Hewlett Packard
Enterprise

HPE Query Manipulation Server

Software Version: 11.0

QMS Administration Guide

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise Development LP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HPE shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HPE required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

HPE Big Data Support provides prompt and accurate support to help you quickly and effectively resolve any issue you may encounter while using HPE Big Data products. Support services include access to the Customer Support Site (CSS) for online answers, expertise-based service by HPE Big Data support engineers, and software maintenance to ensure you have the most up-to-date technology.

To access the Customer Support Site

- go to https://customers.autonomy.com

The Customer Support Site includes:

- **Knowledge Base**. An extensive library of end user documentation, FAQs, and technical articles that is easy to navigate and search.
- **Support Cases**. A central location to create, monitor, and manage all your cases that are open with technical support.
- **Downloads**. A location to download or request products and product updates.
- **Requests**. A place to request products to download or product licenses.

To contact HPE Big Data Customer Support by email or phone

- go to http://www.autonomy.com/work/services/customer-support

Support

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, visit the Knowledge Base on the HPE Big Data Customer Support Site. To do so, go to https://customers.autonomy.com, and then click **Knowledge Base**.

The Knowledge Base contains documents in PDF and HTML format as well as collections of related documents in ZIP packages. You can view PDF and HTML documents online or download ZIP packages and open PDF documents to your computer.

Contents

Part 1: Getting Started

This section describes how to install and set up QMS.

- "Introduction"
- "Install Query Manipulation Server" on page 15
- "Run Query Manipulation Server"
- "Configure Query Manipulation Server"

Chapter 1: Introduction

This section provides an overview of QMS and the IDOL product suite.

Query Manipulation Server

QMS allows you to modify queries and results, and to manage promotions. You can use QMS to:

- generate promotions from search results.
- query for promotions.
- ensure that documents appear at desired locations in a results list.
- boost results.
- expand queries to include results from synonymous terms.
- remove terms from queries.

You can use QMS as a stand-alone component or as a part of IDOL Search Optimizer.

QMS must have access to the IDOL Server Content component and the Promotion Agentstore.

For more information on how to use QMS with IDOL Search Optimizer, refer to the *IDOL Search Optimizer User Guide*.

OEM Certification

Query Manipulation Server works in OEM licensed environments.

HPE's IDOL Platform

At the core of QMS is HPE's *Intelligent Data Operating Layer* (IDOL). IDOL gathers and processes unstructured, semi-structured, and structured information in any format from multiple repositories using IDOL connectors and a global relational index. It can automatically form a contextual understanding of the information in real time, linking disparate data sources together based on the concepts contained within them. For example, IDOL can automatically link concepts contained in an e-mail message to a recorded phone conversation, which can be associated with a stock trade. This information is then imported into a format that is easily searchable, adding advanced retrieval, collaboration, and personalization to an application that integrates the technology.

For more information on IDOL, refer to the *IDOL Getting Started Guide*.

QMS Architecture

QMS uses the Autonomy Content Infrastructure (ACI) Client API to communicate with custom applications that retrieve data by using HTTP commands. QMS communicates over HTTP by using XML, and can adhere to SOAP.

Query Manipulation Server architecture
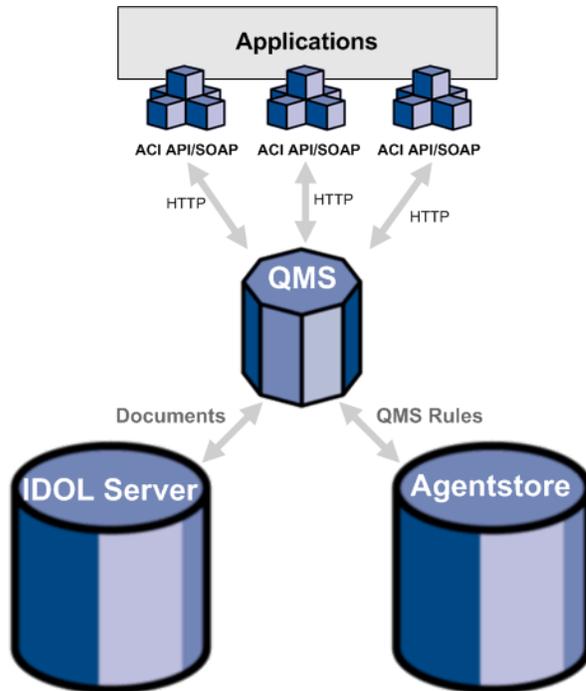
When you use QMS, you send queries to QMS rather than directly to IDOL Server. QMS forwards the queries to IDOL Server and the Promotion Agentstore component, which contains the QMS rules.

The Promotion Agentstore returns any rules that apply to the current query. QMS can then perform one of the following actions:

- modify the current query. For example, QMS can expand the query text with a list of synonyms before it sends the query to IDOL Server.
- modify the result. For example, QMS can insert a document at a particular position in the set of results returned by IDOL Server.

QMS can do either, both, or none of these actions depending on the configuration and the particular query. You can also query for promotions.

QMS can accept HTTP requests sent from a Web browser.

Related Topics

- "Send Actions to Query Manipulation Server" on page 24

Query Manipulation Server Components

An IDOL installation for query manipulation includes the following components:

- **Query Manipulation Server**. QMS manages the queries that you send to IDOL Server. It forwards queries to the IDOL Server data index, and retrieves QMS rules from the Promotion Agentstore. It then modifies the query or the results according to the rules.

- **Data Index**. QMS queries the IDOL data index to retrieve documents that match the original query. Depending on rules that you set up, QMS also retrieves promotion documents from the data index.

- **Promotion Agentstore**. The Promotion Agentstore component stores the QMS rules that you set up. QMS queries the Promotion Agentstore to return rules that apply to a query.

Chapter 2: Install Query Manipulation Server

This section describes how to install QMS by using the IDOL Server installer.

Install Query Manipulation Server on Windows

Use the following procedure to install Query Manipulation Server on Microsoft Windows operating systems, by using the IDOL Server installer.

The IDOL Server installer provides the major IDOL components. It also includes License Server, which Query Manipulation Server requires to run.

To install Query Manipulation Server

1. Double-click the appropriate installer package:

   `IDOLServer_VersionNumber_Platform.exe`

   where:

   `VersionNumber` is the product version.

   `Platform`   is your software platform.

   The Setup dialog box opens.

2. Click **Next**.

   The License Agreement dialog box opens.

3. Read the license agreement. Select **I accept the agreement**, and then click **Next**.

   The Installation Directory dialog box opens.

4. Specify the directory to install Query Manipulation Server (and optionally other components such as License Server) in. By default, the system installs on

   `C:\HewlettPackardEnterprise\IDOLServer-VersionNumber`. Click to choose another location. Click **Next**.

   The Installation Mode dialog box opens.

5. Select **Custom**, and then click **Next**.

   The License Server dialog box opens. Choose whether you have an existing License Server.

   - To use an existing License Server, click **Yes**, and then click **Next**. Specify the host and ACI port of your License Server, and then click **Next**.

- To install a new instance of License Server, click **No**, and then click **Next**. Specify the ports

  that you want License Server to listen on, and then type the path or click and navigate to the location of your HPE license key file (`licensekey.dat`), which you obtained when you purchased Query Manipulation Server. Click **Next**.

  The Component Selection dialog box opens.

6. Click **Next**.

7. Select the check boxes for the components that you want to install, and specify the port information for each component, or leave the fields blank to accept the default port settings.

   For the QMS, you can specify the following ports:

ACI Port	The port that client machines use to send ACI actions to the QMS. Default: 16000
Service Port	The port that client machines use to send service requests to the QMS. Default: 16002

   If you do not specify a value, the installer uses the specified default ports.

   Click **Next** or **Back** to move between components.

8. After you have specified your settings, the Summary dialog box opens. Verify the settings you made and click **Next**.

   The Ready to Install dialog box opens.

9. Click **Next**.

   The Installing dialog box opens, indicating the progress of the installation. If you want to end the installation process, click **Cancel**.

10. After installation is complete, click **Finish** to close the installation wizard.

Install an IDOL Component as a Service on Windows

On Microsoft Windows operating systems, you can install any IDOL component as a Windows service. Installing a component as a Windows service makes it easy to start and stop the component, and you can configure a component to start automatically when you start Windows.

Use the following procedure to install Query Manipulation Server as a Windows service from a command line.

To install a component as a Windows service

1. Open a command prompt with administrative privileges (right-click the icon and select **Run as administrator**).

2. Navigate to the directory that contains the component that you want to install as a service.

3. Send the following command:

   Component.exe -install

   where *Component*.exe is the executable file of the component that you want to install as a service.

   The -install command has the following optional arguments:

`-start {[auto] | [manual] | [disable]}`	The startup mode for the component. `Auto` means that Windows services automatically starts the component. `Manual` means that you must start the service manually. `Disable` means that you cannot start the service. The default option is `Auto`.
`-username` *UserName*	The user name that the service runs under. By default, it uses a local system account.
`-password` *Password*	The password for the service user.
`-servicename` *ServiceName*	The name to use for the service. If your service name contains spaces, use quotation marks (") around the name. By default, it uses the executable name.
-displayname DisplayName	The name to display for the service in the Windows services manager. If your display name contains spaces, use quotation marks (") around the name. By default, it uses the service name.
`-depend` *Dependency1* `[,`*Dependency2* `...]`	A comma-separated list of the names of Windows services that Windows must start before the new service. For example, if you are installing a Community component, you might want to add the Agentstore component and Content component as dependencies.

For example:

```
content.exe -install -servicename ContentComponent -displayname "IDOL Server
Content Component" -depend LicenseServer
```

After you have installed the service, you can start and stop the service from the Windows Services manager.

When you no longer require a service, you can uninstall it again.

To uninstall an IDOL Windows Service

1. Open a command prompt.
2. Navigate to the directory that contains the component service that you want to uninstall.
3. Send the following command:

   Component`.exe -uninstall`

   where *Component*`.exe` is the executable file of the component service that you want to uninstall.

   If you did not use the default service name when you installed the component, you must also add the `-servicename` argument. For example:

   Component`.exe -uninstall -servicename` *ServiceName*

Install Query Manipulation Server on UNIX

Use the following procedure to install Query Manipulation Server in text mode on UNIX platforms.

To install Query Manipulation Server on UNIX

1. Open a terminal in the directory in which you have placed the installer, and enter the following command:

   `./IDOLServer_VersionNumber_Platform.exe --mode text`

   where:

`VersionNumber`	is the product version
`Platform`	is the name of your UNIX platform

   Note: Ensure that you have execute permission for the installer file.

   The console installer starts and displays the Welcome screen.

2. Read the information and then press the `Enter` key.

   The license information is displayed.

3. Read the license information, pressing `Enter` to continue through the text. After you finish reading the text, type **Y** to accept the license terms.

4. Type the path to the location where you want to install the servers, or press `Enter` to accept the default path.

   The Installation Mode screen is displayed.

5. Press `2` to select the Custom installation mode.

   The License Server screen opens. Choose whether you have an existing License Server.

   - To use an existing License Server, type **Y**. Specify the host and port details for your License Server (or press `Enter` to accept the defaults), and then press `Enter`. Go to Step 7.

   - To install a new instance of License Server, type **N**.

6. If you want to install a new License Server, provide information for the ports that the License Server uses.

   a. Type the value for the ACI Port and press `Enter` (or press `Enter` to accept the default value).

ACI Port	The port that client machines use to send ACI actions to the License Server.

   b. Type the value for the Service Port and press `Enter` (or press `Enter` to accept the default value).

Service Port	The port by which you send service actions to the License Server. This port must not be used by any other service.

   c. Type the location of your HPE license key file (`licensekey.dat`), which you obtained when you purchased Query Manipulation Server. Press `Enter`.

7. The Component Selection screen is displayed. Press `Enter`. When prompted, type **Y** for the components that you want to install. Specify the port information for each component, and then press `Enter`. Alternatively, leave the fields blank and press `Enter` to accept the default port settings.

For the QMS, you can specify the following ports:

ACI Port	The port that client machines use to send ACI actions to the QMS. Default: 16000
Service Port	The port that client machines use to send service requests to the QMS. Default: 16002

If you do not specify a value, the installer uses the specified default ports.

> **Note:** These ports must not be used by any other service.

The Init Scripts screen is displayed.

8. Type the user that the server should run as, and then press `Enter`.

> **Note:** The installer does not create this user. It must exist already.

9. Type the group that the server should run under, and then press `Enter`.

> **Note:** If you do not want to generate init scripts for installed components, you can simply press `Enter` to move to the next stage of the installation process without specifying a user or group.

The Summary screen is displayed.

10. Verify the settings that you made, then press `Enter` to begin installation.

    The Installing screen is displayed.

    This screen indicates the progress of the installation process.

    The Installation Complete screen is displayed.

11. Press `Enter` to finish the installation.

Install an IDOL Component as a Service on Linux

On Linux operating systems, you can configure a component as a service to allow you to easily start and stop it. You can also configure the service to run when the machine boots. The following procedures describe how to install Query Manipulation Server as a service on Linux.

> **Note:** To use these procedures, you must have `root` permissions.

The procedure that you must use depends on the operating system and boot system type.

- For Linux operating system versions that use systemd (including Centos 7, and Ubuntu version 15.04 and later), see "Install a Component as a Service for a systemd Boot System" on the next page.

- For Linux operating system versions that use System V, see "Install a Component as a Service for a System V Boot System" on the next page.

Install a Component as a Service for a systemd Boot System

To install an IDOL component as a service

1. Run the appropriate command for your Linux operating system environment to copy the init scripts to your `init.d` directory.

    -   Red Hat Enterprise Linux (and Centos)

        ```
        cp IDOLInstallDir/scripts/init/systemd/componentname /etc/systemd/system/
        ```

    -   Debian (including Ubuntu):

        ```
        cp IDOLInstallDir/scripts/init/systemd/componentname /lib/systemd/system/
        ```

    componentname is the name of the init script that you want to use, which is the name of the component executable (without the file extension).

    For other Linux environments, refer to the operating system documentation.

2. Run the following commands to set to appropriate access, owner, and group permissions for the component:

    -   Red Hat Enterprise Linux (and Centos)

        ```
        chmod 755 /etc/systemd/system/componentname
        chown root /etc/systemd/system/componentname
        chgrp root /etc/systemd/system/componentname
        ```

    -   Debian (including Ubuntu):

        ```
        chmod 755 /lib/systemd/system/componentname
        chown root /lib/systemd/system/componentname
        chgrp root /lib/systemd/system/componentname
        ```

    For other Linux environments, refer to the operating system documentation.

    where *componentname* is the name of the component executable that you want to run (without the file extension).

3. (Optional) If you want to start the component when the machine boots, run the following command:

    ```
    systemctl enable componentname
    ```

Install a Component as a Service for a System V Boot System

To install an IDOL component as a service

1. Run the following command to copy the init scripts to your `init.d` directory.

    ```
    cp IDOLInstallDir/scripts/init/systemv/componentname /etc/init.d/
    ```

    where *componentname* is the name of the init script that you want to use, which is the name of the component executable (without the file extension).

2. Run the following commands to set to appropriate access, owner, and group permissions for the component:

```
chmod 755 /etc/init.d/componentname
chown root /etc/init.d/componentname
chgrp root /etc/init.d/componentname
```

3. (Optional) If you want to start the component when the machine boots, run the appropriate command for your Linux operating system environment:

   - Red Hat Enterprise Linux (and CentOS):

     ```
     chkconfig --add componentname
     chkconfig componentname on
     ```

   - Debian (including Ubuntu):

     ```
     update-rc.d componentname defaults
     ```

   For other Linux environments, refer to the operating system documentation.

Chapter 3: Run Query Manipulation Server

This section describes how to start and stop QMS, and how to send actions.

Start Query Manipulation Server

The following sections describe the different ways that you can start QMS.

Before you can start the QMS, you must start the License Server.

Start QMS on Microsoft Windows

- Double-click the `QMS.exe` file in your component installation directory.
- Start the QMS service from a system dialog box. QMS must be installed as a Windows Service. See "Install an IDOL Component as a Service on Windows" on page 16.
  a. Display the Windows **Services** dialog box.
  b. Select the **QMS** service for the component, and click **Start** to start the component.
  c. Click **Close** to close the **Services** dialog box.

> **Tip:** You can also configure the Windows Service to run automatically when you start the machine.

- Start QMS from the command line. For more information, refer to the *IDOL Getting Started Guide*.

Start QMS on UNIX

- Start the IDOL component service from the command line. The component must be installed as a service. See "Install an IDOL Component as a Service on Linux" on page 19. You can use one of the following commands to start the service:
  - On systemd Linux platforms:

    ```
    systemctl start QMS
    ```

  - On System V Linux platforms:

    ```
    service QMS start
    ```

- On Solaris platforms (using System V):

  ```
  /etc/init.d/QMS start
  ```

  > **Tip:** You can also configure the service to run automatically when you start the machine.

- Start QMS from the command line. For more information, refer to the *IDOL Getting Started Guide*.
- Use the start script (`start-QMS.sh`).

  > **Note:** In most cases, HPE recommends that you use the provided init scripts instead.

Stop Query Manipulation Server

You can stop QMS from running in several different ways.

- (All Platforms) Send the `Stop` service action to the component service port:

  ```
  http://host:servicePort/action=stop
  ```

  where *host* is the name or IP address of the host on which QMS is running, and *servicePort* is the component service port (which is specified in the `[Service]`section of the Query Manipulation Server configuration file).

- On Windows platforms, when the component is installed as a service, you can use the system dialog box to stop the service:

  a. Display the Windows **Services** dialog box.

  b. Select the **QMS** service, and click **Stop** to stop Query Manipulation Server.

  c. Click **Close** to close the **Services** dialog box.

- On UNIX platforms, when the component is installed as a service, you can run one of the following commands to stop the service:

  - On systemd platforms:

    ```
    systemctl stop QMS
    ```

  - On system V platforms:

    ```
    service QMS stop
    ```

  - On Solaris platforms (using System V):

    ```
    /etc/init.d/QMS stop
    ```

- On UNIX platforms, you can also use the stop script, `stop-QMS.sh`.

