1. Create a new Java file implementing the UrlRewriter interface `open`, `close`, and `rewrite` methods. A rewriter, `SampleRewriter.java`, is available for reference under `$ORACLE_HOME/ultrasearch/extension/`.
2. Compile the rewriter Java file into a class file. For example:

   ```
   /jdk1.3.1/bin/javac -O -classpath $ORACLE_HOME/ultrasearch/lib/ultrasearch.jar
   SampleRewriter.java
   ```

3. Package the rewriter class file into a jar file under the `$ORACLE_HOME/ultrasearch/lib/agent/` directory. For example:

   ```
   /jdk1.3.1/bin/jar cv0f $ORACLE_HOME/ultrasearch/lib/agent/sample.jar
   SampleRewriter.class
   ```

4. Specify the rewriter class name and jar file name (for example, `SampleRewriter` and `sample.jar`) in the administration tool in step 2 of "Creating Web Sources" on page 8-18 or in the crawler parameters page of an existing Web data source.
5. Enable the UrlRewriter option from Web Sources page in the administration tool.
6. Crawl the target Web data source by launching the corresponding schedule. The crawler log file confirms the use of the URL rewriter with the message *Loading URL rewriter "SampleRewriter"...*

Note: URL rewriting is available for Web data sources only.

See Also:

- *Oracle Ultra Search API Reference* for the API (`oracle.ultrasearch.crawler` package)
- The URL rewriter `SampleRewriter.java` under `$ORACLE_HOME/ultrasearch/extension/`
- "Web Sources" on page 8-18

Oracle Ultra Search Document Service API

The Document Service crawler agent API allows generation of attribute data based on the document contents. It accepts robot metatag instructions from the agent for the target document, and it transforms the original document contents for indexing control.

The document service agent is a user-implemented Java module that implements the `DocumentService` Java interface such that it can interact with the crawler to provide more control on the crawled document. It is a Java interface in the form of a crawler agent that allows callout during crawling for user-defined document processing. It has the following features:

- Returns processed results, by user's choice, to the crawler as document attributes (for example, classification, theme, and gist) to be indexed and made searchable
- Allows robot metatag instruction from the agent to ignore the target document, not following link, and indexing it or not indexing it
- Allows transformation and filtering of the original document contents for indexing
- Allows assigning one document service agent for all data sources or for a particular data source
- Supports definition of document service agent

Each crawling thread has a copy of the service agent object. The document service agent jar file or class file must be under the `$ORACLE_HOME/ultrasearch/lib/agent/` directory.

APIs and Classes

The `DataSourceParams` interface and `AgentException` class are used by the interface introduced here. They are used by the Oracle Ultra Search user to implement their own service agent. The agent Java code should import the following classes:

```
import oracle.ultrasearch.crawler.UrlData;

import oracle.ultrasearch.crawler.DocAttributes;

import oracle.ultrasearch.crawler.DataSourceParams;

import oracle.ultrasearch.crawler.AgentException;
```

Interface DocumentService

Interface DocumentService processes documents for summarization, classification, or any transformation function that takes the document text as input and produces some kind of text output.

boolean open(DataSourceParams params, PrintWriter log) throws AgentException;

This is always the first method crawler called when the agent is loaded. It lets the agent perform initialization work.

The crawler passes the agent parameters through the `DataSourceParams` interface. The agent should verify the parameter and raise an agent fatal exception if any error is detected.

`log` is the crawler log file where the agent can output any information to it.

A document service session is established.

void close() throws AgentException;

This is always the last method called by the crawler. It lets the agent perform clean up work.

The document service session is terminated.

int doService(String documentUrl, number urlId, Reader docReader) throws AgentException;

Requesting service on the submitted document. This is invoked right before extraction of links and attributes.

This function returns a status code indicating what kind of result it has for this document. The possible codes are:

`NO_CHANGE`: No further information about this document

`FOLLOW_UP`: Call `getRobotsControl`, `getAttribute`, and `getContents` to retrieve the value

Any other status code value is treated as `NO_CHANGE`.

An agent exception is thrown if there is a problem processing the document. Fatal agent exceptions stop the crawler. Warning agent exceptions are treated as `NO_CHANGE` with the exception printed to the crawler log.

UrlData getAttribute(number urlId);

This is called only when `doService` returns `FOLLOW_UP` status code.

The agent returns a UrlData object that contains attribute data for this document. If there is no attribute to be added, then the agent can simply return null. The attribute will be automatically registered if it has not been registered.

int getRobotControl (number urlId);

This is called only when `doService` returns `FOLLOW_UP` status code.

This function returns a status code indicating what kind of robots control it has for this document. The possible codes are:

`USE_CURRENT`: Use existing setting

`FOLLOW_AND_INDEX`: Follow link, and index the document

`FOLLOW_AND_NO_INDEX`: Follow links, but do not index this document

`NO_FOLLOW_AND_INDEX`: No link extraction, but index the document

`NO_FOLLOW_AND_NO_INDEX`: No link extraction, and do not index

Reader getContents(number urlId);

This is called only when `doService` returns `FOLLOW_UP` status code.

The returned Reader object contains the new document contents in HTML to be indexed. The original document contents will be discarded.

The crawler closes the Reader when the crawler finishes reading it. The returned Reader should not be a filter Reader based on the original Reader passed in from `doService`, because the original Reader is closed when `getContents` return a new Reader.

If the return is null, then no contents replacement will happen.

Checksum of the original document is not changed.

void received(number urlId) throws AgentException;

This is called after finishing work with the target document. It allows the agent to perform clean up work associated with its service.

`doServie` call and `received` are always paired.

The crawler is guaranteed not to hold on to the UrlData object returned by the agent. This allows the agent to reuse the object.

Agent Registration Client Interface

Registration of a document service agent is through PL/SQL API from the `wkds_adm` package. You register a document service agent, define an agent instance, and assign it to one or all data sources in the form of crawler preference. The detail of the API can be referenced in `wk0ds.pkh` under `$ORACLE_HOME/ultrasearch/admin/`.

After a document service agent instance is created, use the following API to assign it to be loaded for all data source:

```
wk_crw.update_crawler_config(wk_crw.CRAWLER_COMMON,'CC_AGENT_INSTANCE','<agent
instance name>');
```

Use the following if it is only to be used for a particular data source:

```
wk_crw.update_crawler_config(<data source id>,'CC_AGENT_INSTANCE','<agent instance
name>');
```

To remove the agent instance, use the null value for the instance name:

```
wk_crw.update_crawler_config(wk_crw.CRAWLER_COMMON,'CC_AGENT_INSTANCE',null);
```

Note: These are internal APIs, which are subject to change.

Example of Setting Up the Document Service Agent

This example assumes that `DocServiceAgent.java` has been compiled into `DocServiceAgent.class` file and is archived in a jar file called `wkagent.jar`.

```
declare
    g_tid1 number;
    g_dsid0 number;
    g_pid0 number;
    g_pid1 number;
  begin
    -- All API calls must start with wk_adm.use_instance
    wk_adm.use_instance('<INSTANCE NAME>');
    -- register the agent
    -- agent class name is 'DocServiceAgent', located in wkagent.jar under
    -- $OH/ultrasearch/lib/agent/
    g_tid1 := WKDS_ADM.new_agent('Simple Agent','Document Service Test Agent',
              'DocServiceAgent','wkagent.jar',wkds_adm.DOC_SERVICE_TYPE);
    -- define agent parameters
    g_pid0 := wkds_adm.add_agent_param(g_tid1,'Admin_user','The user');
    g_pid1 := wkds_adm.add_agent_param(g_tid1,'Password','password','Y'); --
encrypted
    -- define an agent instance based on the registered agent
    g_dsid0 := WKDS_ADM.new_agent_inst('Simple Agent Instance',g_tid1);
    -- set agent parameter value for instance 'Simple Agent Instance'
    wkds_adm.set_agent_param_value(g_dsid0,g_pid0,'WK_TEST'); -- user name
    wkds_adm.set_agent_param_value(g_dsid0,g_pid1,'WK_TEST'); -- password
    -- Associate the agent instance with all data sources
    wk_crw.update_crawler_config(wk_crw.CRAWLER_COMMON,'CC_AGENT_INSTANCE','Simple
Agent Instance')
;

    -- Or associate it with one particular data source:
    -- wk_crw.update_crawler_config(<Data Source id>,'CC_AGENT_INSTANCE','Simple
Agent Instance');
  exception when others then wk_err.raise;
  end;
  /
```

Oracle Ultra Search Query Applications

Oracle Ultra Search provides several query applications and a sample crawler agent. Use the query applications as examples for creating your own query application. The query applications are written as J2EE-compliant Web applications. Your query application uses the Oracle Ultra Search query API. You can also use the sample crawler agent to create your own crawler agent.

> **Note:** Pointers to the query applications and the sample crawler agent Java source code, as well as their corresponding readmes, are in the Oracle Ultra Search welcome page: `http://hostname.domainname:port/ultrasearch/index.html`

The query application has been designed to showcase keyword in context and highlighting features. These changes are made to `search.jsp` and its dependent files.

Keyword in context shows a section of the original document that contains the search terms. Highlighting shows the entire document with the search terms in a different color. For highlighting to work, the crawler must be configured to keep cache file as one of its settings. Highlighting is implemented in `cache.jsp` and can be customized by the customer.

