Oracle® Ultra Search

User's Guide

Release 9.0.3

August 2002

Part No. B10043-01

ORACLE®

Oracle Ultra Search User's Guide, Release 9.0.3

Part No. B10043-01

Primary Author: Michele Cyran

Contributors: Sandeepan Banerjee, Stefan Buchta, Eddy Chee, Chung-Ho Chen, Will Chin, Jack Chung, Cindy Hsin, Yasuhiro Matsuda, Colin McGregor, Paul Yang, Steve Yang, David Zhang

Graphic Designer: Valarie Moore

Contents

3 Post-Installation and Upgrade Information

4 Understanding the Ultra Search Crawler and Data Sources

5 Understanding the Ultra Search Administration Tool

6 Ultra Search Developer's Guide and API Reference

E Loading Metadata into Ultra Search

F Altering the Crawler Java Classpath

G Customizing the Query Syntax Expansion 9.0.1

Index

Send Us Your Comments

Oracle Ultra Search User's Guide, Release 9.0.3

Part No. B10043-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. 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?

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

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7410 Attn: Oracle Ultra Search Documentation Manager
- Postal service:
  Oracle Corporation
  Server Technologies Documentation
  500 Oracle Parkway, Mailstop 4op11
  Redwood Shores, CA 94065
  USA

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

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

Preface

Oracle Ultra Search User's Guide describes how to install, configure, and use Ultra Search. It also describes how to use Ultra Search APIs to create custom applications.

This preface contains these topics:

Audience

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

- Administer Oracle Ultra Search installations and 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.

Organization

This document contains:

"What's New in Ultra Search?"

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

Chapter 1, "Introduction to Ultra Search"

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

Chapter 2, "Installing and Configuring Ultra Search"

This chapter describes how to install and configure Ultra Search.

Chapter 3, "Post-Installation and Upgrade Information"

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

Chapter 4, "Understanding the Ultra Search Crawler and Data Sources"

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

Chapter 5, "Understanding the Ultra Search Administration Tool"

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

Chapter 6, "Ultra Search Developer's Guide and API Reference"

This chapter explains the following Ultra Search APIs: query API, crawler agent API, email API, and URL rewriter API. It also provides related API information, such as details about the sample query applications, the query tag library, and query syntax expansion customization.

Appendix A, "Tuning the Web Crawling Process"

This appendix describes web crawling stategies, URL looping, and ways to monitor the crawling process.

Appendix B, "Tuning Query Performance"

This appendix contains suggestions on how to improve the performance of the Ultra Search query.

Appendix C, "Using the Remote Crawler"

This appendix explains why and how to use the remote crawler.

Appendix D, "Table Data Source Synchronization"

This appendix describes the logging mechanism Ultra Search uses to optimize the crawling of table sources.

Appendix E, "Loading Metadata into Ultra Search"

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

Appendix F, "Altering the Crawler Java Classpath"

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

Appendix G, "Customizing the Query Syntax Expansion 9.0.1"

This appendix describes how to customize the query syntax expansion for a previous release of Ultra Search: release 9.0.1.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Database Concepts*
- *Oracle9i Database Administrator's Guide*
- *Oracle9i Database Performance Tuning Guide and Reference*

Many books in the documentation set use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle9i Sample Schemas* for information on how these schemas were created and how you can use them yourself.

In North America, printed documentation is available for sale in the Oracle Store at

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

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

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

`http://otn.oracle.com/admin/account/membership.html`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/docs/index.htm`

To access the database documentation search engine directly, please 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

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.	*Oracle9i 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 executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as 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 open 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.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`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
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	DECIMAL (*digits* [, *precision*])
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	{ENABLE | DISABLE}
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	CREATE TABLE ... AS *subquery*; SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;

Convention	Meaning	Example
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. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates 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 >	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 file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32`

Convention	Meaning	Example
`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 scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\"` `C:\>imp SYSTEM/`*`password`* `FROMUSER=scott TABLES=(emp, dept)`
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 Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default used one of the following names: ▪ `C:\orant` for Windows NT ▪ `C:\orawin98` for Windows 98 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`. 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\ora`*nn*, where *nn* is the latest release number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle9i Database Getting Started for 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.

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 Corporation 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 Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

What's New in Ultra Search?

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

Note: Ultra Search release 9.0.3 is part of the Oracle Collaboration Suite release 9.0.3.

Ultra Search release 9.2 was part of Oracle9*i* release 9.2. Ultra Search release 1.0.3 was part of Oracle9*i* release 9.0.1.

Ultra Search release 9.0.2 was part of Oracle9*i* Application Server (9*i*AS) release 9.0.2.

- **Integration with Oracle9i Application Server**

  Although Ultra Search in the Oracle9*i* Application Server (9*i*AS) is the same product as Ultra Search in the Oracle9*i* database, there are a couple differences:

  – Ultra Search is integrated with 9*i*AS Portal. This lets Portal users add powerful multi-repository search to their Portal pages. It also has the capability to crawl and make searchable Portal's own repository. This is only available with 9*i*AS.

  – 9*i*AS includes a single sign-on (SSO) server. Therefore, SSO users can log on once for all components of the 9*i*AS product, and the Ultra Search administrative interface allows user management operations on either database users or SSO users. Authenticated SSO users never see the Ultra Search login screen. Instead, they can immediately choose an instance. If the SSO user does not have permissions to manage Ultra Search (set in the Users Page), then the SSO user receives an error. This is only available with 9*i*AS.

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

- **Extensible Crawler and Crawler Agents**

  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:

  - "Ultra Search Crawler Agent API" on page 6-3
  - *Oracle Ultra Search Javadoc*

- **Sample Query Applications**

  Ultra Search includes fully functional sample query applications to query and display search results. The sample query applications include a sample search portlet. The sample Ultra Search portlet demonstrates how to write a search portlet for use in Oracle 9*i*AS Portal. This same portlet is installed as a feature of the Oracle 9*i*AS Portal product.

  See Also: "Ultra Search Query API" on page 6-2

- **Sample Search Portlet**

  Ultra Search provides a search portlet that can be embedded in Oracle Portal pages. It is implemented as a Java Server Page application.

  The Ultra Search search portlet supports most of the functionality provided by the Query API Complete Sample application.

  See Also:

  - The Oracle 9*i*AS Portal documentation for more information about portlets
  - Oracle Ultra Search Sample Query Applications Readme for more information about the Query API Complete Sample application

- **Query API**

  Oracle Ultra Search offers a flexible API to incorporate search functionality to your portal site. The new functionalities in query API include the following:

  – 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:

  - "Ultra Search Query API" on page 6-2
  - *Oracle Ultra Search Javadoc*

- **URL Rewrite**

  The URL rewriter is a user-supplied java module for implementing the 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 5-20
- "Ultra Search URL Rewriter API" on page 6-13
- *Oracle Ultra Search Javadoc*

- **Robot Exclusions**

  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 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 5-20

- **Display URL Support**

  When gathering information from a database-based Web application, Ultra Search lets you specify a URL to display the data retrieved 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: "Display URL and Access URL" on page 4-3

- **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.

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: "Document Attributes" on page 4-3

- **Metadata Loader**

  Ultra Search provides a command-line tool to load metadata into an Ultra Search database. If you have a large amount of data, 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 relevance boosting and document loading

  See Also: Appendix E, "Loading Metadata into Ultra Search"

- **Document Relevance Boosting**

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

  See Also: "Queries Page" on page 5-32

- **Data Harvesting Mode**

  For initial planning purposes, you might want the crawler 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 5-27

- **Instance Snapshot Support**

  You can create a read-only snapshot of a master 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 5-7

- **Single Sign-On Authentication**

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

  – A single sign-on (SSO) user managed in the Oracle Internet Directory (OID) and authenticated with the SSO server

  – A local database schema user in the Ultra Search database (non-SS0 mode)

  – An Enterprise Manager `IAS_ADMIN` user

  Note: Single sign-on (SSO) is available only with the Oracle9*i* Application Server (9*i*AS) release. It is not available with the Oracle9*i* database release.

  See Also: "Logging On to Ultra Search" on page 5-4

- **Query Syntax Expansion**

  In previous releases, the code for the default query syntax expansion implementation was contained in the `WK_QUERYEXP` PL/SQL package. Now, the Contains query 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 'title' StringAttribute.

  See Also: "Customizing the Query Syntax Expansion" on page 6-28

1

Introduction to Ultra Search

This chapter contains the following topics:

Ultra Search Overview

Oracle Ultra Search is built on the Oracle database server 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.

Ultra Search uses a 'crawler' to index documents; the documents stay in their own repositories, and the crawled information is used to build an index that stays within your firewall in a designated Oracle database. Ultra Search also provides APIs for building content management solutions.

Ultra Search offers the following:

- A complete text query language for text search inside the database
- Full integration with the Oracle database server and the SQL query language
- Advanced features like concept searching and theme analysis
- Indexing of all popular file formats (150+)
- Full globalization, including support for Chinese, Japanese and Korean (CJK), and Unicode

Ultra Search Components

Ultra Search is made up of the following components:

- Ultra Search Crawler
- Ultra Search Server Component
- Ultra Search Administration Tool
- Ultra Search APIs and Sample Applications

Ultra Search Crawler

The 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 Portal page groups, or user-defined data sources.

The crawler maps links and analyzes relationships. The crawler schedule is integrated with and driven from 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 filter and index the document.

> **See Also:** Chapter 4, "Understanding the Ultra Search Crawler and Data Sources"

Ultra Search Server Component

The Ultra Search server component consists of an 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 server component is not visible to users; it indexes information from the crawler and serves up the query results.

> **See Also:** "Installing the Ultra Search Server Component" on page 2-4

Ultra Search Administration Tool

The administration tool is a JSP Web application to configure and schedule the Ultra Search crawler. The administration tool is typically installed on the same machine as your Web server. You can access the administration tool from any browser in your intranet. The administration tool is independent from the Ultra Search query application. Therefore, the administration tool and query application can be hosted on different machines to enhance security and scalability.

> **See Also:** Chapter 5, "Understanding the Ultra Search Administration Tool"

Ultra Search APIs and Sample Applications

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 email Java API accesses archived emails and is used by the query application to display emails. 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.

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

See Also:

- Chapter 6, "Ultra Search Developer's Guide and API Reference"
- *Oracle Ultra Search Javadoc*

Ultra Search System Configuration

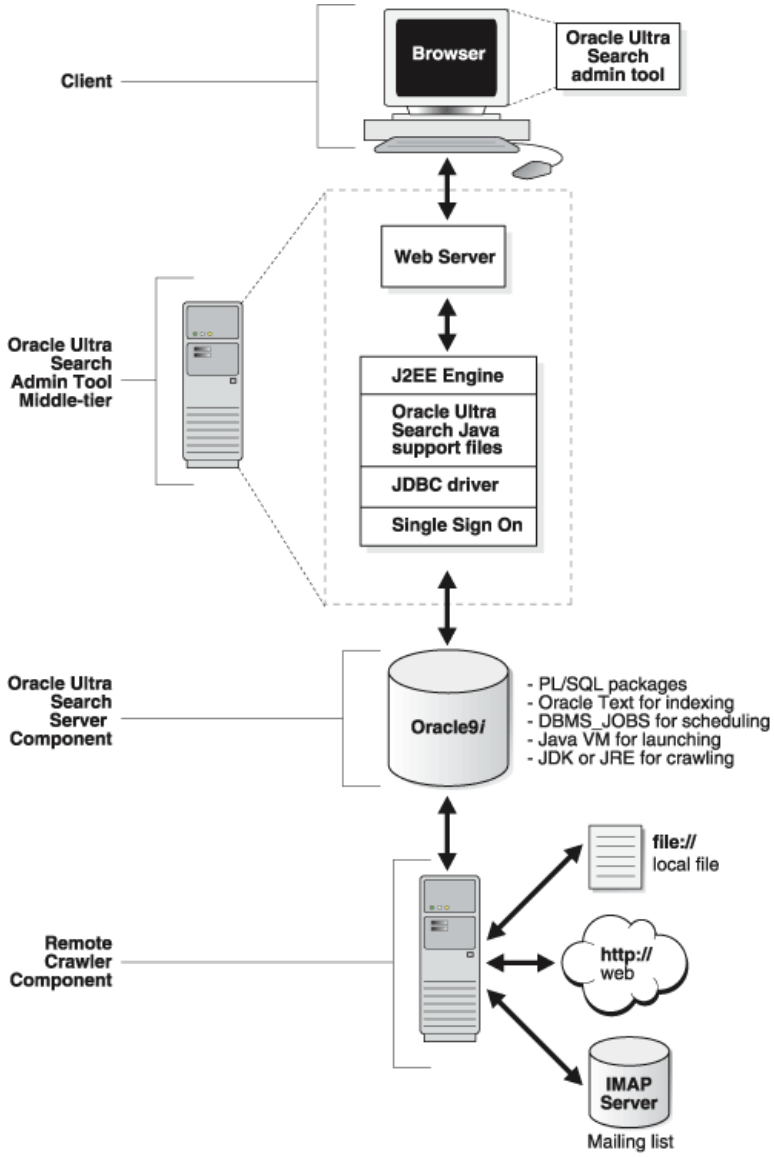
Ultra Search is a client program to the Oracle server at run time. It can be deployed in two configurations: in the server tier or in the middle tier.

The Ultra Search default query interface and the administration tool run in 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 machine as the one that runs the database server, or a different one, running Oracle9*i* AS. The Ultra Search database server component consists of the 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 machine or remotely on another machine.

See Also: Chapter 2, "Installing and Configuring Ultra Search" for more information about the components

Figure 1–1 illustrates the Ultra Search system configuration.

Figure 1–1 Oracle Ultra Search System Configuration



2

Installing and Configuring Ultra Search

This chapter contains the following topics:

Ultra Search Requirements

To use Oracle Ultra Search, you must install the following components with the Oracle Universal Installer:

- Ultra Search server component
- Ultra Search middle tier component

> **Note:** Ultra Search can be installed to crawl in both local and remote configurations. In remote configurations, the Ultra Search server component is also installed on one or more additional machines for scalability.

The Ultra Search server component can be installed on any already existing database that is Oracle database release 9.0.1 or higher with Oracle Text installed. To do this, invoke OPCA (Oracle Portal Configuration Assistant) from a middle tier host, and specify a database host.

> **See Also:** "Installing the Server Component on an Existing Database or Metadata Repository" on page 2-5

Conventions

`ORACLE_HOME` refers to the Oracle home directory where the Ultra Search server component is installed. `ORACLE_HOME` also refers to the Oracle home directory where the Ultra Search middle tier component is installed.

`REMOTE_ORACLE_HOME` refers to the Oracle home directory where the Ultra Search server component is installed on a remote crawler host.

Ultra Search System Requirements

The following section describes the Ultra Search system requirements.

Hardware Requirements

Sufficient RAM The Ultra Search indexing engine runs within Oracle. Hence, it is important that the system have enough memory to accommodate a large Oracle installation. The Oracle instance system global area should be a minimum of 50MB.

> **See Also:** *Oracle9i Database Performance Tuning Guide and Reference*

The Ultra Search Web crawler runs as a separate Java process on the same host. Allocate 50MB of memory for the Web crawler alone.

The Ultra Search administration tool is a J2EE 1.2 standard Web application. Therefore, it can be installed and run on a separate host from the Ultra Search server component. However, you might want to install and run this component on the same host as the Ultra Search server component. Regardless of your choice, allocate enough memory for the J2EE engine. Oracle Corporation recommends using the Oracle HTTP server with the Oracle J2EE container. Allocate enough memory for the HTTP server as well as the JDK that runs the J2EE engine.

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

- Approximately 3 GB of disk space for the *i*AS infrastructure or database and the Ultra Search server component.
- For each Web server host, enough space to install the Oracle HTTP server and related products. In addition, you need 15MB of disk space for the Ultra Search middle tier component.
- For each remote crawler host, the same amount of disk space as needed to install the Ultra Search server component.
- Disk space for a large `TEMPORARY` tablespace. As a general guideline, create a `TEMPORARY` tablespace as large as available depending on the RAM on your host.
- Disk space for the Ultra Search instance user's tablespace.
  - The Ultra Search instance user is a database user that you need to explicitly create. All data that is collected and processed as part of the crawling and indexing process is stored in this user's schema.
  - As a general guideline, create the tablespace as large as the total amount of data that you want to index. For example, if you approximate that the total amount of data to be crawled and indexed is 10GB, then create a tablespace that is at least 10GB for the Ultra Search instance user. Make sure to assign that tablespace as the default tablespace of the Ultra Search instance user.