Send Actions to Query Manipulation Server

You send actions to QMS from your Web browser. The general syntax of these actions is:

```
http://QMSHost:port/action=action&parameters
```

where,

QMSHost	The IP address or name of the machine where QMS is installed.
port	The ACI port by which you send actions to QMS (set by the `Port` parameter in the `[Server]` section of the QMS configuration file). Refer to the *QMS Reference* for more information.
action	The name of the action to run (for example, `addTask`).
parameters	The required and optional parameters for the action.

Note: Separate individual parameters with an ampersand (&).

Verify Query Manipulation Server Runs Correctly

After you set up QMS, you can run the following actions to verify that QMS runs correctly.

GetRequestLog (GRL)

Send a `GetRequestLog` or `GRL` action to QMS to return a log of the requests that have been made to it, including:

- the date and time that a request was made.
- the client IP address that made the request.
- the internal thread that handled the action.

For example:

`http://QMSHost:port/action=GRL`

Related Topics

- "Send Actions to Query Manipulation Server" on the previous page

GetStatus

You can use the `GetStatus` action to verify that the QMS service is running.

For example:

`http://QMSHost:port/action=GetStatus`

Related Topics

- "Send Actions to Query Manipulation Server" on the previous page

Chapter 4: Configure Query Manipulation Server

This section describes how to configure QMS.

Edit the Configuration File

You configure QMS by using a standard configuration file (`.cfg`). The configuration file must be in the same directory as the QMS executable, and must have the same name as the executable. For example:

- `QMS.exe`
- `QMS.cfg`

You can create a configuration file by using a text editor.

Refer to the *QMS Reference* for more information.

Modify Configuration Parameter Values

You modify Query Manipulation Server configuration parameters by directly editing the parameters in the configuration file. When you set configuration parameter values, you must use UTF-8.

> **Caution:** You must stop and restart Query Manipulation Server for new configuration settings to take effect.

This section describes how to enter parameter values in the configuration file.

Enter Boolean Values

The following settings for Boolean parameters are interchangeable:

```
TRUE = true = ON = on = Y = y = 1

FALSE = false = OFF = off = N = n = 0
```

Enter String Values

To enter a comma-separated list of strings when one of the strings contains a comma, you can indicate the start and the end of the string with quotation marks, for example:

ParameterName=**cat,dog,bird,"wing,beak",turtle**

Alternatively, you can escape the comma with a backslash:

ParameterName=**cat,dog,bird,wing\,beak,turtle**

If any string in a comma-separated list contains quotation marks, you must put this string into quotation marks and escape each quotation mark in the string by inserting a backslash before it. For example:

ParameterName=**"<font face=\"arial\" size=\"+1\"><b>","<p>"**

Here, quotation marks indicate the beginning and end of the string. All quotation marks that are contained in the string are escaped.

Include an External Configuration File

You can share configuration sections or parameters between ACI server configuration files. The following sections describe different ways to include content from an external configuration file.

You can include a configuration file in its entirety, specified configuration sections, or a single parameter.

When you include content from an external configuration file, the `GetConfig` and `ValidateConfig` actions operate on the combined configuration, after any external content is merged in.

In the procedures in the following sections, you can specify external configuration file locations by using absolute paths, relative paths, and network locations. For example:

```
../sharedconfig.cfg
K:\sharedconfig\sharedsettings.cfg
\\example.com\shared\idol.cfg
file://example.com/shared/idol.cfg
```

Relative paths are relative to the primary configuration file.

> **Note:** You can use nested inclusions, for example, you can refer to a shared configuration file that references a third file. However, the external configuration files must not refer back to your original configuration file. These circular references result in an error, and Query Manipulation Server does not start.
>
> Similarly, you cannot use any of these methods to refer to a different section in your primary configuration file.

Include the Whole External Configuration File

This method allows you to import the whole external configuration file at a specified point in your configuration file.

To include the whole external configuration file

1. Open your configuration file in a text editor.

2. Find the place in the configuration file where you want to add the external configuration file.

3. On a new line, type a left angle bracket (<), followed by the path to and name of the external configuration file, in quotation marks (""). You can use relative paths and network locations. For example:

   ```
   < "K:\sharedconfig\sharedsettings.cfg"
   ```

4. Save and close the configuration file.

Include Sections of an External Configuration File

This method allows you to import one or more configuration sections from an external configuration file at a specified point in your configuration file. You can include a whole configuration section in this way, but the configuration section name in the external file must exactly match what you want to use in your file. If you want to use a configuration section from the external file with a different name, see "Merge a Section from an External Configuration File" on the next page.

To include sections of an external configuration file

1. Open your configuration file in a text editor.

2. Find the place in the configuration file where you want to add the external configuration file section.

3. On a new line, type a left angle bracket (<), followed by the path to and name of the external configuration file, in quotation marks (""). You can use relative paths and network locations. After the configuration file name, add the configuration section name that you want to include. For example:

   ```
   < "K:\sharedconfig\extrasettings.cfg" [License]
   ```

   Note: You cannot include a section that already exists in your configuration file.

4. Save and close the configuration file.

Include a Parameter from an External Configuration File

This method allows you to import a parameter from an external configuration file at a specified point in your configuration file. You can include a section or a single parameter in this way, but the value in the external file must exactly match what you want to use in your file.

To include a parameter from an external configuration file

1. Open your configuration file in a text editor.

2. Find the place in the configuration file where you want to add the parameter from the external configuration file.

3. On a new line, type a left angle bracket (<), followed by the path to and name of the external configuration file, in quotation marks (""). You can use relative paths and network locations. After the configuration file name, add the name of the configuration section name that contains the

parameter, followed by the parameter name. For example:

```
< "license.cfg" [License] LicenseServerHost
```

To specify a default value for the parameter, in case it does not exist in the external configuration file, specify the configuration section, parameter name, and then an equals sign (=) followed by the default value. For example:

```
< "license.cfg" [License] LicenseServerHost=localhost
```

4. Save and close the configuration file.

Merge a Section from an External Configuration File

This method allows you to include a configuration section from an external configuration file as part of your Query Manipulation Server configuration file. For example, you might want to specify a standard SSL configuration section in an external file and share it between several servers. You can use this method if the configuration section that you want to import has a different name to the one you want to use.

To merge a configuration section from an external configuration file

1. Open your configuration file in a text editor.

2. Find or create the configuration section that you want to include from an external file. For example:

```
[SSLOptions1]
```

3. After the configuration section name, type a left angle bracket (<), followed by the path to and name of the external configuration file, in quotation marks (""). You can use relative paths and network locations. For example:

```
[SSLOptions1] < "../sharedconfig/ssloptions.cfg"
```

If the configuration section name in the external configuration file does not match the name that you want to use in your configuration file, specify the section to import after the configuration file name. For example:

```
[SSLOptions1] < "../sharedconfig/ssloptions.cfg" [SharedSSLOptions]
```

In this example, Query Manipulation Server uses the values in the [SharedSSLOptions] section of the external configuration file as the values in the [SSLOptions1] section of the Query Manipulation Server configuration file.

> **Note:** You can include additional configuration parameters in the section in your file. If these parameters also exist in the imported external configuration file, Query Manipulation Server uses the values in the local configuration file. For example:
>
> ```
> [SSLOptions1] < "ssloptions.cfg" [SharedSSLOptions]
> SSLCACertificatesPath=C:\IDOL\HTTPConnector\CACERTS\
> ```

4. Save and close the configuration file.

The Query Manipulation Server Configuration File

The QMS configuration file contains settings that determine the basic details that the QMS needs to run. It includes location and port details for QMS, logging details, and details of the servers that QMS connects to. For details on all available configuration parameters, refer to the *QMS Reference.*

Configuration File Sections

The QMS configuration file contains several sections, which represent different areas that you can configure. Refer to the *QMS Reference* for more information.

[IDOL] Section

The `[IDOL]` configuration section contains settings that allow QMS to contact the DAH, IDOL Server, or Content component that contains the data index. For example:

```
[IDOL]
Host=localhost
Port=20000
```

[Logging] Section

The `[Logging]` section lists the log streams that you set up to create separate log files for different log message types (action and application). It also contains a subsection for each of the listed log streams, in which you configure the settings that determine how to log each stream. For example:

```
[Logging]
LogDirectory=./logs
LogTime=True
LogEcho=True
LogLevel=full

0=ACTION_LOG_STREAM
1=APP_LOG_STREAM

[ACTION_LOG_STREAM]
LogFile=action.log
LogTime=True
LogEcho=False
LogMaxSizeKBs=1024
LogTypeCSVs=query
LogLevel=normal
LogExpireAction=datestamp

[APP_LOG_STREAM]
LogFile=app.log
LogTime=True
LogEcho=True
```

```
LogMaxSizeKBs=1024
LogTypeCSVs=application
LogLevel=full
LogExpireAction=previous
```

[PromotionAgentStore] Section

The [PromotionAgentStore] configuration section contains settings that allow QMS to contact the Promotion Agentstore component. For example:

```
[PromotionAgentStore]
Host=localhost
Port=20050
```

[Server] Section

The [Server] configuration section contains general settings for QMS. For example:

```
[Server]
Port=16000
UserClients=*.*.*.*
AdminClients=*.*.*.*
DefaultLanguage=ENGLISH
DefaultLanguageType=GenericUTF8
AllowedQueryParams=Text,MatchResults
EngineTimeout=180
MaxResultsLimit=5000
```

> **Note:** QMS obtains language settings from the Content component on startup. However, if this is not possible because of the security settings of your Content ACI port, or if you want to use sentence breaking libraries for query text processing, you can use the DefaultLanguageType parameter in the [Server] section of the QMS configuration file to set the default language type from within QMS itself. For more information, refer to the *QMS Reference.*
>
> You can also list the languages that you use and define some generic parameters in the [LanguageTypes] section in the QMS configuration file. For more information on how to configure language types, refer to the *IDOL Server Reference.*

[Service] Section

The [Service] configuration section contains settings that determine which machines can use and control the IDOL Server service. For example:

```
[Service]
ServicePort=16002
ServiceStatusClients=*.*.*.*
ServiceControlClients=*.*.*.*
```

[SSLOption*N*] Section

The [SSLOptionN] configuration section contains settings that allow QMS to receive SSL connections from clients. For example:

```
[Server]
SSLConfig=SSLOptions1

...

[SSLOptions1] //SSL options for incoming connections
SSLMethod=SSLV23
SSLCertificate=host1.crt
SSLPrivateKey=host1.key
SSLCACertificate=trusted.crt
```

[StatisticsServer] Section

The [StatisticsServer] configuration section contains settings that allow QMS to contact a statistics server that stores promotion and synonym statistics. For example:

```
[StatisticsServer]
Host=localhost
Port=19871
```

Example Configuration File

```
[Service]
ServicePort=16002
ServiceStatusClients=*.*.*.*
ServiceControlClients=*.*.*.*

[Server]
Port=16000
UserClients=*.*.*.*
AdminClients=*.*.*.*
DefaultLanguage=ENGLISH
AllowedQueryParams=Text,MatchResults
EngineTimeout=180
MaxResultsLimit=5000

[IDOL]
Host=localhost
Port=20000

[PromotionAgentStore]
Host=localhost
Port=20050

[StatisticsServer]
Host=localhost
```

```
//Port should be the Event Port of the stats server
Port=19871

[Logging]
LogDirectory=./logs
LogTime=True
LogEcho=True
LogLevel=full

0=ACTION_LOG_STREAM
1=APP_LOG_STREAM

[ACTION_LOG_STREAM]
LogFile=action.log
LogTime=True
LogEcho=False
LogMaxSizeKBs=1024
LogTypeCSVs=query
LogLevel=normal
LogExpireAction=datestamp

[APP_LOG_STREAM]
LogFile=app.log
LogTime=True
LogEcho=True
LogMaxSizeKBs=1024
LogTypeCSVs=application
LogLevel=full
LogExpireAction=previous
```

Configure Child Servers

To run QMS, you must configure it to communicate with the IDOL Server data index, and the Promotion Agentstore.

You can use one of the following components to access the IDOL Server data index:

- Distributed Action Handler (DAH), which distributes the actions to child IDOL Servers.
- The IDOL Server IDOL Proxy component, which forwards the actions to the IDOL Server Content component.
- The IDOL Server Content component, which stores the data index.

To configure child servers for QMS

1. Open the QMS configuration file in a text editor.
2. Find the [IDOL] section, or create one if it does not exist.
3. In the [IDOL] section, set the Host parameter to the host name or IP address of the server that you use to query the IDOL Server data index (DAH, IDOL Proxy, or the Content component). Set the Port parameter to the ACI port of this server.

For example:

```
[IDOL]
Host=127.0.0.1
Port=9000
```

4. Find the [PromotionAgentStore] section, or create one if it does not exist.

5. In the [PromotionAgentStore] section, set the Host parameter to the host name or IP address of the machine where the Promotion Agentstore component is installed. Set the Port parameter to the ACI port of the Promotion Agentstore.

   For example:

```
[PromotionAgentStore]
Host=127.0.0.1
Port=9050
```

6. If you want to store statistics for QMS, find the [StatisticsServer] section, or create one if it does not exist.

7. (Optional) Specify the query parameters that QMS can pass through to child servers as the value of the AllowedQueryParameters parameter in the [Server] section of the QMS configuration file. Separate multiple parameters with a comma.

8. Save and close the QMS configuration file. Restart QMS for your changes to take effect.

Refer to the *QMS Reference* for more information.

Related Topics

- "Edit the Configuration File" on page 27

Use a Community Component

If you want to use the QMS intent ranked query or intent based promotions functionality to promote results that are close to the interests of the user to the top of the result set, QMS must be able to communicate with a Community component. To set up communication with a Community component, you must perform one of the following tasks:

- Configure the [IDOL] section with the host and port of an IDOL server. If you use this option, no further configuration is required.

- Create a [Community] section in the configuration file, with Host and Port parameters that specify the host and port of the Community component that you want to use.

```
[Community]
Host=...
Port=...
```

   Use this option if the [IDOL] section points to a DAH or IDOL Content server.

For more information on intent ranked queries and intent based promotions, see "Intent Based Ranking" on page 76 and "Intent Based Promotions" on page 57, and refer to the *QMS Reference*.

Use Statistics Server

You can optionally configure QMS to communicate with a statistics server. If you configure a statistics server, it records statistics about promotions and synonyms results.

For details of Statistics Server configuration, refer to the *IDOL Server Administration Guide* and the *QMS Reference*.

To configure QMS to use Statistics Server

1. Open the QMS configuration file in a text editor.

2. Find the `[StatisticsServer]` section, or create one if it does not exist.

3. In the `[StatisticsServer]` section, set the `Host` parameter to the host name or IP address of the machine where the statistics server is installed. Set the `Port` parameter to the event port of the statistics server.

   Note: This port must be the Statistics Server `EventPort`, not the ACI port.

4. Save and close the QMS configuration file. Restart QMS for your changes to take effect.

Related Topics

- "Edit the Configuration File" on page 27

Use a Query Cooker

If you are using QMS with Autonomy Business Console (ABC), you can use a *query cooker* to manipulate queries. A query cooker is a JavaScript application that you can create in the ABC user interface. Query cookers modify or enhance queries, for example to add extra query criteria to boost certain results. They provide more flexibility and allow you to use more complex rules to manipulate queries.

Query cookers are for advanced users who want to use the JavaScript API functions.

To use a query cooker

1. Create the query cooker in ABC.

2. Add a `[QueryCooker]` section to the QMS configuration file, which specifies the host and port of the query cooker server. For example:

   ```
   [QueryCooker]
   Host=12.3.4.56
   Port=8080
   ```

3. Send queries to QMS with the `Cook` parameter set to the name of the query cooker. You can add any additional parameters for the query cooker by setting `QueryCookingParameters` to a comma-separated list of *key:value* pairs. For example:

   ```
   action=Query&Text=Cat&Cook=cook1&QueryCookingParameters=key:value
   ```

For more details about how to create and set up query cookers, refer to the *Autonomy Business Console User Guide* and the *QMS Reference*.

Related Topics

-

Use a Request Cooker

A request cooker is similar to a query cooker, but it modifies the whole action, rather than only the query text. For example, you can use a request cooker to add extra parameters to the query that users send to QMS.

You can configure the request cooker to be an external service, or you can use a Lua script.

When you use a request cooker, QMS sends the original action to the cooker. The cooker returns the modified request, and QMS applies whitelist, then blacklist, and then query text processing to it as necessary.

The cooker must specify the whole action to use.

Request Cooker Lua Scripts

The Lua script must provide a globally accessible `cook_request` function, which accepts a string representation of the request as its only argument. The function must return a Lua table. The keys of this table are the parameter names to use in the action (the parameter names must be in lower case). The table values are the corresponding request values.

QMS loads the Lua script for every request that it cooks. This means that any changes to the script are reflected immediately in the query behavior.

The following example request cooker Lua script sets `MaxResults` to **10** for all queries:

```
-- load module that provides string to table request parser
aci = require "autn_aci"
-- Set maxresults to 10 on every query
function cook_request(request_string)
  cooked_request = aci.parse_request_string(request_string)
  cooked_request["maxresults"] = 10
  return cooked_request
end
```

Configure the Request Cooker

The following procedure describes how to configure QMS to use a request cooker from an external service, or a Lua script.

To use a request cooker

1. Open the QMS configuration file in a text editor.

2. Add a `[RequestCooker]` section to the QMS configuration file.

3. If you want to use a request cooker on an external server, specify the host and port information of the server and set `Mode` to **legacy** in the `[RequestCooker]` section of the configuration file. For

example:

```
[RequestCooker]
Host=12.3.4.56
Port=8080
Mode=legacy
```

4. If you want to use a custom Lua script for request cooking, set `Mode` to **lua** in the `[RequestCooker]` section of the configuration file, and set `Script` to the path to the script that you want to use. For example:

```
[RequestCooker]
Script=qms\lua\cookrequest-maxresults.lua
Mode=lua
```

5. Send queries to QMS with the `CookRequest` parameter set to **True**.

   If you want to cook all requests that you send to QMS, set the `CookAllRequests` configuration parameter to **True**.