Note that framed HTML documents contain only the frame layout and frame content specification, not the actual content. Therefore, the cached version of these documents appears blank in a browser.

The query applications are shipped as a deployed J2EE Web application (`sample.ear`). This component depends on a J2EE container to host the Web pages, a JDBC driver, and Java Mail API for displaying e-mail results. After the `sample.ear` file is deployed by the Oracle Containers for J2EE (OC4J), you see a set of JSP files that demonstrate the query API usage.

The query applications include a search portlet. The Oracle Ultra Search portlet demonstrates how to write a search portlet for use in Oracle Application Server Portal.

When the user issues a query in any of the query applications, a result list containing query results is returned. The user can select a document to view from the result list. A result list can include HTML documents, files, database table content, archived e-mails, or Oracle Application Server items. The Oracle Ultra Search query applications also incorporate an e-mail browser for reading and browsing e-mails.

The Oracle Ultra Search administration tool and the Oracle Ultra Search query applications are part of the Oracle Ultra Search middle tier. However, the Oracle Ultra Search administration tool is independent from the Oracle Ultra Search query applications. Therefore, they can be hosted on different computers to enhance security or scalability.

If you do not want to use the query applications, you can build your own query application by directly invoking the Oracle Ultra Search Java query API. Because the API is coded in Java, you can invoke the API methods from any Java-based application, such as from a Java servlet or a JavaServer Page (as in the case of the provided query applications). For rendering e-mails that have been crawled and indexed, you can also directly invoke the Oracle Ultra Search Java e-mail API methods.

Query Applications

The query applications are located in the `$ORACLE_HOME/ultrasearch/sample` directory.

JavaServer Page Concepts

As mentioned earlier, you can use JSP code and the supplied Java APIs to create your Web application. Typically, your Web application runs in an application server, such as Oracle Application Server. The application server typically runs on a separate computer from the Oracle server for performance and scalability reasons. The Oracle server holds the Oracle Ultra Search indexes.

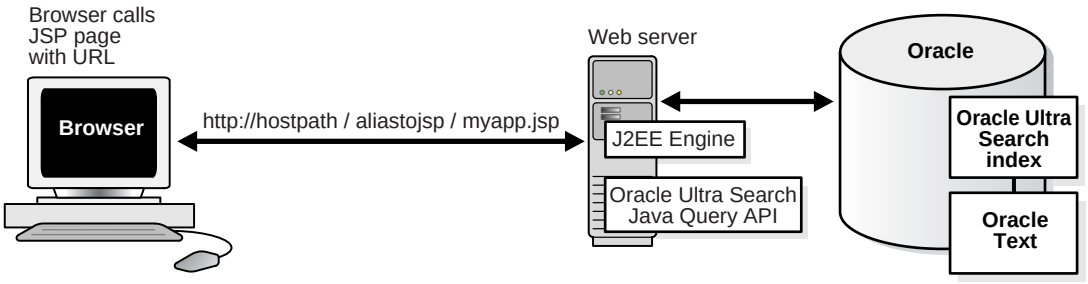
JSP applications are compiled into Java servlets at runtime. The compiled servlets run in one or more Java Virtual Machine processes. The JSP application communicates with the Oracle server through the Oracle JDBC driver.

As in any Java application, you must include the following files in your servlet engine classpath to use the Java query and e-mail APIs:

1. $ORACLE_HOME/ultrasearch/lib/ultrasearch_query.jar
2. $ORACLE_HOME/lib/mail.jar
3. $ORACLE_HOME/lib/activation.jar

Figure 9–1 shows how your Web query application calls the Oracle Ultra Search Java query API.

Figure 9–1 Calling JavaServer Pages



10

Tuning and Performance in Oracle Ultra Search

This chapter contains the following sections:

Tuning the Web Crawling Process

The Oracle Ultra Search crawler is a powerful tool for discovering information on Web sites in an organization's intranet. This feature is especially relevant to Web crawling. The other data sources (for example, table or e-mail data sources) are defined such that the crawler does not follow any links to other documents that you might not be aware of.

Web Crawling Strategy

Your Web crawling strategy can be as simple as identifying a few well-known sites that are likely to contain links to most of the other intranet sites in your organization. You could test this by crawling these sites without indexing them. After the initial crawl, you have a good idea of the hosts that exist in your intranet. You could then define separate Web sources to facilitate crawling and indexing on individual sites.

However, in reality, the process of discovering and crawling your organization's intranet is an interactive one characterized by periodic analysis of crawling results and modification to crawling parameters to direct the crawling process somewhat. For example, if you observe that the crawler is spending days crawling one Web host, then you might want to exclude crawling at that host or limit the crawling depth.

Monitoring the Crawling Process

Monitor the crawling process by using a combination of the following methods:

- Monitoring the schedule status with the administration tool
- Monitoring the real time schedule progress with the administration tool
- Monitoring the crawler statistics with the administration tool
- Monitoring the log file for the current schedule

URL Looping

URL looping refers to the scenario where, for some reason, a large number of unique URLs all point to the same document. One particularly difficult situation is where a site contains a large number of pages, and each page contains links to every other page in the site. Ordinarily, this would not be a problem, because the crawler eventually analyzes all documents in the site.

However, some Web servers attach parameters to generated URLs to track information across requests. Such Web servers might generate a large number of unique URLs that all point to the same document.

For example, `http://mycompany.com/somedocument.html?p_origin_page=10` might refer to the same document as `http://mycompany.com/somedocument.html?p_origin_page=13` but the `p_origin_page` parameter is different for each link, because the referring pages are different. If a large number of parameters are specified and if the number of referring links is large, then a single unique document could have thousands or tens of thousands of links referring to it. This is an example of how URL looping can occur.

Monitor the crawler statistics in the Oracle Ultra Search administration tool to determine which URLs and Web servers are being crawled the most. If you observe an inordinately large number of URL accesses to a particular site or URL, then you might want to do one of the following:

- Exclude the Web Server: This prevents the crawler from crawling any URLs at that host. (You cannot limit the exclusion to a specific port on a host.)
- Reduce the Crawling Depth: This limits the number of levels of referred links the crawler will follow. If you are observing URL looping effects on a particular host, then you should take a visual survey of the site to find out an estimate of the depth of the leaf pages at that site. Leaf pages are pages that do not have any links to other pages. As a general guideline, add three to the leaf page depth, and set the crawling depth to this value.

Be sure to restart the crawler after altering any parameters in the Crawler Page. Your changes take effect only after restarting the crawler.

Tuning Query Performance

This section contains suggestions on how to improve the performance of the Oracle Ultra Search query. Query performance is generally affected by response time and throughput.

- Tune the `DB_CACHE_SIZE` initialization parameter.

  The database buffer cache keeps frequently accessed data read from datafiles. Efficient usage of the buffer cache can improve Oracle Ultra Search query performance. The cache size is controlled by the `DB_CACHE_SIZE` initialization parameter.

  > **See Also:** *Oracle Database Performance Tuning Guide* for information on how to tune this parameter

- Optimize the index.

  Optimize the Oracle Ultra Search index after the crawler has made substantial updates. To do so, schedule index optimization on a regular basis. Make sure index optimization is scheduled during off-peak hours, because query performance is significantly degraded during index optimization.

See Also: "Index Optimization" on page 8-30

- Optimize the index based on tokens.

  Optimize the Oracle Ultra Search index by basing it on frequently searched tokens. To log queries, use the administration tool to turn on query statistics collection. The frequently searched tokens then can be passed to `CTX_DDL.OPTIMIZE_INDEX` in token mode. The Oracle Ultra Search index name is `WK$DOC_PATH_IDX`.

  See Also: *Oracle Text Reference* for more information on `OPTIMIZE_INDEX`

- Simplify query expansion.

  The search response time is directly influenced by the Oracle Text query string used. Although Oracle Ultra Search provides a default mechanism to expand user input into a Text query, simpler expansions can greatly reduce search time.

  See Also:

  - "Customizing the Query Syntax Expansion" on page 9-3
  - *Oracle Ultra Search API Reference* for the `oracle.ultrasearch.query.Query` interface

- Size the shared pool.

  The shared pool stores the library cache and the dictionary cache. The library cache stores recently run SQL and PL/SQL code. A cache miss on the data dictionary cache or library cache is more expensive than a miss on the buffer cache. For this reason, the shared pool should be sized to ensure that frequently used data is cached. The shared pool size is controlled by the `SHARED_POOL_SIZE` initialization parameter.

  See Also: *Oracle Database Performance Tuning Guide* for information on tuning this parameter

- Define JDBC connection pooling.

  The Oracle Ultra Search middle tier connects to the database through JDBC. Because creation of a connection is an expensive operation in JDBC, a pool of open connections is used to improve the response time of queries. With Oracle Application Server, OC4J can manage the connection pool for the applications.

  The minimum size, maximum size, and allocation algorithm of the pool can be specified in the `data-sources.xml` configuration file of OC4J.

  The following is an example of a data source definition, with minimum 2 and maximum 30 open-connections. Each connection closes after 30 seconds of inactivity, and new connections are created dynamically according to load. The other caching schemes are `FIXED_WAIT_SCHEME` and `FIXED_RETURN_NULL_SCHEME`.

