



Hewlett Packard
Enterprise

HPE IDOL Server

Software Version: 11.0

IDOL Server Administration Guide

Document Release Date: March 2016
Software Release Date: March 2016

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

© Copyright 2016 Hewlett Packard Enterprise Development LP

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 I: Introduction	21
Chapter 1: Introduction to IDOL Server	23
IDOL Server Operations	23
Agents	23
Alerts	24
Automatic Query Guidance	24
Categorization	24
Category Matching	24
Channels	24
Cluster Information	25
Collaboration	25
Dynamic Clusters	25
Dynamic Thesaurus	25
Education	25
Expertise	25
Hyperlinks	26
Email Users	26
Profiles	26
Search and Retrieval	26
Conceptual Matches	26
Advanced Keyword Search	27
Boolean and Bracketed Boolean Search	27
Exact Phrase Search	27
Field Restrictions	27
FieldText Search	27
Fuzzy Search	27
Parametric Search	27
Proper Names Search	28
Proximity Search	28
Soundex Keyword Search	28
Synonym Search	28
Spell Check	28
Summarization	28
Taxonomy Generation	28
Automatic Taxonomy Based on Cluster Result	29
Automatic Taxonomy to Category Generation	29
View Documents	29
Getting Started	29
Send Actions to IDOL Server	29
Display Online Help	30

Part II: Store Content in IDOL Server 33

Chapter 2: Configure Content Storage 35

- Edit the Configuration File 35
- Stored Content 35
 - Disable Content Storage 35
 - Store IDOL Server Data Files on Multiple Disks 36
 - Compress the Data Index 36
 - Allocate Files to IDOL Server Databases 37
- Set up the Field Index Process 38
 - Index XML Attributes 39
- Configure IDOL Server for Language and to Encode 40
- Optimize Index Process 41
 - Index Process 41
 - Delayed Synchronization 41

Chapter 3: Index Data 43

- Index Overview 43
- Process Data before you Index 44
- DREADD: Index IDX and XML Files Directly 44
 - DREADD Parameters 45
 - DREADD Examples 47
 - Specify Field Names 48
- DREADDATA: Index Data over a Socket 50
 - DREADDATA Parameters 50
 - Send Data with a POST Method 51
 - Use the cURL Command-Line Tool 51
 - Use a Script 52
 - DREADDATA Examples 53
- Index Stop Words 54
- Index Nonalphanumeric Characters 54
 - Term Separators 55
 - Index Nonalphanumeric Characters for Retrieval 56
 - Hyphenated Terms 57
 - Character Tokenization 58
- Prevent Duplicate Documents 58
 - Deduplication Options—KillDuplicates 59
 - Enable Deduplication for all Index Jobs 61
 - Limit ReferenceType Fields used for Deduplication 61
 - Use KillDuplicatesChecksumField to Prevent Unnecessary Indexing 62
 - Use KillDuplicatesPreserveFields to Preserve a Field 62
 - Enable Deduplication for Individual Index Jobs 63
 - Use KeepExisting to Minimize the Index Load 63
 - Enable Deduplication for Connector Index Jobs 63
 - Deduplication Constraints 64
 - Use the Combine Operation 64
 - Use Deduplication with DIH Reference-Based Indexing 64

Use Deduplication with DIH Field-Based Indexing	64
Add Metadata to Documents	65
Check Index Status	65
IndexerGetStatus Status Codes	67
Tag Documents into Clusters	69
Chapter 4: Fields	73
About Fields	73
Configure a Field Process	76
Update Field Configuration	79
Update Fields in the Configuration File	81
Update Field Configuration with an Index Action	82
Update Field Configuration with IDOL Admin	83
Index Fields	83
Configure the Number Index Process	85
NumericDateType Fields	86
NumericType Fields	87
FieldCheckType Fields	88
ReferenceType Fields	90
Set up ReferenceType Fields	90
Use KillDuplicates and Combine on ReferenceType Fields	91
Highlight Fields	93
BitFieldType Fields	94
Edit Set Information after Indexing	95
Find Documents within a Set	95
Metadata Fields	96
Change Field Values	97
Chapter 5: Language Support	99
IDOL Language Support Concepts	99
Run IDOL Server in Multiple Languages	101
Determine the Languages that are Enabled	102
Define Language Types	103
Associate Language Types with Documents	104
Documents that Contain a Language Type Field	105
Documents that Contain Field Data that can Identify Language	106
Add LanguageType Fields to Documents	107
Define a Default Language Type	108
Define a General Language	108
Enable Automatic Language Detection	109
Specify the Language Type of a Query	110
Convert Results to a Specific Encoding	110
Text Queries	110
Text-Free Queries	111
Return Documents in Multiple Languages	111
International Stop List	112
Return Documents in a Specific Language	113
Create a Custom Stem File for a Language	113