Refer to the *QMS Reference* for more information on how to configure query cooking.

Customize Logging

You can customize logging by setting up your own *log streams*. Each log stream creates a separate log file in which specific log message types (for example, action, index, application, or import) are logged.

To set up log streams

1. Open the Query Manipulation Server configuration file in a text editor.

2. Find the `[Logging]` section. If the configuration file does not contain a `[Logging]` section, add one.

3. In the `[Logging]` section, create a list of the log streams that you want to set up, in the format *N=LogStreamName*. List the log streams in consecutive order, starting from 0 (zero). For example:

```
[Logging]
LogLevel=FULL
LogDirectory=logs
0=ApplicationLogStream
1=ActionLogStream
```

   You can also use the `[Logging]` section to configure any default values for logging configuration parameters, such as `LogLevel`. For more information, see the *Query Manipulation Server Reference*.

4. Create a new section for each of the log streams. Each section must have the same name as the log stream. For example:

```
[ApplicationLogStream]
[ActionLogStream]
```

5. Specify the settings for each log stream in the appropriate section. You can specify the type of logging to perform (for example, full logging), whether to display log messages on the console, the

maximum size of log files, and so on. For example:

```
[ApplicationLogStream]
LogTypeCSVs=application
LogFile=application.log
LogHistorySize=50
LogTime=True
LogEcho=False
LogMaxSizeKBs=1024

[ActionLogStream]
LogTypeCSVs=action
LogFile=logs/action.log
LogHistorySize=50
LogTime=True
LogEcho=False
LogMaxSizeKBs=1024
```

6. Save and close the configuration file. Restart the service for your changes to take effect.

Configure Query Manipulation Server to Receive SSL Connections

You can configure QMS to receive SSL communication from client servers.

To configure SSL connections

1. Open the QMS configuration file in a text editor.

2. In the `[Server]` section, set the `SSLConfig` parameter to the name of the configuration file section where you specify details of the SSL connections, for example *SSLOptionN*.

3. Create a configuration section with the name that you have specified.

4. Set any SSL configuration parameters.

   For example:

   ```
   [Server]
   SSLConfig=SSLOptions1
   ...

   [SSLOptions1] //SSL options for incoming connections
   SSLMethod=SSLV23
   SSLCertificate=host1.crt
   SSLPrivateKey=host1.key
   SSLCACertificate=trusted.crt
   ```

5. Save and close the QMS configuration file. Restart QMS for your changes to take effect.

Part 2: Query Manipulation Server Operations

This section describes the operations that you can perform using QMS.

- "Query Manipulation Server Rules"
- "Query Manipulation Server Promotions"
- "Modify Queries"
- "Manipulate Results"
- "Use QMS TypeAhead"

Chapter 5: Query Manipulation Server Rules

This section describes how to create QMS rules and index them into the Promotion Agentstore component.

Query Manipulation Server Rules Overview

QMS rules contain the information that QMS uses to modify user queries. You store the rules in the Promotion Agentstore.

Promotion Agentstore is an IDOL Server Content component. It stores rules in the same way that the Content component stores documents, or the Agentstore components stores agents and categories.

QMS rules are IDX or XML documents that contain the fields that the Promotion Agentstore and QMS need to find and apply rules. You can create QMS rules in IDOL Search Optimizer, or you can create the IDX documents manually and index them into the Promotion Agentstore.

For details of how to use QMS in IDOL Search Optimizer, refer to the *IDOL Search Optimizer User Guide*.

You can create the following types of QMS rules:

- **Static promotions**. Return a specified document or set of documents as a promotion.
- **Dynamic promotions**. Return the results of a query as a promotion.
- **Synonyms**. Modify the query with synonymous terms.
- **Whitelists**. Modify the query to remove terms that do not occur in the whitelist.
- **Blacklists**. Modify the query to remove terms that occur in the blacklist.
- **Boost rules**. Add extra query FieldText.
- **Cardinal placements**. Add a document to a specified position in the results list.
- **Parametric cardinal placements**. Add a parametric value to a specified position in the results list.

Configure Promotion Agentstore

The Promotion Agentstore is a Content component that stores the QMS rules. You can use the same configuration as the IDOL Server Agentstore component. However, you can improve performance by adding some configuration settings to the Promotion Agentstore component.

AgentBoolean Fields

QMS rules match queries by using AgentBoolean and FieldText expressions. To optimize this matching process, you can configure the Promotion Agentstore to cache the values of the QMS rules fields that store the expressions.

For details about AgentBoolean fields, refer to the *IDOL Server Administration Guide*.

To configure AgentBoolean fields

1. Open the Promotion Agentstore configuration file in a text editor.

2. In the `[Server]` section, add or edit the following configuration parameters:

   ```
   [Server]
   AgentBooleanCacheField=*/QMSAGENTBOOL
   FieldTextCacheField=*/QMSFIELDTEXT
   ```

3. Restart the Promotion Agentstore for your changes to take effect.

> **Note:** If you change these settings after you index rules into Promotion Agentstore, you must re-index the rules.

Field Processing

Promotion Agentstore can process different fields in the same way as IDOL Server. You can configure the Promotion Agentstore to process and store the QMS rules fields to optimize their retrieval.

For details about field processing, refer to the *IDOL Server Administration Guide*.

You can use the following fields and field properties to optimize performance:

Fields	Description	Field property
/QMSTYPE,/QMSVALUE2	Fields that QMS uses to sort results.	SortType
*/QMSTYPE	Fields that QMS must match exactly.	MatchType

For example:

```
[SetSortFields]
Property=SortFields
PropertyFieldCSVs=*/QMSTYPE,*/QMSVALUE2

[SetMatchFields]
Property=MatchFields
PropertyFieldCSVs=*/QMSTYPE

[SortFields]
SortType=True

[MatchFields]
MatchType=True
```

Create Query Manipulation Server Rules

The Promotion Agentstore component stores QMS rules in a similar way to the way in which IDOL Server stores agents and categories.

The QMS rules documents must contain text that defines the queries that they affect. They also define which actions are performed on the queries, and the results for any transformations. For some QMS rules you can also apply a schedule, so that the rule is active only at certain times.

You create QMS rules as IDX documents, which you index into the Promotion Agentstore. If you do not use QMS with IDOL Search Optimizer, you must manually create IDX documents and index them. For details on how to manually create an IDX document, refer to the *IDOL Server Administration Guide*.

Create a QMS Rule IDX Document

A QMS document is a text file that includes the required IDX fields, saved with the file name extension .IDX.

The table describes the fields that each QMS rule contains.

Required IDX fields for all QMS rule documents

Field	Description
#DREREFERENCE	A unique reference string to identify the rule.
#DRETITLE	A title to identify the rule.
#DREFIELD QMSTYPE="N"	The type of rule. Each rule type has a different value of N. See "Values of QMSTYPE" on the next page.
#DREENDDOC	The end of the rule document. Add this field after all the rule fields.

In addition to these standard fields, you must add different fields depending on the type of rule. For example, you must add fields to specify the queries that a rule applies to, and to specify how QMS must modify the query or results.

Add each additional field in the following format:

`#DREFIELD FIELDNAME="FieldValue"`

where,

FIELDNAME is the name of the field.

FieldValue is the value that this field contains.

For example:

`#DREFIELD QMSAGENTBOOL="panda AND bear"`

If you include FieldText restrictions in the QMSFIELDTEXT field (or another field), you must also add any fields that you use in the field restrictions as empty fields. For example:

```
#DREFIELD QMSFIELDTEXT="MATCH{poodle}:BREED"
#DREFIELD BREED=""
```

Related Topics

-

-

-

-

-

-

-

Values of QMSTYPE

QMS and the Promotion Agentstore use the value of the QMSTYPE field in each rule document to determine the type of rule that the document describes. You can use the following values for the QMSTYPE field.

1	Cardinal placement (insertion)	Adds or moves a document to a specified position in a results list.
2	Static promotion	Returns a promotion document or set of documents.
3	Dynamic promotion	Returns the results of a query as promotion documents.
4	Synonyms	Modifies the query to include synonymous terms.
5	Whitelist	Removes query terms that do not match the words in the whitelist.
6	Blacklist	Removes query terms that match the words in the blacklist.
7	Boost rules	Expands the query `FieldText` to include extra terms.
8	Parametric cardinal placement (parametric insertion)	Adds or moves a parametric value to a specified position in a results list.

Related Topics

-

-

-

-

-

-

-

Schedules for QMS Rules

For rules with a QMSTYPE of 1, 2, or 3 (document cardinal placement or promotions), you can set a schedule to determine when the rule applies.

IDX fields for schedules

Field	Description	Content type or value
ALWAYSACTIVE	Whether the rule is always active. If this field is absent or contains the value False, the SCHEDULE field must contain a valid value.	Boolean value
SCHEDULE	The schedule for when the rule applies.	iCalendar format schedule data

- To apply the rule all the time, add the ALWAYSACTIVE document field with the value True.
- To apply a schedule, add the SCHEDULE document field. This field is required if the ALWAYSACTIVE field is absent, or if it contains the value False.

You must define the schedule in iCalendar format. For details about this format, refer to:

http://datatracker.ietf.org/doc/rfc5545/

Separate the parameters and values with colons (:). Separate each parameter and value pair with a plus sign and two backslashes (+\\). For example:

```
SCHEDULE="BEGIN:VCALENDAR+\\PRODID:-//Autonomy//schedule
1.0//EN+\\VERSION:2.0+\\CALSCALE:GREGORIAN+\\BEGIN:VEVENT+\\DTSTAMP:20080319T194426
Z+\\SUMMARY:+\\DTSTART:20080321T220000+\\DTEND:20080322T060000+\\RRULE:FREQ=WEEKLY;
UNTIL=20080530T000000Z;INTERVAL=2;BYDAY=TU,FR;WKST=SU+\\END:VEVENT+\\END:VCALENDAR
+\\"
```

QMS automatically detects whether the iCalendar schedule is percent-encoded.

> **Note:** The following iCalendar formats are not available in QMS rules:
>
> - "BYSECOND", "BYMINUTE", "BYHOUR", "BYYEARDAY", "BYWEEKNO" and "BYSETPOS" recurrences.
> - Weekday numerics, for example:
>
>   "Last Friday of every month => [FREQ=MONTHLY;BYDAY=-1FR]"
>
>   "Second Tuesday of every year => [FREQ=YEARLY;BYDAY=2TU]"

Related Topics

- "Static Promotion Rules" on page 52
- "Dynamic Promotion Rules" on page 54
- "Cardinal Placement Rules" on page 72

Example QMS Rule

The following example of a QMS rule IDX document is a parametric cardinal placement rule that places the value `giant panda` at the top of a parametric list of values of the `species` field.

```
#DREREFERENCE Rule
#DRETITLE PromotePanda
#DREFIELD QMSTYPE="8"
#DREFIELD QMSAGENTBOOL="*"
#DREFIELD QMSFIELDTEXT="MATCH{2002}:born"
#DREFIELD BORN=""
#DREFIELD QMSFIELDNAME="species"
#DREFIELD QMSFIELDVALUE="giant panda"
#DREFIELD QMSVALUE="1"
#DREFIELD ALWAYSMATCH="1"
#DRECONTENT
#DREENDDOC
```

Related Topics

Index QMS Rules

You publish QMS rules by indexing them into the `Activated` database in the Promotion Agentstore.

To index rules into the Promotion Agentstore

- Send a `DREADD` action. This action takes the following form:

  `http://AgentStoreHost:IndexPort/DREADD?FileName=FileName.idx&DREDbName=Database`

  where,

AgentStoreHost	is the IP address or host name of the machine on which the Promotion Agentstore is installed.
IndexPort	is the index port of the Promotion Agentstore (specified in the `IndexPort` parameter in the `[Server]` section of the Promotion Agentstore configuration file).

FileName	is the name of the IDX file that contains your rules.
Database	is the name of the database in the Promotion Agentstore that contains QMS rules. QMS searches for rules in the `Activated` database.

For details about the `DREADD` index action, refer to the *IDOL Server Administration Guide*.

Related Topics

- "Create a QMS Rule IDX Document" on page 45

Match QMS Rules in Queries

The Promotion Agentstore component retrieves rules in the same way that the IDOL Server Agentstore component retrieves categories and agents. You must set up your rules correctly to ensure that your QMS rules return for relevant queries.

When Promotion Agentstore matches query text against rules, it uses the following matching order:

1. It matches the query text against the `Index` fields in the rules (for example `DRECONTENT`).
2. For rules that match in step 1, it matches the query text against the Boolean restrictions in the `QMSAGENTBOOL` field of the rule.
3. For rules that match in step 2, it matches the query text against the FieldText restrictions in the `QMSFIELDTEXT` field in the rule (if this field exists).

Promotion Agentstore checks the Boolean restriction only if the rule content matches the query.

To ensure that rules match query text, the `DRECONTENT` field (or another index field) must contain terms that match the AgentBoolean expression. For example, you could add each term from the AgentBooelan expression to the `DRECONTENT` field.

> **Note:** In boost rules, you must configure and use an `AlwaysMatchType` field. QMS uses this value to retrieve these rules, and you do not need to add extra terms.
>
> In whitelist and blacklist rules, you do not need to set `DRECONTENT`, because QMS retrieves the lists directly by using the document reference.

You can optimize the performance of matching rules in the same ways that you optimize AgentBoolean agents and categories in IDOL Server. For more information, refer to the *IDOL Server Administration Guide*.

Related Topics

- "Boost Rules" on page 67
- "Whitelists and Blacklists" on page 65

Monitor Rule Usage

You can configure QMS to store data each time it activates a particular rule in the Promotion Agentstore.

QMS uses IDOL document tracking functionality to log a message when a rule is activated. You can then use a script to load the information from your logs to Statistics Server to monitor the most frequently accessed rules. For more information, refer to the *IDOL Server Administration Guide*.

To configure QMS to log rule usage

1. Open the QMS configuration file in a text editor.

2. Create a [DocumentTracking] configuration section.

3. In the [DocumentTracking] section, set the Backend parameter to **Log**. For example:

```
[DocumentTracking]
Backend=Log
```

4. Find the [Logging] configuration section.

5. Add a new log stream for your document tracking log. For example:

```
[Logging]
...
0=ApplicationLogStream
1=ActionLogStream
2=DocumentTrackingLogStream
```

6. Create a new configuration section for the new log stream. For example:

```
[DocumentTrackingLogStream]
```

7. In the log stream configuration section, set LogTypeCSVs to **Events**. For example:

```
[DocumentTrackingLogStream]
LogTypeCSVs=Events
```

You can also configure any additional logging parameters. For details, refer to the *QMS Reference*.

8. Save and close the QMS configuration file. Restart QMS for your changes to take effect.

Chapter 6: Query Manipulation Server Promotions

This section describes how to use QMS to manage promotions.

Promotions Overview

A *promotion* is targeted content that you want to display to users but is not included in the search results, such as advertisements. QMS promotion rules allow you to return and display promotion documents that are similar to a user query.

You can create promotions in QMS to ensure that certain results return for a particular query. For example, you might want to return a particular set of products when a user queries for phones.

There are two types of promotions:

- *Static promotions* list a specific document or set of documents.
- *Dynamic promotions* contain a query. The promotion documents are the results that return for this query.

When you query QMS for promotions, QMS forwards the query to the Promotion Agentstore. The Promotion Agentstore returns a list of promotions that match the query.

- The static promotions contain a list of document references, which QMS retrieves from IDOL Server and returns as promotions.
- The dynamic promotions contain a query, which QMS sends to IDOL Server. It then returns the results of this query as promotions.

QMS stores promotions in a similar way to cardinal placements. However, when you query for promotions, only the promotion documents return.

When you query the Promotion Agentstore, you can also use *scope rules* to filter the list of results that return according to specific values. For more information on how to set up scope rules and use them in promotions queries, see "Scope Rules" on page 56.

Related Topics

- "Cardinal Placement" on page 71
- "Static Promotions" on the next page
- "Dynamic Promotions" on page 54
- "Query for Promotions" on page 56

Static Promotions

A *static promotion* returns a specific promotion document or group of documents when a user sends a matching query to IDOL.

The static promotion rule contains a list of document references. QMS retrieves the promotion documents from IDOL Server and returns them to the client.

Static Promotion Rules

The table shows all the fields that you must include in the IDX file for static promotion rules.

Required IDX fields for static promotion rules

Field	Description	Content type or value
#DREREFERENCE	The reference for the rule.	String
#DRETITLE	The title of the rule.	String
QMSTYPE	The type of the QMS rule.	2
QMSAGENTBOOL	The AgentBoolean expression used to match the original query text.	Boolean matching expression
QMSFIELDTEXT	The FieldText rules to match the original query text.	FieldText expression
QMSVALUE1	The references of the promotion documents.	Comma-separated list of strings
PROMOTION_ NAME	The value that QMS must return in the autn:promotionname element. ABC uses this field value as the promotion type. Use one of the following values: • **Top Promotions**. A list of documents that are being promoted internally. Top Promotions are not advertisements. • **Sponsored**. A paid advertisement. • **Hotwire**. Promoted documents that are shown at the top of a results list for a particular keyword, but are noticeably distinct from the list.	String
QMSUSERTYPE	The value that QMS must return in the autn:usertype element.	String
QMS_ PROMOTION_	Data for promotions that do not exist in the content database.	String list. Separate items

Required IDX fields for static promotion rules, continued

Field	Description	Content type or value
ITEMS		in the list with \n and then percent-encode the list.
QMS_ PROMOTION_ ITEM_COUNT	The number of promotions included in the QMS_PROMOTION_ ITEMS field.	Number
FIELD_ CRITERIA _ DATABASEMATCH	(Optional) When specified, QMS uses the value of this field as the value of the DatabaseMatch parameter in the query to the Content component, and treats it as a database restriction.	
QMS_SCOPE_ RULE	(Optional) You can use this field with the PromotionsScope query parameter to return only promotions that match the text that you specify as the scope rule. This allows you to filter the list of returned promotions by user-defined criteria such as a specific department or job function.	String list. Separate items in the list with a comma.
DRECONTENT	Terms to match the query text.	String

You must also add either:

- the ALWAYSACTIVE field with the value **True**.
- a SCHEDULE field with iCalendar data.