  Note: `DYNAMIC_SCHEME` = 1, `FIXED_WAIT_SCHEME` = 2, and `FIXED_RETURN_NULL_SCHEME` = 3

```
<data-source
        class="oracle.jdbc.pool.OracleConnectionCacheImpl"
        name="UltraSearchDS"
        location="jdbc/UltraSearchPooledDS"
        username="user"
        password="pass"
        url="jdbc:oracle:thin:@hostname:1521:oracle_sid"
        min-connections="2"
        max-connections="30"
        inactivity-timeout="30" >
       <property name="cacheScheme" value="1" />
 </data-source>
```

- Pin the query package in memory.

  Pin frequently used packages in the shared memory pool. When a package is pinned, it remains in memory, no matter how full the pool gets or how frequently you access the package. You can pin packages using the supplied package DBMS_SHARED_POOL.

  The PL/SQL package used for Oracle Ultra Search query is WKSYS.WK_QRY.

  See Also: *PL/SQL Packages and Types Reference*

Using the Remote Crawler

Without the Oracle Ultra Search remote crawler, you must run the Oracle Ultra Search crawler on the same host as the Oracle Database. For large data sets, you can improve performance by running the Oracle Ultra Search crawler on one or more separate hosts from the Oracle Database. The Oracle Ultra Search crawler is a pure Java application, and it communicates with the Oracle Database through JDBC. Because the Oracle Ultra Search scheduling mechanism runs within the Oracle Database, it automatically uses the database's high availability features.

The Oracle Database uses one of two mechanisms to send launch requests to the remote crawler hosts. The first is Java remote method invocation (RMI). The second is Java database connectivity (JDBC). Both mechanisms establish a launching sub-system on the remote host. You can conceptualize the launching sub-system as a process that uses either RMI or JDBC to listen for launch requests. (This chapter refers to the launching sub-system as the "launcher").

Upon receipt of a launch request, the launcher spawns a new Java process. It is this process that is the actual remote crawler.

You should use JDBC-based remote crawling if you do not want the dependency on RMI (for example, because of network restrictions).

Understanding the Launcher

The launcher is the sub-system that listens for launch requests and launches remote crawler processes. When you register a remote crawler (either RMI-based or JDBC-based), you are actually registering the launcher with the Oracle Ultra Search backend.

By registering a launcher, you make it available to be used by all Oracle Ultra Search instances within an Oracle Ultra Search installation. Thus, after registration, an administrator or an Oracle Ultra Search instance may subsequently choose to associate the launcher and assign schedules to be launched with that launcher.

There is no way to restrict a launcher to specific Oracle Ultra Search instances. Once registered, all Oracle Ultra Search instances may use it. However, the launcher is only a sub-system for launching remote crawler processes. Having multiple instances use the same launcher for launching purposes poses no security problems for most customers.

Both RMI and JDBC launchers are simply Java processes themselves. They are started from the command line. Oracle provides scripts for starting these launchers, described in the following section.

Also, the JDBC launcher must establish JDBC connections to the Oracle Ultra Search backend database to listen for launch events. You must specify the launch user (or role) at registration time. Oracle strongly recommends that you create a new database user (or role) specifically for the purposes of launching remote crawlers. You should not use this user (or role) for any other purposes.

RMI-Based Remote Crawling

RMI-based remote crawling depends on the standard RMI infrastructure. Therefore, each remote crawler host must have an RMI registry and an RMI daemon running. These are started when you run the scripts to start the RMI-based launcher.

- When a crawling schedule is activated, the Oracle Ultra Search scheduler launches a Java program as a separate process on the database host. This Java program is known as the ActivationClient.
- This program attempts to connect to the remote crawler host through the RMI registry port specified at installation time. If successful, then the ActivationClient receives a remote reference to a Java object running on the remote host. This remote Java object is known as the ActivatableCrawlerLauncher.
- The ActivationClient then instructs the ActivatableCrawlerLauncher to launch the Oracle Ultra Search crawler on the remote host. The ActivatableCrawlerLauncher launches the Oracle Ultra Search crawler as a separate Java process on the remote host.

The RMI registry and daemon ports are inflexible. Therefore, if you have other RMI services running on the same host, you will not be able to use RMI-based remote crawling. Also, you cannot run two RMI-based launchers, because they will both conflict on the RMI ports.

JDBC-Based Remote Crawling

JDBC-based remote crawling requires that the launcher be up and running.

- When a crawling schedule is activated, the Oracle Ultra Search scheduler sends a message to the launcher.
- If the launcher is running and properly connected to the database as the appropriate launch user (or role), then it can receive the launch messages. Otherwise, the message times out after 30 seconds and launch failure is reported.
- The launcher then deciphers the launch message and spawns an Oracle Ultra Search crawler as a separate Java process on the remote host.

The launcher maintains two permanent JDBC connections to the backend database. If either connection goes down at any time, then the JDBC launcher attempts to reestablish it. The number of attempts to reestablish connections is configurable as a command line parameter. The wait time between attempts is also configurable.

Note: The JDBC launcher can be configured to periodically trigger a "keep-alive" signal. This is useful to prevent inadvertent closing of the JDBC connections by firewalls. The time between signals is configurable with a command line parameter.

Security With Remote Crawlers

When launching a remote crawler, the Oracle Ultra Search backend database communicates with the remote computer through Java remote method invocation (RMI) or JDBC.

Oracle Ultra Search encrypts all RMI communication. However, the JDBC launcher uses the Oracle Thin JDBC driver. If security is a concern, then encrypt all JDBC traffic by securing the Oracle Thin JDBC driver.

See Also: *Oracle Advanced Security Administrator's Guide* for more information on Thin JDBC support

Scalability and Load Balancing

Each Oracle Ultra Search schedule can be associated with exactly one crawler. The crawler can run locally on the Oracle database host or on a remote host. There is no limit to the number of schedules that can be run. Similarly, there is no limit to the number of remote crawler hosts that can be run. However, each remote crawler host requires that the Oracle Ultra Search middle tier be installed on its host.

By using several remote crawler hosts and carefully allocating schedules to specific hosts, you can achieve scalability and load balancing of the entire crawling process.

Installation and Configuration Sequence

1. Make sure that you have installed the Oracle Ultra Search Backend server component as well as a Server component on each host that is to be used to run remote crawlers.

   See Also: Chapter 3, "Installing Oracle Ultra Search"

2. Understand the cache and mail archive directories.

   All remote crawlers must cache crawled data into a common file system location, that is, accessible by the backend Oracle Ultra Search database. Likewise, when crawling e-mail sources, all e-mails must be saved in a common, central location. The simplest way to achieve this is by ensuring that the cache and mail archive directories seen by the remote crawler uses are mounted through NFS to point to the cache and mail directories used by the Oracle Ultra Search backend database.

   For example, your Oracle Ultra Search installation might consist of four hosts: one database server (host X) running Solaris on which the Oracle Ultra Search backend is installed; one remote crawler host (host Y1) running on Windows; one remote crawler host (host Y2) running on Solaris; and one remote crawler host (host Y3) running on Linux.

   In this scenario, export the shared directories on host X using the UNIX `export` command. Then use the UNIX `mount` command on hosts Y2 and Y3 to mount the exported directories. For host Y1, you must purchase a third-party NFS client for Windows and use that to mount the shared directories. If host X is a Linux server,

you can create Samba shares and thereby mount those shares on Windows without needing any third party software.

If for some reason there is no shared file system between the database and remote crawler hosts, you can instruct the remote crawler to transfer all cache and mail archive data across JDBC to the database host. The files are then saved locally on the database host. You can choose this option by selecting through JDBC connection for the Cache file access mode setting in the next step.

3. Configure the remote crawler with the administration tool.

   To edit the remote crawler profile, navigate to the **Crawler**: **Remote Crawler Profiles** page and click **Edit** for the remote crawler profile you want to edit. Edit that profile by manually entering all mount points for the shared crawler resources that you defined.

   Cache and mail archive directories. If the backend database host and remote crawler host are on a shared file system (such as NFS), select "through mounted file system" for the Cache file access mode setting. Then specify values for the following parameters:

   - Mount point for cache directory path as seen by the remote crawler
   - Mount point for mail archive path as seen by the remote crawler (if you are using the Oracle Ultra Search mailing list feature)

   Otherwise, if there is no shared file system between the remote crawler host and the backend database host, you must select through JDBC connection for the Cache file access mode setting. Then, specify values for the following parameters:

   - Local cache directory as seen by local crawlers on the backend database host
   - Local mail archive directory as seen by local crawlers on the backend database host

   Crawler log directory. It is not necessary that the remote crawler log directory be an NFS mount a central location accessible by the backend Oracle Ultra Search database. However, it is beneficial to do so if you want to be able to monitor all crawler logs (local as well as all remote crawlers) in one central location.

   Additionally, you must specify the following crawler parameters before you can begin crawling:

   - number of crawler threads that the remote crawler uses for gathering documents
   - number of processors on the remote crawler host
   - initial Java heap size
   - maximum Java heap size

   Java classpath. It is not usually necessary to specify this classpath. The classpath that remote crawler processes use is inherited from the RMI subsystem. The RMI subsystem classpath is configured by the scripts used to launch it. You will need to modify the classpath only in special circumstances where you require the classpath to be different from the RMI subsystem classpath.

4. Complete the crawler configuration with the administration tool.

   Create schedules and data sources. Assign one or more data sources to each schedule.