Software Requirements

The Ultra Search middle tier components are Web applications. Therefore, they require a Web server to run. Oracle Corporation recommends the Oracle HTTP Server and Jserv or Oracle HTTP Server and the Oracle J2EE container.

Installing the Ultra Search Server Component

The Ultra Search server component is included as part of the Oracle database. It is installed together with the Ultra Search middle tier component during the installation. It is installed in the same Oracle home directory as the database server tier.

The Ultra Search server component consists of the following:

- Ultra Search data dictionary and PL/SQL packages
- Ultra Search crawler Java classes
- Ultra Search remote crawler
- Ultra Search product libraries

Installing and Configuring the Server Component

Step 1: Install 9*i*AS Infrastructure and the Ultra Search Server Component

Database Release: Start up the Oracle Universal Installer (OUI) on the relevant host. After choosing the destination Oracle home name and full path, choose the option "Oracle9*i* Server." Ultra Search server tier is installed with the Oracle9*i* database by default.

***i*AS Infrastructure:** Start up the OUI on the relevant host. After choosing the destination Oracle home name and full path, choose the option "Oracle9*i*AS Infrastructure 9.0.2.0.0." Ultra Search server tier is installed with the Oracle *i*AS infrastructure by default.

During the installation of *i*AS or the Oracle database server, the Ultra Search server component is installed. The following activity occurs during this process:

- All Ultra Search server component files are copied into a directory named "ultrasearch". This directory resides immediately under the `ORACLE_HOME` of the designated database installation.

- The database user WKSYS with password wksys is created. You should change this password later for security purposes. All Ultra Search database objects are installed in this user's schema.
- Various PL/SQL scripts are run against the database as user WKSYS. These scripts install and create various database objects.

> **See Also:** Your installation guide for information on setting necessary environment variables

Step 2: Configure the Oracle Database for Ultra Search

After you have installed all Ultra Search components, you must configure the Oracle database. This is a post-installation operation.

> **See Also:** "Configuring the Oracle Server for Ultra Search" on page 3-2

Installing the Server Component on an Existing Database or Metadata Repository

Ultra Search can be installed on an Oracle9*i* database, or an *i*AS metadata repository, that is currently in use.

> **Note:** This feature is supported on the *i*AS release only.

Installing Ultra Search on an existing database or metadata repository includes transferring files, creating Ultra Search super user WKSYS, loading Ultra Search PL/SQL database packages, and associating Oracle9*i* Application Server (9*i*AS) with Ultra Search.

Unlike the infrastructure database installation, Ultra Search packages are loaded from the middle tier installation. By choosing to install Oracle Portal to your database, Oracle Portal Configuration Assistant (OPCA) loads the Ultra Search packages onto the database host you choose. You then manually transfer necessary files to the database tier's file system.

> **See Also:** http://portalstudio.oracle.com/ for more information on Oracle Portal and on how to use OPCA

Database Requirements

- The Java development kit (JDK) must be release 1.2.207 or higher.
- Oracle database must be release 9.0.1 or higher.
- The initialization file must have `JOB_QUEUE_PROCESSES` set to greater than two.
- Oracle Text must be installed in the database. Oracle Text provides the text indexing and search capabilities required to index and query data retrieved from your data sources, such Web sites or database tables.

> **Note:** To run the crawler, Ultra Search requires a Java 1.2 compliant runtime environment (JDK) to be installed on the database machine. JDK 1.2.2 07 should already be installed with the Oracle HTTP server.

Installing the Server Component on an Existing Oracle9i Database

1. Create user `WKSYS`, load Ultra Search PL/SQL packages, and associate Ultra Search with 9*i*AS.

   Ultra Search PL/SQL database packages are loaded during the middle tier (9*i*AS) installation. During this installation, after the Oracle Universal Installer copies the necessary files to the middle tier host, OPCA associates Ultra Search with 9*i*AS. After specifying the database tier host, Oracle SID, and port, Ultra Search packages area loaded by OPCA to the database on the database tier host.

   If OPCA detects that the Ultra Search superuser `WKSYS` already exists, and the Ultra Search release is 1.0.3 (based on Oracle9*i* database 9.0.1), then it asks the user to choose from the following three options:

   - Deinstall/Reinstall: This drops the existing `WKSYS` user, creates a new `WKSYS` user, and reloads the Ultra Search packages. All data collected by Ultra Search is deleted.
   - Migrate: This asks the user to follow the instructions on the Ultra Search migration document for manual migration.

     See Also: "Upgrade from Ultra Search 1.0.3 (Oracle9i Database 9.0.1) to 9.0.3" on page 3-10

- Abort: This stops OPCA from loading Ultra Search PL/SQL database packages into the existing database.

Note: The default password for WKSYS is wksys. You should immediately change the password to avoid any security issues. OPCA then configures 9*i*AS to work with Ultra Search. Configuration involves the following files: mod_oc4j.conf, server.xml, and ultrasearch.properties. For details, see "Installing the Ultra Search Middle Tier on Web Server Hosts" on page 2-9.

---

2. Transfer files and configure the database. Ultra Search requires that certain files exist on the database tier's file system. Because the database tier might exist in a physically unreachable place, and because neither the Oracle Universal Installer nor OPCA have access to the remote database tier's file system, you must manually copy the necessary files. In the ORACLE_HOME/ultrasearch/setup/ directory, there are several files including setupDB.jar, setupDB.bat, setupDB.csh, and setupDB.sh. Copy these files to the ORACLE_HOME of your remote database tier.

3. Set the files to the correct location by running the following scripts for your operating system.

   For Windows NT: Edit setupDB.bat using any text editor. After the line "set ORACLE_HOME=" put the directory name where you installed the Oracle database. Save the change and run setupDB.bat by double clicking setupDB.bat or by running setupDB in the command prompt.

   For Solaris: Run setupDB.sh under the Bourne shell prompt. The system sets the current directory as ORACLE_HOME and asks for the path to your JDK_HOME. If the default path to the jar executable or Java executable is not correct, then the system asks for the path to JDK. If you do not have JDK installed, then download the latest JDK and run setupDB.sh again.

Note: To run this script, the executable bit of setupDB.sh should be turned on.

Installing the Server Component on an Existing iAS Metadata Repository

1. Create user WKSYS, load Ultra Search PL/SQL packages, and associate Ultra Search with 9*i*AS.

Ultra Search PL/SQL packages are loaded during the middle tier (9*i*AS) installation. During this installation, after the Oracle Universal Installer copies the necessary files to the middle tier host, OPCA associates Ultra Search with 9*i*AS. After specifying the metadata repository tier host, Oracle SID, and port, Ultra Search packages area loaded by OPCA to the metadata repository on the metadata repository tier host.

If the detected Ultra Search release is 9.0.2 or later, then it asks the user to choose from the following three options:

- Deinstall/Reinstall: This drops the existing `WKSYS` user, creates a new `WKSYS` user, and reloads the Ultra Search packages. All data collected by Ultra Search is deleted.
- Upgrade to 9.0.3: This upgrades the existing Ultra Search packages to release 9.0.3. All data collected with the existing Ultra Search remain intact.
- Abort: This stops OPCA from loading Ultra Search PL/SQL packages into the existing metadata repository.

Note: The default password for `WKSYS` is wksys. You should immediately change the password to avoid any security issues. OPCA then configures 9*i*AS to work with Ultra Search. Configuration involves the following files: `mod_oc4j.conf`, `server.xml`, and `ultrasearch.properties`. For details, see "Installing the Ultra Search Middle Tier on Web Server Hosts" on page 2-9.

---

2. Transfer files and configure the metadata repository. Ultra Search requires that certain files exist on the metadata repository tier's file system. Because the metadata repository tier might exist in a physically unreachable place, and because neither the Oracle Universal Installer nor OPCA have access to the remote metadata repository tier's file system, you must manually copy the necessary files. In the `ORACLE_HOME/ultrasearch/setup/` directory, there are several files including `setupDB.jar`, `setupDB.bat`, `setupDB.csh`, and `setupDB.sh`. Copy these files to the `ORACLE_HOME` of your remote metadata repository tier.
3. Set the files to the correct location by running the following scripts for your operating system.

   For Windows NT: Edit `setupDB.bat` using any text editor. After the line "set ORACLE_HOME=" put the directory name where you installed the Oracle

metadata repository. Save the change and run `setupDB.bat` by double clicking `setupDB.bat` or by running `setupDB` in the command prompt.

For Solaris: Run `setupDB.sh` under the Bourne shell prompt. The system sets the current directory as `ORACLE_HOME` and asks for the path to your `JDK_HOME`. If the default path to the jar executable or Java executable is not correct, then the system asks for the path to JDK. If you do not have JDK installed, then download the latest JDK and run `setupDB.sh` again.

Note: To run this script, the executable bit of `setupDB.sh` should be turned on.

Installing the Ultra Search Middle Tier on Web Server Hosts

The Ultra Search middle tier component includes the following:

- Ultra Search administration tool
- Ultra Search Java query API
- Ultra Search Java server page sample query applications

For the *i*AS release, the Ultra Search middle tier component is part of the Application Server install. You must choose the "Oracle 9*i*AS Portal and Wireless" option from the Oracle Universal Installer menu to install and configure the Ultra Search middle tier component with the Application Server install.

For the database release, the Ultra Search middle tier component is installed with the Ultra Search server component during the database server install. This component is also part of the database client install. However, Oracle database client install does not install Oracle HTTP server, which is a requirement for the Ultra Search middle tier component. You should install Oracle HTTP server before installing the Ultra Search middle tier component. Do so by choosing the "server" install option from the Oracle Universal Installer menu. Then, perform a custom install and choose the Oracle HTTP Server for installation.

The following section describes how to install and configure the Ultra Search middle tier component on a host that you have designated to be a Web server. You can install the Ultra Search middle tier component on multiple Web server hosts to handle a large number of incoming queries by end users.

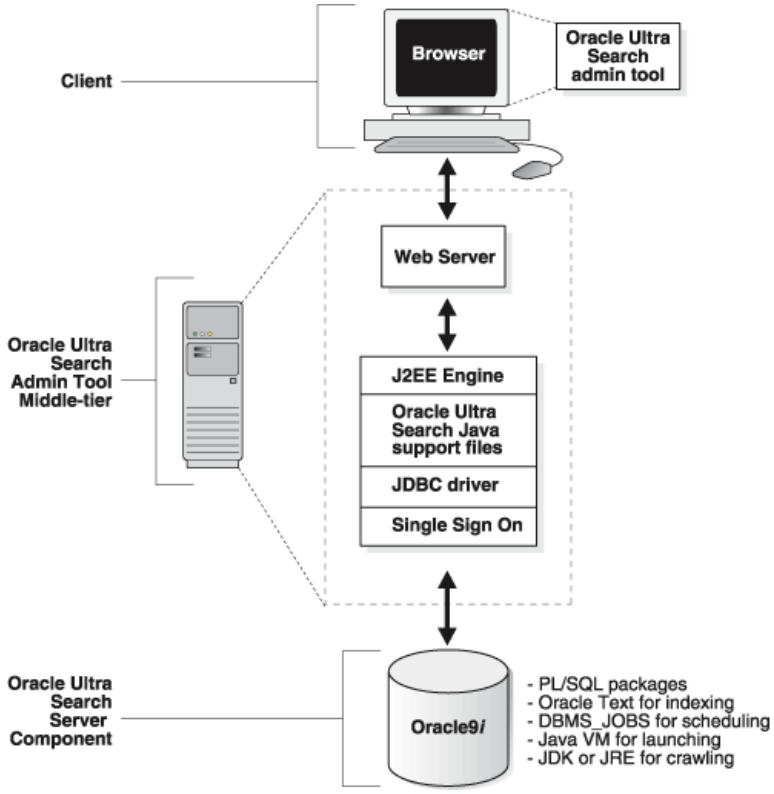
Web Application Concepts

The Ultra Search administration tool and the Ultra Search JSP query applications are Java Server Page Web applications. These are three tier architecture applications. Figure 2–1 shows the relationship between the browser interface (the first tier), the Web server and the JSP engine (the middle tier), and the Oracle database (the third tier). Typically, you install JSP applications on the same host as your Web server. The Ultra Search JSP applications consist of Ultra Search Java libraries, JSP files, and the `ultrasearch.properties` configuration file. This file specifies which database the JSP applications connect to.

Your Web server accepts requests from the browser and forwards all administration tool JSP page requests to the JSP engine for processing. To store and retrieve crawler configuration data, the JSP engine communicates with the Oracle database through the JDBC driver, as in Figure 2–1.

To invoke the Ultra Search administration tool or the Ultra Search JSP query applications, enter a URL that invokes those Web applications. The URLs are described in the following section. Use the Ultra Search administration tool to define Ultra Search instances, manage users, configure and schedule the Ultra Search crawler, and set query options. Use the Ultra Search JSP query applications to search for documents gathered and indexed by the Ultra Search server component.

Figure 2–1 Ultra Search Architecture



Browser Requirements

To use the administration tool, your browser must be Netscape version 4.0 or Microsoft Internet Explorer version 4.0 or higher.

Installing the Middle Tier Component

Installing with the Oracle Database

There are two ways to configure the Ultra Search middle tier component:

Option 1 Install Ultra Search middle tier component into an Oracle home with Oracle HTTP Server preinstalled. If the Oracle HTTP server is preinstalled, then the

installer can automatically configure it to run the Ultra Search middle tier component. To obtain the Oracle HTTP server, choose the "Server" install option from the Oracle Universal Installer (OUI) menu. Then, perform a custom install and choose the Oracle HTTP server for installation.

Option 2 Install without Oracle HTTP server preinstalled.

If the Oracle HTTP Server is not preinstalled, then you must manually perform all configuration work that the installer does. If you use a different Web server, then you must also manually configure the Web server accordingly.

Choose the installation option. Start up OUI on the relevant host. Choose to install the Oracle9*i* client. Make sure to choose the "Administrator Install" or the "Custom Install" option. The OUI prompts for an Oracle home directory in which to install the middle tier component. This directory is referred to as `$ORACLE_HOME`.

Installing with the Oracle Application Server

Start the OUI on the relevant host. Choose the destination Oracle home name and full path, and do the following steps:

1. Choose the option "Oracle 9iAS Application Server 9.0.2.0.0", and click next.
2. Choose the option "B. Portal and Wireless", and click next.
3. On the "Available Product Components" screen, make sure "Oracle Ultra Search Extension for EMD 9.0.2.0.0" is checked. Click next.
4. On the "Component Configuration" screen, make sure "Oracle9iAS Portal" is checked. This allows Oracle Portal Configuration Assistant to configure Oracle HTTP server and OC4J with Ultra Search. If you uncheck this option, then you must follow the instructions listed under Configuring Ultra Search Middle Tier Component with Oracle HTTP Server and OC4J to set up Oracle HTTP server and OC4J manually.
5. Continue with the installation until Oracle9*i*AS is successfully installed.

Note: If you decide to use a third party J2EE container or a servlet engine, then uncheck the option "Oracle9iAS Portal" on the "Component Configuration" screen of Oracle Installer, and see the "Deploying the Ultra Search EAR File on a Third Party Middle Tier" on page 2-20. Upon completion of this step, all middle tier component files are copied under the `$ORACLE_HOME`.

Configuring the Middle Tier Component

Installing with the Oracle Database

If you chose Option 1, then OUI automatically configures the Ultra Search middle tier component with Oracle HTTP server and Jserv. Proceed to "Editing the ultrasearch.properties File" on page 2-26.

If you chose Option 2, then you must manually perform the following steps to configure your existing Web server. The default Web server setup of Oracle 9.2 is Oracle HTTP server with Jserv. You can also deploy the Ultra Search middle tier component on Oracle HTTP server with Oracle J2EE container (OC4J). However, Oracle J2EE container is only available with Oracle9*i* Application Server. Proceed to the section corresponding to the Web server you decide to use to configure the Ultra Search middle tier component.