See "Create a QMS Rule IDX Document" on page 45.

Related Topics

- "Create Query Manipulation Server Rules" on page 45
- "Create a QMS Rule IDX Document" on page 45
- "Scope Rules" on page 56

Example Static Promotion Rule

The following QMS rule returns when you send a query for promotions that contains one or more of the terms *news*, *drama*, or *data* in the query text. QMS then returns the IDOL Server document with the reference 4567389246372.

```
#DREREFERENCE static_promotion
#DRETITLE url_redirect_1
#DREDBNAME ACTIVATED
#DREOUTPUTENCODING UTF8
#DRELANGUAGETYPE englishUTF8
#DREFIELD QMSAGENTBOOL="news OR drama OR data"
#DREFIELD PROMOTION_NAME="static_promotion_1"
```

```
#DREFIELD QMSTYPE="2"
#DREFIELD QMS_PROMOTION_ITEMS="0"
#DREFIELD QMS_PROMOTION_ITEM_COUNT="0"
#DREFIELD QMSVALUE1="4567389246372"
#DREFIELD ALWAYSACTIVE="True"
#DRECONTENT
NEWS DRAMA DATA
#DREENDDOC
```

For example, the following query to QMS returns this promotion:

```
action=Query&Text=drama&Promotions=True
```

Dynamic Promotions

A *dynamic promotion* returns documents that match a particular promotion query. When you send a query to QMS that matches the rule, QMS sends a promotion query to IDOL Server or the Content component. QMS then returns the results documents for this query as a promotion.

Dynamic Promotion Rules

The table shows all the fields that you must include in the IDX file for dynamic promotion rules.

Required IDX fields for dynamic promotion rules

Field	Description	Content type or value
#DREREFERENCE	The reference for the rule.	String
#DRETITLE	The title of the rule.	String
QMSTYPE	The type of the QMS rule.	3
QMSAGENTBOOL	The AgentBoolean expression used to match the original query text.	Boolean matching expression
QMSFIELDTEXT	The FieldText rules to match the original query text.	FieldText expression
PROMOTION_NAME	The value that QMS must return in the autn:promotionname tag. ABC uses this field value as the promotion type. Use one of the following values: • **Top Promotions**. A list of documents that are being promoted internally. Top Promotions are not advertisements. • **Sponsored**. A paid advertisement.	String

Required IDX fields for dynamic promotion rules, continued

Field	Description	Content type or value
	• **Hotwire**. Promoted documents that are shown at the top of a results list for a particular keyword, but are noticeably distinct from the list.	
QMSUSERTYPE	The value that QMS must return in the autn:usertype tag. ABC uses this value.	String
DYNAMIC_QUERY_TEXT	The query text to send for additional promotions.	String
FIELDTEXTRESTRICTION	The optional field text restriction for the query to send for additional promotions.	Percent-encoded string.
DYNAMIC_ DATABASEMATCH	The databases to query for additional promotions.	String
QMS_SCOPE_RULE	(Optional) You can use this field with the PromotionsScope query parameter to return only promotions that match the text that you specify as the scope rule. This allows you to filter the list of returned promotions by user-defined criteria such as a specific department or job function.	String list. Separate items in the list with a comma.

You must also add the ALWAYSACTIVE field with the value **True**, or a SCHEDULE field with iCalendar data. See "Create a QMS Rule IDX Document" on page 45.

Related Topics

- "Create Query Manipulation Server Rules" on page 45
- "Create a QMS Rule IDX Document" on page 45
- "Scope Rules" on the next page

Example Dynamic Promotion Rule

The following QMS rule returns when you send a query for promotions that contains one or more of the terms *orange*, *melon*, or *fruit* in the query text. QMS then returns the results of an IDOL Server query for the text *pumpkin* from the food database.

```
#DREREFERENCE dynamic_promotion
#DRETITLE dynamic_promo_1
#DREDBNAME ACTIVATED
#DREFIELD QMSAGENTBOOL="orange OR melon OR fruit"
#DREFIELD DYNAMIC_DATABASEMATCH="food"
#DREFIELD PROMOTION_NAME="Top promotions"
#DREFIELD QMSTYPE="3"
#DREFIELD DYNAMIC_QUERY_TEXT="pumpkin"
```

```
#DREFIELD FIELDTEXTRESTRICTION="MATCH{fruit}:TYPE"
#DREFIELD ALWAYSACTIVE="True"
#DRECONTENT
ORANGE MELON FRUIT
#DREENDDOC
```

For example, the following query to QMS returns this promotion:

```
action=Query&Text=melon&Promotions=True
```

It sends the following query to IDOL Server to retrieve the promotion documents:

```
action=Query&Text=pumpkin&FieldText=MATCH{fruit}:TYPE&DatabaseMatch=food
```

> **Note:** For queries where a synonym rule database restriction applies, you can specify multiple databases in the `DatabaseMatch` request parameter by separating them with a plus symbol or a space.

Scope Rules

You can use scope rules with "Static Promotion Rules" and "Dynamic Promotion Rules" to specify that a promotion is relevant to a particular set of users. You can then specify in your promotions queries that you want to return only promotions that match the text that you specified as the scope rule. For example, you might want set up a scope rule for promotions that are of particular relevance to the Human Resources and Legal departments of your company; you can then specify in your query that only promotions that match that scope rule should return.

To set up scope rules

- Add the `QMS_SCOPE_RULE` field to the IDX file for your promotion rule. Set the value of the field to the term that you want to match on when you query the Promotion Agentstore. You can create multiple `QMS_SCOPE_RULE` fields to specify multiple values. For example:

  ```
  #DREFIELD QMS_SCOPE_RULE="HR"
  #DREFIELD QMS_SCOPE_RULE="Legal"
  ```

To use scope rules in your queries

- Send queries with the `PromotionsScope` parameter set to a comma-separated list of terms that you want to match on when you query the Promotion Agentstore, as specified in the `QMS_SCOPE_RULE` fields.

  QMS applies only rules that match at least one of the values in the specified `PromotionsScope`.

Query for Promotions

Use the following procedure to query both static and dynamic promotions.

To query for promotions

- Send queries with the `Promotions` parameter set to **True** to return promotions.

  For example:

```
action=Query&Text=phone&Promotions=True
```

This query searches all published promotions and returns all promotions that match the word *phone*.

When you query for promotions, only the promotion documents return.

To use scope rules in your query

- Send queries with the `PromotionsScope` parameter set to the value of the scope rules that you want to match on.

  For example:

  ```
  action=Query&Text=contract&Promotions=True&PromotionsScope=Legal
  ```

  This query searches all published promotions and returns all promotions where `QMS_SCOPE_RULE` is set to **Legal** that contain the text *contract*. `PromotionsScope=Legal` translates to `FieldText=MATCH{Legal}:QMS_SCOPE_RULE` in the query.

Refer to the *QMS Reference* for more information.

Related Topics

- "Static Promotions" on page 52
- "Dynamic Promotions" on page 54
- "Scope Rules" on the previous page

Intent Based Promotions

QMS supports intent based promotions, whereby documents from promotions that are similar to the interests of the user are automatically boosted in the results.

> **Note:** Intent based ranking is licensed functionality, and is not available by default. Contact HPE for further details.

To run an intent based promotions query, set `IntentRankedQuery` to **True**, and set `Username` to the user name of the user that you want to target. For example:

```
action=Query&Text=phone&Promotions=True&IntentRankedQuery=True&Username=jsmith
```

In this case, QMS queries the Promotion Agentstore a second time, using the profile terms for `jsmith`, against the set of promotion rules originally returned. QMS then returns the IDOL Server documents with the references from the `QMSVALUE1` fields of the static promotion rules that match this second query as intent ranked promotions.

To use intent based promotions, QMS must be able to communicate with a Community component and a Promotions Agentstore component. You can enable this feature in one of the following ways:

- Configure the `[IDOL]` section of the QMS configuration file with the host and port of an IDOL server.
- Create a `[Community]` section and a `[PromotionAgentStore]` section in the QMS configuration file, with `Host` and `Port` parameters that specify the host and port of the Community and Promotion Agentstore components that you want to use. Use this option if the `[IDOL]` configuration file section points to a DAH or IDOL Content component.

Related Topics

- "Static Promotions" on page 52

Chapter 7: Modify Queries

This section describes how to use QMS to modify queries.

For example, QMS can modify queries to add synonymous terms or increase the relevance weighting of certain results.

When QMS receives a query, it forwards the query to the Promotion Agentstore. Any rules that match the query return, and QMS modifies the query according to the rules.

Synonyms, Hyponyms, and Hypernyms

You can use QMS to expand queries to include other terms that are synonymous with the other query terms.

The synonym QMS rules allow you to determine a set of keyword terms that the rule applies to, and then to specify how QMS must expand the query. For example, you can use the following types of rules:

- **Synonym rule**. Include the original query term and add a list of synonymous terms. For example, you could expand a query for *dog* to a query for *dog OR hound OR canine*.
- **Hyponym rule**. Exclude the original query term, and replace it with a list of terms that are hyponyms of this term (that is, the original term could describe all the new terms). For example, you could replace a query for *dog* with a query for *poodle OR retriever OR labrador* and so on.
- **Hypernym rule**. Exclude the original query term, and replace it with a single term that is the hypernym of this term (that is, the new term could describe the original term). For example, you could replace a query for *poodle* with a query for *dog.*

Create Synonym Rules

In all synonym rules, the IDX rule document has the same basic form.

The table shows all the fields that you must include in the IDX file for synonym rules. The value of the `KEYWORDS` and `CONCEPT` fields determines whether the rule is a synonym, hyponym, or hypernym rule.

Required IDX fields for synonym rules

Field	Description	Content type or value
#DREREFERENCE	The reference for the rule.	String
#DRETITLE	The title of the rule.	String
QMSTYPE	The type of the QMS rule.	**4**

Required IDX fields for synonym rules, continued

Field	Description	Content type or value
QMSAGENTBOOL	The AgentBoolean expression used to match the original query text.	Boolean matching expression
KEYWORDS	A list of words to match the query text.	Comma-separated list of words. Percent-encode spaces and punctuation in the keywords.
CONCEPT	The expanded query text that replaces the matched query text.	Replacement string. Percent-encode spaces and punctuation in the string.
DATABASERESTRICTION	The databases to query.	String

To create a synonym rule

- Set the KEYWORDS field to a list of all the synonymous terms, and set the CONCEPT field to the same list.

  QMS expands any query that matches one keyword to include all the other keyword terms.

To create a hyponym rule

- Set the KEYWORDS field to the parent term, and set the CONCEPT field to the list of terms that the parent term describes (the hyponyms).

  QMS alters a query that matches the parent term to include all the hyponyms, but not the original term.

To create a hypernym rule

- Set the KEYWORDS field to a list of the terms that the hypernym describes, and set the CONCEPT field to the parent term (hypernym).

  QMS alters any query that matches one of the list of child terms to query for the hypernym (and not the original term).

Related Topics

Field Dependent Synonym Rules

You can create a synonym rule that matches queries only when certain FieldText is present. For example, you might have a synonym list that applies only for game products, and you do not want to apply the synonyms for DVDs.

Required IDX fields for field dependent synonym rules

Field	Description	Content type or value
#DREREFERENCE	The reference for the rule.	String
#DRETITLE	The title of the rule.	String
QMSTYPE	The type of the QMS rule.	4
QMSAGENTBOOL	The AgentBoolean expression used to match the original query text.	Boolean matching expression
KEYWORDS	A list of words to match the query text.	Comma-separated list of words
CONCEPT	The expanded query text that replaces the matched query text.	Replacement string
DATABASERESTRICTION	The databases to query.	String
QMSFIELDTEXT	The FieldText expression that must match the original query.	FieldText expression
Referenced Fields	All fields that the QMSFIELDTEXT field references must be present in the rule as a field with blank content.	Empty field

Send Synonym Queries

QMS can expand a query to include all terms that are synonymous with the query text. For each query, QMS queries the Promotion Agentstore for lists of synonyms for the query text. It adds the synonyms to the query that it sends to IDOL Server.

For example, if a synonym rule states that *dog* is synonymous with *hound*, QMS expands the query for *dog* to *(dog OR hound)* before it sends it to IDOL Server.

Refer to the *QMS Reference* for more information.

To use synonym queries

1. Create QMS rules containing the synonym lists.
2. Send queries with the ExpandQuery parameter set to **True**.

   If your synonym rules contain a DATABASERESTRICTION field, you must set the SynonymDatabaseMatch parameter to **True** to apply the restriction. By default, QMS does not apply the database restriction.

For example:

```
action=Query&Text=phone&ExpandQuery=True
```

In this query, QMS searches for synonyms that contain the word *phone*. It includes all synonymous terms in the query and returns results for all terms.

Send Synonym Queries that Replace Query Text

You can use QMS to replace all the text in a query with the result of a synonym rule.

By default, QMS replaces terms that match the synonym rule `KEYWORDS` field with all terms that match the synonym rule `CONCEPT` field. Other terms in the query are not changed.

You can set the `SynonymReplace` action parameter in the query to replace all the query text with the `CONCEPT`.

For example, if you have set:

```
#DREFIELD KEYWORDS="labrador"
#DREFIELD CONCEPT="dog"
```

The following query:

```
action=Query&Text=I walk my labrador in the park&ExpandQuery=True
```

expands to:

```
action=Query&Text=I walk my dog in the park
```

However, the following query:

```
action=Query&Text=I walk my labrador&ExpandQuery=True&SynonymReplace=True
```

expands to:

```
action=Query&Text=dog
```

Check Synonym Queries

The XML response from QMS includes the tag `autn:expandedQuery`, which shows the expanded query text that QMS sends to IDOL Server.

For example:

```
<autn:expandedQuery>dog hound canine</autn:expandedQuery>
```

You can use the value of this tag to check that QMS applies the synonym rules correctly.

Related Topics

- "Synonyms, Hyponyms, and Hypernyms" on page 59

Example Synonym Rules

The following rule expands any query for the terms *dog*, *hound*, or *canine* to include all three terms.

```
#DREREFERENCE 695120425336110405
#DRETITLE dog synonyms
#DREDBNAME ACTIVATED
#DREFIELD QMSTYPE="4"
```

```
#DREFIELD QMSAGENTBOOL="dog hound canine"
#DREFIELD KEYWORDS="dog,hound,canine"
#DREFIELD CONCEPT="dog hound canine"
#DREFIELD DATABASERESTRICTION=""
#DRECONTENT dog hound canine
#DREENDDOC
```

This synonym rule expands the following query:

```
action=Query&Text=hound&ExpandQuery=True
```

to:

```
action=Query&Text=dog hound canine
```

The following rule expands any query for the term *dog*, *hound*, or *canine* to include all three terms when the FieldText restricts the results to those that contain the term pets in the Type field.

```
#DREREFERENCE 695120425336110405
#DRETITLE dog synonyms
#DREDBNAME ACTIVATED
#DREFIELD QMSTYPE="4"
#DREFIELD QMSAGENTBOOL="dog hound canine"
#DREFIELD KEYWORDS="dog,hound,canine"
#DREFIELD CONCEPT="dog hound canine"
#DREFIELD DATABASERESTRICTION=""
#DREFIELD QMSFIELDTEXT="MATCH{pets}:Type"
#DREFIELD Category=""
#DRECONTENT dog hound canine
#DREENDDOC
```

This synonym rule expands the following query:

```
action=Query&Text=dog&FieldText=MATCH{pets}:Type&ExpandQuery=True
```

to:

```
action=Query&Text=dog hound canine&FieldText=MATCH{pets}:Type&ExpandQuery=True
```

Related Topics

- "Create Synonym Rules" on page 59

Example Hyponym Rule

The following rule changes any query for the term *dog* to include the names of different breeds of dog, but removes the term *dog*.

```
#DREREFERENCE 695120425336110405
#DRETITLE dog hyponyms
#DREDBNAME ACTIVATED
#DREFIELD QMSTYPE="4"
#DREFIELD QMSAGENTBOOL="dog"
#DREFIELD KEYWORDS="dog"
```

```
#DREFIELD CONCEPT="poodle retriever hound boxer labrador terrier spaniel alsatian
chihuahua bloodhound collie"
#DREFIELD DATABASERESTRICTION=""
#DRECONTENT
dog
#DREENDDOC
```

This hyponym rule expands the following query:

```
action=Query&Text=dog&ExpandQuery=True
```

to:

```
action=Query&Text=poodle retriever hound boxer labrador terrier spaniel alsatian
chihuahua bloodhound collie
```

Related Topics

- "Create Synonym Rules" on page 59

Example Hypernym Rule

The following query changes any query for one of the specific breeds of dog to include only the term
dog.

```
#DREREFERENCE 695120425336110405
#DRETITLE dog hypernyms
#DREDBNAME ACTIVATED
#DREFIELD QMSTYPE="4"
#DREFIELD QMSAGENTBOOL="poodle retriever hound boxer labrador terrier spaniel
alsatian chihuahua bloodhound collie"
#DREFIELD
KEYWORDS="poodle,retriever,hound,boxer,labrador,terrier,spaniel,alsatian,chihuahua,
bloodhound,collie"
#DREFIELD CONCEPT="dog"
#DREFIELD DATABASERESTRICTION=""
#DRECONTENT
poodle retriever hound boxer labrador terrier spaniel alsatian chihuahua bloodhound
collie
#DREENDDOC
```

This hypernym rule changes the following query:

```
action=Query&Text=chihuahua&ExpandQuery=True
```

to:

```
action=Query&Text=dog
```

Related Topics

- "Create Synonym Rules" on page 59

Whitelists and Blacklists

QMS can apply rules to remove certain words from queries.