   Each schedule must then be assigned to a remote crawler or the local crawler. (The local crawler is the crawler that runs on the local Oracle database host itself). To

assign the a schedule to a remote crawler host or the local database host, click the host name of a schedule in the Schedules page.

You can also turn off the remote crawler feature for each schedule, thereby forcing the schedule to launch a crawler on the local database host, instead of the specified remote crawler host. To turn off the remote crawler feature, click the host name of a schedule in the **Synchronization Schedules** page. If a remote crawler host is selected, the RMI-based remote crawler hostname (or JDBC-Based launcher name) will be displayed. Change this to the local database host in order to disable remote crawling.

> **See Also:** Chapter 8, "Understanding the Oracle Ultra Search Administration Tool"

5. Start the remote crawler launching sub-system on each remote crawler host.

   Use the helper scripts in `$ORACLE_HOME/tools/remotecrawler/scripts/operating_system` to do this.

   - If the remote crawler is running on a UNIX platform, then source the `$ORACLE_HOME/tools/remotecrawler/scripts/unix/runall.sh` Bourne shell script for RMI-based remote crawling. Source `runall_jdbc.sh` for JDBC-based remote crawling.
   - If the remote crawler is running on a Windows host, then run the `%ORACLE_HOME%\tools\remotecrawler\scripts\winnt\runall.bat` file for RMI-based remote crawling. Run`runall_jdbc.bat` for JDBC-based remote crawling.

   For RMI-based remote crawling, the `runall` scripts perform the following tasks in sequence:

   1. `define_env` is invoked to define necessary environment variables.
   2. `runregistry` is invoked to start up the RMI registry.
   3. `runrmid` is invoked to start up the RMI daemon.
   4. `register_stub` is invoked to register the necessary Java classes with the RMI subsystem.

   > **Note:** You can invoke `runregistry`, `runrmid`, and `register_stub` individually. However, you must first invoke `define_env` to define the necessary environment variables.

   For JDBC-based remote crawling, the `runall_jdbc` scripts perform the following tasks in sequence:

   1. `define_env` is invoked to define necessary environment variables
   2. The JDBC launcher is started with a command line Java process invocation. There are the following command line arguments for the JDBC launcher:

   - `-name`: name of launcher (that you used to register this launcher)
   - `-url`: JDBC connection URL to the backend Oracle Ultra Search database
   - `-user`: database user to connect
   - `-password`: database user password
   - `-rw`: wait time (in seconds) between attempts to reestablish JDBC connections

- `-ra`: maximum number of attempts to reestablish JDBC connections
- `-kw`: wait time (in milliseconds) between keep-alive signals

You must edit the contents of the `runall_jdbc` script and specify the values for each parameter before running it.

6. Launch the remote crawler from the administration tool, and verify that it is running.

   The state of the schedule is listed in the **Schedules** page. The remote crawler launching process takes up to 90 seconds to change state from LAUNCHING to FAILED if failure occurs.

   To view the schedule status, click the crawler status in the schedules list. To view more details, especially in the event of failure, click the schedule status itself. This brings up a detailed schedule status.

   The RMI-based remote crawler fails to launch if any one of the following requirements are not met:

   - The RMI registry is not running and listening on the port specified at installation.
   - The RMI daemon is not running and listening on the port specified at installation.
   - The necessary Java objects have not been successfully registered with each RMI registry.

   The JDBC-based remote crawler fails to launch if any one of the following requirements are not met:

   - The JDBC launcher is not running.
   - The JDBC launcher is running, but the connect user (or role) specified is incorrect.

   After a remote crawler is launched, verify that it is running with one or more of the following methods:

   - For RMI-based crawling, check for active Java processes on the remote crawler host. A simple way to confirm that remote crawler is running on the remote crawler host is to use an operating system command, such as `ps` on UNIX systems. Look for active Java processes.
   - For JDBC-based crawling, check that the launcher is up and running and that there are no errors. When you start the JDBC-based launcher, it will output text to standard output. You may optionally redirect output to a file. Monitor this output for any errors.
   - Monitor the contents of the schedule log file. If the remote crawler is running successfully, then you should see the contents of the schedule log file changing periodically. The schedule log file is located in the shared log directory.

Oracle Ultra Search on Real Application Clusters

Oracle Ultra Search can crawl on one fixed node or on any node, depending on the storage access configuration of the Real Application Clusters system. PL/SQL APIs are provided to specify which node should run the crawler, if needed. For Oracle Ultra Search administration and the Oracle Ultra Search query application, you can configure the connection string to connect to any node of Real Application Clusters.

See Also: The documentation for Oracle Database Real Application Clusters

Configuring Storage Access

The disk of any node in a Real Application Clusters system can be shared (cluster file system) or not shared (raw disk). For Real Application Clusters on a cluster file system (CFS), the cache files generated by the crawler on any node are visible to any Oracle instance and can be indexed by any Oracle instance that performs index synchronization. If the disk is not shared, then the crawler must run on one particular Oracle instance to ensure that all cache files can be indexed.

This is due to the nature of Oracle Text indexing, where rows inserted into one table by different sessions go to the same pending queue, and whoever initiates index synchronization attempts to index all of the inserted rows. Because of this limitation, on a CFS, Oracle Ultra Search is configured to launch the crawler on any database instance. If it is not on a CFS, then Oracle Ultra Search launches the crawler on the database instance where `INSTANCE_NUMBER` = 1.

The Oracle Ultra Search administrator can configure which instance runs the crawler with the following PL/SQL API:

```
WK_ADM.SET_LAUNCH_INSTANCE(instance_name, connect_url);
```

where `instance_name` is the name of the launching instance (or the database name if it is to be launched on any node) and `connect_url` is the connect descriptor.

For connection to a single database instance, the descriptor can be in the short form "*host*:*port*:*SID*" or the connect descriptor (Oracle Net keyword-value pair). For example:

```
(DESCRIPTION=(ADDRESS_
LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=cls02a)(PORT=3999)))(CONNECT_DATA=(
  SERVICE_NAME=acme.us.com)))
```

To connect to any database instance, the full database connect descriptor must be used. For example:

```
(DESCRIPTION=(LOAD_BALANCE=yes)(ADDRESS_
LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=cls02a)(PORT=3999
  ))(ADDRESS=(PROTOCOL=TCP)(HOST=cls02b)(PORT=3999)))(CONNECT_DATA=(SERVICE_
NAME=acme.us.com)))
```

See Also: *Oracle Database JDBC Developer's Guide and Reference* for configuration details.

You cannot configure Oracle Ultra Search to launch the crawler on any node on a non-cluster file system.

To query on the existing launching instance configuration, use the following PL/SQL API:

```
WK_ADM.GET_LAUNCH_INSTANCE RETURN VARCHAR2;
```

This returns the name of the launching instance or the database name if any node can launch the crawler.

Remote Crawler File Cache

The Oracle Ultra Search remote crawler requires that the remote file system be mounted on the Oracle instance for indexing.

For cluster file system Real Application Clusters, the file system of the remote computer should be NFS mounted to all nodes of the system.

For non-cluster file system Real Application Clusters, the NFS mount can be limited to the specific node where the Oracle instance is serving the remote crawler. There is no advantage to mounting the remote file system to all nodes--it could lead to stale NFS handles when nodes go down. When there is a configuration change to move to a different Oracle instance, the remote file system should be NFS mounted to the new node accordingly.

Logging on to the Oracle Instance

All components of Oracle Ultra Search use the JDBC Thin Driver with the connect string consisting of "*hostname*:*port*:*SID*" or the full connect descriptor as seen in `tnsnames.ora`.

The administration middle tier connects to the Oracle database with a JDBC connection specified in the ultrasearch.properties file. If the client serving node is down, then you must manually edit the ultrasearch.properties file to connect to a different Oracle instance.

Query Search Application for Read Application Clusters

Query components should fully utilize Real Application Clusters. You can specify the JDBC connection string as a database connect descriptor so that it can connect to any Oracle instance in Real Application Clusters. For example:

```
"jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=yes)(ADDRESS_
LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=cls02a)(PORT=3999
  ))(ADDRESS=(PROTOCOL=TCP)(HOST=cls02b)(PORT=3999)))(CONNECT_DATA=(SERVICE_
NAME=acme.us.com)))"
```

See Also: *Oracle Database JDBC Developer's Guide and Reference*

Java Crawler

The connect string used by Oracle Ultra Search crawler is initialized during installation and can be changed with the `WK_ADM.SET_LAUNCH_INSTANCE` API. When there is a system configuration change, such as adding or dropping a node, the connect string is changed automatically.

Choosing a JDBC Driver

The Oracle Ultra Search administrator optionally can configure the local crawler to use the JDBC OCI driver to log on to the database. This is done with the following PL/SQL API:

```
WK_ADM.SET_JDBC_DRIVER(driver_type)
```

Where

- Thin driver (default) `driver_type` = 0
- OCI driver `driver_type` = 1

This API requires super-user privileges. The change affects all Oracle Ultra Search instances.

Note: The OCI driver requires that environment variables, such as `LD_LIBRARY_PATH` and `NLS_LANG`, be set properly on the launching database instance. The crawler inherits the environment setting from the Oracle process. Therefore, you must configure them appropriately before starting Oracle.

See Also: *Oracle Database JDBC Developer's Guide and Reference* for configuration details on using the OCI driver.