Installing with the Oracle Application Server

If you checked the "Oracle9iAS Portal" option on the "Component Configuration" Oracle Installer screen, then the configuration steps in the following section are automatically performed by OPCA. Proceed directly "Editing the data-sources.xml File" on page 2-24.

If not, then you must manually perform the steps under "Configuring the Middle Tier Component with Oracle HTTP Server and Jserv" on page 2-13 to configure your existing Web server.

You can also deploy Ultra Search Web applications using Oracle Enterprise Manager.

> **See Also:** *Oracle9i Database Administrator's Guide* for more information on Enterprise Manager

Configuring the Middle Tier Component with Oracle HTTP Server and Jserv

1. Oracle Universal Installer (OUI) automatically creates a Web server alias for the Ultra Search online documentation accessible through the administration tool. The Installer edits `$ORACLE_HOME/Apache/jsp/conf/ojsp.conf` to create an alias for the Ultra Search document root. The following line is added to that file:

   ```
   Alias /ultrasearch/doc/ "$ORACLE_HOME/ultrasearch/doc/"
   ```

2. OUI automatically creates a Web server alias for the Ultra Search administration tool. The Installer edits `$ORACLE_HOME/Apache/jsp/conf/ojsp.conf` to

create an alias for the Ultra Search JSP tree root. The following line is added to that file:

```
Alias /ultrasearch/admin/ "$ORACLE_HOME/ultrasearch/webapp/isearch_admin/"
```

3. OUI automatically creates a Web server alias for the Ultra Search JSP query applications and welcome page: The Installer edits `$ORACLE_HOME/Apache/jsp/conf/ojsp.conf` to create an alias for the Ultra Search JSP tree root. The following line is added to that file:

   ```
   Alias /ultrasearch/ "$ORACLE_HOME/ultrasearch/sample/"
   ```

   Note: This alias must be positioned last.

4. OUI automatically adds the Ultra Search middle tier library, the Ultra Search Java query API library, and the JGL object library to the servlet engine Java classpath. The Installer edits `$ORACLE_HOME/Apache/Jserv/etc/jserv.properties` to include those files. The following five lines are added:

   ```
   wrapper.classpath=$ORACLE_HOME/ultrasearch/lib/ultrasearch_midtier.jar
   wrapper.classpath=$ORACLE_HOME/ultrasearch/lib/ultrasearch_query.jar
   wrapper.classpath=$ORACLE_HOME/ultrasearch/lib/jgl3.1.0.jar
   wrapper.classpath=$ORACLE_HOME/lib/mail.jar
   wrapper.classpath=$ORACLE_HOME/lib/activation.jar
   ```

5. OUI automatically adds the Oracle9*i* JDBC thin driver to the servlet engine Java classpath. The Installer edits `$ORACLE_HOME/Apache/Jserv/etc/jserv.properties` to include that directory. The following three lines are added:

   ```
   wrapper.classpath=$ORACLE_HOME/jlib/uix2.jar
   wrapper.classpath=$ORACLE_HOME/jlib/share.jar
   wrapper.classpath=$ORACLE_HOME/jlib/regexp.jar
   ```

6. OUI automatically adds the Oracle9*i* JDBC thin driver to the servlet engine Java classpath. The Installer edits `$ORACLE_HOME/Apache/Jserv/etc/jserv.properties` to include that directory. The following line is added:

   ```
   wrapper.classpath=$ORACLE_HOME/jdbc/lib/classes12.zip
   ```

7. OUI automatically adds the directory containing the `ultrasearch.properties` file to the servlet engine classpath. The Installer

edits $ORACLE_HOME/Apache/Jserv/etc/jserv.properties to include that directory. The following line is added:

```
wrapper.classpath=$ORACLE_HOME/ultrasearch/webapp/config/
```

Note: On Windows NT, the path is $ORACLE_HOME/Apache/Jserv/conf/jserv.properties.

---

8. The current version of Ultra Search middle tier component requires version 2.1 or higher of servlet.jar. To configure Jserv with version 2.2 of servlet.jar, edit $ORACLE_HOME/Apache/Jserv/etc/jserv.properties file, and add the following line:

   ```
   wrapper.classpath=$ORACLE_HOME/lib/servlet.jar
   ```

9. Proceed to "Editing the ultrasearch.properties File" on page 2-26.

Configuring the Middle Tier Component with Oracle HTTP Server and OC4J

Note: This is available with *i*AS only.

To deploy Ultra Search Web applications, you must have a J2EE 1.2 container. Oracle Corporation recommends using Apache Web server and Oracle Containers for J2EE (OC4J).

See Also: "Deploying the Ultra Search EAR File on a Third Party Middle Tier" on page 2-20 if you use a third party J2EE container or servlet engine

1. For OC4J configuration, modify the following OC4J configuration files: server.xml, application.xml, and default-web-site.xml in $ORACLE_HOME/j2ee/OC4J_Portal/config/. The configuration of OC4J works with Ultra Search J2EE applications.

   See Also: OC4J documentation for more information on deploying EAR and WAR applications and for the more advanced functionality of OC4J.

- For `server.xml`, under <application-server> tag, add the following:

```
<application name="ultrasearch_admin" path="$ORACLE_
HOME/ultrasearch/webapp/ultrasearch_admin.ear" />

<application name="ultrasearch_query" path="$ORACLE_
HOME/ultrasearch/sample.ear" />

<application name="ultrasearch_portlet" path="$ORACLE_
HOME/ultrasearch/webapp/ultrasearch_portlet.ear" />
```

Note: These lines let OC4J know that it must deploy the Ultra Search EAR file, as well as define where this EAR files is. `Ultrasearch_admin.ear` contains the Ultra Search administration tool Web application. The `sample.ear` file contains the sample query JSP pages. After OC4J deploys `sample.ear`, you can see the `$ORACLE_HOME/ultrasearch/sample` directory. Use the JSPs in this directory to create you own query Web pages. For more information on this directory, see "Testing the Ultra Search JSP Sample Query Applications" on page 2-28.

- For `application.xml`, under <orion-application> tag, add the following:

```
<library path="$ORACLE_HOME/ultrasearch/lib/ultrasearch_query.jar" />
<library path="$ORACLE_HOME/ultrasearch/webapp/config" />
<library path="$ORACLE_HOME/jlib/uix2.jar" />
<library path="$ORACLE_HOME/jlib/share.jar" />
<library path="$ORACLE_HOME/jlib/regexp.jar" />
<library path="$ORACLE_HOME/lib/mail.jar" />
<library path="$ORACLE_HOME/lib/activation.jar" />
<library path="$ORACLE_HOME/lib/xmlparserv2.jar" />
<library path="$ORACLE_HOME/jdbc/lib/nls_charset12.zip" />
<library path="$ORACLE_HOME/jdbc/lib/classes12.jar" />
```

The preceding libraries are required for the Ultra Search administration tool and query Web applications to run.

Note: `$ORACLE_HOME/ultrasearch/webapp/config` contains the `ultrasearch.properties` file. For more information, see "Editing the ultrasearch.properties File" on page 2-26.

- For `default-web-site.xml`

  For <web-site> tag, set port="<ajp13 port>" and add protocol="ajp13" as a attribute for <web-site> tag. Under <web-site> tag, add the following:

```
<web-app application="ultrasearch_admin" name="admin"
root="/ultrasearch/admin" />

<web-app application="ultrasearch_query" name="query"
root="/ultrasearch/query" />

<web-app application="ultrasearch_portlet" name="query"
root="/provider/ultrasearch" />
```

  The preceding lines describe which Web application (WAR file) in the Ultra Search EAR files are deployed.

  – The application field describes the application name. It should match the application name in `server.xml`.

  – The name field describes the Web application name. This should match the WAR file name within the EAR file corresponding to the application.

  – Root specify the virtual path for this Web application. The virtual path is the path under the URL. For the administrative Web application, access it using `http://<hostname.domainname>:<HTTPport>/ultrasearch/admin/`.

  Note: The virtual path for a particular Web application is defined in three files: `default-web-site.xml`, `mod_oc4j.conf`, and `application.xml` in the `META-INF` directory of the EAR file. (The `META-INF` is created by extracting the EAR file.) You must modify the root attribute of web-app in `default-web-site.xml`, and the value enclosed by tag context-root in `application.xml` to change the virtual path point to each Web application.

2. Modify modOC4J configuration files. Add the following to `$ORACLE_HOME/Apache/Apache/conf/mod_oc4j.conf`:

```
Oc4jMount /ultrasearch/admin/* ajp13://<host>:<ajp13 port>
Oc4jMount /ultrasearch/query/* ajp13://<host>:<ajp13 port>
Oc4jMount /provider/ultrasearch/* ajp13://<host>:<ajp13 port>
```

Where <host> is the host name of the computer where the OC4J is installed, and <ajp13 port> is the port for ajp13 protocol. It must match the <ajp13 port> listed under `default-web-site.xml`, and the OC4Jmount must match the root of web-app in `default-web-site.xml`.

Note: With this modOC4J configuration, you must start the OC4J server manually with the command "java -jar oc4j.jar -config $ORACLE_HOME/j2ee/OC4J_Portal/config/server.xml" in the `$ORACLE_HOME/j2ee/home/` directory. Module OC4J offers more functionality, such as load balancing and fail over. Turn on this functionality based on your needs. See the official module OC4J documentation for the configuration of this functionality.

---

3. Add alias to Oracle HTTP Web server. Modify the `$ORACLE_HOME/Apache/jsp/conf/ojsp.conf` file to deploy the Ultra Search online help documentation and the Ultra Search welcome page:

   Alias /ultrasearch/doc/ "$ORACLE_HOME/ultrasearch/doc/" for Solaris

   Alias /ultrasearch/doc/ "$ORACLE_HOME\ultrasearch\doc/" for NT

   Alias /ultrasearch/ "$ORACLE_HOME/ultrasearch/sample/" for Solaris

   Alias /ultrasearch/ "$ORACLE_HOME\ultrasearch\sample/" for NT

   If you choose to install Ultra Search on an already existing database, then in addition to the preceding steps, the installation of Ultra Search on a customer's database is required.

   See Also: "Installing the Server Component on an Existing Database or Metadata Repository" on page 2-5

   Follow instructions in the Configuring the Administration Tool with Single Sign-On Server; otherwise proceed to "Editing the data-sources.xml File" on page 2-24.

4. Ultra Search sample pages require JDBC connections to the database as the instance owner. Due to Jserv limitations, the username, password and connection string used to create the JDBC connection are hard-coded inside the sample's JSP code. To configure the JSP to query a specific instance, edit the JSP source code, and replace the username, password, and connection string values.

All sample JSP source code are in $ORACLE_HOME/ultrasearch/sample/query directory.

The following files contain username, password, and connection string values:

- common_customize_instance.jsp (used by search.jsp)
- usearch.jsp
- tag/tsearch.jsp
- 9i/gsearch.jsp
- 9i/display.jsp
- 9i/gsearchf.jsp
- 9i/gutil.jsp
- 9i/mail.jsp

Note: The 9*i* JSP files are being deprecated. It is not necessary to configure them if you do not plan to use them.

Configuring the Administration Tool with Single Sign-On Server

Note: This is available with *i*AS only.

To configure the Ultra Search administration tool with Oracle single sign-on (SSO) server, you must also follow the steps in the following section in addition to the configuration in "Configuring the Middle Tier Component with Oracle HTTP Server and OC4J" on page 2-15.

1. For OC4J configuration, modify the following OC4J configuration files: application.xml, and default-web-site.xml in $ORACLE_HOME/j2ee/OC4J_Portal/config/.

   - For application.xml, under <orion-application> tag, add the following:

```
<library path="$ORACLE_HOME/jlib/repository.jar" />
<library path="$ORACLE_HOME/jlib/jndi.jar" />
<library path="$ORACLE_HOME/jlib/ldapjclnt9.jar" />
<library path="$ORACLE_HOME/j2ee/home/jazn.jar" />
<library path="$ORACLE_HOME/j2ee/home/jaas.jar" />
```

- For `default-web-site.xml`, under <web-site> tag, add the following:

```
<web-app application="ultrasearch_admin" name="admin"
root="/ultrasearch/admin_sso" />
```

2. Modify modOC4J configuration files. Add the following to `mod_oc4j.conf`:
`Oc4jMount /ultrasearch/admin_sso/* ajp13://<host>:<ajp13 port>`
3. Modify Oracle HTTP server configuration file `httpd.conf` under `$ORACLE_HOME/Apache/Apache/conf` directory, add the following lines:

```
<IfModule mod_osso.c>
<Location /ultrasearch/admin_sso>
require valid-user
authType Basic
</Location>
</IfModule>
```

Access URL `http://<hostname.domainname>:<HTTPport>/ultrasearch/admin_sso/` to log on to Ultra Search administration tool using SSO authentication.

Proceed to "Editing the data-sources.xml File" on page 2-24.

Deploying the Ultra Search EAR File on a Third Party Middle Tier

Because Ultra Search EAR files contain only Web applications (WAR files), they can be made to deploy on any J2EE 1.2 container. To do so, you must know the Ultra Search WAR file name, the predefined URL root, and the Java library required. The following section explains the Ultra Search EAR files that you need to know to deploy in a standard J2EE 1.2 container. It does not contain information on the configuration of each J2EE 1.2 container.

See Also:

- The documentation of the third party J2EE container for its configuration.
- "Configuring the Middle Tier Component" on page 2-13 for the exact configuration of Oracle HTTP server and Oracle J2EE container

Deploying the Administration Tool Ultra Search administration tool is a Web application contained in `$ORACLE_HOME/ultrasearch/webapp/ultrasearch_admin.ear`

file. The following is the file structure of ultrasearch_admin. Extract the archived file by running the command "jar -xvf ultrasearch_admin.ear".

```
ultrasearch_admin.ear
            META-INF/
                      application.xml
                      orion-application.xml
            admin.war
            admin_sso.war
            ultrasearch_midtier.jar
            jgl3.1.0.jar
```

Note: admin.war is the Ultra Search administration tool, and admin_sso.war is the Ultra Search administration tool with SSO (single sign-on) support. Both ultrasearch_midtier.jar and jgl3.1.0.jar are Java libraries used by admin.war and admin_sso.war. They are not EJBs. The context root for admin.war and admin_sso.war are /ultrasearch/admin and /ultrasearch/admin_sso. They are defined in META-INF/application.xml.

The following is the structure of admin.war and admin_sso.war. They contain the same set of files. You can see it by running the command "jar -xvf admin.war" and "jar -xvf admin_sso.war".

```
admin_sso.war and admin.war
            WEB-INF/
                      lib/
                      web.xml
            index.jsp
            */*.jsp
```

To make Ultra Search EAR file and WAR file J2EE 1.2 compliant, you must move the ultrasearch_midtier.jar and jgl3.1.0.jar to the WEB-INF/lib/ directory of admin.war and admin_sso.war. Then, rejar admin.war, admin_sso.war, and ultrasearch_admin.ear using the command "jar -cvf <war file or EAR file name> <files to jar>".

Note: The new admin.war file created is a servlet 2.2 standard Web application. Deploy admin.war alone using any servlet 2.2 engine.

The following Java libraries are needed for Ultra Search administration tool. They should be added to the library path of J2EE container or the servlet engine used to deploy the Ultra Search administration tool.

```
$ORACLE_HOME/ultrasearch/webapp/config
$ORACLE_HOME/jlib/uix2.jar
$ORACLE_HOME/jlib/share.jar
$ORACLE_HOME/jlib/regexp.jar
$ORACLE_HOME/jdbc/lib/nls_charset12.zip
$ORACLE_HOME/jdbc/lib/classes12.jar
$ORACLE_HOME/lib/xmlparserv2.jar
```

To configure Ultra Search with SSO, you must have the Oracle *i*AS infrastructure installed and Oracle Internet Directory turned on. In addition to this library path, the following library paths are also needed:

```
$ORACLE_HOME/jlib/repository.jar
$ORACLE_HOME/jlib/jndi.jar
$ORACLE_HOME/jlib/ldapjclnt9.jar
$ORACLE_HOME/j2ee/home/jazn.jar
$ORACLE_HOME/j2ee/home/jaas.jar
```

Deploying the Sample Query Applications Ultra Search sample query applications are Web applications contained in the `$ORACLE_HOME/ultrasearch/sample.ear` file. This file is already compliant to the J2EE 1.2 standard. You should not have to change this file to deploy it.