A *whitelist* is a list of all words that are allowed in query text. QMS removes all other words from the query.

A *blacklist* is a list of words that are not allowed in query text. QMS removes them from the query.

You can use whitelists and blacklists for the `Query` and the `GetQueryTagValues` action, and QMS applies the rules to the terms in the `Text` parameter.

Create Whitelist and Blacklist Rules

A *whitelist* is a list of words that are allowed in queries. QMS removes any words that are not in the whitelist from the query before it sends the query to IDOL Server. If a rule defines a whitelist, it must have a `QMSTYPE` field with the value **5**.

Conversely, a *blacklist* is a list of words that are not allowed in queries. QMS removes any words on the blacklist from the query before it sends the query to IDOL Server. If a rule defines a blacklist, it must have a `QMSTYPE` field with the value **6**.

The whitelist or blacklist rule must contain a list of words that form the list. The table shows all the fields that you must include in the IDX file for whitelist and blacklist rules.

Required IDX fields for whitelist and blacklist rules

Field	Description	Content type or value
`#DREREFERENCE`	The reference for the rule.	String
`#DRETITLE`	The title of the rule.	String
`QMSTYPE`	The type of the QMS rule.	**5** (whitelist rule) **6** (blacklist rule)
`KEYWORDS`	The set of words that form the list.	Comma-separated list of words. Percent-encode spaces and punctuation in the words.

Related Topics

- "Example Blacklist Rule" on the next page
- "Example Whitelist Rule" on page 67

Send Whitelist and Blacklist Queries

When you query QMS, you can add a whitelist or blacklist to the query to modify the query.

To use a whitelist or blacklist

- Send queries with one of the following action parameters:

  - the `Whitelist` parameter set to the name of the whitelist to use.

  - the `Blacklist` parameter set to the name of the blacklist to use.

For example:

```
action=Query&Text=phone&Whitelist=AllowList
```

```
action=GetQueryTagValues&Text=phone&FieldName=MODEL&Whitelist=AllowList
```

These actions apply the `AllowList` whitelist, and remove any terms from the specified `Text` that are not in the whitelist.

```
action=Query&Text=mobile phone&Blacklist=BanList
```

```
action=GetQueryTagValues&Text=phone&FieldName=MODEL&Blacklist=BanList
```

These actions applies the `BanList` blacklist, and remove any terms from the specified `Text` that appear in the blacklist.

Refer to the *QMS Reference* for more information.

Check Whitelist and Blacklist Rules

You can check that QMS has applied a whitelist or blacklist correctly by sending the `GetRequestLog` (or `GRL`) action to IDOL Server:

```
http://localhost:9000/action=GRL
```

This action returns a list of all the actions that have been sent to IDOL Server. You can match the query that you sent to QMS to the query that IDOL Server receives from QMS, and check that the correct terms are retained or removed.

You can also check the results list to ensure that the `<autn:links>` tags in each result include only terms that are allowed according to the list that you used. This tag includes a list of all query terms that match in the result document.

Example Blacklist Rule

The following rule removes the terms *beer* and *chicken* from a query, if they are present.

```
#DREREFERENCE 664253145850447559
#DRETITLE blacklist01
#DREDBNAME ACTIVATED
#DREFIELD THRESHOLD="20"
#DREFIELD KEYWORDS="beer,chicken"
#DREFIELD QMSTYPE="6"
#DREFIELD DRELANGUAGETYPE="English"
#DREFIELD DREOUTPUTENCODING="UTF8"
#DRECONTENT
#DREENDDOC
```

This blacklist alters the following query:

```
action=Query&Text=beer wine lemonade&Blacklist=blacklist01
```

to:

```
action=Query&Text=wine lemonade
```

Related Topics

- "Create Whitelist and Blacklist Rules" on page 65

Example Whitelist Rule

The following rule allows the terms *cow*, *goat*, and *deer* in a query, and removes all other terms.

```
#DREREFERENCE 5710712692228798215
#DRETITLE whitelist01
#DREDBNAME ACTIVATED
#DREFIELD THRESHOLD="20"
#DREFIELD KEYWORDS="cow,goat,deer"
#DREFIELD QMSTYPE="5"
#DREFIELD DRELANGUAGETYPE="English"
#DREFIELD DREOUTPUTENCODING="UTF8"
#DRECONTENT
#DREENDDOC
```

This whitelist alters the following query:

```
action=Query&Text=cow goat sheep chicken&Whitelist=whitelist01
```

to:

```
action=Query&Text=cow goat
```

Related Topics

- "Create Whitelist and Blacklist Rules" on page 65

Boost Rules

QMS can add FieldText to queries that it sends to IDOL Server, for example to boost results for a particular product. Boost rules are also known as FieldText expansion rules.

QMS forwards queries to the Promotion Agentstore, which returns any boost rules that match the query. QMS then appends the FieldText to the query that it sends to IDOL Server.

Enable Boost Rules

To use boost rules, you must enable them in the QMS configuration file.

To enable boost rules

1. Open the QMS configuration file in a text editor.

2. In the [Server] section, set the ExpandFieldText parameter to **True**.

3. Save and close the QMS configuration file. Restart QMS for your changes to take effect.

Refer to the *QMS Reference* for more information.

Create Boost Rules

Boost rules allow you to add FieldText to queries to IDOL Server. For example, if a user searches for *televisions*, you can add FieldText to the query that boosts results for a particular brand of television.

The table shows all the fields that you must include in the IDX file for boost rules.

> **Note:** You must add any fields that are referenced in the FieldText expressions to the document as empty fields.

Required IDX fields for boost rules

Field	Description	Content type or value
#DREREFERENCE	The reference for the rule.	String
#DRETITLE	The title of the rule.	String
QMSTYPE	The type of the QMS rule.	7
QMSAGENTBOOL	The AgentBoolean expression used to match the original query text.	Boolean matching expression. Set this field to ∗ to match all query text.
DATABASERESTRICTION	Restrict the query to these databases.	String
QMSFIELDTEXT	The FieldText rules to match against.	FieldText expression
Referenced Fields	All fields that the QMSFIELDTEXT field references must be present in the rule as a field with blank content.	Empty field
CONCEPT	FieldText to append.	FieldText expression
AlwaysMatchType field	An AlwaysMatchType field to match the rule.	Any non-empty value of anything other than 0

Use Boost Rules in Queries

QMS does not apply boost rules by default. You must enable them in individual queries to QMS.

To use boost rules in your query

- Send queries with the `ExpandQuery` parameter set to `True`.

  > **Note:** The query must have a `FieldText` parameter to match boost rules. This `FieldText` can use only the `MATCH` and `EQUAL` field operators.

For example:

```
action=Query&Text=phone&FieldText=MATCH{UK}:place&ExpandQuery=True
```

This query searches for boost rules that match the word *phone,* and that contain the `FieldText` expression `MATCH{UK}:place`. QMS then adds the additional FieldText from the rule to the query to IDOL Server.

> **Note:** This query also searches for synonymous terms.

If your boost rules contain a `DATABASERESTRICTION` field, set the `SynonymDatabaseMatch` parameter to `True` to apply the restriction. By default, QMS does not apply the database restriction.

Refer to the *QMS Reference* for more information.

Related Topics

- "Create Boost Rules" on the previous page
- "Create Query Manipulation Server Rules" on page 45
- "Synonyms, Hyponyms, and Hypernyms" on page 59

Check Boost Rules

You can check that QMS has applied a boost rule correctly by sending the `GetRequestLog` (or `GRL`) action to IDOL Server:

```
http://IDOLhost:port/action=GRL
```

where,

`IDOLhost`	is the IP address or name of the machine on which IDOL Server is installed.
`port`	is the ACI port by which you send actions to IDOL Server (set by the `Port` parameter in the `[Server]` section of the IDOL Server configuration file).

This action returns a list of all the actions that have been sent to IDOL Server.

You can match the query that you sent to QMS to the query that IDOL Server receives from QMS, and check that the FieldText has been correctly appended.

Example Boost Rule

The following boost rule adds the FieldText `BIASVAL{chicken,70}:MAIN_INGREDIENT` to a query that contains the FieldText `MATCH{American}:CUISINE`.

```
#DREREFERENCE BOOST_RULE_27_1254181469986_eawn
#DRETITLE boost rule 5
```

```
#DREFIELD QMSTYPE="7"
#DREFIELD DATABASERESTRICTION="Recipes"
#DREFIELD QMSAGENTBOOL="*"
#DREFIELD QMSFIELDTEXT="MATCH{American}:CUISINE"
#DREFIELD COUNTRY=""
#DREFIELD CONCEPT="BIASVAL{chicken,70}:MAIN_INGREDIENT"
#DRECONTENT
AUTNMATCHCAT
#DREENDDOC
```

This rule expands the following query:

```
action=Query&Text=fried&ExpandQuery=True&SynonymDatabaseMatch=True&DatabaseMatch=Re
cipes&FieldText=MATCH{American}:CUISINE
```

to:

```
action=Query&Text=fried&DatabaseMatch=Recipes&FieldText=MATCH{American}:CUISINE AND
BIASVAL{chicken,70}:MAIN_INGREDIENT
```

Chapter 8: Manipulate Results

This section describes how to use QMS to manipulate the results that IDOL Server returns. QMS can manipulate results so that they always appear in a specific position in the results list. You can also configure QMS to use intent based ranking so that results that are close to the interests of the user are automatically promoted to the top of the results list.

Cardinal Placement

You can use QMS to ensure that a query or parametric query result appears at a specific position in the results list. This process is known as *cardinal placement*.

There are two different types of cardinal placement that you can use in QMS:

- **Cardinal placement for documents**. QMS places a document at a specific location in a results list. For example, if a user queries for *phone*, you can display a result for a specific phone model at the top of the results list.
- **Parametric cardinal placement**. QMS orders a parametric list with an item at a specific position in the list. For example, if a user queries for *phone* and you provide a list of phone models to narrow the search, you can display a specific model at the top of the list.

QMS automatically queries for cardinal placement and parametric cardinal placement rules. If these rules exist, cardinal placements return by default.

Related Topics

- "Cardinal Placement for Documents" below
- "Parametric Cardinal Placement" on page 74

Cardinal Placement for Documents

QMS can place selected documents at certain positions in a results list, regardless of document relevance or other factors. For example, you might want a certain car advertisement to be the first result at all times for the query *"sports car"*. Cardinal placement can ensure that this placement occurs.

QMS forwards all queries to IDOL Server or the Content component to retrieve results. It also forwards the queries to the Promotion Agentstore component, which returns any cardinal placement rules for the specified query. When QMS receives the results, it inserts any cardinal placement documents at the correct position in the results list before it returns results to the client.

QMS moves the cardinal placement document to the correct position if it already exists in the result list from IDOL Server.

Cardinal Placement Rules

Cardinal placement rules allow you to insert documents at a specified location in a results list for a query.

The table shows all the fields that you must include in an IDX file for cardinal placement rules.

Required IDX fields for cardinal placement rules

Field	Description	Content type or value
#DREREFERENCE	The reference for the rule.	String
#DRETITLE	The title of the rule.	String
QMSTYPE	The type of the QMS rule.	**1**
QMSAGENTBOOL	The AgentBoolean expression used to match the original query text.	Boolean matching expression
QMSVALUE1	The reference of the document to insert.	String
QMSVALUE2	The position at which to insert the cardinal placement document in the results.	Number (result position)
#DRECONTENT	The content to match the Boolean field.	Keywords from the AgentBoolean string

You must also add the ALWAYSACTIVE field with the value **True**, or a SCHEDULE field with iCalendar data. See "Create a QMS Rule IDX Document" on page 45.

Related Topics

- "Create Query Manipulation Server Rules" on page 45
- "Create a QMS Rule IDX Document" on page 45

Field-Dependent Cardinal Placement Rules

You can use a cardinal placement rule to insert a document in a results list based on FieldText in the original query. For example, you can insert a value at the top of results when users search for all products in a particular category.

The table shows the fields that you must add to your cardinal placement rule IDX document to include field-dependent rules.

Required IDX fields for field-dependent cardinal placement rules

Field	Description	Content type or value
#DREREFERENCE	The reference for the rule.	String
#DRETITLE	The title of the rule.	String

Required IDX fields for field-dependent cardinal placement rules, continued

Field	Description	Content type or value
QMSTYPE	The type of the QMS rule.	**1**
QMSAGENTBOOL	The AgentBoolean expression used to match the original query text.	Boolean matching expression
QMSVALUE1	The reference of the document to insert.	String
QMSVALUE2	The position at which to insert the cardinal placement document in the results.	Number (result position)
#DRECONTENT	The content to match the Boolean field.	Keywords from the AgentBoolean string
QMSFIELDTEXT	FieldText that the query must contain to match this rule.	FieldText expression.
Referenced Fields	All fields that the QMSFIELDTEXT field references must be present in the rule as a field with blank content.	Empty field

Example Cardinal Placement Rules

The following cardinal placement rule inserts the document with the reference 21764 as the first result for any query that contains the terms cod, halibut, or haddock.

```
#DREREFERENCE 659768056580090736
#DRETITLE cardinal_placement01
#DREFIELD QMSTYPE="1"
#DREFIELD ALWAYSACTIVE="True"
#DREFIELD QMSVALUE1="21764"
#DREFIELD QMSVALUE2="1"
#DREFIELD QMSAGENTBOOL="cod OR halibut OR haddock"
#DRECONTENT
cod halibut haddock
#DREENDDOC
```

This rule inserts the document with the reference 21764 at the top of the response to the following query:

```
action=Query&Text=halibut
```

The following cardinal placement rule inserts the document with the reference FilmBoxSet as the first result for any query for all products that have DVD in the Category field.

```
#DREREFERENCE 746297379561765222
#DRETITLE cardinal_placement02
#DREFIELD QMSTYPE="1"
#DREFIELD ALWAYSACTIVE="True"
#DREFIELD QMSVALUE1="FilmBoxSet"
```

```
#DREFIELD QMSVALUE2="1"
#DREFIELD QMSAGENTBOOL="*"
#DREFIELD ALWAYSMATCH="1"
#DREFIELD QMSFIELDTEXT="MATCH{DVD}:Category"
#DREFIELD Category=""
#DRECONTENT
#DREENDDOC
```

This rule inserts the document with reference `FilmBoxSet` at the top of the response to the following query:

```
action=Query&Text=*&FieldText=MATCH{DVD}:Category
```

Check Cardinal Placements

When a cardinal placement document returns, the XML response includes a `<autn:qmsstate>`, which indicates how many cardinal placement documents were added to the results.

The cardinal placement document result also includes a `<DOCUMENT>` tag section that specifies that the document is a cardinal placement, and the reference of the rule. For example:

```
<DOCUMENT>
        <INJECTEDPROMOTION>TRUE</INJECTEDPROMOTION>
        <QMSID>cardinal_doc_placement01</QMSID>
</DOCUMENT>
```

This XML section specifies that the result is a cardinal placement, according to the rule `cardinal_doc_placement01`.

Parametric Cardinal Placement

You can create cardinal placement rules for parametric queries.

In a parametric query, IDOL Server returns a list of all possible values for a field. For example, if documents in IDOL Server contain a `model` field, a parametric query for `model` returns all the possible values of this field that occur in documents. Users can refine their queries by selecting one of the available models.

Parametric Cardinal Placement Rules

Parametric cardinal placements allow you to insert a parametric value at a particular position in a list. For example, if users restrict their queries by using a list of models of car, a parametric cardinal placement can ensure that a particular model appears at the top of the list.

The table shows all the fields that you must include in the IDX file for parametric cardinal placement rules.

Required IDX fields for parametric cardinal placement rules

Field	Description	Content type or value
#DREREFERENCE	The reference for the rule.	String
#DRETITLE	The title of the rule.	String
QMSTYPE	The type of the QMS rule.	8
QMSAGENTBOOL	The AgentBoolean expression used to match the original query text.	Boolean matching expression
QMSFIELDNAME	The name or the parametric field.	String
QMSFIELDVALUE	The field value to insert in the list of results.	String
QMSVALUE	The position at which to insert the cardinal placement value in the results.	Number (result position)
#DRECONTENT	The content to match the Boolean field.	Keywords from the AgentBoolean string

You can also create parametric cardinal placement rules with field text. In this case, you must add fields to determine the FieldText rules to match against in addition to the fields for other parametric cardinal placement rules. You must also add any fields that are referenced within the FieldText as empty fields.

FieldText fields to add for parametric cardinal placement rules

Field	Description	Content type or value
QMSFIELDTEXT	The FieldText rules to match against.	FieldText expression
Referenced Fields	All fields that are referenced in the QMSFIELDTEXT field must be present in the rule as a field with blank content.	Empty field

Related Topics

- "Create Query Manipulation Server Rules" on page 45

Example Parametric Cardinal Placement Rule

The following parametric cardinal placement rule inserts the value spinach at the top of a query for the values of the field VEGETABLE that includes the terms *carrot* or *broccoli*.

```
#DREREFERENCE automation_cardinal_placement
#DRETITLE cardinal placement 1
#DREFIELD QMSTYPE="8"
#DREFIELD QMSAGENTBOOL="carrot OR broccoli"
#DREFIELD QMSFIELDNAME="VEGETABLE"
```

```
#DREFIELD QMSFIELDVALUE="spinach"
#DREFIELD QMSFIELDTEXT=""
#DREFIELD QMSVALUE="1"
#DRECONTENT
carrot broccoli
#DREENDDOC
```

For example, this rule inserts the value spinach at the top of the list of values in the following query:

```
action=GetQueryTagValues&Text=broccoli&FieldName=VEGETABLE
```

Check Parametric Cardinal Placements

You can check that a cardinal placement has occurred correctly by comparing the response from QMS with the result that IDOL Server returns when you send the same query directly to it.

You can also use the request log for the Promotion Agentstore to check that a rule returned. Send the GRL action to the Promotion Agentstore. You can click the query in the log to run it, and see if a parametric cardinal placement rule returns.