The following PL/SQL API determines which kind of JDBC drivers are used currently:

```
WK_ADM.GET_JDBC_DRIVER RETURN NUMBER;
```

Oracle Ultra Search Failover in a RAC Environment

When RAC uses the cluster file system (CFS), the Oracle Ultra Search crawler can be launched from any of the RAC nodes, as long as at least one RAC node is up and running.

When RAC is not using CFS, the Oracle Ultra Search crawler always runs on a specified node. If this node stops operating, then you must run the `wk0reconfig.sql` script to move Oracle Ultra Search to another RAC node.

```
sqlplus wksys/wksys_passwd
ORACLE_HOME/ultrasearch/admin/wk0reconfig.sql
instance_name connect_url
```

where:

`instance_name` is the name of the RAC instance that Oracle Ultra Search uses for crawling. After connecting to the database, simply `SELECT instance_name FROM v$instance` to get the name of the current instance.

`connect_url` is the JDBC connection string that guarantees a connection only to the specified instance. For example:

```
"(DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=TCP)
              (HOST=<nodename>)
              (PORT=<listener_port>)))
   (CONNECT_DATA=(SERVICE_NAME=<service_name>)))"
```

When preserving the crawler cache, if Oracle Ultra Search is switched from one RAC node to another, you loose the contents of the cache. Force a re-crawl of the documents after switching instances.

Table Data Source Synchronization

Oracle Ultra Search crawls database tables in the local Oracle Database instance where Oracle Ultra Search is installed. Additionally, it can crawl remote databases if they have been linked to the main Oracle Database. Remote databases are linked to the main Oracle instance with database links.

See Also: *Oracle Database Administrator's Guide* for instructions on how to create database links

Oracle Ultra Search provides a logging mechanism to optimize crawling of table sources. Using this logging mechanism, only newly updated documents are revisited during the crawling process. If the source database is not an Oracle database, then you must perform a sequence of steps to use this feature.

Synchronizing Crawling of Oracle Databases

Before creating log tables and log triggers, make sure that the Oracle Ultra Search instance schema has the CREATE ANY TABLE and CREATE ANY TRIGGER system privileges. For tables in Oracle databases, data definition language (DDL) statements are provided to create the following:

Create Log Table

The log table stores changes that have occurred in the base table. The Oracle Ultra Search crawler uses the change information to figure out which rows need to be recrawled. For example, a log table generated by Oracle Ultra Search could be named WK$LOG.

The structure of the log table conforms to the following rules:

1. For every primary key column of the base table, a column must be created in the log table.
2. There can be up to only eight primary key columns in the base table.
3. Each column in the log table that corresponds to a primary key column must be named Kx, where x is a number from one to eight.
4. Each column in the log table that corresponds to a primary key column must be of type VARCHAR2(1000).
5. There must be exactly one column named mark that has type CHAR(1).
6. The column named mark must have a default value F.

For example, the base table employees has the following structure:

Column Name	Column Type
ID	NUMBER
NAME	VARCHAR2(200)
ADDRESS	VARCHAR2(400)
TELEPHONE	VARCHAR2(10)
USERNAME	VARCHAR2(24)

If the primary key of the employees table comprises of the ID and NAME columns, then a log table WK$LOG (whose name is generated automatically) is created with the following structure:

Column Name	Column Type
K1	NUMBER
K2	VARCHAR2(200)

The SQL statement for creating the log table is as follows:

```
CREATE TABLE WK$LOG(
K1 VARCHAR2(1000),
K2 VARCHAR2(1000),
MARK CHAR(1) default 'F');
```

Create Log Triggers

An INSERT trigger, UPDATE trigger, and DELETE trigger are created. The Oracle trigger definitions are as follows:

INSERT Trigger Statement Every time a row is inserted into the employees base table, the INSERT trigger inserts a row into the log table. The row in the log table records the new values of the id and the name into the k1 and k2 columns. An F is inserted into the mark column to signal the crawler that work needs to be done for this row.

For example:

```
CREATE TABLE employees (id NUMBER, name VARCHAR2(10));

CREATE OR REPLACE TRIGGER wk$ins
AFTER INSERT ON employees
FOR EACH ROW

BEGIN
  INSERT INTO WK$LOG(k1,k2,mark)
    VALUES(:new.id,:new.name,'F');
END;
/
```

UPDATE Trigger Statement Every time a row is updated in the employees base table, the UPDATE trigger inserts two rows into the log table. The first row in the log table records the old values of the id and the name into the k1 and k2 columns. An F is inserted into the mark column to signal the crawler that work needs to be done for this row. The second row in the log table records the new values of the id and the name into the k1 and k2 columns.

For example:

```
CREATE OR REPLACE TRIGGER wk$upd
AFTER UPDATE ON employees
FOR EACH ROW

BEGIN
  INSERT INTO WK$LOG(k1,k2,mark)
    VALUES(:old.id,:old.name,'F');
  INSERT INTO WK$LOG(k1,k2,mark)
    VALUES(:new.id,:new.name,'F');
END;
/
```

DELETE Trigger Every time a row is deleted from the employees base table, the DELETE trigger inserts a row into the log table. The row in the log table records the old values of the id and the name into the k1 and k2 columns. An F is inserted into the mark column to signal the crawler that work needs to be done for this row.

For example:

```
CREATE OR REPLACE TRIGGER wk$del
AFTER DELETE ON employees
FOR EACH ROW

BEGIN
  INSERT INTO WK$LOG(k1,k2,mark)
    VALUES(:old.id,:old.name,'F');
END;
/
```

Synchronizing Crawling of Non-Oracle Databases

For tables in non-Oracle remote databases, you must perform the following steps:

1. Manually create the log table. The log table must conform to the rules for log tables described earlier. Also, it must reside in the same schema and database instance as the base table.
2. Create three triggers that record inserts, updates, and deletes on the base table. These triggers must exhibit the same behavior as the triggers described earlier for Oracle tables.
3. Associate the log table. When you have completed these tasks, choose the "Enable logging mechanism (non-Oracle tables)" option during the creation of an Oracle Ultra Search table data source. By choosing that option, the Oracle Ultra Search administration tool prompts you for the name of the log table in the remote database. Oracle Ultra Search associates this log table with the base table. Oracle Ultra Search assumes that you have correctly performed steps 1 and 2.

11

Oracle Ultra Search Administration PL/SQL APIs

This chapter provides reference information on PL/SQL APIs available for use with Oracle Ultra Search. The APIs are grouped as follows:

- Instance-Related APIs
- Schedule-Related APIs
- Crawler Configuration APIs

The following tables show the contents of each API group.

- Table 11–1 shows the Instance-Related APIs.

Table 11–1 Instance-Related APIs

Name	Function
CREATE_INSTANCE	create an Oracle Ultra Search instance
DROP_INSTANCE	drop an Oracle Ultra Search instance
GRANT_ADMIN	grant instance administrator privileges
REVOKE_ADMIN	revoke instance administrator privileges
SET_INSTANCE	operate on an Oracle Ultra Search instance

- Table 11–2 shows the Schedule-Related APIs.

Table 11–2 Schedule-Related APIs

Name	Function
CREATE_SCHEDULE	create a crawler schedule
DROP_SCHEDULE	drop a crawler schedule
INTERVAL	generate a schedule interval string
SET_SCHEDULE	run, resume, or stop a schedule
UPDATE_SCHEDULE	update a crawler schedule

- Table 11–3 shows the Crawler Configuration APIs.

Table 11–3 Crawler Configuration APIs

Name	Function
IS_ADMIN_READONLY	check if a crawler configuration setting is read-only
SET_ADMIN_READONLY	make a read-only crawler configuration
UPDATE_CRAWLER_CONFIG	update crawler configurations

Instance-Related APIs

This section provides reference information for using the instance-related APIs.

CREATE_INSTANCE

Use this procedure to create an Oracle Ultra Search instance.

Syntax

```
OUS_ADM.CREATE_INSTANCE(
  inst_name     IN VARCHAR2,
  schema_name   IN VARCHAR2,
  password      IN VARCHAR2 DEFAULT NULL,
  lexer         IN VARCHAR2 DEFAULT NULL,
  stop_list     IN VARCHAR2 DEFAULT NULL,
  data_store    IN VARCHAR2 DEFAULT NULL,
  snapshot      IN NUMBER DEFAULT ous_adm.NO,
);
```

inst_name
The name of the instance.

schema_name
The name of the schema.

password
The password for the schema.

lexer
The Oracle Text index lexer preference.

stop-list
The Oracle Text index stoplist preference.

data_store
The Oracle Text index datastore preference.

snapshot
If OUS_ADM is set to YES, create an instance for the snapshot.

Example

```
OUS_ADM.CREATE_INSTANCE('Scott instance','scott','tiger');
```

DROP_INSTANCE

Use this procedure to drop an Oracle Ultra Search instance.

Syntax

```
OUS_ADM.DROP_INSTANCE(
  inst_name IN VARCHAR2
);
```

inst_name
The name of the instance to drop.

Example

```
OUS_ADM.DROP_INSTANCE('Scott instance')
```

GRANT_ADMIN

Use this procedure to grant instance administrator privileges to the specified user either for the current instance or all instances.