The following is the file structure of `sample.ear`. Extract the archived file by running the command "jar -xvf sample.ear".

```
sample.ear
            META-INF/
                      application.xml
            query.war
            agent/
            index.html
```

All the query JSP pages are contained in `query.war`. This file is a servlet 2.2 compliant Web application. Deploy it alone with any servlet 2.2 engine. The context root for `query.war` is `/ultrasearch/query`. It is defined in the `META-INF/application.xml` of the `sample.ear` file. You can change it by editing this file.

The following are the Java libraries needed for Ultra Search sample query application:

```
$ORACLE_HOME/ultrasearch/webapp/config
$ORACLE_HOME/jdbc/lib/classes12.jar
$ORACLE_HOME/jdbc/lib/nls_charset12.zip
$ORACLE_HOME/lib/xmlparserv2.jar
$ORACLE_HOME/lib/activation.jar
$ORACLE_HOME/lib/mail.jar
```

Ultra Search query applications also use the connection pooling functionality of J2EE container. You must define a container authenticated data source. This data source must return an Oracle connection. Oracle Corporation recommends using the Java class equal to `oracle.jdbc.pool.OracleConnectionCacheImpl` for this data source.

In addition, the data source should contain the field location equal to jdbc/UltraSearchPooledDS, username, password equal to the Ultra Search instance owner's database username, and password and URL equal to the JDBC connection string in the form of "jdbc:oracle:thin:@<database host>:<oracle port>:<oracle sid>".

> **See Also:** "Editing the data-sources.xml File" on page 2-24 for the data source configuration of the Oracle J2EE container

Deploying the Ultra Search Portlet Ultra Search Portlet is a Web application contained in the `$ORACLE_HOME/ultrasearch/webapp/ultrasearch_portlet.ear` file. This file is compliant to the J2EE 1.2 standard. This file is similar to `sample.ear` in terms of file structure. Extract the archived file by running the command "jar -xvf ultrasearch_portlet.ear".

```
ultrasearch_portlet.ear
            META-INF/
                      application.xml
            query.war
            agent/
            index.html
```

All the query JSP pages are contained in `query.war`. This file is a servlet 2.2 compliant Web application. You can deploy it alone with any servlet 2.2 engine. The context root for `query.war` is `/provider/ultrasearch/`. It is defined in the `META-INF/application.xml` of the `ultrasearch_portlet.ear` file. You can change it by editing this file.

The following Java libraries are needed for the Ultra Search Portlet:

```
$ORACLE_HOME/jdbc/lib/classes12.jar
$ORACLE_HOME/jdbc/lib/nls_charset12.zip
$ORACLE_HOME/lib/xmlparserv2.jar
$ORACLE_HOME/lib/activation.jar
$ORACLE_HOME/lib/mail.jar
```

Ultra Search Portlet uses the connection pooling functionality of J2EE container. You must define a container authenticated data source. This data source must return an Oracle connection. Oracle Corporation recommends using the Java class equal to `oracle.jdbc.pool.OracleConnectionCacheImpl` for this data source.

In addition, the data source should contain the field location equal to jdbc/UltraSearchPooledDS, username, password equal to the Ultra Search instance owner's database username, and password and URL equal to the JDBC connection string in the form of "jdbc:oracle:thin:@<database host>:<oracle port>:<oracle sid>".

> **See Also:** "Editing the data-sources.xml File" on page 2-24 for the data source configuration of Oracle J2EE container

Editing the data-sources.xml File

Note: This is available with *i*AS only.

Caution: Storage of clear database schema passwords in `data-sources.xml` is insecure. For secure storage, you must implement your own solution.

The Ultra Search *i*AS 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`. 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>"
/>
```

where <username> and <password> are the Ultra Search instance owner's database username and password, <database_host> is the host name of the back end database machine, <oracle_port> is the port to the user's Oracle database, and <oracle_sid> is the SID of the user's Oracle database. In addition to username, password, and JDBC URL, `data-sources.xml` also allows configuration of the connection cache size, as well as 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:5521:isearch"
   min-connections="3"
   max-connections="30"
   inactivity-timeout="30">
 <property name="cacheScheme"  value="1"/>
</data-source>
```

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)))"`

There are three types of caching schemes:

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

See Also: *Oracle9iAS Containers for J2EE Services Guide*

Editing the ultrasearch.properties File

The `ultrasearch.properties` file specifies which database the Web application and JSP applications connect to. Edit `$ORACLE_HOME/ultrasearch/webapp/config/ultrasearch.properties` to specify the hostname, port, and SID of the Oracle instance and listener.

To do this, edit the line that begins with "connection.url" to read: "connection.url=jdbc:oracle:thin:<hostname>:<port>:<SID>"

- For hostname, enter the full host name of the Oracle base instance running Ultra Search.
- For port, enter the listener port number for the Oracle9*i* database instance.
- For SID, enter the Oracle9*i* database instance ID.

Here is an example `connection.url` string:

```
connection.url=jdbc:oracle:thin:@ultrasearch.us.oracle.com:1521:myInstance
```

If you chose to configure the Ultra Search middle tier component with Oracle HTTP Server and Jserv, then you must also edit the line that begins with "admin.srchome" to read: "admin.srchome=<jsp_src_home>".

- For Solaris, enter "admin.srchome=*$ORACLE_HOME*/ultrasearch/webapp/isearch_admin"
- For NT, enter "admin.srchome=*$ORACLE_HOME*\\ultrasearch\\webapp\\isearch_admin"

Where *$ORACLE_HOME* is the Oracle home directory where the Ultra Search server component is installed.

This is the location of the Ultra Search administration tool JSP pages.

Note: You should not need to change the first line, because it is the name of the Oracle JDBC driver.

See Also: Your documentation on the Oracle JDBC driver

Starting the Web Server

With the *i*AS release only, if you allow Oracle Portal Configuration Assistant to configure the Web server for you, then you should start the Web server using the Enterprise Manager console.

> **See Also:** *Oracle9iAS Administrator's Guide* for information on the Enterprise Manager console

With the database release, or if you choose to configure the Oracle HTTP server and OC4J manually, do the following:

- Start Oracle HTTP server by invoking `$ORACLE_HOME/Apache/Apache/bin/spachectl stop` followed by `$ORACLE_HOME/Apache/Apache/bin/spachectl start`.
- The default HTTP port is 7777. Change this by editing `$ORACLE_HOME/Apache/Apache/conf/httpd.conf`
- To test if the Oracle HTTP server is started, visit the Ultra Search welcome page: `http://<hostname.domainname>:<HTTPport>/ultrasearch/index.html`. This page provides general information about Ultra Search, and it also contains links to the Ultra Search administration tool, as well as Ultra Search sample query JSP page. However, the Ultra Search administration tool, Ultra Search sample query page, and the directory `$ORACLE_HOME/ultrasearch/sample` are not available after OC4J is started.
- If you deploy Ultra Search middle tier with OC4J, start OC4J by invoking `java -jar $ORACLE_HOME/j2ee/home/oc4j.jar -config $ORACLE_HOME/j2ee/OC4J_Portal/config/server.xml`
- Configuration can be made for OC4J to start automatically when Oracle HTTP server starts. See the OC4J documentation for more information.

Testing the Ultra Search Administration Tool

Check that the Web Server is running.

Test your changes by attempting to log on to the administration tool:

- Visit: `http://<hostname.domainname>:<HTTPport>/ultrasearch/admin/index.jsp`

  where <hostname.domainname> is the full name of the host where you have installed the Ultra Search middle tier component, and <HTTPport> is the default Web server port. If you are running the Web browser on the same host, then enter "localhost".
- During the installation of the Ultra Search server component, you should have created a new Ultra Search instance owner. The instance owner is created in Step 4 of the Ultra Search server component installation process. Log on to the

Ultra Search administration tool by entering the Ultra Search instance owner's database username and password.

- The nature of JSP pages is such that the first time any page is accessed, it takes a few seconds to compile. Subsequent accesses are much faster.
- If you log on to the Ultra Search administration tool successfully, then you have completed the Ultra Search administration tool configuration process.

Testing the Ultra Search JSP Sample Query Applications

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

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

- Visit `http://<hostname.domainname>:<HTTPport>/ultrasearch/query/search.jsp`
- Follow the links in the Ultra Search welcome page: `http://<hostname.domainname>:<HTTPport>/ultrasearch/index.html`

> **See Also:** "Configuring the Middle Tier Component with Oracle HTTP Server and OC4J" on page 2-15 for information about configuring the JSP to query a specific instance

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

The root query directory is `$ORACLE_HOME/ultrasearch/sample/query/`.

The URL root for the query is `http://<hostname.domainname>:<HTTPport>/ultrasearch/query/`.

The 9*i*AS query (query sample JSP pages that use the 9*i*AS query API and include `usearch.jsp` and `search.jsp`) is in `$ORACLE_HOME/ultrasearch/sample/query/`.

The URL root for the 9*i*AS query is in `http://<hostname.domainname>:<HTTPport>/ultrasearch/query/`.

(For example: `access search.jsp` with `http://<hostname.domainname>:<HTTPport>/ultrasearch/query/search.jsp`.)

The 9*i* query (query JSP that uses the 9*i* query API and includes `gsearch.jsp`) is in `$ORACLE_HOME/ultrasearch/sample/query/9i/`.

The URL root for the 9*i* query is in `http://<hostname.domainname>:<HTTPport>/ultrasearch/query/9i/`.

Portlet is in `$ORACLE_HOME/ultrasearch/sample/query/portlet/`.

The URL root for Portlet is in `http://<hostname.domainname>:<HTTPport>/ultrasearch/query/portlet/`.

Taglib is in `$ORACLE_HOME/ultrasearch/sample/query/tag/`.

The URL root for taglib is in `http://<hostname.domainname>:<HTTPport>/ultrasearch/query/tag/`

Installing the Server Component on Remote Crawler Hosts

The 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 Ultra Search database.

Installing the Server Component on Remote Crawler Hosts

The Ultra Search remote crawler is part of the Ultra Search server component. Therefore, the installation procedure is the same as installing the Ultra Search server component.

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

If you choose not to install the Oracle HTTP server during the Oracle9*i* Application Server (9*i*AS) installation, then you must perform the following steps manually for remote crawling:

- Locate `$REMOTE_ORACLE_HOME/ultrasearch/tools/remotecrawler/scripts/unix/define_env` on a UNIX system or `$REMOTE_ORACLE_`

HOME/ultrasearch/tools/remotecrawler/scripts/winnt/define_env.bat on a Windows NT system.

- 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 Server Component on Remote Crawler Hosts

The only configuration needed for an Ultra Search remote crawler host is to register the host with the Ultra Search system. The registration process is done by running a SQL script on the Ultra Search remote crawler host. The SQL script connects over SQL*Plus to the 9*i*AS middle tier and registers the remote crawler host.

1. Locate the correct Oracle home.

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

2. Locate the WKSYS superuser password.

   You must run the registration script as the WKSYS superuser or as a database user that has been granted the WKADMIN role.

3. Start SQL*Plus.

   The SQL script is located in /ultrasearch/tools/remotecrawler/scripts/common/register.sql under $REMOTE_ORACLE_HOME.

   You must now run 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 platforms, 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 Ultra Search database. To do this, you might need to configure an Oracle Net service setting for the Ultra Search database.

> **See Also:** *Oracle9i Net Services Administrator's Guide* for information on how to configure a service setting

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

4. Invoke the registration script.

   Starting up SQL*Plus as the WKSYS superuser and enter the following:

   ```
   @<full path name of registration script>
   ```

   For example, if value for $REMOTE_ORACLE_HOME on a UNIX host is /home/oracle9i, then enter the following at the SQL*Plus prompt:

   ```
   @/home/oracle9i/ultrasearch/tools/remotecrawler/scripts/unix/register.sql
   ```

   If you are running SQL*Plus on Windows NT, and $REMOTE_ORACLE_HOME is in d:\Oracle\Oracle9i, then enter the following at the SQL*Plus prompt:

   ```
   @d:\Oracle\Oracle9i\ultrasearch\tools\remotecrawler\scripts\winnt\register.sql
   ```

   The registration script prompts you for two variables. The following is a list of the variables and their descriptions:

   REMOTE_CRAWLER_HOSTNAME: The DNS host name of the remote crawler host.

   ORACLE_HOME: The Oracle home located in Step 1. For example, /u01/oracle9i on a UNIX host or D:/u01/oracle9i on a Windows NT host. (Be careful to use forward slashes for Windows NT hosts.)

   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.

5. 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, log on to the Ultra Search administration tool. Click the Remote Crawler Profiles subtab in the Crawler tab. You should see the host name of the remote crawler host you have just registered in the

remote crawler Profile List. Click Edit to complete the configuration process for the remote crawler profile.

What To Do if You Make a Mistake

If you enter any wrong values for the `register.sql` script, then you must unregister the remote crawler using the `unregister.sql` script. Invoke the unregister script the same way as you invoke 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.

3

Post-Installation and Upgrade Information

This chapter contains the following topics:

Configuring the Oracle Server for Ultra Search

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

This document lists the necessary steps for the following:

- Configuring the Oracle server for Ultra Search
- Creating tablespaces and users for each Ultra Search instance

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 10Mb or more.

The following section lists some quick 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:

- *Oracle9i Database Performance Tuning Guide and Reference*
- *Oracle9i 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 10m;
ALTER DATABASE ADD LOGFILE '<redo log directory>/newredo2.log' size 10m;
ALTER DATABASE ADD LOGFILE '<redo log directory>/newredo3.log' size 10m;
```

   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 actions of transactions, primarily before they are committed. Oracle refers to these records collectively as undo. The undo space created by the Oracle Installer is likely to be too small.

Historically, Oracle has used rollback segments to store undo. Oracle now offers another method of storing undo that eliminates the complexities of managing rollback segment space, and enables DBAs to exert control over how long undo is retained before being overwritten. This method uses an undo tablespace.

Oracle Corporation recommends that you use automatic undo management and increase the undo space using an UNDO_TABLESPACE.

> **See Also:** *Oracle9i Database Administrator's Guide* for details on using automatic undo management

Tune the Oracle Initialization Parameters in the init<SID>.ora File

PROCESSES: Increase this to 50 or more.

SORT_AREA_SIZE: Increase this to 5MB or more.

SORT_AREA_RETAINED_SIZE: Increase this to 5MB or more.

JOB_QUEUE_PROCESSES: Increase this to three. (Set it to at least one.) This is needed because the 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 Ultra Search crawler will fail.

For the latest information on initialization parameters, see the Ultra Search Readme.

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

The starter database created by the Oracle Installer most likely creates 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 a raw tablespace.

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:** *Oracle9i Database Administrator's Guide* for information on how to create a temporary tablespace

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

For each Ultra Search instance, you must create a tablespace large enough for containing all data obtained during the crawling and indexing processes. This amount is naturally 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 later, Ultra Search is able to resume crawling after you have added more datafiles to the instance tablespace.

Pay attention to the STORAGE clause in your CREATE TABLESPACE statement. The amount of data to be stored in the tablespace can potentially 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 raw tablespaces.

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

See Also:

- *Oracle9i SQL Reference* for more information on creating tablespaces and managing storage settings
- *Oracle9i Database Administrator's Guide* for information on how to create a tablespace

Step 4: Create and Configure New Database Users for Each Ultra Search Instance

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

Important: The Ultra Search system requires that each 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 Ultra Search instance.

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

You must grant certain roles and privileges to each Ultra Search user. For convenience, the WKUSER role has all necessary privileges. The instance owner must have been granted the WKUSER role.

See Also: "Users Page" on page 5-35

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 Ultra Search instance owner]

[and <password> = password of the Ultra Search instance owner]

[and <default_tbs> = default tablespace for the 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 Ultra Search superuser can create an Ultra Search instance on this user schema.

If you want this user to have the general administrative privilege or the superuser privilege of Ultra Search, log in as an Ultra Search superuser or WKSYS and click the Users tab to grant the appropriate privilege.

For the database release: After the database is installed, all the user schema accounts are locked. To login as user WKSYS, unlock WKSYS by running the following statements as the SYSTEM or SYS database user:

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

For the *i*AS release: After the infrastructure database is installed, all the user schema passwords are randomized. To login as user WKSYS, change the WKSYS schema password by running the following statement as the SYSTEM or SYS database user:

```
ALTER USER WKSYS IDENTIFIED BY <password>;
```

Step 5: Gather Statistics for the Tables

If you notice performance degradation on the crawler, it might be because statistics have not been gathered for the tables. You should gather statistics for the following tables: WK$URL, WK$DOC, and DRWKDOC_PATH_IDX$I. Statistics for the WK$URL table are the most important. You must regularly gather statistics, because statistics are used by the cost-based optimizer to generate the best execution plan. Make sure that the crawler is not running during the performance tuning period to avoid interference.

Use the DBMS_STATS PL/SQL package or the ANALYZE procedure to gather statistics. The DBMS_STATS package can be run on either the table level or the schema level. Running on the schema level computes the statistics for the all the objects in the schema including the tables and indexes. Oracle Corporation recommends using the DBMS_STATS package.

Connect to the schema owning the Ultra Search instance. For example:

```
EXEC DBMS_STATS.GATHER_TABLE_STATS('<schema_name>', '<table_name>', null, DBMS_
STATS.AUTO_SAMPLE_SIZE);

EXEC DBMS_STATS.GATHER_SCHEMA_STATS('<schema_name>', DBMS_STATS.AUTO_SAMPLE_
SIZE);
```

or

```
ANALYZE TABLE <table_name> ESTIMATE STATISTICS SAMPLE 20 percent;
```

where <schema_name> is the owner of the Ultra Search instance and <table_name> is the table you want to gather statistics for (for example, wk$url).

Occasionally rebuilding the B-tree indexes could also improve performance by freeing disk space. For example:

```
ALTER INDEX <index_name> REBUILD;
```

where <index_name> with the index that you want to rebuild.

To get a list of the indexes, run the following statement:

```
SELECT index_name FROM user_indexes WHEREindex_type='NORMAL';
```

Step 6: Alter the Index Preferences

This step is optional.

An empty index is created when an 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 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*

Managing Stoplists

Every 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 Ultra Search Stoplist

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

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

Modifying Instance Stoplists

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.

Modifications made to the default stoplist are reflected in all other instance stoplists created after the time of modification.

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

In both cases, the result is that the Ultra Search instance stoplist is modified and defined before initial crawling. This means that all documents collected by the 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 again.

Modifying Instance Stoplists Before Initial Crawling

1. Modifying the default stoplist before creating the instance:

   To add the stopword "web" to the default stoplist, log in 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 in 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 in 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 in 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');
   ```

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 operation is not an expensive operation.

   To add the stopword "web" to the instance stoplist, log in 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 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.

Upgrading to the Most Recent Ultra Search Release

Ultra Search supports the following upgrades:

- Upgrade from Ultra Search 1.0.3 (Oracle9i Database 9.0.1) to 9.0.3
- Upgrade from Ultra Search 9.0.2 to 9.0.3
- Upgrade from Ultra Search 9.2 to 9.0.3

Upgrade is based on the server component only. Upgrade on the middle tier is not supported. Install the 9.0.3 middle tier in a separate Oracle Home.

> **See Also:** "What's New in Ultra Search?"describes the Ultra Search release numbering

Upgrade from Ultra Search 1.0.3 (Oracle9*i* Database 9.0.1) to 9.0.3

To upgrade Ultra Search 1.0.3 to 9.0.3, perform the following steps:

1. Launch Oracle Portal Configuration Assistant (OPCA) in 'ALL' mode, and it invokes Ultra Search Configuration Assistant (USCA). USCA checks the Ultra Search release in the existing database.
2. If USCA detects that the Ultra Search release is 1.0.3, then it asks you to select one of the following three options:

- Deinstall / reinstall
- Migrate
- Abort

3. Select the option "Migrate". You must run the upgrade script and perform some manual steps.

   The Ultra Search upgrade script first verifies the version of the current system, then upgrades the system and migrates user data. User data includes all dictionary and table data, such as information about the metadata, data sources, mappings, crawler schedules, authentication, and query statistics.

   All crawler schedules and jobs created in the older version are disabled before data and system migration. **When migration is complete, the system administrator should re-activate the crawling schedule to re-index the document.** You do not need to reconfigure the system or re-enter any data. Users can still query documents that were crawled and indexed by the previous version.

   See Also: "Installing the Server Component on an Existing Database or Metadata Repository" on page 2-5

Ultra Search Migration Approaches

There are two approaches to migrate user data: the in-place approach and the ETL (extract-transform-load) approach. With the in-place approach, the current `ORACLE_HOME` is used. With the ETL approach, a new `ORACLE_HOME` is created.

Ultra Search In-Place Migration In-place migration upgrades existing configurations and user data to the latest Ultra Search release. Upgraded files are left in place, and the source installation is modified. The benefit to this approach is that it might conserve disk space. With the in-place approach, data migration involves the following six steps:

1. Back up user data
2. Deinstall previous database objects
3. Install new database objects
4. Re-create user instances
5. Restore data
6. Rebuild index

Use the SQL script `wk0upgrade.sql` to run the in-place migration steps one through five, listed in the preceding section. The script is located in the `%ULTRASEARCH_HOME%/admin/` directory. It requires the following input parameters:

- `SYSPW`: password of the user `SYS`
- `WKSYSPW`: password of the user `WKSYS`
- `HOST`: database host machine
- `PORT`: database port number
- `ORACLE_SID`: database SID
- `WK_TABLESPACE`: tablespace for Ultra Search
- `WK_TEMPTABLESPACE`: temporary tablespace
- `CONN_STRING`: database connect string
- `ORACLE_HOME`: the path of Oracle home
- `JAVA_EXE_PATH`: Java executable file path
- `PATH_SEPARATOR`: Java classpath separator; use ':' for UNIX or ';' for Windows NT

The sixth step requires the system administrator to re-activate all crawling schedules through the Ultra Search administration tool.

Ultra Search Extract-Transform-Load Migration Extract-transform-load (ETL) migration extracts the useful subset of configuration data from the source installation, transforms necessary data, and loads or merges this data into a new installation of Ultra Search. This approach might require more disk space, but it offers the following benefits:

- No destabilization of the source installation
- Stability of target installation
- No installer integration requirement

With the ETL approach, data migration involves the following five steps:

1. Install the new system (for example, 9.0.3) in a new ORACLE_HOME
2. Re-create user instance schemas and related database objects
3. Re-create user instances
4. Restore data
5. Rebuild index

The first two steps in the ETL approach must be done manually:

- Install Ultra Search 9.0.3 in a separate ORACLE_HOME, either on the same machine or on a different machine. If the new 9.0.3 system is installed in the same machine as the old 9.0.1 system, then the database listener port number should be configured to a different number than the old 9.0.1 database. This lets both the old and the new database run at the same time.
- Re-create all Ultra Search 1.0.3 user instance schemas in the new database. Also, for each table data source created in Ultra Search 1.0.3, if the base table is located in the local database, then you must copy the base table to the new database. If the table data source base table is set to a remote database table, then you must to re-create the database link from the new database to the remote database.

Use the SQL script wk0migrate.sql to run the ETL migration steps three and four. The script is located in the %ULTRASEARCH_HOME%/admin/ directory. It requires the following input parameters:

- WKSYSPW: password of the user WKSYS
- CONN_STRING: database connect string
- SRC_WKSYSPW: password of the source database (9.0.1 database) user WKSYS
- SRC_CONN_STRING: source database connect string

The fifth step requires the system administrator to re-activate all crawling schedules through the Ultra Search administration tool.

Note: The upgrade script does not roll back the Ultra Search system to the old version if an unexpected error occurs, such as a power failure or system failure. For in-place migration, back up the database before starting migration. For ETL migration, because all previous data is kept, you can switch back to the previous (for example, 9.0.1) system

Ultra Search Migration Logs

The upgrade script provides log files to show which actions the migration has taken. The upgrade script writes the following contents to the log file:

- The current execution step
- Any error message raised from the stored procedures
- Number of data records backup
- Number of data records copied or migrated

For in-place migration, the `wk0upgrade.sql` script writes the execution logs to the file `wk0upgrade.log` in the `%ULTRASEARCH_HOME%/admin/` directory.

For ETL migration, the `wk0migrate.sql` script writes the execution logs to the file `wk0migrate.log` in the `%ULTRASEARCH_HOME%/admin/` directory.

Upgrade from Ultra Search 9.0.2 to 9.0.3

To upgrade Ultra Search 9.0.2 to 9.0.3, perform the following steps:

1. Launch Oracle Portal Configuration Assistant (OPCA) in 'ALL' mode, and it invokes Ultra Search Configuration Assistant (USCA). USCA checks the Ultra Search release in the existing database.
2. If USCA detects that the Ultra Search release is 9.0.2, then it asks you to select one of the following three options:
   - Deinstall / reinstall
   - Upgrade to 9.0.3
   - Abort
3. Select the option "Upgrade to 9.0.3". USCA upgrades the existing Ultra Search server components to release 9.0.3 automatically. There is no manual step required, and all data collected with the existing Ultra Search remains intact.

4. After USCA finishes the upgrade, the Ultra Search 9.0.3 server components are installed into the existing database.

Upgrade from Ultra Search 9.2 to 9.0.3

Because Ultra Search 9.2 uses the same database schema as Ultra Search 9.0.2, the upgrade procedure is the same.

See Also: "Upgrade from Ultra Search 9.0.2 to 9.0.3" on page 3-14

4

Understanding the Ultra Search Crawler and Data Sources

This chapter contains the following topics:

See Also: Appendix B, "Tuning Query Performance"

Ultra Search Crawler Overview

The 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 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. Some parameters, like the log file directory and the temporary directory, have no default value, so you must set them before crawling. To do so, use the Crawler Settings Page in the administration tool.

See Also: "Crawler Page" on page 5-11

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
- Oracle9*i*AS 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, 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 Ultra Search crawler, which enqueues it for later crawling. For more information on defining a new data source type, see the User-Defined sub-tab 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 Ultra Search index is kept up to date with existing information in the associated data sources. To define a synchronization schedule, use the Schedules 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 end 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 5-19

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 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.

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 attributes of different names but used for the same purpose; for example, "version" and "revision". It can also have the same attribute name for different purposes; for example, "language" as in natural language in one data source but as "programming language" in another.

Search attributes are created in three 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 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 4–1 and Figure 4–2 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 4–1 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 to 5.

Figure 4–2 illustrates how the crawler caches Web pages. This figure correspond to Steps 6 to 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. Figure 4–1.
2. Crawler initiates multiple crawling threads.
3. Crawler thread removes the next URL in the queue.
4. Crawler thread fetches the document from the Web. The document is usually an HTML file containing text and hypertext links.
5. 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. Crawler caches the HTML file in the local file system. Figure 4–2.
7. Crawler registers URL in the document table.
8. Crawler thread starts over by repeating Step 3.

Fetching a document, as shown 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 is still there. Cleanup is done through index optimization, which can be scheduled separately.

Figure 4–1 Queuing URLs

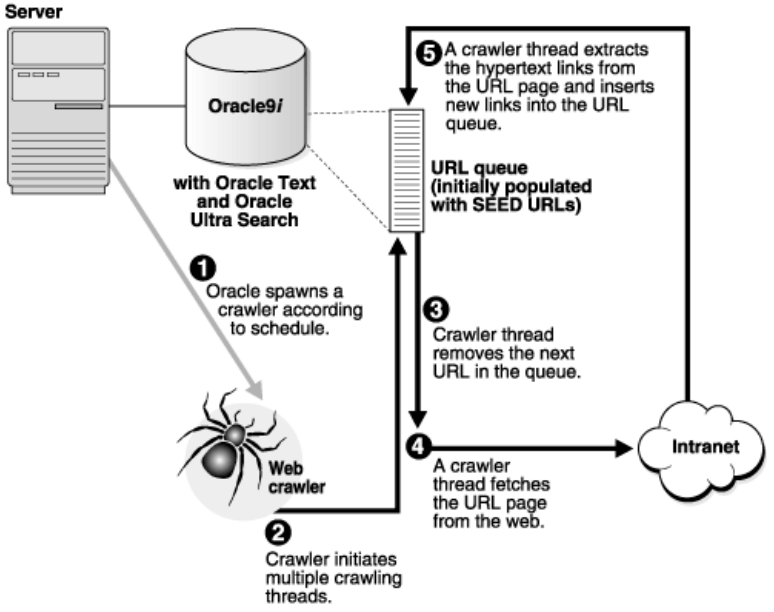
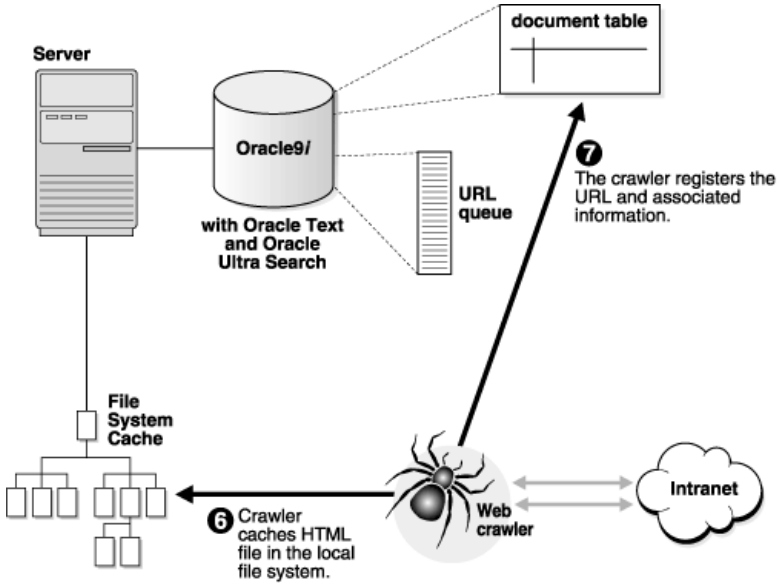


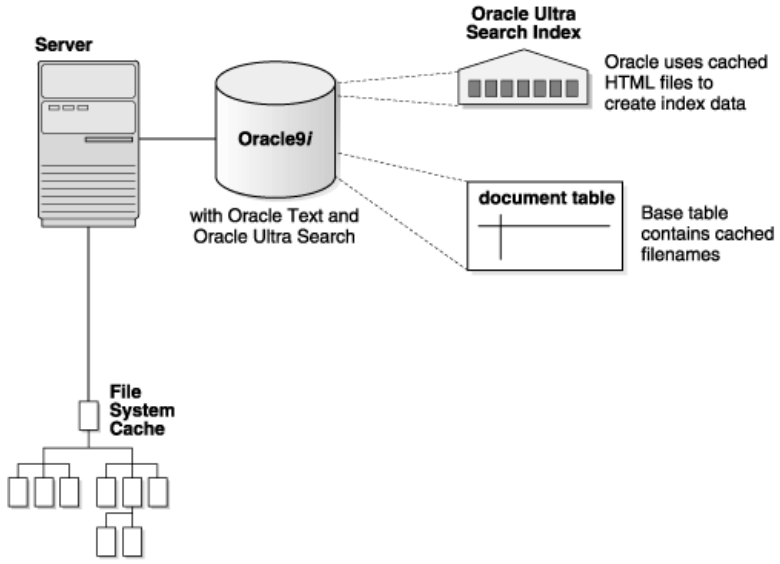
Figure 4–2 Caching URLs



Indexing Documents

When the file system cache is full (default maximum size is 20 megabytes), document caching stops and indexing begins. In this phase, Ultra Search augments the Oracle Text index using the cached files referred to by the document table. See Figure 4–3.

Figure 4–3 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 so 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. Crawler initiates multiple crawling threads.
3. Crawler thread removes the next URL in the queue.
4. Crawler thread fetches the document from the Web. The page is usually an HTML file containing text and hypertext links.

5. 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 crawler goes to step 3. Otherwise, the crawler moves to the next step.
6. Crawler thread scans the document for hypertext links and inserts new links into the URL queue. Duplicate links already in the document table are discarded.
7. Crawler caches the document in the local file system. See Figure 4–2.
8. Crawler registers 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 by repeating Step 3.