Intent Based Ranking

QMS can support intent based ranking, whereby results that are close to the interests of the user are automatically promoted to the top of the results list.

> **Note:** Intent based ranking is licensed functionality, and is not available by default. Contact HPE for further details.

To use intent based ranking, QMS must be able to communicate with a Community component. You can enable this feature in one of the following ways:

- Configure the [IDOL] section of the QMS configuration file with the host and port of an IDOL server.
- Create a [Community] section in the QMS configuration file, with Host and Port parameters that specify the host and port of the Community component that you want to use. Use this option if the [IDOL] configuration file section points to a DAH or IDOL Content component.

Intent Ranked Query Parameters

You can optionally configure the following parameters for intent ranked queries.

Parameter	Description	Required
DefaultIRQCorpusSize	The default value for the IRQCorpusSize action parameter in intent ranked queries.	Yes
IntentRankedQuery	Set IntentRankedQuery to True to activate intent based ranking.	No
IRQCorpusSize	The size of the pool of intent ranked results.	No

Parameter	Description	Required
RegenerateUsersTermCache	Whether to retrieve the user terms from the Community component for this query.	No
SoftCacheMaxSize	The size (in bytes) of the QMS soft state cache, which is used to store terms for intent ranking.	Yes
Username	For intent ranked queries, the user name of the user that you want to rank the queries for.	Yes

Refer to the *QMS Reference* for more information.

Related Topics

- "Configuration File Sections" on page 31

Chapter 9: Use QMS TypeAhead

This section describes how to use QMS to provide query completion. You can provide a string to the QMS `TypeAhead` action, and it returns a list of suggested completions for the string. For example, if you type *go*, it might return *government* and *golf* as possible options.

Index Mode

In Index Mode, the `TypeAhead` action uses the configured IDOL Server data index to retrieve suggestion values. The action sends the string that you provide in a `TermExpand` action to the index Content component or DAH, and returns the expansions as the suggested values.

You can add the `Stemming`, `Type`, and `Expansion` action parameters from the IDOL `TermExpand` action to your `TypeAhead` action to modify how QMS requests the terms. You can also add the `FieldRestriction` parameter to the `TypeAhead` action to specify the fields that you want the expanded values to come from. For more information, refer to the *QMS Reference*.

Related Topics

- "Filter Results" on page 82

Dictionary Mode

This section describes how to set up and use QMS query completion with a dictionary file.

Related Topics

- "Filter Results" on page 82

Create a Dictionary File

To use the `Dictionary` mode for the `TypeAhead` action, you must provide a dictionary file. This file contains a list of the words that you want to use as suggestions. It optionally also contains a score for the words, to determine the order in which they return in a results list.

The dictionary file must have one word on each line, in the format:

word,score

where *word* is the word that you want to use as a suggestion, and *score* is the score for that word. The score can be a positive or negative integer (terms with a higher score return first in a results list). If you do not add a score value, the default score is zero. The minimum allowed score value is -2^{30}, and the maximum allowed score is $2^{30}-1$

For example:

```
gold,15
government,25
golf,20
golfer,19
gopher,12
gotcha,-10
gone
```

Configure the Dictionary File

After you create a dictionary file, you must configure the location of the file in the QMS configuration file.

To configure the dictionary file

1. Open the QMS configuration file in a text editor.

2. Add a `[TypeAhead]` configuration section.

3. In the `[TypeAhead]` section, set the `DictionaryFile` parameter to the path to the dictionary file that you want to use. For example:

   ```
   [TypeAhead]
   DictionaryFile=C:\Dictionaries\QMS_Typeahead.txt
   ```

4. Save and close the QMS configuration file. Restart QMS for your changes to take effect.

Manage Dictionaries

After you have created and configured the dictionary file, you can use the `TypeAheadManage` action to modify the dictionary file. This action ensures that you do not have to restart the server when you want to update the dictionary.

Add Values

To add values to the dictionary, send a `TypeAheadManage` action with:

- `Mode` set to **Dictionary**.
- `ManageMode` set to **Add**.
- `Text` set to a list of the *word,score* pairs that you want to add to the dictionary. Separate each pair with a semicolon. The score value is optional. If you do not add a score, the term is added with a score of zero. If you include a score, the minimum allowed score value is -2^{30}, and the maximum allowed score is 2^{30}-1.

For example:

```
action=TypeAheadManage&Mode=Dictionary&ManageMode=Add&Text=goal,25;gothic,5
```

You can also upload the changes to the dictionary in a file. The file must contain the words and scores that you want to add to the dictionary. It must have the same format as the dictionary file (see "Create a Dictionary File" on the previous page). You upload the file by using the `ManageFile` parameter. For

more information about uploading a file by using the `ManageFile` parameter, refer to the *QMS Reference*.

Modify Values

To modify the values in the dictionary, send a `TypeAheadManage` action with:

- `Mode` set to **Dictionary**.
- `ManageMode` set to **Edit**.
- `Text` set to a list of the *word,score* pairs that you want to modify in the dictionary. Separate each pair with a semicolon. The score value is optional. If you do not specify a score, the existing score for the term is replaced with the default value of zero. If you include a score, the minimum allowed score value is -2^{30}, and the maximum allowed score is 2^{30}-1.

For example:

```
action=TypeAheadManage&Mode=Dictionary&ManageMode=Edit&Text=government,15;golf
```

You can also upload the changes to the dictionary in a file. The file must contain the words and scores that you want to modify in the dictionary. It must have the same format as the dictionary file (see "Create a Dictionary File" on page 79). You upload the file by using the `ManageFile` parameter. For more information about uploading a file by using the `ManageFile` parameter, refer to the *QMS Reference*.

Remove Values

To remove values to the dictionary, send a `TypeAheadManage` action with:

- `Mode` set to **Dictionary**.
- `ManageMode` set to **Remove**.
- `Text` set to a list of the *word,score* pairs that you want to remove from the dictionary. Separate each pair with a semicolon. The score value is optional. If you include a score, the minimum allowed score value is -2^{30}, and the maximum allowed score is 2^{30}-1.

For example:

```
action=TypeAheadManage&Mode=Dictionary&ManageMode=Remove&Text=gold;golfer
```

You can also upload the changes to the dictionary in a file. The file must contain the words that you want to remove from the dictionary. It must have the same format as the dictionary file (see "Create a Dictionary File" on page 79). You upload the file by using the `ManageFile` parameter. For more information about uploading a file by using the `ManageFile` parameter, refer to the *QMS Reference*.

Save Changes to the Dictionary

When you have finished making modifications to the dictionary file, you must send another `TypeAheadManage` action to save those changes to disk. If you do not persist the changes, QMS continues to use your updated values for the dictionary, but if you restart the server, the changes are lost.

To save dictionary changes to disk, send a `TypeAheadManage` action with:

- `Mode` set to **`Dictionary`**.
- `ManageMode` set to **`Persist`**.

For example:

```
action=TypeAheadManage&Mode=Dictionary&ManageMode=Persist
```

Filter Results

In some cases, you might want to restrict the results that a user receives as suggestions. You can apply filters to the `TypeAhead` action, depending on the mode that you are using.

- In **Index** mode, you can use a `SecurityInfo` string with the `TypeAhead` action. QMS forwards this string to the IDOL index, and returns only terms that occur in documents that the user is permitted to see.
- In both **Index** and **Dictionary** mode, you can provide a Lua script that filters the suggestions that the `TypeAhead` action returns.

Use a SecurityInfo String

When you are using the data index to provide suggestions, you can use the IDOL security functionality to ensure that users see only suggestions that occur in at least one document that they are permitted to see.

You can set the `SecurityInfo` parameter in the `TypeAhead` action. QMS forwards this value to the IDOL Content component, which filters the results according to user permissions. You can generate the `SecurityInfo` string for a user by using the `UserRead` action for the IDOL Server Community component. For more information, refer to the *IDOL Server Reference*.

Use a Lua Script to Filter Results

You can configure and use a Lua script to filter the suggestions according to your own filtering criteria.

The Lua script that you create to filter results must contain a valid Lua function named `typeahead_filter` that accepts a string value and returns a Boolean value.

QMS calls the Lua script once for each suggestion that the `TypeAhead` action returns. It passes the suggestion to the `prefix` argument of the `typeahead_filter` function. The following script is a very simple example, which filters out the suggestion *gold*, if it returns from QMS:

```
function typeahead_filter (prefix)
   if prefix == "gold" then
      return false
   end
   return true
end
```

When you have a script configured, the `TypeAhead` action returns only suggestions for which the `typeahead_filter` function returns **`True`**. In the example above, the suggestion *gold* returns **`False`**, so QMS does not show it in the response for the `TypeAhead` action.

To configure QMS to use a Lua script for filtering

1. Open the QMS configuration file in a text editor.

2. Find the `[TypeAhead]` configuration section, or create one if it does not exist.

3. Set the `Script` parameter to the name of the script file that you want to use. For example:

```
[TypeAhead]
Script=filter_script.lua
```

4. Save and close the QMS configuration file. Restart QMS for your changes to take effect.

Part 3: Appendixes

This section includes the following appendixes:

- "Record Statistics with Statistics Server"

Appendix A: Record Statistics with Statistics Server

This appendix describes how to set up and use the Statistics Server, and lists its parameters.

About Statistics Server

The IDOL Statistics Server monitors interactions between one or more IDOL databases and end users. Interactions can occur in two ways:

- Users can send actions directly to IDOL through a Web browser.
- Users can interact with IDOL by using a front-end application such as Find, or a third-party application.

The Statistics Server monitors IDOL log files for queries or actions that users send to the database, then uses that data to report statistics.

You can determine what statistics the server measures by defining statistical criteria in the Statistics Server configuration file. These statistics can include the following examples:

- actions that do not return any results.
- the top 25 queries in the past day, week, month, and so on.
- the average number of queries in a particular time period.
- the total number of hits for a specific term.

Statistics are entirely user-defined: a flexible set of parameters allows you to measure a wide range of statistics. You can also specify how often the statistics are reported.

When a user interacts with IDOL, a script or the front-end application sends the Statistics Server an XML record, known as an *XML event*. If the event matches the criteria of any of the configured statistics, it is included in the statistical tally.

You can use statistics for many reasons; to construct a profile of end users, to see which queries or terms are most popular, to refine promotions, and so on.

You can install Statistics Server when you install QMS with the IDOL Server Installer. It is primarily intended for use with IDOL Search Optimizer.

> **Tip:** You can view information on ACI requests, errors, and other statistical information about your server or component in the IDOL Admin user interface. For more information, refer to the *IDOL Admin User Guide*.

Configuration

To configure Statistics Server, you must:

- configure XML events to be sent to the Statistics Server
- configure basic settings
- configure statistics

> **Note:** There is no default configuration file for Statistics Server; you must create a `statsserver.cfg` file in the same directory as the `statsserver.exe` file.

Create XML Events

Before you can configure statistics, you must configure the XML events to send to Statistics Server. Specifically, you must set up fields that contain values that you want to measure; you can then configure statistics to measure these fields.

There are two ways to configure XML events:

- Write a script to generate the events. For a sample PERL script, see "Sample XML Event Script" on page 94.
- Use a third-party front end that can generate events.

After you configure the events, the XML files that Statistics Server receives must contain the desired fields. For example:

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<events>
  <queryinfo>
    <ver>0.1</ver>
    <id>10385792</id>
    <url>
     <![CDATA[http://content:19352/action=query&text=dog&numhits=6]]>
    </url>
    <action>query</action>
    <terms><term>dog1</term></terms>
    <duration>10</duration>
    <numhits>5</numhits>
    <type>16</type>
    <user>user_name</user>
    <ip>127.0.0.1</ip>
  </queryinfo>
</events>
```

In this example, any of the fields in the `<queryinfo>` tag can be used for statistics.

Configure Statistics Server Information

You must configure the Statistics Server information in the `[License]`, `[Service]`, `[Server]`, and optionally, the `[Path]` sections of the Statistics Server configuration file.

To configure basic settings

1. Open the Statistics Server configuration file (the default file name is `statsserver.cfg`).

2. Specify the license information in the `[License]` section of the Statistics Server configuration file.For a list of standard `[License]` parameters, refer to the *Query Manipulation Server Reference*.

3. Specify the service information in the `[Service]` section of the Statistics Server configuration file. For a list of standard `[Service]` parameters, refer to the *Query Manipulation Server Reference*.

4. If you are recording statistics from multiple IDOL components, you must specify the IDOL information in the `[IDOLStatistics]` section of the Statistics Server configuration file. For more information, see "Record Statistics from Multiple IDOL Servers" on page 91.

5. Configure Statistics Server information in the `[Server]` and `[Path]` sections of the Statistics Server configuration file, including which port receives events, and which clients can send events to Statistics Server. For more information, see "Statistics Server Parameters" on page 100.

6. Save the configuration file.

Define Statistical Criteria

After you configure the Statistics Server information, you must define criteria for each of the statistics that you want to measure.

To configure statistics

1. Open the QMS configuration file in a text editor.

2. Find the `[Statistics]` section, and list the statistics that you want Statistics Server to process.

   For example:

   ```
   [Statistics]
   0=count_queries_hour
   1=topn_zerohits_week
   2=count_corrections_month
   3=topn_suggestions_day
   ```

3. For each statistic that you are using, create a section using the name of the statistic.

   In this section, specify the criteria that define the statistic. For details on the configuration parameters that you can use, see "Statistical Criteria Parameters" on page 106.

4. You must add the `Field`, `Operation`, and `Period` parameters to each section. If you are recording statistics from multiple IDOL components, you must also set the `IDOLName` parameter.

   For example:

   ```
   [count_queries_hour]
   Operation=count
   Field=queryinfo/terms/term
   ```

```
Period=3600
IDOLName=myIDOL
```

5. Save the configuration file.

Record and View Statistics

After you have configured the Statistics Server, you can begin to record statistics and view statistical results. You can start the Statistics Server either through a command prompt, or as a Windows service.

To install Statistics Server as a Windows service, run `statsserver.exe -install` at the command prompt. To uninstall it as a service, run `statsserver.exe -uninstall`. For more information, refer to the *IDOL Getting Started Guide*.

Record Statistics

To record statistics, you must ensure that IDOL, Statistics Server, and the front-end application are all running.

To record statistics

1. Start IDOL.

2. Start Statistics Server with one of the following options:

    -   Open a command prompt in the Statistics Server installation directory, and run `statsserver.exe`.

    -   Open Windows Services, and start the `statsserver` service.

3. Run the XML event script either from the front-end application (if the option exists) or from the command line. If you run the script from the command line, ensure that you identify the IDOL log files that you want to monitor, as in the following example using a PERL script:

    `perl XMLscript.pl datafile.log`

    where `XMLscript.pl` is the script and `datafile.log` is the IDOL log file.

    > **Note:** Certain applications might generate XML events or run scripts automatically.

4. If you are using a front-end application, start it.

    The Statistics server begins recording statistical data.

View Statistical Results

There are two ways to view the statistical results recorded by Statistics Server:

Use the Front-end Application

The way in which the data appears depends on the application that you are using. To configure or change the display, refer to the documentation for the application.

Use the `StatResult` Action

If you are not using a front-end application, or you simply want to view results in a Web browser, you can use an action.

Related Topics

- "StatResult" on page 115.

Record Statistics from Multiple IDOL Servers

If you are working with multiple Query Manipulation Servers, you might want to collect statistics from some or all of them.

To configure Statistics server for multiple Query Manipulation Servers

1. Ensure that the XML event output has a field that contains the name of the Query Manipulation Server.
2. Create an `[IDOLStatistics]` section in the Statistics Server configuration file.
3. In the `[IDOLStatistics]` section, set the `Number` and `EventField` parameters.
4. In the `[Statistics]` section, set the `IDOLName` parameter for each statistic.
5. Save the configuration file.

Related Topics

- "Create XML Events" on page 88
- "Configuration Parameter Reference" on page 100

For example:

```
[IDOLStatistics]
Number=2
EventField=IDOLServerUsed

[Statistics]
0=querycount1
1=querycount2

[querycount1]
IDOLName=IDOL1
Field=spellingquery
Operation=count
Period=3600

[querycount2]
```

```
IDOLName=IDOL2
Field=spellingquery
Operation=count
Period=3600
```

In this example, Statistics Server collects information from two Query Manipulation Servers. The XML events have an `<IDOLServerUsed>` field that identifies the Query Manipulation Server. The Statistics Server measures two nearly identical statistics, but `querycount1` records only requests sent to the IDOL1 server, and `querycount2` records only requests sent to the IDOL2 server.

Preserve Data during Service Interruptions

Safe mode preserves unreported statistics in case the Statistics Server unexpectedly shuts down or stops responding. Each statistic has a configured time period that determines how often the statistical data is recorded into the report file. If Statistics Server stops responding during the interval, the intermediate data is usually lost. Safe mode preserves this data.

To enable safe mode, set `SafeModeActivated` to **True** in the Statistics Server configuration file. The amount of data preserved in safe mode depends on the time period of the most frequently recorded statistic.

Intermediate data is stored temporarily after every tenth of an interval. For example, if the most frequent statistic has a `Period` value of one hour, intermediate data for all statistics is stored every six minutes. If Statistics Server shuts down unexpectedly, all information up to the most recent of these intermediate intervals is preserved.

When you restart Statistics Server, it resumes from the end of the most recent intermediate interval. For example, if the most frequent statistic `Period` is one hour and the Statistics Server shuts down at 34 minutes, it resumes recording information from the 30-minute mark when it is restarted; all data received between 30 and 34 minutes is lost.

Related Topics

Sample Files

This section contains a sample configuration file and a sample PERL script that generates XML events.

Sample Configuration File

```
[license]

[service]
ServicePort=19873
ServiceControlClients=127.0.0.1
ServiceStatusClients=127.0.0.1
```

```
[server]
EventClients=*.*.*.*
Port=19870
Threads=100
EventPort=19871
EventThreads=8

[statistics]
0=test1
1=test2
2=test3
3=querycount
4=topterm
5=avgdurationquery

[test1]
Operation=count
Field=numhits
Period=60
NEqualStat=numhits,0

[test2]
Operation=topn,100
Field=spellingterms
Period=86400
NRangeStat=numhits,10,20,duration,0,1000

[test3]
Operation=topn,10
Field=spellingterms
Period=60
ARangeStat=term,aardvark,monkey

[querycount]
Operation=count
Field=queryinfo/url
Period=5

[topterm]
Operation=topn,100
Field=queryinfo/terms/term
Period=5

[avgdurationquery]
Operation=average
Field=duration
AEqualStat=action,query
Period=5
```

```
[logging]
LogEcho=True
LogTime=True
LogHistorySize=50
LogMaxSizeKBs=10240
0=APP_LOG_STREAM
1=EVENT_LOG_STREAM

[app_log_stream]
LogFile=application.log
LogTypeCSVs=application
LogLevel=normal
LogExpireAction=datestamp

[event_log_stream]
LogFile=event.log
LogTypeCSVs=event
LogLevel=full
LogExpireAction=datestamp
```

Sample XML Event Script

The following example event script monitors a specified log file.

The IDOL Server installer provides additional example scripts.