Syntax

```
OUS_ADM.GRANT_ADMIN (
 user_name     IN VARCHAR2,
 user_type     IN NUMBER DEFAULT DB_USER,
 scope         IN NUMBER DEFAULT CURRENT_INSTANCE,
 grant_option  IN NUMBER DEFAULT NO_OPTION
);
```

user_name
The name of the user to whom the administrator privilege should be assigned.

user_type
The user type (`OUS_ADM.DB_USER`: database user, `OUS_ADM.LDAP_USER`: lightweight `single sign-on` user).

scope
The scope of the granting; `CURRENT_INSTANCE` or `ALL_INSTANCE`.

grant_options
Options for granting privileges: `NO_OPTION` or `WITH_GRANT`, which allows the grantee to grant the privilege to other users.

Example

```
OUS_ADM.GRANT_ADMIN('scott',ous_adm.DB_USER, ous_adm.ALL_INSTANCE);
```

REVOKE_ADMIN

Use this procedure to revoke instance administrator privileges from the specified user.

Syntax

```
OUS_ADM.REVOKE_ADMIN (
 user_name  IN VARCHAR2,
 user_type  IN NUMBER DEFAULT DB_USER,
 scope      IN NUMBER DEFAULT CURRENT_INSTANCE
);
```

user_name
The name of the user whose privileges are to be revoked.

user_type
The user type (`OUS_ADM.DB_USER`: database user, `OUS_ADM.LDAP_USER`: lightweight `single sign-on` user).

scope
The scope of the granting; `CURRENT_INSTANCE` or `ALL_INSTANCE`.

Example

```
OUS_ADM.REVOKE_ADMIN('scott',ous_adm.DB_USER);
```

SET_INSTANCE

Use this procedure to operate on an Oracle Ultra Search instance. Almost all OUS_ADM APIs require SET_INSTANCE be called first.

Syntax

This procedure takes two forms. In the first, you specify the name of the instance to set.

```
OUS_ADM.SET_INSTANCE(
  inst_name IN VARCHAR2
);
```

inst_name
The name of the instance to set.

In the second form, you specify the ID of the instance to set.

```
OUS_ADM.SET_INSTANCE(
  inst_id IN NUMBER
);
```

inst_id
The ID of the instance to set.

Example

```
OUS_ADM.SET_INSTANCE('Scott Instance');
```

Schedule-Related APIs

This section provides reference information for using the schedule related APIs.

CREATE_SCHEDULE

Use this procedure to create a crawler schedule. It returns an ID for the schedule.

Syntax

```
OUS_ADM.CREATE_SCHEDULE (
 name            IN VARCHAR2,
 interval        IN VARCHAR2,
 crawl_mode      IN NUMBER DEFAULT REGULAR_CRAWL,
 recrawl_policy  IN NUMBER DEFAUL RECRAWL_WHEN_MODIFIED,
 crawler_id      IN NUMBER DEFAULT LOCAL_CRAWLER
) return number;
```

name
The name of the schedule to create.

interval
The schedule interval. This is a string generated from the OUS_INTERVAL function.

crawl_mode
The crawl mode can be `REGULAR_CRAWL`, `CRAWL_ONLY`, or `INDEX_ONLY`.

recrawl_policy
The recrawl condition can be `RECRAWL_WHEN_MODIFIED` or `RECRAWL_ON_EVERYTHING`.

crawler_id
The ID of the crawler used to run the schedule. This can be `LOCAL_CRAWLER` or the remote crawler ID.

Example

This example creates a crawler schedule that mandates only crawling a marketing Web site with no indexing; it is started every 6 hour by the local crawler.

```
schedule_id := OUS_ADM.CREATE_SCHEDULE('marketing site schedule',
            OUS_ADM.INTERVAL(ous_adm.HOURLY,6), ous_adm.CRAWL_ONLY);
```

DROP_SCHEDULE

Use this procedure to drop a crawler schedule.

Syntax

```
OUS_ADM.DROP_SCHEDULE (
 name IN VARCHAR2
);
```

name
The name of the schedule to drop.

Example

```
OUS_ADM.DROP_SCHEDULE('marketing site schedule');
```

INTERVAL

Use this function to generate a schedule interval string.

Syntax

```
OUS_ADM.INTERVAL (
 type        IN NUMBER,
 frequency   IN NUMBER DEFAULT 1,
 start_hour  IN NUMBER DEFAULT 1,
 start_day   IN NUMBER DEFAULT 1
) return varchar2;
```

type
The schedule interval type. This allowed values are defined as package constants: HOURLY, DAILY, WEEKLY, MONTHLY, and MANUAL.

frequency
The schedule frequency. This depends on the interval type; it can be "every *x* number of hours / days / weeks / months." Not used for MANUAL interval type.

start_hour
The schedule's launching hour, in 24-hour format, where 1 represents 1 AM. Not used for HOURLY and MANUAL schedules.

start_day
The schedule's start day; this parameter is only used for WEEKLY and MONTHLY intervals. The day of the week is specified as 0 through 6, where 0 is Sunday; the day of the month is specified as 1 through 31.)

Examples

This specifies an interval of every 5 days starting at 6 PM:

```
OUS_ADM.INTERVAL(OUS_ADM.DAILY, 5, 18);
```

This specifies launch-on-demand:

```
OUS_ADM.INTERVAL(OUS_ADM.MANUAL);
```

This specifies every 2 weeks on Monday, starting at 6 AM:

```
OUS_ADM.INTERVAL(type=>OUS_ADM.WEEKLY,frequency=>2,start_day=>2,start_hour=>6);
```

This specifies every 3 months, starting on the first day of the month at 11 PM:

```
OUS_ADM.INTERVAL(OUS_ADM.MONTHLY, 3, 23, 1);
```

SET_SCHEDULE

Use this procedure to run, resume, or stop a schedule.

Syntax

```
OUS_ADM.SET_SCHEDULE (
 name       IN VARCHAR2,
 operation  IN NUMBER
);
```

name
The name of the schedule.

operation
This may be EXECUTE, RESUME, or STOP.

Example

```
OUS_ADM.SET_SCHEDULE('marketing site schedule', ous_adm.EXECUTE);
```

UPDATE_SCHEDULE

Use this procedure to update a crawler schedule.

Syntax

```
OUS_ADM.UPDATE_SCHEDULE (
 name       IN VARCHAR2,
 operation  IN NUMBER,
 value      IN VARCHAR2 DEFAULT null
);
```

name
The name of the schedule to update.

operation
The desired update operation.

Some operations may need a value. Possible values include RENAME, ADD_DS, REMOVE_DS, SET_INTERVAL, CRAWL_MODE, RECRAWL_POLICY, and SET_CRAWLER. Values that are not allowed include ENABLE_SCHEDULE and DISABLE_SCHEDULE.

value
This parameter is context-sensitive to the update operation. It can be a new schedule name (RENAME), a data source name (ADD_DS or REMOVE_DS), an interval string (SET_INTERVAL), a crawl mode value (CRAWL_MODE), a recrawl policy (RECRAWL), or a crawler ID (SET_CRAWLER).

Examples

```
OUS_ADM.UPDATE_SCHEDULE('marketing site schedule', ous_adm.SET_INTERVAL,
       OUS_ADM.INTERVAL(ous_adm.HOURLY,3);
)
OUS_ADM.UPDATE_SCHEDULE('marketing site schedule', OUS_ADM.RENAME,
       'marketing site');

OUS_ADM.UPDATE_SCHEDULE('marketing site', OUS_ADM.ADD_DS,
       'marketing primary site');

OUS_ADM.UPDATE_SCHEDULE('marketing site', ous_adm. DISABLE_SCHEDULE);

OUS_ADM.UPDATE_SCHEDULE('marketing site',
       OUS_ADM.RECRAWL_POLICY, ous_adm.RECRAWL_ON_EVERYTHING);
```

In this example, 1001 is the ID of a remote crawler:

```
OUS_ADM.UPDATE_SCHEDULE('marketing site', ous_adm.CRAWLER_ID, 1001);
```

Crawler Configuration APIs

This section provides reference information for using the crawler configuration APIs.

IS_ADMIN_READONLY

Use this function to check whether a crawler configuration setting is read-only or not. IS_ADMIN_READONLY returns 1 if the configuration is read-only, 0 if it is not.

Syntax

```
OUS_ADM.IS_ADMIN_READONLY (
  config_name  IN NUMBER,
  crawler_id   IN NUMBER DEFAULT LOCAL_CRAWLER
) return number;
```

config_name
The name of the crawler configuration. Possible values are:

Configuration Name	Description
CC_CACHE_DIRECTORY	crawler cache directory path
CC_CACHE_SIZE	size of the cache in megabytes
CC_CACHE_DELETION	enable / disable removing cache files after indexing
CC_LOG_DIRECTORY	crawler log file location

crawler_id
The ID of the crawler whose configuration you are checking. This may be set either to LOCAL_CRAWLER or the ID of a remote crawler.

Example

```
If OUS_ADM.IS_ADMIN_READONLY(ous_adm.CC_CACHE_DIRECTORY) then

  …

end if;
```

SET_ADMIN_READONLY

Use this procedure to prevent a crawler configuration setting from being modified from the administration GUI page. This procedure is useful when a setting, such as the location of a cache directory, should not be controlled from the administration GUI; this might be the case, for example, when the people managing the server machine do not also manage Oracle Ultra Search.