Ultra Search Remote Crawler

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

To configure a remote crawler, you must first install the Ultra Search middle tier components module on a machine other than the database host. During installation, the remote crawler is registered with the Ultra Search system, and a profile is created for the remote crawler. After installing the Ultra Search middle tier components module, you must log on to the 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.

Caution: When launching a remote crawler, the Ultra Search back end database communicates with the remote machine through Java RMI (remote method invocation). By default, RMI sends data over the network unencrypted. Using the remote crawler to perform crawling introduces a potential security risk. A malicious entity within the enterprise could steal the Ultra Search instance schema and password by listening to packets going across the network. Refrain from using the remote crawler feature if this security risk is unacceptable.

5

Understanding the Ultra Search Administration Tool

The Ultra Search administration tool lets you configure and schedule the Ultra Search crawler. This chapter contains the following topics:

- Ultra Search Administration Tool
- Logging On to Ultra Search
- Logging On and Managing Instances as SSO Users
- Instances Page
- Crawler Page
- Web Access Page
- Attributes Page
- Sources Page
- Schedules Page
- Queries Page
- Users Page
- Globalization Page

> **See Also:** The Oracle Ultra Search online help for detailed information about using the Ultra Search administration tool

Ultra Search Administration Tool

The administration tool is a Web application for configuring and scheduling the Ultra Search crawler. The administration tool is typically installed on the same machine as your Web server. You can access the administration tool from any browser in your intranet, directly as an Ultra Search database user, or as a single sign-on (SSO) user with a SSO server.

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

With the administration tool, you use to do the following:

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

See Also: Chapter 2, "Installing and Configuring Ultra Search" for information about how to deploy the administration tool

Setting Crawler Parameters

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

- Set crawler parameters, such as the number of crawler threads. 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 crawler data sources. Data sources can be Web pages, database tables, files, email mailing lists, Oracle9*i* Application Server (9*i*AS) Portals, 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. On this 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 a crawler synchronization schedule. The crawler uses the synchronization schedule to reconcile the 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 to document attributes and data groups.

Attributes

Search attributes can be mapped to HTML metatags, table columns, document attributes, and email headers. Some attributes, such as author and description, are predefined and need no configuration. However, you can customize your own attributes. To set custom document 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

Ultra Search provides context-sensitive online help, based on the language setting in the Users Page. If the translated help files are not installed on the local machine, then English online help files are used.

To download the latest online help files, visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at `http://technet.oracle.com/membership/index.htm`.

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at `http://technet.oracle.com/docs/index.htm`.

Logging On to Ultra Search

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

- Users managed by the Oracle Internet Directory (OID). These are authenticated by the single-sign-on server (SSO)
- Users existing in the database on which Ultra Search lives (non-SSO mode) [applicable in iAS]
- Enterprise Manager `IAS_ADMIN` users [applicable in iAS]

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

- For non-SSO mode: `http://<hostname>:<port>/ultrasearch/admin/index.jsp`
- For SSO 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 `IAS_ADMIN` Enterprise Manager user [applicable in *i*AS]
- The `PORTAL` SSO user belonging to the default company [applicable in *i*AS]
- The `ORCLADMIN` SSO user belonging to the default company [applicable in *i*AS]

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

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

See Also:

- Chapter 2, "Installing and Configuring Ultra Search"
- "Instances Page" on page 5-7 for more information about creating Ultra Search instances
- "Users Page" on page 5-35 for more information about granting permission to other users
- "Logging On and Managing Instances as SSO Users" on page 5-5 for more information about how Ultra Search handles SSO users

Logging On and Managing Instances as SSO Users

Note: Single sign-on (SSO) is available only with the Oracle9*i* Application Server (9*i*AS) release. It is not available with the Oracle9*i* database release.

Logging On to Ultra Search through Oracle Portal

When single sign-on (SSO) users log in to the SSO-protected Ultra Search administration tool through the Oracle Portal administration page, one of the following occurs:

- If the SSO user has been granted super-user privileges in Ultra Search, then the Ultra Search administration tool presents a pull-down list allowing the SSO user to select an instance to manage.
- If the SSO user does not have super-user privileges, but has been explicitly granted permission to manage one or more Ultra Search instances, then the Ultra Search administration tool also presents a pull-down list allowing the SSO user to select an instance to manage.
- If the SSO user does not have super-user privileges and does not have privileges to manage any Ultra Search instances, then the Ultra Search administration tool displays an error message indicating that the user has no privileges and that they should contact the appropriate authority to be granted privileges.

Granting privileges to SSO users

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

Note: An SSO user is uniquely identified by Ultra Search with an SSO-nickname / subscriber-nickname combination.

- In non-hosted mode, the subscriber-nickname is not required when granting privileges to an SSO user. This is because there is exactly one subscriber in Oracle Portal in non-hosted mode.
- In hosted mode, the subscriber-nickname is required when granting privileges to an SSO user. This is because there can be more than one subscriber in Oracle Portal, and two or more users with the same SSO-nickname (for example, PORTAL) could be distinct SSO users distinguished by their subscriber-nickname. When running 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 Ultra Search.
  - When logging in to SSO 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 Ultra Search.

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

See Also: *Hosting Developer's Guide* at `http://otn.oracle.com/`.

Instances Page

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

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

Creating an Instance

To create an instance, select the Create tab on the Instances Page. This takes you to another page with links for creating a regular instance (a master instance) and creating a read-only snapshot instance. Only Ultra Search super-users can create new instances.

Note: If the search domains of Ultra Search instances overlap, then there could be crawling conflict for table data sources with logging enabled, email 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 Ultra Search instance exists in one and only one database user / schema. To create a new Ultra Search instance, you first must have a database user that has been configured for Ultra Search and that does *not* already contain an Ultra Search instance.

   The database user you create to house the 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 Ultra Search" on page 3-2 for information and instructions on configuring database users for Ultra Search
- "Creating a Snapshot Instance" on page 5-9
- *Oracle9i Database Administrator's Guide* for details on using transportable tablespaces

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

   From the main instance creation page, select the "Create instance" link, 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 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. Try to 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 3-8

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. Also, after the master instances 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:

- Query Processing

  Two 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 gets 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 reenter 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 that has been configured for Ultra Search and that does *not* already contain an Ultra Search instance.

2. Copy the data from the master instance.

   This is done with the transportable tablespace mechanism, which does not allow renaming of tablespaces. Therefore, snapshot instances 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 Ultra Search administration tool.

   From the main instance creation page, select the "Create read-only snapshot instance" link, 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

After providing this information, click Apply.

See Also:

- "Configuring the Oracle Server for Ultra Search" on page 3-2 for information and instructions on configuring database users for Ultra Search
- *Oracle9i Database Administrator's Guide* for details on using transportable tablespaces

Selecting an Instance

You can have multiple Ultra Search instances. For example, an organization could have separate 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 the Select tab 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 the Delete tab on the Instances Page.
2. Select an instance from the pull-down menu.
3. Click Apply.

Note: To delete an Ultra Search instance, the user must be assigned the `WKADMIN` role.

Editing an Instance

To edit an instance, click the Edit tab on the Instances Page. You can change the instance mode (make it 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 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 Ultra Search crawler is a Java application that spawns threads to crawl defined data sources, such as Web sites, database tables, or email archives. Crawling occurs at regularly scheduled intervals, as defined in the Schedules Page.

With this page, you can do the following:

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 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 Ultra Search crawler specify the language. For documents with no language specification, the 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 (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 (Chinese, Japanese, Korean, and so on.).

The crawler determine 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. Multi-lexer is the only lexer used for 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 when a Web document does not have a specified language, 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:

- English
- Brazilian portuguese
- Danish
- Dutch
- French
- German
- Italian
- Japanese
- Korean
- Portuguese
- Simplified Chinese
- Spanish
- Swedish
- Traditional Chinese

Crawling Depth

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

> **See Also:** Appendix A, "Tuning the Web Crawling Process" for more information on the importance of the crawling depth

Crawler Timeout Threshold

Specify in seconds a crawler timeout. 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.

Temporary Directory Location and Size

Specify a temporary directory and size. The crawler uses the temporary directory for intermittent storage during indexing. Specify the absolute path of the temporary directory. The size is the maximum temporary space in megabytes that will be used by the crawler.

The size of the temporary directory is important because it affects index fragmentation. The smaller the size, the more fragmented the index. As a result, the query will be slower, and index optimization needs to be performed more frequently. Increasing the directory size reduces index fragmentation, but it also reduces crawling throughput (total number of documents crawled each hour). This is because it takes longer to index a bigger temporary directory, and the crawler needs to wait for the indexing to complete before it can continue writing new documents to the directory.

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.

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=5521)...))"
```

See Also: *Oracle9i JDBC Developer's Guide and Reference*

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 Ultra Search crawler on a remote machine other than the Ultra Search database. A remote crawler profile is identified by the hostname. The profile includes the cache, log, and mail directories that the remote crawler shares with the database machine.

To set these parameters, click Edit. Enter the shared directory paths as seen by the remote crawler. You must ensure that these directories are shared or mounted appropriately.

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

Use this page to set up basic authentication and proxies.

URL Authentication

The Ultra Search crawler provides basic authentication information to hosts that require it. Basic authentication is based on the model that the client must authenticate itself with a username and a password for each realm. A realm is a string that identifies a set of protected URLs on a Web server. Enter the host, realm, username, and password, and click Add.

Proxies

Specify a proxy server if the search space includes Web pages that reside outside your organization's firewall. Specifying a proxy server is optional. Currently, only the HTTP protocol is supported.

Note: The crawler cannot use a proxy server that requires proxy authentication.

You can also set domain exceptions.

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 could be multiple translations of the attribute display name.

To define a search attribute, use the Search Attributes subtab. Ultra Search provides some system-defined attributes, such as "Author" and "Description." 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: Ultra Search provides a command-line tool to load metadata, such as search attribute LOVs and display names into an 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 E, "Loading Metadata into Ultra Search".

Search Attributes

Search attributes are attributes exposed to the query user. Ultra Search provides system-defined attributes, such as "Author" and "Description." 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.

Update Policy

To update the policy value, click the Manage LOV icon for any attribute.

A data source-specific LOV can be updated in three ways:

1. 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 user-defined 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 which global search attribute the document attribute is mapped to.

For Web 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. You should avoid this step, unless necessary.

Note: After you define a search attribute mapping, you cannot remove that mapping.

Note: There are no user-managed mappings for email sources. There are two predefined mappings for emails. The "From" field of an email is intrinsically mapped to the Ultra Search "Author" attribute. Likewise, the "Subject" field of an email is mapped to the Ultra Search "Subject" attribute. The abstract of the email 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 email inbox. The Ultra Search crawler retrieves data from one or more data sources.

The different types of sources are:

- Web Sources
- Table Sources
- Email Sources
- File Sources
- User-Defined Sources (requires a crawler agent)
- Oracle9iAS Portal Sources

See Also:

- "Schedules Page" on page 5-27 to assign one or more data sources to a synchronization schedule
- "Queries Page" on page 5-32 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 HTML content on a specific Web site. Web sources differ from other data source types because they exist specifically to 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.
2. Override the default crawler settings for each Web source. This step is optional. You can change the crawling depth, the character set, the language, the number of crawler threads, and the crawler timeout threshold. You can also enable or disable robots exclusion, language detection, and the UrlRewriter. You can also edit those in Edit Web Sources. If you change any of the default settings, click Update.

   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 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.

   The URL Rewriter is a user-supplied java module for implementing the 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.

See Also:

- "Ultra Search URL Rewriter API" on page 6-13
- "Display URL and Access URL" on page 4-3
- "Crawler Page" on page 5-11 for information on default languages

3. Enter a starting address. This is the URL for the crawler to begin crawling.
4. Set URL boundary rules to refine the crawling space. You can include or exclude hosts and URL paths. For example, an inclusion domain of oracle.com limits the Ultra Search crawler to hosts belonging to Oracle Corporation worldwide. (This is a suffix inclusion, so anything ending with oracle.com is crawled; however, http:/ / www.oracle.com.tw is not crawled.) An exclusion domain uk.oracle.com prevents the crawler from crawling Oracle hosts in the United Kingdom. You cannot include or exclude port numbers. In other words, you cannot crawl www.acme.com:7777 but not www.acme.com:8888. Use UrlRewriter for this. Exclusion rules always override inclusion rules.
5. Specify the types of documents the Ultra Search crawler should process for this source. HTML and plain text are default document types that the crawler will always process.
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.

Table Sources

A table source represents content in a database table or view. The database table or view can reside in the Ultra Search database instance or in a remote database. Ultra Search accesses remote databases using database links.

See Also: "Limitations With Database Links" on page 5-23

Creating Table Sources

To create a table source, click Create new table source, and follow these steps:

1. Specify a table source name, and the name of the database link, schema, and table. 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 5-11.
3. Verify the information about your table source.
4. Decide whether or not to use the Ultra Search logging mechanism to optimize the crawling of table data sources. With this logging mechanism, 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 document attributes. Each table column can be mapped to exactly one document 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. Ultra Search uses a default text viewer for table data sources. If you specify display URL, then Ultra Search uses the Web URL defined to display the table data retrieved. If display URL column is available, then 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.

   The Table Column to Key Mappings section provides mapping information. 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 Ultra Search attributes. Do not map the text column.

See Also: *Oracle9i SQL Reference* for more on format models

Editing Table Sources

Click Edit to change the name of the table source; 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. 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 comprised of a join of one or more tables
  - A view 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 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 or view name>;
  ```

- The base table or view cannot have text columns of type `BFILE`.

Email Sources

An email source derives its content from emails sent to a specific email address. When the Ultra Search crawler searches an email source, it collects all emails that have the specific email address in any of the "To:" or "Cc:" email header fields.

The most popular application of an email source is where an email source represents all emails sent to a mailing list. In such a scenario, multiple email sources are defined where each email source represents an email list.

To crawl email sources, you need an IMAP account. At present, the Ultra Search crawler can only crawl one IMAP account. Therefore, all emails 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 email account and subsequently crawled. The Ultra Search crawler is IMAP4 compliant.

When the Ultra Search crawler retrieves an email message, it deletes the email message from the IMAP server. Then, it converts the email message content to HTML and temporarily stores that HTML in the cache directory for indexing. Next, the Ultra Search crawler stores all retrieved messages in a directory known as the archive directory. The email files stored in this directory are displayed to the search end-user when referenced by a query hit.

To crawl email sources, you must specify the username and password of the email account on the IMAP server. Also specify the IMAP server hostname and the archive directory.

Creating Email Sources

To create email sources, you must enter an email address and a description. The description can be viewed by all search end-users, so you should specify a short but meaningful name. When you create (register) an email source, the name you use is the email of the mailing list. If the emails are not sent to one of the registered mailing lists, then those emails are not crawled.

You can specify email address aliases for an email source. Specifying an alias for an email source causes all emails sent to the main email address, as well as the alias address, to be gathered by the crawler.

File Sources

A file source is the set of documents that can be accessed through the file protocol on the Ultra Search database machine or on a remote crawler 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.
2. Designate files or directories to be crawled. If a URL represents a single file, then the 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.

4. Specify the types of documents the Ultra Search crawler should process for this file source. HTML and plain text are default document types that the crawler will always process.

Ultra Search displays file data sources in text format by default. However, if you specify display URL for the file data source, then 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. 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 / archive / <sub_dir_name> / <file_ name> and the display URL is https:/ / host:7777 / private / <sub_dir_name> / <file_ name>, then you can specify the file URL prefix to file:/ / / home / archive and the display URL prefix to https:/ / host:7777 / private.

User-Defined Sources

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 Ultra Search crawler, which enques it for later crawling.

See Also: "Ultra Search Crawler Agent API" on page 6-3

To define a new data source, you first define a data source type to represent it. You define the type name, the crawler agent java class / jar file, and parameters to be

used, such as starting address. After you define your data type, define a new data source by specifying parameter values.

Creating User-Defined Data Sources

To create a new user-defined data source, click Create new source. To create, edit, or delete data source types, click Manage types.

To create a user-defined data source type:

1. Specify data source type name, description, and crawler agent Java class file or jar file name. The crawler agent Java class path 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 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 need to 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.