```perl
#!/usr/bin/perl

use strict;
use warnings;
use File::Tail;
use Encode;
use URI::Escape;

#--------------------------------------------------

# Script to monitor a specified log file ($ARGV[0]) generated by any IDOL
# servers, constantly tailing it. When a query appears in the logs it
# gathers the appropriate information and sends it to the Stats server
# at the specified host ($ARGV[1]) and port ($ARGV[2]).
#
# The log file has the following format:
# <date> <time> [<thread number>] <log level> <information>
#
# For example:
# (...)
# 12/12/2007 08:22:21 [7] 30-Normal:
#/action=suggest&maxresults=6&reference=doc_123123 (127.0.0.1)
```

```perl
# 12/12/2007 08:22:23 [7] 30-Normal: Returning 6 matches
# 12/12/2007 08:22:23 [7] 30-Normal: Suggest complete
# (...)
#
# Parameters: Four main parameters are required to run the script.
#
# - logfile: The full path and name of the log file.
# - stathost: Host name of the stats server.
# - statport: Event port of the Stats server to which the script can
#send XML events.
# - idolname: IDOL Server name generating the log file that you want to
# monitor. The value must be the same one specified in the Stats server
# configuration file when setting the "IDOLName" stat parameter.
#
# For example: idolname = "myIDOLServer"
# If only one IDOL Server is used, you do not have to use idolname.
#
# Run the script as follows:
#
# perl stats.pl <logfile> <host> <port> <idolname>
#
#----------------------------------------------------------------

use constant XML_HEADER => "<?xml version='1.0' encoding='ISO-8859-
1'?>\n<events>\n";
use constant START_FROM_LOGFILE_END => 0;
use constant READ_ENTIRE_LOGFILE => -1;

if (!defined $ARGV[2]) {
 print "Syntax: $0 logfile host port [idolname]\n";
 exit(0);
}

my $name = $ARGV[0]; # The log file to monitor
my $statshost = $ARGV[1]; # Host of the Stats Server
my $statsport = $ARGV[2]; # Event port of the Stats Server
my $idolname = $ARGV[3]; # IDOL Server name. Set an IDOL Server name if Stats
Server is running in Multiple IDOL Server mode.

my $BATCHSIZE=1; # Wait until we have this many events and send them all at once
my $tailn = START_FROM_LOGFILE_END; # Start tailing from the end
my $resetTail=0; # Start tailing after the file has been automatically closed and
reopened
my $tail; # Structure returned while tailing a file
my %threadqueries; # The 'current query' on each thread
my %threadips; # The 'current ip' on each thread
my %threadtime; # The 'current time' on each thread
my $total = 0; # Number of XML events generated by the script
my %escapes = (); # Hash mapping characters to hex equivalent
```

```perl
my $data = ""; # Data being sent to the StatsServer
my $events = 0;
my $bIsWindows = 0;

sub buildQueryXML($$$$$@); # return event XML for a particular query with matches
and terms
sub postEventsData($$$); # Send XML events to the Stats server
sub processLine($); #Processes a line returned from the log file
sub releaseAndReopenHandle($);

#Are we running on Windows?
eval
{
 require Win32::Process;
};

if ($@) {
 $bIsWindows = 0;
} else {
 $bIsWindows = 1;
}

#-------------------------------
# Main loop
#-------------------------------
for (;;) {
    eval {
        # Tail the specified log file
        $tail = new File::Tail(name => $name, maxinterval => 2, interval => 1, tail
=> $tailn, adjustafter => 1);
        $tailn = START_FROM_LOGFILE_END;

        # Event Creating loop
        for (;;) {
            $data = XML_HEADER;
            $events = 0;

            # Event Writing loop
            while ($events < $BATCHSIZE) {
                # Wait for new input in the log file.
                my ($nfound, $timeleft, @pending) = File::Tail::select(undef,
undef, undef, 1, $tail);

                # Exit if no new line is found
                last if $events > 0 && !$nfound;

                # Returns one line from the log file
                my $line = $tail->read();
```

```perl
                    processLine($line);
                }

                # End of the XML event
                $data .= "</events>\n";

                print STDERR "DATA:\n$data\n";

                # End of the current batch. Post the response to the statsserver.
                if ($events) {
                    $total += $events;

                    eval {
                        postEventsData($statshost, $statsport, $data);
                    };

                    print "Posting events failed: $@" if $@;

                    if ($total % 1000 == 0) {
                        print STDERR "Total events: $total\n";
                    }

                    $data = "";

                    #The file handle used by File::Tail is used permanently until it
                    #($tail) goes out of scope. On Windows, this means it keeps a lock
                    #on the log file we are tailing: therefore, the log file has no
                    #opportunity to roll over after it exceeds the maximum permitted
                    #size. The following call closes the handle, sleeps, and reopens
                    #the handle. This allows the rollover to occur correctly if logging
                    #is made to the log file during the sleep, and the log file size
                    #has overstepped the limit.

                    if ($bIsWindows) {
                        releaseAndReopenHandle($tail);
                    }
                }
            }
        };

        warn unless $tailn;
        $tailn = READ_ENTIRE_LOGFILE;
        sleep 1;
    } #end of the Main loop

    sub processLine($) {
        my $line = shift;

        # Check whether the line contains the following format (that is, 16/09/2008
```

```perl
14:20:00 [1] 30-Normal: ....)
    if ($line =~ m,^(\d\d/\d\d/\d{4} \d\d:\d\d:\d\d) \[(\d+)\] \d\d-
(Full|Normal|Always|Warning|Error): (.*),) {
        # A log line (beginning with a time)
        my $timeLog = $1; # Get the time when the IDOL Server received the query
        my $thread = $2; # The thread number the query is on
        my $message = $4; # The rest of the log line; $3 captures the log level
(Full, Normal, and so on)

        # Check whether the message contains a query (action=termgetbest&...)
        if ($message =~ m,^/?(a.*?=.+) \(([\d\.]+)\)$,) {
            # The initial 'action received'
            $threadqueries{$thread} = $1; # Get the query
            $threadips{$thread} = $2; # Get IP address of the IDOL Server that sent
the query
            $threadtime{$thread} = $timeLog; # Get the query time
        }

        # Check whether the message contains the number of hits, returned matches
or completed query
        elsif ($threadqueries{$thread} && ($message =~ /^Completed Action,
returning (\d+) hits$/ || $message =~ /^Returning (\d+) match/ || $message =~ /^.*
complete$/)) {
            # The end of the query. Form the event xml and save
            my $matches = $1 || 0;
            my $query = $threadqueries{$thread};
            my $ip = $threadips{$thread};

            # If the query format is correct, generate the data for the XML event
            if (defined $query && defined $ip && $query =~ m!/?a.*?=(\w+)([?&].*
(?<=[?&])text=([^?&]*))?!) {
                $events++;
                my $terms = $3 || "";
                $data .= buildQueryXML($idolname, $query, $1, $matches, $ip, split
(/ /,uri_unescape($terms)));
            }

            print STDERR "$threadtime{$thread} $threadqueries{$thread}\n";

            # Reset
            $threadqueries{$thread} = "";
            $threadips{$thread} = "";
            $threadtime{$thread} = "";
        } # end of the action
    } # end of the line
}
#Windows-only: close and reopen the log file handle, with a suitable pause in
#between, to allow the log file to be rolled over.
sub releaseAndReopenHandle($) {
```

```perl
    my $tail = shift;
    my $logfile = $tail->input();
    close($tail->{'handle'});
    sleep(5);
    open($tail->{'handle'}, "<$logfile") or die "Cannot reopen logfile handle:
$!\n";
}

sub stripControlChars($) {
    my $x = shift;
    $x =~ tr/\x00-\x1F//d;
    return $x;
}


#----------------------------------------------------------------------
# Extract information from a query and build the XML to send
#----------------------------------------------------------------------
sub buildQueryXML($$$$$@) {
    my $idolname = shift;
    my $query = shift;
    my $action = shift;
    my $matches = shift;
    my $ip = shift;
    my @terms = @_;
    my $xml = "<queryinfo>\n<ver>0.1</ver>\n<url><![CDATA[$query]]></url>\n";
    $xml .= "<action>$action</action>\n";
    $xml .= "<terms><term>" . uri_escape(stripControlChars($_)) .
"</term></terms>\n" for @terms;
    $xml .= "<numhits>$matches</numhits>\n";
    $xml .= "<ip>$ip</ip>\n";

    if ($idolname) {
        $xml .= "<idolname>$idolname</idolname>\n";
    }

    $xml .= "</queryinfo>\n";
    return $xml;
}


#----------------------------------------------------------------------
# Post a batch of events data to the specified host and port
#----------------------------------------------------------------------
sub postEventsData($$$) {
    my $host = shift;
    my $port = shift;
    my $data = shift;
    my $nConnectTry = 3;

    use Socket;
```

```
    socket(INDEXSOCK, Socket::PF_INET, Socket::SOCK_STREAM, getprotobyname('tcp'))
|| print STDERR "Socket Failure: $!";
    my $inet_addr = Socket::inet_aton($host) || print STDERR "Internet_addr
Failure: $1\n";
    my $paddr= Socket::sockaddr_in($port, $inet_addr) || print STDERR "Sockaddr_in
Failure: $!\n";

    while (!connect(INDEXSOCK, $paddr) && $nConnectTry > 0) {
        $nConnectTry--;
    }

    $nConnectTry > 0 or die "Connect problem: $|\n";
    select INDEXSOCK; $| =1;
    select STDOUT;
    print INDEXSOCK "POST ";
    print INDEXSOCK "/stats";
    print INDEXSOCK " HTTP/1.0\r\n";
    print INDEXSOCK "Content-Length: ".length($data)."\r\n\r\n";
    print INDEXSOCK "$data";
    my $buffer;
    read(INDEXSOCK, $buffer, 100);
    close INDEXSOCK; return(1);
}
```

Configuration Parameter Reference

Statistics Server supports standard logging parameters and log streams. For more information, see "Customize Logging" on page 38. It also supports several unique parameters that you must use when you set up the system and configure statistics.

Statistics Server Parameters

Set general parameters to configure machine information, time display options, and data storage locations.

ActionEvent

Set this parameter to `True` to enable the `Event` action.

Type:	Boolean
Default:	False
Required:	No
Configuration Section:	**[Server]**

Example:	`ActionEvent=True`
See Also:	"Event" on page 113

DateString

Set this parameter to `False` to display time in Epoch time, or `True` to display time in YYYY-MM-DD format.

Type:	Boolean
Default:	False
Required:	No
Configuration Section:	**[Server]**
Example:	`DateString=True`

EventClients

Specify the IP addresses or host names of machines that are permitted to send events to the Statistics Server. Separate multiple values with a comma.

Type:	String
Default:	127.0.0.1
Required:	No
Configuration Section:	**[Server]**
Example:	`EventClients=*.*.*.*`

EventField

Specify the field in the XML event that contains the name of the IDOL Server. Use this parameter if you are using multiple IDOL Servers.

Type:	String
Default:	None
Required:	No
Configuration Section:	**[IDOLStatistics]**

Example:	`EventField=idolname`
See also:	"IDOLName" on the next page
	"Number" on page 104

EventPort

Specify the number of the port that receives events.

Type:	Integer
Default:	None
Required:	No
Configuration Section:	**[Server]**
Example:	`EventPort=19871`

EventThreads

Specify the number of event threads.

Type:	Integer
Default:	8
Required:	No
Configuration Section:	**[Server]**
Example:	`EventThreads=80`

ExternalClock

The `ExternalClock` parameter determines when Statistics Server begins recording events. If you do not use this parameter, the server begins recording events as soon as it starts. If you set a value (in seconds), the Statistics Server uses that value as a timestamp for each statistic. The first time the server receives an XML event with a `<timestamp>` value equal to or greater than the `ExternalClock` value, it begins recording statistics from that point onward. Each subsequent event must have a `<timestamp>` value greater than the previous one to be recorded.

Note: To use `ExternalClock`, configure a `<timestamp>` field in the XML events. For more information, see "Create XML Events" on page 88.

Type:	Integer

Default:	0
Required:	No
Configuration Section:	**[Server]**
Example:	ExternalClock=100

History

Specify the directory in which statistical results are stored.

Type:	String
Default:	./history
Required:	No
Configuration Section:	**[Path]**
Example:	History=./results

IDOLName

Specify the name of the IDOL component from which the statistic is received. Set this parameter if you are using multiple IDOL components.

Note: To use IDOLName, you must configure a field that contains the name of the IDOL component in the XML events. For more information, see "Create XML Events" on page 88.

Type:	String
Default:	None
Required:	No
Configuration Section:	**[*MyStatistic*]**
Example:	IDOLName=IDOL1
See also:	"EventField" on page 101
	"Number" on the next page

Main

Specify the directory in which database files are stored. The database files contain information about the fields defined in the Statistics server configuration file.

Type:	String
Default:	`./main`
Required:	No
Configuration Section:	**[Path]**
Example:	`Main=./dbfiles`

Number

Specify the number of IDOL components from which to measure statistics. Set this parameter if you are using multiple IDOL components.

Type:	Integer
Default:	None
Required:	No
Configuration Section:	**[IDOLStatistics]**
Example:	`Number=2`
See also:	"EventField" on page 101 "IDOLName" on the previous page

Port

Specify the Statistics server ACI port number.

Type:	Integer
Default:	None
Required:	Yes
Configuration Section:	**[Server]**
Example:	`Port=19873`

SafeModeActivated

Set this parameter to `True` to activate safe mode. See "Preserve Data during Service Interruptions" on page 92.

Type:	Boolean
Default:	False
Required:	No
Configuration Section:	**[Server]**
Example:	SafeModeActivated=True

StatLifetime

Specify how long you want to keep statistical data stored on disk. After the specified time period has elapsed, all statistical data is removed. You can specify the period that you want to elapse before deleting statistical data by using the following format:

<number><timeUnit>

where *<number>* is the number of time units that you want to elapse, and *<timeUnit>* is the time unit to specify. The following units are available:

- second
- seconds
- minute
- minutes
- hour
- hours
- day
- days
- week
- weeks
- month
- months

If you do not specify a *timeUnit*, Statistics Server reads the specified number as seconds. You can also enter -1 for an unlimited time (no data is deleted).

Type:	String
Default:	-1 (no data is deleted)
Required:	No
Configuration Section:	**[Server]**
Example:	StatLifetime=1week

TemplateDirectory

Specify the directory that contains an XSL template file, used for reporting.

Type:	String
Default:	None
Required:	No
Configuration Section:	**[Paths]**
Example:	TemplateDirectory=C:\IDOL\IDOLServer\IDOL\templates

Threads

Specify the number of ACI threads.

Type:	Integer
Default:	4
Required:	No
Configuration Section:	**[Server]**
Example:	Threads=100

XSLTemplates

Set this parameter to **True** to enable XSL reporting.

Type:	Boolean
Default:	False
Required:	No
Configuration Section:	**[Server]**
Example:	XSLTemplates=True

Statistical Criteria Parameters

Use the following parameters to configure the statistics that you want to measure.

When you configure XML fields for statistic restrictions (`AEqualStat`, `ARangeStat`, `NEqualStat`, `NRangeStat`, and `BitANDStat`) in the configuration file, you must provide either the full XML path (excluding the root element) or the last branch name.

For example, to create a statistic for the following XML event:

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<events>
  <queryinfo>
    <appIDs>
      <appID>DOMATCH</appID>
    </appIDs>
  </queryinfo>
</events>
```

You can configure the statistic either by using the full XML path:

```
AEqualStat=queryinfo/appIDs/appID,ARCHIVE
```

or by using the last XML branch:

```
AEqualStat=appID,ARCHIVE
```

> **Note:** `AEqualStat=appIDs/appID/,ARCHIVE` is an invalid restriction. As a result, all statistic results are stored on disk, ignoring the restrictions.
>
> On startup, if a statistic restriction setting contains an XML path, a warning message is logged in the `application.log` file to remind the user to check the settings to ensure that it uses a valid path.

AEqualStat

Specify the field and the string value that the field must contain to trigger the statistic.

Enter multiple field-value pairs as comma-separated variables in the form $fieldN,valueN$:

```
AEqualStat=field1,value1,field2,value2
```

Type:	String
Default:	None
Required:	No
Configuration Section:	**[*MyStatistic*]**
Example:	`AEqualStat=action,query`

ARangeStat

Specify the field and the alphabetical range into which a value must fall to trigger the statistic. To set the range, enter the lower and upper limits as comma-separated values.

Enter multiple field-value combinations as comma-separated variables in the form
fieldN,*valueN*1,*valueN*2:

`ARangeStat=field1,value1_1,value1_2,field2,value2_1,value2_2`

Type:	String
Default:	None
Required:	No
Configuration Section:	**[*MyStatistic*]**
Example:	`ARangeStat=term,captain,lieutenant`

BitANDStat

Specify the field and bitwise AND value that the field must match to trigger the statistic.