Decompose Compound Words	114
Enable Transliteration	115
Enable Generic Transliteration	115
Enable Transliteration for Individual Languages	116
Chapter 6: Set Up Document Tracking	119
Set up Document Tracking with an SQL Back End	119
Set up Document Tracking with PostgreSQL	119
Set up PostgreSQL to Store Tracking Information	119
Install the SQL Database	120
Set up the Database and Table	120
Database Access Permissions	123
Set up the IDOL Host Machines	124
Install the SQL Driver and Manager for PostgreSQL	124
Check the Installed Drivers	124
Configure IDOL Components	125
Set up Document Tracking with Microsoft SQL Server	126
Set up Microsoft SQL Server to Store Tracking Information	126
Configuration Example for Microsoft SQL Server	126
Troubleshoot Connection and Authentication Problems	127
Initialization Commands	127
Set up the IDOL Host Machines	129
Install the SQL Driver and Manager for Microsoft SQL Server	129
Check the Installed Drivers	130
Configure IDOL Components	130
Verify the Setup	131
Check IDOL Configuration	132
Index Content	132
Query Your Document	133
Clean Results	133
Set up Document Tracking with a Log Back End	133
Configure Event Storage	134
Document Tracking Event Definitions	135
Part III: IDOL Server Operations	137
Chapter 7: Agents	139
About Agents	139
Manipulate Agents	139
Create an Agent	139
Edit an Agent	140
Retrain an Agent	140
Copy an Agent	140
View Agent Details	140
Delete an Agent	141
Query with Agents	141
Modify Document References for an Agent	141

Collaboration and Expertise with Agents	142
Collaboration	142
Expertise	142
Chapter 8: Categorization	143
Introduction to Categorization	143
Create a Hierarchical Category Structure	144
Create Categories from Scratch	144
Create Categories from Clusters	145
Create Categories from Legacy Topic Sets	145
Create Categories by Copying Categories	145
Create Categories when you Generate a Taxonomy	146
Create Categories from XML	146
Create Categories from Partitions	146
Create Categories for Sentiment Analysis	147
Train Categories	147
Retrain Categories	147
Move Categories	148
View and Administer Categories	148
View Category Details	148
View Category Hierarchy Details	149
View Category Terms and Weights	149
View Category Training	149
Change Category Fields	149
Reset Category Fields	149
Change Category Term Weights	150
Remove Category Term Weights	150
Replace Categories	150
Activate or Deactivate Categories	151
Build Categories	151
Delete Categories	151
Delete Category Training	151
Export Categories to XML	152
Synchronize IDOL Server with Stored Categories	152
Categorize Data	152
Suggest Categories	152
Suggest Categories for Documents	152
Suggest Categories for Text	153
Suggest Categories for Categories	153
Suggest Categories with Confidence Values	153
Match Categories	153
Create Taxonomies	153
Generate Taxonomies Automatically	154
Generate a Taxonomy from Clusters	154
Generate a Taxonomy from Query Results	155
Schedule Taxonomy Generation	155
Create Named Taxonomies	155