To create a user-defined data source:

1. Specify a name, data source type, and default language for the data source. Each data source is created based on data source type definition. For information on default languages, see "Crawler Page" on page 5-11.
2. Enter parameter values, such as starting point.
3. 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, specify it here. The crawler picks up attributes that have been returned by the crawler agent or specified here.

Edit user-defined data sources by changing the name, type, default language, or starting address.

Oracle9*i*AS Portal Sources

Ultra Search supports the crawling and indexing of Oracle9*i* Application Server (9*i*AS) Portal installations. This enables searching across multiple portal

installations. To crawl a 9*i*AS Portal, you must first register your portal with Ultra Search. To register your portal:

1. Select a name and portal URL base for the portal source. After it is created, the portal URL base is not updatable. For information on default languages, see "Crawler Page" on page 5-11.
2. Click Register Portal. Ultra Search attempts to contact the Oracle 9*i*AS Portal instance and retrieve information about it.

After registering your portal, select the Oracle 9*i*AS Portal page groups you want to index. Each page group chosen is created as a 9*i*AS portal source.

You can edit the types of documents the Ultra Search crawler should process for a portal source. HTML and plain text are default document types that the crawler will always process. Edit document types by clicking the edit icon of the portal source after it has been created.

See Also: The Oracle 9*i*AS Portal documentation.

Schedules Page

Use this page to schedule data synchronization and index optimization. Data synchronization means keeping the 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 4-3

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. Select a schedule frequency and determine whether the schedule should automatically accept all URLs for indexing or examine URLs before indexing. You can also associate the schedule with a remote crawler profile.
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. 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, executing, scheduled, and so on.

After you click the link for a specific host, you see list of document URLs that have been crawled for the host. You can delete document URLs in this section.

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, a user wants to crawl only text and HTML on a Web site. Later, the user also wants to crawl Microsoft word and Adobe PDF documents. The user must modify the document types for the source, edit the schedule to select "process all documents," then reexecute 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. For example, a new mimetype document is added, new host/path inclusion rules are added, or crawling depth increases (for example, from 5 to 7). Every document must be reparsed to pick up links that had been dropped before due to depth limit.

Note: "Process all documents" does not help when the crawling scope has been narrowed. For example, if crawling depth was reduced from 7 to 5, the PDF mimetype was deleted, or a host inclusion was removed, then the user must remove the affected documents manually in a SQL*Plus session.

- 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.
  - 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 document that have changed" and "index only" re-indexes only changed documents.

Launching Synchronization Schedules

You can launch a synchronization schedule in the following ways:

- Set a schedule frequency and wait for the predetermined launch time.
- Execute it immediately. To do so, click Status, then Execute immediately.

Note: Launching a synchronization schedule could 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(s) to be crawled by the schedule are copied over into a queue table. Depending on the number of URLs associated with that data source, the copy operation can potentially take a long time. The administration tool displays the schedule state as 'Launching' during the entire 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 the statistics icon.

The crawling progress contains the following information:

- 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 is the sum of "Document non-indexable", "Document conversion failure", and "Documents indexed"
- Document fetch failures: This could be an HTTP server timeout or another HTTP server error
- Documents rejected: The document is not within the URL boundary rule.
- Documents discovered: This is the sum of "Documents to fetch", "Documents fetched", "Document fetch failures", and "Documents rejected"
- Documents indexed
- Documents non-indexable: This could be a file directory, a portal page that is a discovery node, or a robot metatag that says no index.
- Document conversion failures: The binary file filter failed.

Index Optimization

Index Optimization

To ensure fast query results, the Ultra Search crawler maintains an active index of all documents crawled over all data sources. This page 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 temporary 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 will not exceed this limit, but it could be shorter. Specifying a

longer optimization time will result in a more optimized index. Alternatively, you can specify that the optimization continue until it is finished.

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 source 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 to search from. Use this page to define these data groups.

A data group consists of one or more data sources. 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 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 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. It also makes them easier to find.

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 hostname 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 will get an error message on this page if no Web data source is defined.

Note: Ultra Search provides a command-line tool to load metadata, such as document relevance boosting, into an 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 E, "Loading Metadata into 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 that occurred 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 made 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 that occurred 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 end-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: Appendix B, "Tuning Query Performance"

Users Page

Use this page to manage Ultra Search administrative users. You can assign a user to manage an Ultra Search instance. You can also select a language preference.

Preferences

This section lets you set preference options for the 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.

To grant super-user administrative privileges to another user, specify the user name and type. 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 this page. Instance owners must have been granted the `WKUSER` role.

Granting general administrative privileges to a user allows that user to modify general settings for this instance. To do this, specify the user name and type. Specify also 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 usernames from the list of current administrators and click Remove.

General administrative privileges do not include the ability to:

- Create an instance
- Delete an instance

These privileges belong to super-users.

> **See Also:** "Step 4: Create and Configure New Database Users for Each Ultra Search Instance" on page 3-5

Globalization Page

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. The pull-down menu lists the language options.

Data Group Name

This section lets you translate data group display names to different languages. The pull-down menu lists the language options.

6

Ultra Search Developer's Guide and API Reference

This chapter explains the Ultra Search APIs and related information. This chapter contains the following topics:

- Overview of Ultra Search APIs
- Ultra Search Query API
- Ultra Search Crawler Agent API
- Ultra Search Java Email API
- Ultra Search URL Rewriter API
- Ultra Search Sample Query Applications
- Ultra Search Query Tag Library
- Customizing the Query Syntax Expansion

See Also: *Oracle Ultra Search Javadoc*

Overview of Ultra Search APIs

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 email API is used by the Ultra Search query application to display emails. 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.

Ultra Search also includes highly functional query applications to query and display search results. The query applications are based on Java Server Pages (JSP) and work with any JSP1.1 compliant engine.

Ultra Search Query API

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 Java server pages (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:

- Basic search form
- Advanced search form
- Query result display
- Help page
- Feedback page
- Register URL

You embed Ultra Search query functionality in your Web application with the supplied 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 methods that retrieve HTML code support features such as allowing you to embed query input boxes and result lists in your Web application. 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.

Some features of the Ultra Search Java query API:

- Lets you to retrieve query results
- Lets you to set query properties, such as the total number of hits to return, and so on.
- Lets you to set the query session language
- Lets you to access Ultra Search tables to retrieve Ultra Search dictionary data, such as all defined data groups and attributes
- Lets you 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.
- Lets you allow the search end user to submit URLs to the seed URL list
- The Ultra Search Java query API is encapsulated in the `oracle.ultrasearch.query` package.

See Also: Appendix B, "Tuning Query Performance"

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 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 Ultra Search crawler, which enqueues it for later crawling. The crawler agent must be implemented in Java using the Ultra Search crawler agent API.

Ultra Search provides a sample implementation of user-defined crawler agents using the 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. This agent performs the following task:

- Reads data source parameters
- Connects to the database that contains the data source
- Initializes fetching document URL and attributes from the data source
- Fetches document URL and attributes from the data source
- Disconnects 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 attribute
- Maps the document attribute to a common attribute name used by end users
- Provides a "flatten" view of the data source such that document is 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, the agent invocation process is repeated with the original last crawl time stamp. If the crawler already finished enqueueing URLs fetched from the agent and is half way through 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.
- 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.
- Deletes the URLs that are still marked for garbage collection.
- 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 can be also created such that the list of collected attributes are configurable. Ultra search automatically

registers attributes returned by the agent. The agent can decide which attributes to return for a document.

Crawler Agent Functionality

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 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 Ultra Search server component, *not* the middle tier component, 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 max.) and a description (4000 bytes maximum). By default, a parameter is not encrypted.
- Encryption: Should the value of this parameter be encrypted when stored.

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.

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 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 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 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 what document attributes Ultra Search should keep. Any attribute not defined in Ultra Search is automatically registered.
- 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 need to 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/sample/`.

UrlData: The crawler agent uses this interface to populate document properties and attribute values. 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/sample` directory. You can directly view the sample agent source code using your preferred text editor.

There is a `sample_agent_readme.htm` file and a `SampleAgent.java` file. This is for the sample crawler agent implementation using agent APIs.

Setting up the Sample Crawler Agent

Compiling and Building the Agent Jar File

The Java source code for the sample agent must be first 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 Ultra Search server component, *not* the middle tier component, is installed.

The classes needed for compilation are the JDK class (`classes.zip`), Oracle JDBC thin driver (`classes12.zip`), and `ultraserach.jar`. For example:

```
   javac -J-ms16m -J-mx96m -O -classpath /jdk1.2.2
05/lib/classes.zip:/lib/classes12.zip:
   $ORACLE_HOME/ultrasearch/lib/ultrasearch.jar SampleAgent.java
```

To build the `sampleAgent.jar` file:

```
  /jdk1.2.2_05/bin/jar cv0f /oracle/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/`

Ultra Search Java Email API

Ultra Search provides a Java API for accessing archived emails. The API is used by the Ultra Search query application to display emails addressed to mailing lists that have been indexed by the 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 the look-and-feel can be completely customized to your needs.

Email documents contain valuable information, but they are not structured to find specific relevant information easily. Ultra Search lets you retrieve and index emails on a server that supports the IMAP4 protocol.

An email source is a data source that derives its content from emails sent to a specific email address. When the Ultra Search crawler searches an email source, the crawler collects all emails that have the specific email address in any of the "To:" or "Cc:" email header fields.

Note: Ultra Search stores copies of all retrieved emails in the local file system of the Ultra Search server installation.

A possible application of an email source is where an email source represents all emails sent to a mailing list. In such a scenario, multiple email sources are defined where each email source represents an email list.

Ultra Search email crawling and rendering is built on top of the JavaMail API using Sun Microsystems' reference implementation of JavaMail. This enables Ultra Search to provide a Java API for accessing indexed emails. The API is known as the Ultra Search Java Email API. This API lets you retrieve information such as email header information, email body content, and attachments of an email.

Use this API to embed Ultra Search email browsing functionality into Java server page (JSP) or servlet-based Web applications. Ultra Search ships a fully functional JSP Web application that directly uses this API to render indexed emails. Because the source code is viewable, you can use it as an example for building your own customized email browser.

JavaMail Implementation

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 Ultra Search.

Java Email API

The Ultra Search Java Email API is encapsulated in the `oracle.ultrasearch.query` package.

Sample Mailing List Browser Application Files

The sample mailing list browser applications files are located in the `$ORACLE_HOME/ultrasearch/sample/query` directory. You can directly view the sample mailing list browser application source code using your preferred text editor.

The following tables describe all sample mailing list browser application files:

README File and Stylesheets:

File	Description
`README.html`	Readme
`mail.css`	Style sheet for sample email Web application

Sample Java Server 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 email sources (mailing lists) of an Ultra Search instance
`mailmsgs.jsp`	JSP page that displays all emails for an email source (mailing list)
`mailreader.jsp`	JSP page that displays an email
`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`	Ultra Search banner
`images/wsd.gif`	Background image used in sample query application

Setting up the Sample Mailing List Browser Application

For detailed instructions on setting up the sample JSP mailing list browser application, see "Installing the Ultra Search Middle Tier on Web Server Hosts" on page 2-9.

Ultra Search URL Rewriter API

A URL rewriter is a user supplied Java module that implements the 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 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

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 is endless, 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 hit display. This is the URL used when users click the search hit 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 sample rewriter, `SampleRewriter.java`, is available for reference under `$ORACLE_HOME/ultrasearch/sample/`.
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 5-20 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 Javadoc* for the API (`oracle.ultrasearch.crawler` package)
- The sample URL rewriter `SampleRewriter.java` under `$ORACLE_HOME/ultrasearch/sample/`
- "Web Sources" on page 5-20

Ultra Search Sample Query Applications

Ultra Search provides several sample query applications and a sample crawler agent. Use the sample query applications as examples for creating your own query

application. The query applications are written as Java server page (JSP) applications. Your query application will use the Ultra Search query API. You can also use the sample crawler agent to create your own crawler agent.

Note: Pointers to the sample query applications and the sample crawler agent Java source code, as well as their corresponding readmes, are in the Ultra Search welcome page: `http://<hostname.domainname>:<HTTPport>/ultrasearch/index.html`

The sample 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 email 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 sample query applications include a sample search portlet. The sample Ultra Search portlet demonstrates how to write a search portlet for use in Oracle 9*i*AS Portal.

When the user issues a query in any of the query applications, a hit list containing query results is returned. The user can select a document to view from the hit list. A hit list can include HTML documents, files, database table content, archived emails, or Oracle 9*i*AS items. The Ultra Search sample query applications also incorporate an email browser for reading and browsing emails.

The Ultra Search administration tool and the Ultra Search sample query applications are part of the Ultra Search middle tier components module. However, the Ultra Search administration tool is independent from the Ultra Search sample query applications. Therefore, they can be hosted on different machines to enhance security or scalability.

If you do not want to use the sample query applications, you can build your own query application by directly invoking the 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 Java server page (as in the case of the provided sample query applications). For rendering emails that have been crawled and indexed, you can also directly invoke the Ultra Search Java email API methods.

Java Server Page (JSP) Sample Query Applications

The JSP sample query applications are located in the `$ORACLE_HOME/ultrasearch/sample` directory.

Java Server 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 *i*AS. The application server typically runs on a separate machine from the Oracle server for performance and scalability reasons. The Oracle server holds the Ultra Search indexes.

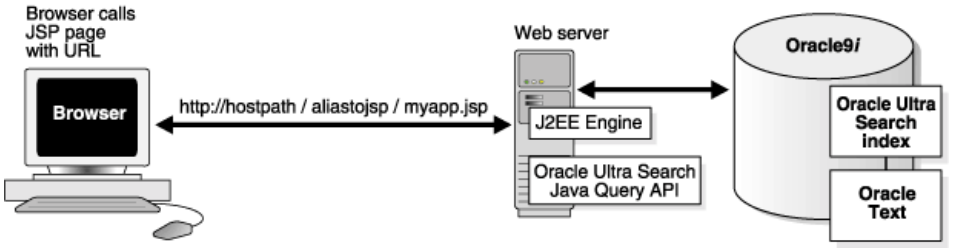
JSP applications are compiled into Java servlets at runtime. The compiled servlets run in one or more Java Virtual Machine processes. The JSP application communicates with the Oracle server through the Oracle JDBC driver.

As in any Java application, you must include the following files in your servlet engine classpath to use the Java query and email APIs:

1. `$ORACLE_HOME/ultrasearch/lib/ultrasearch_query.jar`
2. `$ORACLE_HOME/lib/mail.jar`
3. `$ORACLE_HOME/lib/activation.jar`

Figure 6–1 shows how your Web query application calls the Ultra Search Java query API.

Figure 6–1 Calling Java Server Pages



Ultra Search Query Tag Library

On top of the Java query API, Ultra Search provides a JSP tag library as an alternative for developing search applications. Based on the Sun Microsystems Java

Server Pages specification version 1.1, the 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 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

Ability to iterate through the resulting hit 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 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

Tag	Description	Attributes
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 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 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 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 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 Ultra Search Instance

This tag establishes a connection to an Ultra Search instance. Some basic parameters must be established for this tag to work, such as JDBC connection string, schema username/password, Ultra Search instance name, and so on.

Attribute Name	Description
instanceId="name"	This names the instance defined by this tag. This name is then used by other Ultra Search tags to specify the instance being searched.
username	This creates a database connection.
password	This 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 Ultra Search instance that is owned by the schema user. If the schema user owns only one Ultra Search instance, 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 email.
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 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 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 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 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

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 to search for. 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 hit.
to="number"	This specifies the index of the last hit.
boostTerm="string"	This specifies the search term that will be used for relevance boosting. This is optional.
withCount="true | false"	This specifies whether the result will have an estimate of the total hit 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 straightforwardly, 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: Meta-data 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 simply outputs the hit count to the page.

The following shows the hit 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 used 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>
```

Customizing the Query Syntax Expansion

Ultra Search uses the Oracle Text engine to index and search documents. When an end user specifies a certain query string, Ultra Search takes that string and transforms it into an Oracle Text query expression. This process is called query syntax expansion.

You can customize Ultra Search to use your own implementation of the query syntax expansion. In previous releases, the default query syntax expansion implementation was contained in the WK_QUERYEXP PL/SQL package.