Syntax

```
OUS_ADM.SET_ADMIN_READONLY (
 config_name IN NUMBER,
 read_only   IN NUMBER DEFAULT YES,
 crawler_id  IN NUMBER DEFAULT LOCAL_CRAWLER
);
```

config_name
The name of the crawler configuration setting. Possible values are:

Configuration Name	Description
CC_CACHE_DIRECTORY	crawler cache directory path
CC_CACHE_SIZE	size of the cache in megabytes
CC_CACHE_DELETION	enable / disable removing cache files after indexing
CC_LOG_DIRECTORY	crawler log file location

read_only
Set to YES to prevent the setting from being modified from the GUI.

crawler_id
The ID of the crawler whose configuration you are modifying. This may be set either to LOCAL_CRAWLER or the ID of a remote crawler.

Examples

```
OUS_ADM.SET_ADMIN_READONLY(ous_adm.CC_CACHE_DIRECTORY, ous_adm.YES);

OUS_ADM.SET_ADMIN_READONLY(ous_adm.CC_LOG_DIRECTORY,ous_adm.NO)
```

UPDATE_CRAWLER_CONFIG

Use this procedure to update crawler configurations.

Syntax

```
OUS_ADM.UPDATE_CRAWLER_CONFIG (
 config_name   IN NUMBER,
 config_value  IN VARCHAR2
);
```

config_name
The name of the crawler configuration.

config_value
The configuration value. Possible values are:

Configuration Name	Description	Value
CC_CACHE_DIRECTORY	crawler cache directory path	any valid directory path
CC_CACHE_SIZE	size of the cache in megabytes	any positive integer
CC_CACHE_DELETION	enable / disable removing cache files after indexing	OUS_ADM.DELETE_CACHE or OUS_ADM_KEEP_CACHE
CC_LOG_DIRECTORY	crawler log file location	any valid directory path
CC_DATABASE	connection string to the backend database	any valid JDBC connection string
CC_PASSWORD	database connection password for the crawler to connect to the database	database connect password for the schema that owns the instance
CC_JDBC_DRIVER	JDBC driver type used by the local crawler	OUS_ADM.THIN_DRIVER or OUS_ADM.OCI_DRIVER

Example

```
OUS_ADM.UPDATE_CRAWLER_CONFIG(ous_adm.CC_CACHE_DIRECTORY,
          '/private1/ultrasearch/cache/');

OUS_ADM.UPDATE_CRAWLER_CONFIG(ous_adm.CC_CACHE_SIZE,15);
```

A

Loading Metadata into Oracle Ultra Search

Oracle Ultra Search provides a command-line tool to load metadata into an Oracle Ultra Search database. If you have a large amount of data, then this is probably faster than using the HTML-based administration tool.

The loader tool supports the following types of metadata:

- Search attribute list of values (LOVs) and display names
- Document relevancy boosting and document loading

The metadata loader is a Java application. To use the program, you must put the metadata in an XML file that conforms to the XML schema formats described in the following sections. You then can launch the Java program with the XML filename, the database related parameters, and the loader type parameter. The program parses the XML file and uploads the metadata. Status and error messages are displayed in the terminal console.

See Also: "Document Relevancy Boosting" on page 1-9

Launching the Loading Tool

The loader program binary file is located in the following directory: `%ULTRASEARCH_HOME%/bin/MetaLoader.class`.

Your computer should have Java 1.2 compliant Java Runtime or higher. The following Java libraries should be included in the system Java CLASSPATH:

- Oracle JDBC Thin Driver version 1.2. The filename is `classes12.zip`.
- Oracle XML parser for Java version 2. The filename is `xmlparserv2.jar`.
- Oracle XML schema processor for Java. The filename is `xschema.jar`.
- Oracle Ultra Search Java library. The filename is `ultrasearch.jar`.
- Oracle JDBC globalization support version 1.2. The filename is `nls_charset12.zip`.

Also include the path for Oracle Ultra Search binary files in the system Java CLASSPATH: `%ULTRASEARCH_HOME%/bin` on UNIX or `%ULTRASEARCH_HOME%\bin` on Windows.

To launch the file, enter the following:

```
% java MetaLoader -db database_connection_string -u user_name -p password      -i
instance_name -type loader_type -f input_file
```

Where:

- `-db` is the database connection string
- `-u` is the database schema user name
- `-p` is the database schema password
- `-i` is the Oracle Ultra Search instance name
- `-type` is the loader metadata type:lov or doc
- `-f` is the input metadata XML filename

For example, suppose you use the tool to load attribute LOVs specified in the XML file `test.xml` with the following arguments:

- Database connection string: dlsun576:5521:isearch
- Schema user name: wk_test
- Schema password: welcome
- Oracle Ultra Search instance name: wk_inst

The following statement launches the loader program:

```
% java MetaLoader -db dlsun576:5521:isearch -u wk_test -p welcome -i wk_inst -type
lov -f test.xml
```

Loading Documents and Relevance Scores

To use the loader tool to add documents and their relevancy boosting scores into Oracle Ultra Search, the parameter `-type` value should be doc.

The Input XML File

The document URL and relevance boosting scores are defined in an XML file. You can define one or more documents to be boosted. Each document can have one or more boosting score pairs. The definition of the XML file is stored in the XML schema.

> **See Also:** "XML Schema for Document Relevance Boosting" on page A-4

Example of the Document Relevance Boosting XML File

```
<?xml version = "1.0" encoding = "UTF-8"?>
<doc_list>
  <doc url="http://www.oracle.com" data_source_name="Data Source A">
    <term score="100">database</term>
    <term score="90">internet</term>
    <term score="80">software</term>
  </doc>
  <doc url="http://www-st.us.oracle.com" data_source_name="Data Source B">
    <term score="100">Sever Technology</term>
    <term score="100">ST Web site</term>
    <term score="95">st</term>
  </doc>
</doc_list>
```

In the previous example, the document URL `http://www.oracle.com` is loaded to the data source Data Source A. This is defined in Oracle Ultra Search with relevance boosting term database and score 100, term internet and score 90, term software and score 80.

Note: The data source name is the original data source name, not the data source display name.

Loading Search Attribute LOVs and LOV Display Names

To use loader tool to add LOV entries and display names to Oracle Ultra Search, the parameter -type value should be lov.

The LOV XML File

The LOV entries and display names are defined in a XML file. You can define one or more search attribute LOVs in the XML file. Both default LOV and data source-specific LOVs are put in the XML file. The definition of the XML file is stored in the XML schema.

See Also: "XML Schema for LOVs and LOV Display Names" on page A-4

Example of the LOV XML File

```
<?xml version = "1.0" encoding = "UTF-8"?>
<lov_list>
  <lov search_attr_name="Department" search_attr_type="string">
    <default>
      <lov_values>
        <entry value="100"></entry>
        <entry value="200"></entry>
      </lov_values>
      <lov_display_names lang="en-US">
        <entry value="100" display_name="Human Resource"></entry>
        <entry value="200" display_name="Finance"></entry>
      </lov_display_names>
    </default>
    <data_source name ="data source a">
      <lov_values>
        <entry value="300"></entry>
        <entry value="400"></entry>
      </lov_values>
      <lov_display_names lang="en-US">
        <entry value="300" display_name="Sales"></entry>
        <entry value="400" display_name="Marketing"></entry>
      </lov_display_names>
    </data_source>
    <data_source name ="data source b">
      <lov_values>
        <entry value="500"></entry>
        <entry value="600"></entry>
      </lov_values>
      <lov_display_names lang="en-US">
        <entry value="500" display_name="Production"></entry>
        <entry value="600" display_name="Research"></entry>
      </lov_display_names>
    </data_source>
  </lov>
</lov_list>
```

In the previous example, several LOVs for the string type search attribute Department are loaded to Oracle Ultra Search. They are:

- Default LOV entries for search attribute Department
- Search attribute Department LOV for data source data source a
- Search attribute Department LOV for data source data source b

XML Schema for Document Relevance Boosting

The XML schema for document relevance boosting terms and scores is described as follows:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!--Generated by XML Authority. Conforms to w3c http://www.w3.org/2001/XMLSchema-->
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
         elementFormDefault = "qualified">
  <xsd:element name = "doc_list">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name = "doc" maxOccurs = "unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name = "term" maxOccurs = "unbounded">
                <xsd:complexType>
                  <xsd:simpleContent>
                    <xsd:extension base = "xsd:string">
                      <xsd:attribute name = "score" use = "required" type = "xsd:integer"/>
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name = "url" use = "required" type = "xsd:string"/>
            <xsd:attribute name = "data_source_name" use = "required" type = "xsd:string"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

XML Schema for LOVs and LOV Display Names

The XML schema for LOV entries and display names is described as follows:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!--Generated by XML Authority. Conforms to w3c http://www.w3.org/2001/XMLSchema-->
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
         elementFormDefault = "qualified">
  <xsd:element name = "lov_list">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name = "lov" maxOccurs = "unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name = "default" minOccurs = "0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name = "lov_values" minOccurs = "0">
                      <xsd:complexType>
                        <xsd:sequence>
```