Enter multiple field-value pairs as comma-separated variables in the form `fieldN,valueN`:

`BitANDStat=field1,value1,field2,value2`

Type:	String
Default:	None
Required:	No
Configuration Section:	**[*MyStatistic*]**
Example:	`BitANDStat=MyOption,5`

NEqualStat

Specify the field and the numeric value that the field must contain to trigger the statistic.

Enter multiple field-value pairs as comma-separated variables in the form `fieldN,valueN`:

`NEqualStat=field1,value1,field2,value2`

Type:	String
Default:	None
Required:	No
Configuration Section:	**[*MyStatistic*]**
Example:	`NEqualStat=numhits,0`

NRangeStat

Specify the field and the numeric range into which a value must fall to trigger the statistic. To set the range, enter the lower and upper limits as comma-separated values.

Enter multiple field-value combinations as comma-separated variables in the form fieldN,valueN1,valueN2:

```
NRangeStat=field1,value1_1,value1_2,field2,value2_1,value2_2
```

Type:	String
Default:	None
Required:	No
Configuration Section:	[*MyStatistic*]
Example:	NRangeStat=numhits,10,20

DynamicField

Specify the XML tag that triggers a dynamic statistic. For dynamic statistics, a substatistic is added when a new value of the dynamic field is recorded.

For example, if you set DynamicField to **Location** and Statistics Server records the new value London in this field, it creates a substatistic. This substatistic records the information configured in the dynamic statistic for each event that contains the value London in the Location field.

Type:	String
Default:	None
Required:	No
Configuration Section:	[*MyStatistic*]
Example:	DynamicField=location

Field

Specify the XML tag that triggers the statistic.

Type:	String
Default:	None
Required:	Yes

Configuration Section:	[*MyStatistic*]
Example:	Field=duration

Offset

Add an offset to the time range that is used to collect data. Specify the date and time when one statistics period ends and the next begins. You can use this parameter to align Statistics Server time periods with the periods for which you want to record statistics.

By default, Statistics Server starts a statistics period at a time when the epoch time is divisible by the period length. If you set `Offset` to a date and time, the statistics period starts at that time, rather than the default start time.

> **Note:** Statistics Server starts to collect data immediately when a statistic is created, even if it is in the middle of a period.

You must specify `Offset` in the format `YYYY/MM/DD HH:MM:SS`. This date and time can be in the past.

Type:	String
Default:	0
Required:	No
Configuration Section:	[*MyStatistic*]
Example:	Offset=2010/04/01 12:30
See Also:	"Period" on the next page

Operation

Specify the type of statistic. Enter one of the following values:

- `Count`. The number of results since the previous statistical report.
- `CumulativeCount`. The total number of results.
- `TopN`. The top *N* results in a particular time period. If you enter `TopN`, you must also enter the number you want to measure, separated by a comma. You can enter `all` to list an unlimited number of ranked terms.
- `CumulativeTopN`. The top *N* results for all time. If you enter `CumulativeTopN`, you must also enter the number you want to measure, separated by a comma. You can enter `all` to list an unlimited number of ranked terms.
- `Average`. The average number of results in a particular time period.

Type:	String

Default:	None
Required:	Yes
Configuration Section:	[*MyStatistic*]
Example:	`Operation=TopN,100`

Period

Specify the frequency in seconds of statistical reports.

Type:	Long
Default:	3600
Required:	Yes
Configuration Section:	[*MyStatistic*]
Example:	`Period=3600`

Action and Action Parameter Reference

You can send actions to Statistics Server through a Web browser. The actions use the following format:

`http://host:port/action=ActionName&[Parameters]`

where,

`host`	The IP address or name of the Statistics Server.
`port`	The ACI port specified in the `[Server]` section of the Statistics Server configuration file.
`ActionName`	One of the actions listed below.
`Parameters`	One or more parameters that might be required by an action.

Note: Separate parameters with an ampersand (&).

AddStat

The `AddStat` action creates a new statistic while Statistics Server is running, and adds the statistic to the Statistics Server configuration file.

Parameters

The `AddStat` action supports the following parameters.

Name

The name of the new statistic. A configuration section with the specified name is added to the Statistics server configuration file.

Type:	String
Default:	None
Required:	Yes
Example:	`Name=dynamicTopN`

ConfigurationOption

Configuration parameters to add for the new statistic. Add each configuration option that you want to configure as an action parameter for the `AddStat` action. Statistics Server adds these parameters to the configuration section for the new statistic.

You can add the following configuration parameters:

- "AEqualStat"
- "ARangeStat"
- "BitANDStat"
- "NEqualStat"
- "NRangeStat"
- "DynamicField"
- "Field"
- "Offset"
- "Operation"
- "Period"

Example

```
action=AddStat&Name=NewStat&Period=5&Field=Term&Operation=TopN,all
```

This creates the following section in the configuration file.

```
[NewStat]
Period=5
Field=Term
Operation=TopN,all
```

Event

The `Event` action sends XML events as URL-encoded text strings to the ACI port instead of as XML files to the Event port. This option can be useful if the front-end application uses the IDOL ACI API or something similar.

> **Note:** To use `Event`, you must enable the `ActionEvent` configuration parameter. See "ActionEvent" on page 100.

Parameters

The `Event` action supports the following parameters.

Data

Specify the URL-encoded XML string.

Type:	String
Default:	None
Required:	Yes
Example:	`data=encoded_string` where *encoded_string* is the XML event in the form of a URL-encoded text string.

InflateBytes

If the `Data` string is encoded in Base64 and is compressed with zlib, `InflateBytes` indicates the original size of the string. This value must be specified by the application that compressed the data.

Type:	Integer
Default:	None
Required:	No
Example:	`InflateBytes=1024`

Example

```
action=Event&Data=<?xml version='1.0' encoding='ISO-8859-
1'?><events><queryinfo><ver>0.1</ver><url><![CDATA
[action=Query&Spellcheck=True&Highlight=SummaryTerms]]
></url><action>query</action><terms><term>This</term></terms><terms><term>is</term>
</terms><terms><term>an</term></terms><terms><term>event</term></terms><terms><term
>test.</term></terms><numhits>12</numhits><ip>127.0.0.1</ip><timestamp>1200398400</
timestamp><idolname>myIDOL</idolname></queryinfo></events>
```

GetDynamicValues

The `GetDynamicValues` action returns the dynamic values associated with each statistic. It can either return values for a particular statistic name, or for all statistics.

Parameters

The `GetDynamicValues` action supports the following parameters.

Name

Restrict results to the specified statistics. The statistics that you enter must exist in the Statistics Server configuration file. Separate multiple entries with commas. By default, Statistics Server returns dynamic values for all statistics.

Type:	String
Default:	None
Required:	No
Example:	`Name=DynamicTopN,StandardCount,DynamicCount`

Example

```
action=GetDynamicValues&Name=DynamicTopN,StandardCount
```

GetStatus

The `GetStatus` action returns information about the Statistics Server configuration settings, as well as the names and types of the statistics set in the configuration file.

Parameters

`GetStatus` does not support any parameters.

Example

```
action=GetStatus
```

StatDelete

The `StatDelete` action allows you to delete statistics data, either from a particular statistic name, or from all statistics.

Parameters

The `StatDelete` action supports the following parameters.

Name

Restrict results to the specified statistics. The statistics that you enter must exist in the Statistics Server configuration file. Separate multiple entries with commas.

Type:	String
Default:	None
Required:	No
Example:	`Name=count_queries_hour,count_queries_day`

AllStats

Delete all statistics data.

Type:	Boolean
Default:	False
Required:	No
Example:	`AllStats=True`

Example

```
action=StatDelete&Name=Stat1
```

StatResult

The `StatResult` action returns the results of the configured statistics in XML format.

Parameters

The `StatResult` action supports the following parameters.

Name

Restrict results to the specified statistics. The statistics that you enter must exist in the Statistics Server configuration file. Separate multiple entries with commas.

Type:	String
Default:	None
Required:	No
Example:	`Name=count_queries_hour,count_queries_day`

DateString

Set the timestamp format. Set this parameter to `True` to display the timestamp in human-readable format (DD/MM/YYYY HH:MM:SS), or `False` to display it in epoch time.

Type:	Boolean
Default:	False
Required:	No
Example:	`DateString=True`

DynamicValue

The value of the dynamic field. If the field specified in the `Name` parameter is a dynamic field, specify the value of this field for which you want to return results.

Type:	String
Default:	None
Required:	No
Example:	`DynamicValue=London`

From

Set a minimum timestamp value. Statistical results must have a timestamp of equal or greater value to be returned.

> **Note:** The format in which you must enter a value depends on the `DateString` parameter setting. If it is `False`, you must enter an epoch time value. If it is `True`, you must enter a human-readable format value.

Type:	String
Default:	0 (disabled)
Required:	No
Example:	`From=1220433495`
See also:	"DateString" above "UpTo" below

UpTo

Set a maximum timestamp value. Statistical results must have a timestamp of equal or lesser value to be returned.

> **Note:** The format in which you must enter a value depends on the `DateString` parameter setting. If it is `False`, you must enter the an epoch time value. If it is `True`, you must enter a human-readable format value.

Type:	String
Default:	0 (disabled)
Required:	No
Example:	UpTo=1220433505
See also:	"DateString" on the previous page "From" on the previous page

MaxResults

Return the last N statistical results.

Type:	Integer
Default:	0 (disabled)
Required:	No
Example:	MaxResults=10

MaxValues

Return only the top N values for a `TopN` or `CumulativeTopN` statistic.

Alternatively, set `MaxValues` to `0` to return only the number of unique values for `TopN` or `CumulativeTopN` statistics, without returning the values.

Type:	Integer
Default:	All values are shown
Required:	No
Example:	MaxValues=10

Example

```
action=StatResult&Name=stat1,stat2&MaxResults=25
```

In this example, Statistics Server returns the most recent 25 results of the `stat1` and `stat2` statistics.

Glossary

A

ACI (Autonomy Content Infrastructure)

A technology layer that automates operations on unstructured information for cross-enterprise applications. ACI enables an automated and compatible business-to-business, peer-to-peer infrastructure. The ACI allows enterprise applications to understand and process content that exists in unstructured formats, such as email, Web pages, Microsoft Office documents, and IBM Notes.

ACI Server

A server component that runs on the Autonomy Content Infrastructure (ACI).

ACL (access control list)

An ACL is metadata associated with a document that defines which users and groups are permitted to access the document.

action

A request sent to an ACI server.

active directory

A domain controller for the Microsoft Windows operating system, which uses LDAP to authenticate users and computers on a network.

agent

A process that searches for information about a specific topic. An administrator can create agents for users or allow users to create their own agents.

agent index

An index in IDOL Server that stores agents and profiles.

AgentBoolean field

An IDOL Server field that stores Boolean agents (Boolean or Proximity expressions). You can then query IDOL Server with text and an AgentBoolean field to return categories whose Boolean agent matches this text. QMS rules use Boolean agents for matching queries.

B

blacklist rules

A rule that allows you to specify a list of disallowed words for queries, and removes any terms that appear on this list.

boost rules

A rule that modifies a query to include extra FieldText criteria, for example to boost the relevance of certain results.

C

cardinal placement

A QMS rule that allows you to add a specified result to a specified position in a query results list.

Category component

The IDOL Server component that manages categorization and clustering.

Community component

The IDOL Server component that manages users and communities.

conceptual search

A type of query that allows you to search for documents that match the concept that your query text defines, rather than matching the

particular keywords in your text. See also: query.

connector

An IDOL component (for example File System Connector) that retrieves information from a local or remote repository (for example, a file system, database, or Web site).

Connector Framework Server (CFS)

Connector Framework Server processes the information that is retrieved by connectors. Connector Framework Server uses KeyView to extract document content and metadata from over 1,000 different file types. When the information has been processed, it is sent to an IDOL Server or Distributed Index Handler (DIH).

Content component

The IDOL Server component that manages the data index and performs most of the search and retrieval operations from the index.

D

DAH (Distributed Action Handler)

DAH distributes actions to multiple copies of IDOL Server or a component. It allows you to use failover, load balancing, or distributed content.

data index

An IDOL Server index that stores content data. You can customize how data is stored in the data index by configuring appropriate settings in the IDOL Server configuration file.

database

An IDOL Server data pool that stores indexed information. The administrator can set up one or more databases, and specifies how data is fed to the databases. By default IDOL Server

contains the databases Profile, Agent, Activated, Deactivated, News, and Archive.

DIH (Distributed Index Handler)

DIH allows you to efficiently split and index extremely large quantities of data into multiple copies of IDOL Server or the Content component. DIH allows you to create a scalable solution that delivers high performance and high availability. It provides a flexible way to batch, route, and categorize the indexing of internal and external content into IDOL Server.

Distributed Action Handler

See: DAH

dynamic promotion

A query that returns a document or set of documents that you want to promote. See Also: static promotions.

E

Extensible Markup Language

See: XML

F

field

Fields define different parts of content in IDOL documents, such as the title, content, and metadata information.

field operator

A syntax string that defines a matching criteria in FieldText.

FieldText

A type of query that searches for particular content in a particular document field. See also: query, field.

H

hypernym rule

A rule that modifies query text to replace specific terms with a more general term. For example, flower is a hypernym of rose and lily.

hyponym rule

A rule that modifies query text to include terms that are specific instances of the original terms. For example, poodle and labrador are hyponyms of dog.

I

IDOL

The Intelligent Data Operating Layer (IDOL) Server, which integrates unstructured, semi-structured and structured information from multiple repositories through an understanding of the content. It delivers a real-time environment in which operations across applications and content are automated.

IDOL Proxy component

An IDOL Server component that accepts incoming actions and distributes them to the appropriate subcomponent. IDOL Proxy also performs some maintenance operations to make sure that the subcomponents are running, and to start and stop them when necessary.

IDX

A structured file format that you can index into IDOL Server. You can use a connector to import files into this format, or you can manually create IDX files.

index

The IDOL Server data index contains document content and field information for analysis and retrieval.

index action

An IDOL Server command to index data, or to maintain or manipulate the data index.

index fields

Fields that IDOL Server processes linguistically when it stores them. Store fields that contain text which you want to query frequently as Index fields. IDOL Server applies stemming and stop word lists to text in Index fields before it stores them, which allows IDOL Server to process queries for these fields more quickly. Typically DRETITLE and DRECONTENT are fields that are set up as Index fields.

indexing

The process of storing data in IDOL Server. IDOL Server stores data in different field types (for example, index, numeric, and ordinary fields). It is important to store data in appropriate field types to ensure optimized performance.

Intellectual Asset Protection System (IAS)

An integrated security solution to protect your data. At the front end, authentication checks that users are allowed to access the system that contains the result data. At the back end, entitlement checking and authentication combine to ensure that query results contain only documents that the user is allowed to see, from repositories that the user has permission to access.

K

KeyView

The IDOL component that extracts data, including text, metadata, and subfiles from over 1,000 different file types.

L

LDAP

Lightweight Directory Access Protocol. Applications can use LDAP to retrieve information from a server. LDAP is used for directory services (such as corporate email and telephone directories) and user authentication. See also: active directory, primary domain controller.

License Server

License Server enables you to license and run multiple IDOL solutions. You must have a License Server on a machine with a known, static IP address.

link term

Also referred to as "links". Terms in query text that are also contained in the result documents that IDOL Server returns for this query.

O

OmniGroupServer (OGS)

A server that manages access permissions for your users. It communicates with your repositories and IDOL Server to apply access permissions to documents.

P

parametric cardinal placement

A QMS rule that allows you to place a specified parametric field in a specified position in a list.

primary domain controller

A server computer in a Microsoft Windows domain that controls various computer resources. See also: active directory, LDAP.

Promotion Agentstore

The agent index that stores QMS rules.

promotions

Targeted content that you want to display to users but that is not included in the search results, such as advertisements.

Q

QMS

An ACI server that manipulates queries and results according to user-defined rules.

QMS rules

A document stored in the Promotion Agentstore that defines how QMS manages a query. Rules can return promotion documents, modify the original query, or modify the results of a query.

query

A string that you submit to IDOL Server, which analyzes the concept of the query and returns documents that are conceptually similar to it. You can submit queries to IDOL Server to perform several kinds of search, such as natural language, Boolean, bracketed Boolean, and keyword.

query cooker

A JavaScript application that manipulates queries and query results.

Query Manipulation Server

See: QMS

R

reference

A string that is used to identify a document. This string might be a title or a URL, and allows IDOL to identify documents for retrieval, indexing, and deduplication.

ReferenceType field

Fields used to identify documents. At index time IDOL Server can use ReferenceType fields to eliminate duplicate copies of documents. At query time IDOL Server can use ReferenceType fields to filter results.

relevance

The similarity that a particular query result has to the initial query. IDOL Server assigns results a percentage relevance score according to how closely it matches the query criteria.

S

scope rules

A rule that allows you to return results from a promotions query that are relevant to a particular role, department, or other user-defined characteristic.

static promotion

A specific document or set of documents that you want to return as a promotion. See Also: dynamic promotions.

synonym rule

A rule that modifies query text to include terms that are synonymous with the original terms.

T

tagging

The process of adding extra information to documents. The tag might be a category, or entities returned from Eduction. Tagging usually adds a field to a document, which you can use to search by the name of a tag.

term

The basic entity that IDOL Server stores (for example, a word in a document after IDOL Server applies stemming).

V

View

An IDOL component that converts files in a repository to HTML formats for viewing in a Web browser.

W

whitelist

A rule that allows you to specify a list of allowed words for queries, and to remove any terms that do not appear on this list.

Wildcard

A character that stands in for any character or group of characters in a query.

X

XML

Extensible Markup Language. XML is a language that defines the different attributes of document content in a format that can be read by humans and machines. In IDOL Server, you can index documents in XML format. IDOL Server also returns action responses in XML format.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on QMS Administration Guide (Query Manipulation Server 11.0)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to AutonomyTPFeedback@hpe.com.

We appreciate your feedback!