The Contains query 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 'title' StringAttribute. The syntax for Contains is the same when used on the document content and on StringAttributes.

Customize this syntax by subclassing the Contains query and overriding the expand() method with your own implementation. In fact, you can implement the Query interface and ignore the provided Contains query, because the query API accepts any object that implements the Query interface.

This document describes how to customize the query syntax expansion implementation to suit your organization's preferences.

Default Query Syntax Expansion Implementation

The default query syntax expansion implementation directly affects the following:

- The way the end user enters a query string (known as the *end user query syntax*)
- The way the documents matching the query are scored (known as *scoring*)
- The way the end user's query string is transformed into an Oracle Text query string (known as the *expansion rules*)

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.

  Plus operator [+] This specifies that the token immediately following it must appear in all documents included in the search result.

  Minus operator [-] This specifies that the token immediately following it cannot appear in any document included in the search result.

  Asterisk [*] This 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 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.
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."
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 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>)

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 or 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 to 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%)))`

End User Query String	Expanded Query String Understandable by Oracle Text
[Oracle Applications]	`(((({Oracle Applications})*2,(({Oracle};{Applications})*2,({Oracle},{Application s})))) 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. To do so, you need to understand the requirements of Oracle Text queries. The details of Oracle Text queries are beyond the scope of this document.

See Also:

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

To customize Ultra Search to use your own implementation of the query syntax expansion, use the Contains query. This finds documents that contain some text within its content or its string attributes. The Contains query does not apply to date or number attributes. If no attribute is specified, then Contains operates on the document content, instead of any attribute. A match found in the title attribute of the document will have a higher score than a match in the document content.

Constructors

- Contains(StringAttribute, String, InstanceMetaData)

  ```
  public Contains(StringAttribute att, java.lang.String val, InstanceMetaData instmd)
  ```

  This constructs a contains query on a string attribute.

- Contains(String, InstanceMetaData)

  ```
  public Contains(java.lang.String val, InstanceMetaData instmd)
  ```

  This constructs a contains query on the document content.

Methods

- compile()

  ```
  public java.lang.String compile()
  ```

  This compiles into a query string.

  Specified By:

  compile() in interface Query

  Returns:

  a query string representing this query

- expand(StringAttribute, String, InstanceMetaData)

  ```
  public java.lang.String expand(StringAttribute att, java.lang.String str,
        InstanceMetaData instmd)
  ```

  This translates a user's attribute Contains query string into a text query.

  Parameters:

  att - a string attribute

  str - the main query string

  instmd - the InstanceMetaData object

  Returns:

  the translated Oracle Text query string (contains clause)

- expand(String, InstanceMetaData)

  ```
  public java.lang.String expand(java.lang.String str, InstanceMetaData
  instmd)
  ```

  This translates a user query string into a text query.

  Parameters:

  str - the main query string

  instmd - the InstanceMetaData object

  Returns:

  the translated Oracle Text query string (contains clause)

A

Tuning the Web Crawling Process

The 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 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 will 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, 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 as the crawler eventually complete analyzing 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 in both cases, 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 scenario is one example of how URL looping can occur.

Monitor the crawler statistics in the Ultra Search administration tool to get an idea of what 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, 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, 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 will take effect only after restarting the crawler.

B

Tuning Query Performance

This appendix contains suggestions on how to improve the performance of the Ultra Search query. Query performance is generally impacted by response time and throughput.

1. Tune the `DB_CACHE_SIZE` parameter.

   The database buffer cache keeps frequently accessed data read from datafiles. Efficient usage of the buffer cache can improve Ultra Search query performance. The cache size is controlled by the `DB_CACHE_SIZE` initialization parameter.

   See Also: *Oracle9i Database Performance Tuning Guide and Reference* for information on how to tune this parameter

2. Optimize the index.

   Optimize the Ultra Search index after the crawler has made substantial updates. This is done by scheduling 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 5-31

3. Optimize the index based on tokens.

   Optimize the 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 Ultra Search index name is `WK$DOC_PATH_IDX`.

   See Also: *Oracle Text Reference* for more information on `OPTIMIZE_INDEX`

4. Simplify query expansion.

   The search response time is directly influenced by the Oracle Text query string used. Although 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 6-28
   - *Oracle Ultra Search Javadoc* for the oracle.ultrasearch.query.Query interface

5. Size the shared pool.

   The shared pool stores the library cache and the dictionary cache. The library cache stores recently executed 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: *Oracle9i Database Performance Tuning Guide and Reference* for information on tuning this parameter

6. Define JDBC connection pooling.

   The 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 *i*AS, 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>
```

7. 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 Ultra Search query is WKSYS.WK_QRY.

See Also: *Oracle9i Supplied PL/SQL Packages and Types Reference*

C

Using the Remote Crawler

Without the Ultra Search remote crawler, you must run the Ultra Search crawler on the same host as the Oracle9*i* database. For large data sets, you can improve performance by running the Ultra Search crawler on one or more separate hosts from the Oracle9*i* database. Because the Ultra Search crawler is a pure Java application, it communicates with the Oracle9*i* database through JDBC.

Ultra Search remote crawler instances are always launched by the Oracle9*i* database. The Oracle9*i* database uses Java remote method invocation (RMI) to communicate with the remote crawler hosts. Therefore, each remote host must have an RMI registry and an RMI daemon running.

By launching remote crawlers from the Oracle9*i* database, you can leverage the high-availability of the Oracle9*i* database. The Ultra Search scheduling mechanism runs within the Oracle9*i* database and therefore automatically uses the database's high availability features.

Caution: By default, RMI sends data over the network unencrypted. Using the remote crawler to perform crawling introduces a potential security risk. A malicious entity within the enterprise could steal the Ultra Search instance schema and password by listening to packets going across the network. Refrain from using the remote crawler feature if this security risk is unacceptable.

Scalability and Load Balancing

Each Ultra Search schedule can be associated with exactly one crawler. The crawler can run locally on the Oracle9*i* 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 Ultra Search middle tier component 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.

How is "Launching" Achieved?

Launching is done using Java RMI technology.

1. When a crawling schedule is activated, the Ultra Search scheduler launches a Java program as a separate process on the database host. This Java program is known as the ActivationClient.
2. This program attempts to connect to the remote crawler host through the standard RMI registry and RMI daemon on ports 1098 and 1099. If successful, the ActivationClient receives a remote reference to a Java object running on the remote host. This remote Java object is known as the ActivatableCrawlerLauncher.
3. The ActivationClient then instructs the ActivatableCrawlerLauncher to launch the Ultra Search crawler on the remote host. The ActivatableCrawlerLauncher launches the Ultra Search crawler as a separate Java process on the remote host.

Installation and Configuration Sequence

1. Make sure that you have installed the Ultra Search server components on the Oracle9*i* database, the Ultra Search middle tier component on one or more Web server hosts, and the Ultra Search middle tier component on all remote crawler hosts.

   See Also: Chapter 2, "Installing and Configuring Ultra Search"

2. Export the following common resources on the database host:
   - The temporary directory
   - The log directory
   - The mail archive directory (if you are using the Ultra Search mailing list feature)

   These resources are merely directories that must be accessible by all remote crawler instances over the network. Use whatever mechanism you want to share these resources with a remote crawler host.

The remote crawler code is pure Java. Therefore, it is platform independent. For example, your Ultra Search installation might consist of four hosts: one database server (host X) running Solaris, on which the Ultra Search server component is installed; one remote crawler host (host Y1) running on Windows NT; one remote crawler host (host Y2) running on Solaris; one remote crawler host (host Y3) running on Linux.

In this scenario, you export the shared directories on host X using the UNIX "export" command. You 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 NT and use that to mount the shared directories. (Note: if host X is a Linux server, then you can create Samba shares and thereby mount those shares on Windows NT without needing any third party software).

3. Configure the remote crawler with the administration tool.

   Edit that profile by manually entering all mount points for the shared crawler resources that you defined. 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. Specify values for the following parameters:

   - Mount point for temporary directory path as seen by the remote crawler
   - Mount point for log 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 Ultra Search mailing list feature)

   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

4. Complete the crawler configuration with the administration tool.

   The minimum set of parameters that will likely need to be configured are the following:

   - Seed URLs
   - Web proxy
   - A schedule

Each schedule must be assigned to a remote crawler or the local crawler (the local crawler is the crawler that runs on the local Oracle9*i* 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, then you can enable or disable the remote crawler.

> **See Also:** Chapter 5, "Understanding the Ultra Search Administration Tool"

5. Start up the RMI registry and RMI daemon 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.
   - If the remote crawler is running on a Windows NT host, then run the `%ORACLE_HOME%\tools\remotecrawler\scripts\winnt\runall.bat` file.

   The `runall.sh` and `runall.bat` scripts perform the following tasks in sequence:

   - `define_env` is invoked to define necessary environment variables
   - `runregistry` is invoked to start up the RMI registry
   - `runrmid` is invoked to start up the RMI daemon
   - `register_stub` is invoked to register the necessary Java classes with the RMI subsystem

   You can invoke `runregistry`, `runrmid`, and `register_stub` individually. However, you must first invoke `define_env` to define the necessary environment variables.

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 remote crawler fails to launch if any one of the following requirements are not met:

- The RMI registry is not running and listening on port 1099 of each remote host.
- The RMI daemon is not running and listening on port 1098 of each remote host.
- The necessary Java objects have not been successfully registered with each RMI registry.

After a remote crawler is launched, verify that it is running by one or more of the following methods:

- 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. You should look for active Java processes.

- Monitor the contents of the schedule log file.

  If the remote crawler is running successfully, you should see the contents of the schedule log file changing periodically. The schedule log file is located in the shared log directory.

D

Table Data Source Synchronization

Ultra Search crawls database tables in the main Oracle9*i* database instance where Ultra Search is installed. Additionally, it can crawl remote databases if they have been linked to the main Oracle9*i* database. Remote databases are linked to the main Oracle9*i* instance with database links.

> **See Also:** *Oracle9i Database Administrator's Guide* for instructions on how to create database links.

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 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 Ultra Search crawler uses the change information to figure out which rows need to be recrawled. For example, a log table generated by 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'.

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 (name is generated on the fly) is created with the following structure:

Column Name	Column Type
K1	NUMBER
K2	VARCHAR2(200)

The SQL statement for creating the log table will be 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 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,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 yourself. The log table must conform to the rules for log tables described earlier. Also, they 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 Ultra Search table data source. By choosing that option, the Ultra Search administration tool prompts you for the name of the log table in the remote database. Ultra Search associates this log table with the base table. Ultra Search assumes that you have correctly performed steps 1 and 2.

E

Loading Metadata into Ultra Search

Ultra Search provides a command-line tool to load metadata into an Ultra Search database. If you have a large amount of data, 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 relevance 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.

Launching the Loading Tool

The loader program binary file is located in the following directory: `%ULTRASEARCH_HOME%/bin/MetaLoader.class`.

Your machine should have Java 1.2 compliant Java Runtime. 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 `xmlschema.jar`.
- Ultra Search Java library. The filename is `ultrasearch.jar`.

- Oracle JDBC globalization support version 1.2. The filename is `nls_charset12.zip`.

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 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 username: wk_test
- Schema password: welcome
- 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 relevance boosting scores into 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 E-5

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 Website</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 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 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 E-6

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 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>
```

F

Altering the Crawler Java Classpath

The 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 Ultra Search is installed, the default crawler classpath is stored in the database. Whenever a new 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 may be 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 that the Ultra Search Server component is installed. However, 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: Appendix C, "Using the Remote Crawler"

Altering the Crawler Java Classpath on the Ultra Search Server Host

1. Log on to the host where the Ultra Search Server component is installed. Locate the file `$ORACLE_HOME/ultrasearch/admin/wk0cpath.sql`.
2. Using SQL*Plus, run the `wk0cpath.sql` script as the `WKSYS` superuser or as a database user that has been granted the `WKADMIN` role.
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 Ultra Search instance name if you are attempting to modify an instance-specific classpath. If you are modifying the default classpath, 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, entries earlier on 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 Ultra Search Server component 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 host where the Ultra Search middle tier components are installed on a remote crawler host. If the host is a UNIX machine, locate the file `$ORACLE_HOME/ultrasearch/tools/remotecrawler/scripts/unix/define_env`. If the host is a windows NT machine, locate the file `$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. You must restart the RMI subsystem before these changes can take effect.

See Also: Appendix C, "Using the Remote Crawler" for more details on starting up the RMI subsystem

G

Customizing the Query Syntax Expansion 9.0.1

Oracle Ultra Search uses the Oracle Text engine to index and search documents. When an end 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.

Oracle Ultra Search provides a default query syntax expansion implementation The code for this implementation is contained in the `WK_QUERYEXP` PL/SQL package. It can be viewed in the `$ORACLE_HOME/ultrasearch/admin/wk0queryexp.pkb` file on the Oracle Server host.

This document describes how to customize the query syntax expansion implementation to suit your organization's preferences.

Default Query Syntax Expansion Implementation

The default query syntax expansion implementation directly affects the following.

1. The way the end user enters a query string (known as the *end user query syntax*)
2. The way the documents matching the query are scored (known as *scoring*)
3. The way the end user's query string is transformed into an Oracle Text query string (known as the *expansion rules*)

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.

Plus operator [+] This specifies that the token immediately following it must appear in all documents included in the search result.

Minus operator [-] This specifies that the token immediately following it cannot appear in any document included in the search result.

Asterisk [*] This 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.

Summary

The following table summarizes the rules for the 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.
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."

Rule	Description
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."
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

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 earlier, 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 Ultra Search uses as a default query syntax expansion implementation.

Rules

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>)
```

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 or 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 to 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. To do so, you need to understand

the requirements of Oracle Text queries. The details of Oracle Text queries are beyond the scope of this document.

See Also:

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

To customize Ultra Search to use your own implementation of the query syntax expansion, modify the WK_QUERYEXP package. In that package are two PL/SQL functions that you must edit. The functions are expand_main and expand_attr. Expand_main is applied to the query string entered by the end user. Expand_attr is applied to each search attribute specified in an advanced search. The return value of each expand_attr function is appended to the return value of the expand_main function. This resultant query string is what's given to Oracle Text to query on.

The expand_main Function

This takes the query string entered in the basic search box or advanced search box and converts it to an Oracle Text query string according to your custom query syntax expansion implementation rules.

```
CREATE OR REPLACE FUNCTION expand_main(query varchar2)
RETURN varchar2
AS
  newqry varchar2(4000);
BEGIN
  newqry := <Convert the input query string into an
             Oracle Text query string according to
             your custom rules>
  return newqry;
END;
```

The expand_attr Function

This is applied to each search attribute in an advanced search. It takes each attribute and converts it to an Oracle Text query string according to your custom query syntax expansion implementation rules.

```
CREATE OR REPLACE FUNCTION expand_attr(query varchar2)
RETURN varchar2
AS
  newqry varchar2(4000);
```

```
BEGIN
  newqry := <Convert a search attribute into an
             Oracle Text query string according to
             your custom rules>
  return newqry;
END;
```

Note: All customized functions are instance-specific and should be defined in the schema of the Ultra Search instance user.

Functions should be executed with definers-rights.

Example of Combining the expand_main Return Value with the expand_attr Values

The following example illustrates how the default query syntax expansion implementation converts the end user query string Oracle Applications to an Oracle Text compatible query string. The additional clause added by the introduction of the two search attributes is highlighted in bold.

End User Query String	Expanded Query String Understandable by Oracle Text
[Oracle Applications]	(((({Oracle Applications})*2,(({Oracle};{Applications})*2,({Oracle},{Applications}))) within TITLE__31)*2,(({Oracle Applications})*2,(({Oracle};{Applications})*2,({Oracle},{Applications}))))
[Oracle Applications] with the Title attribute restricted to "MyTitle" and the Author attribute restricted to "MyAuthor"	((((({Oracle Applications})*2,(({Oracle};{Applications})*2,({Oracle},{Applications}))) within TITLE__31)*2,(({Oracle Applications})*2,(({Oracle};{Applications})*2,({Oracle},{Applications}))))&((((MyTitle}) WITHIN TITLE__31)&(({MyAuthor}) WITHIN AUTHOR__32))*10)*10

Index