```
                        <xsd:element name = "entry" maxOccurs = "unbounded">
                          <xsd:complexType>
                            <xsd:attribute name = "value" use = "required" type = "xsd:string"/>
                          </xsd:complexType>
                        </xsd:element>
                      </xsd:sequence>
                    </xsd:complexType>
                  </xsd:element>
                  <xsd:element name = "lov_display_names" minOccurs = "0" maxOccurs =
"unbounded">
                    <xsd:complexType>
                      <xsd:sequence>
                        <xsd:element name = "entry" maxOccurs = "unbounded">
                          <xsd:complexType>
                            <xsd:attribute name = "value" use = "required" type = "xsd:string"/>
                              <xsd:attribute name = "display_name" use = "required" type =
"xsd:string"/>
                          </xsd:complexType>
                        </xsd:element>
                      </xsd:sequence>
                      <xsd:attribute name = "lang" use = "required">
                        <xsd:simpleType>
                          <xsd:restriction base = "xsd:string">
                            <xsd:length value = "5"/>
                              <xsd:pattern value = "[a-zA-Z]{2}\-[a-zA-Z]{2}"/>
                          </xsd:restriction>
                        </xsd:simpleType>
                      </xsd:attribute>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
            <xsd:element name = "data_source" minOccurs = "0" maxOccurs = "unbounded">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name = "lov_values" minOccurs = "0">
                    <xsd:complexType>
                      <xsd:sequence>
                        <xsd:element name = "entry" maxOccurs = "unbounded">
                          <xsd:complexType>
                            <xsd:attribute name = "value" use = "required" type = "xsd:string"/>
                          </xsd:complexType>
                        </xsd:element>
                      </xsd:sequence>
                    </xsd:complexType>
                  </xsd:element>
                  <xsd:element name = "lov_display_names" minOccurs = "0">
                    <xsd:complexType>
                      <xsd:sequence>
                        <xsd:element name = "entry" maxOccurs = "unbounded">
                          <xsd:complexType>
                            <xsd:attribute name = "value" use = "required" type = "xsd:string"/>
                            <xsd:attribute name = "display_name" use = "required" type =
"xsd:string"/>
                          </xsd:complexType>
                        </xsd:element>
                      </xsd:sequence>
                      <xsd:attribute name = "lang" use = "required">
                        <xsd:simpleType>
```

```
                        <xsd:restriction base = "xsd:string">
                          <xsd:length value = "5"/>
                            <xsd:pattern value = "[a-zA-Z]{2}\-[a-zA-Z]{2}"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:attribute>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
              <xsd:attribute name = "name" use = "required" type = "xsd:string"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name = "search_attr_name" use = "required" type = "xsd:string"/>
        <xsd:attribute name = "search_attr_type" use = "required">
          <xsd:simpleType>
            <xsd:restriction base = "xsd:string">
              <xsd:enumeration value = "string"/>
                <xsd:enumeration value = "number"/>
                  <xsd:enumeration value = "date"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
</xsd:schema>
```

B

Altering the Crawler Java Classpath

The Oracle Ultra Search crawler is a pure Java application that runs in a Java virtual machine. A Java virtual machine uses the Java classpath to find classes during runtime. When Oracle Ultra Search is installed, the default crawler classpath is stored in the database. Whenever a new Oracle Ultra Search instance is created, this default classpath is copied and used as the crawler classpath for that specific instance.

Reasons for Altering the Crawler Java Classpath

Usually, you do not need to alter the crawler Java classpath. However, there are certain reasons for you to do so. One reason could be to replace the JavaMail reference implementation with a third party JavaMail implementation.

Difference Between the Crawler Classpath and the Remote Crawler Classpath

The crawler classpath is the classpath of a crawler that runs on the same host as the Oracle Ultra Search backend. However, Oracle Ultra Search allows remote crawlers to be run on other hosts for scalability.

Remote crawler activation uses Java remote method invocation (RMI) technology. As a result, the classpath setting of a remote crawler is inherited from the classpath settings of the RMI registry and RMI daemon.

See Also: "Using the Remote Crawler" on page 10-4

Altering the Crawler Java Classpath on the Oracle Ultra Search Server Host

1. Log on to the host where the Oracle Ultra Search backend is installed. Locate the file `$ORACLE_HOME/ultrasearch/admin/wk0addcpath.sql`.
2. Using SQL*Plus, run the `wk0addcpath.sql` script as the `WKSYS` super-user or as a database user that has been granted the super-user privileges. (This script only updates the `CRAWLER_CONFIG_DEFAULT` table. You also need to reconfigure your crawlers to get the `WK$CRAWLER_CONFIG` table updated correctly.)
3. When prompted, specify whether you want to alter the default classpath or an instance-specific classpath. Altering the default classpath causes all subsequently created instances to use that classpath. Existing instances are not modified.

4. When prompted, enter the Oracle Ultra Search instance name if you are attempting to modify an instance-specific classpath. If you are modifying the default classpath, then you do not need enter anything here.
5. When prompted, specify whether you want to update the entire classpath or append to it. Appending to a classpath adds entries to the beginning of it. Usually, earlier entries in the classpath override later entries in the case of duplicate classes.
6. When prompted, enter the new classpath if updating the entire classpath. If you are appending one or more directories or library files to the classpath, then enter these separated by the classpath separator for the platform where the Oracle Ultra Search backend is installed (the colon on UNIX platforms, or the semicolon on Windows).

Altering the Crawler Java Classpath on a Remote Crawler Host

1. Log on to the remote crawler host where the Oracle Ultra Search middle tier is installed. On a UNIX computer, locate and open the file `$ORACLE_HOME/ultrasearch/tools/remotecrawler/scripts/unix/define_env`. On a Windows computer, locate and open the file `$ORACLE_HOME/ultrasearch/tools/remotecrawler/scripts/winnt/define_env.bat`.
2. The `define_env` file specifies all environment settings used by the RMI subsystem. To alter the classpath, use a text editor to modify the `APPLICATION_CLASSPATH` variable.
3. Restart the RMI subsystem for these changes to take effect.

> **See Also:** "Using the Remote Crawler" on page 10-4 for more details on starting up the RMI subsystem

C

Oracle Ultra Search Views

This appendix lists all of the views provided by Oracle Ultra Search. The system provides the following views:

- OUS_INSTANCES
- OUS_SCHEDULES
- OUS_DEFAULT_CRAWLER_SETTINGS
- OUS_CRAWLER_SETTINGS

OUS_INSTANCES

This view displays all instance information. Any user can query it.

Column Name	Type	Description
INS_ID	NUMBER	Instance ID
INS_NAME	VARCHAR2(100)	Instance name
INS_SCHEMA	VARCHAR2(30)	Instance owner
INS_MODE ONLY	VARCHAR2(30)	Instance mode; UPDATABLE / READ

OUS_SCHEDULES

This view displays schedule data for the current instance.

Column Name	Type	Description
SCH_ID	NUMBER	Schedule ID
SCH_NAME	VARCHAR2(100)	Schedule name
SCH_INTERVAL	VARCHAR2(30)	Schedule interval
SCH_LAST_RUN	DATE	Time stamp of last run
SCH_NEXT	DATE	Next scheduled run
SCH_STATUS	VARCHAR2(100)	Schedule status
SCH_ERROR	VARCHAR2(4000)	Schedule error if failed
SCH_CRAWLER_LOG_NAME	VARCHAR2(100)	Log file name of current running crawler
SCH_CRAWLER_ID	NUMBER	ID of the crawler to run this schedule
SCH_FORCE_RECRAWL	NUMBER	Force recrawl of the data source
SCH_CRAWL_HOME	VARCHAR2(30)	Schedule crawling mode

OUS_DEFAULT_CRAWLER_SETTINGS

This view shows default crawler settings like crawling depth and log file directory.

Column Name	Type	Description
DCS_NAME	VARCHAR2(30)	Crawler setting name
DCS_VALUE	VARCHAR2(4000)	Crawler setting value

OUS_CRAWLER_SETTINGS

This view shows crawler setting at the data source level for the current instance.

Column Name	Type	Description
CWS_DS_ID	NUMBER	Data source ID
CWS_NAME	VARCHAR2(30)	Crawler setting name
CWS_VALUE	VARCHAR2(4000)	Crawler setting value

D

URL Crawler Status Codes

The crawler uses a set of codes to indicate the result of the crawled URL. Besides the standard HTTP status code, it uses its own code for non-HTTP related situations. Only URLs with status 200 will be indexed.

The following table lists the URL status codes.

Code	Description
200	URL OK
400	Bad request
401	Authorization required
402	Payment required
403	Access forbidden
404	Not found
405	Method not allowed
406	Not acceptable
407	Proxy authentication required
408	Request timeout
409	Conflict
410	Gone
414	Request URI too large
500	Internal server error
502	Bad gateway
503	Service unavailable
504	Gateway timeout
505	HTTP version not supported
902	Timeout reading document
903	Filtering failed
904	Out of memory error
905	IOEXCEPTION in processing URL
906	Connection refused
907	Socket bind exception

Code	Description
908	Filter not available
909	Duplicate document detected
910	Duplicate document ignored
911	Empty document
951	URL not crawled (this can happen if robots.txt specifies that a certain document should not be indexed)
952	URL crawled
953	Metatag redirection
954	HTTP redirection
955	Black list URL
956	URL is not unique
957	Sentry URL (URL as a place holder)
958	Document read error
959	Form login failed
1001	Datatype is not TEXT / HTML
1002	Broken network data stream
1003	HTTP redirect location does not exist
1004	Bad relative URL
1005	HTTP error
1006	Error parsing HTTP header
1007	Invalid URL table column name
1008	JDBC driver missing
1009	Binary document reported as text document
1010	Invalid display URL

Index