- Categorization Example 155
- Chapter 9: Document Classification 157
 - Introduction to Document Classification 157
 - Use Document Classification 157
 - Choose Feature Fields 157
 - Create a Classifier 158
 - Create and Train Classes 159
 - Choose Training Documents for Classes 160
 - Train the Classifier 160
 - Classify Documents 161
 - View and Administer Classifiers 162
 - List and View Classifiers 162
 - Retrain a Class 162
 - Delete a Class 163
 - Delete a Classifier 163
- Chapter 10: Binary Categories 165
 - About Binary Categories 165
 - Create and Administer Binary Categories 165
 - Create a Binary Category 165
 - Train a Binary Category 166
 - Delete Training From a Binary Category 166
 - Change Binary Category Details 166
 - View Binary Category Details 166
 - List Binary Categories 167
 - Delete a Binary Category 167
 - Query with Binary Categories 167
 - Binary Category Example 167
- Chapter 11: AgentBoolean Agents and Categories 169
 - AgentBoolean Agents and Categories 169
 - Examples 169
 - Match Specific Concepts 170
 - Use Field Restrictions 170
 - Use Term Occurrence Restrictions 170
 - Categorize Documents before Indexing 170
 - Alert Users to Documents that Match Their Agents 171
 - Configure IDOL Server for Text Parse Queries 171
 - Create AgentBoolean Agents and Categories 172
 - Perform AgentBoolean Queries 172
 - Optimize AgentBoolean Matching 174
 - Configure AgentBoolean Cache Fields 174
 - Index a Dummy IDX 175
 - Customize Agent Content 175
 - Use a Minimal List of Rare Terms 176
 - Use AlwaysMatchType Fields 177
- Chapter 12: Cluster Information 179

Generate Snapshots	179
Generate Spectrograph Data	181
Generate WhatsNew and WhatsHot Information	182
WhatsNew Information	183
WhatsHot Information	183
Generate a Cluster Map after You Cluster	183
Configure Clusters	184
Change the Number and Size of Clusters	184
Build Seeds	184
Group Seeds into Clusters	184
Configuration Parameters	185
Set up Schedules	187
Partition Documents into Clusters	189
Chapter 13: Profiles	191
About Profiles	191
Profile a User	191
Create an Interest Profile for a User	191
Create an Expertise Profile for a User	192
Manipulate Profiles	192
Edit a Profile	193
Query with a Profile	193
View Profile Details	193
Delete a Profile	193
Collaboration and Expertise with Profiles	193
Collaboration	193
Expertise	194
Part IV: Results	195
Chapter 14: Search and Retrieve	197
Actions	197
Asynchronous Actions	198
Configure Asynchronous Actions	199
Send Asynchronous Actions	199
Retrieve Results for Asynchronous Actions	199
Conceptual Matches	200
Types of Matches	200
Example Queries	200
Agent or Category Query	201
Profile Query	202
Text Query	202
Suggest Query	202
SuggestOnText Query	202
Keyword Search	202
Keyword Occurrence Search	202
Exact Keyword Search	203

Case-Sensitive Exact Keyword Search	204
Paragraph and Sentence Keyword Search	204
Keyword Search Examples	204
Phrase Search	208
Phrase Occurrence Search	208
Default Phrase Search	208
Exact Phrase Search	209
Case-Sensitive Exact Phrase Search	209
Phrase Search Examples	210
Boolean and Proximity Search	212
Boolean Search Operators	212
Proximity Search Operators	214
WHEN and NOTWHEN Search Operators	217
Specify the Number of Levels from the XML Root	220
Precedence of Search Operators	221
Simple Field Restricted Search	221
FieldText Search	222
FieldText Query Guidelines	222
Field Specifiers for Common Restrictions	224
Fields whose Value Exactly Matches One or More Strings	224
Fields that Contain a Number	225
Fields that Contain a Date	228
Fields whose Value Matches Wildcard Strings	231
Field Specifiers for Advanced Restrictions	232
Fields whose Value Falls within a Specific Alphabetical Range	233
Fields with a Nonzero Value for Bitwise AND	234
Fields that Contain BitFieldType Information	237
Fields whose Values are Boolean Agents	238
Fields that are within a Specified Distance from a Specified Point	239
Fields that Do Not Exist or Contain No Value	240
Specific Fields, Irrespective of their Value	240
Fields whose Values are Similar to a Specified String	241
At Least One Field Instance Matches a Specified String or Number	241
All Field Instances Match a Specified String or Number	243
Fields that Contain a Specified ReferenceMemoryMappedType Field	244
Fields that Do Not Contain a Specified Value	245
Fields that Contain Coordinates within a Specified Area	248
Fields that Contain a Specified String	249
Fields whose Values Match Specific Terms or Phrases	251
Field Specifiers to Bias Result Scores	255
Field Specifier for Linked Queries	256
Fuzzy Search	256
Fuzzy Query Syntax	256
Adjust the Tolerance Level of a Fuzzy Search	256
Parametric Search	257
Configure IDOL Server for Parametric Fields	257

Perform a Parametric Search	258
GetTagValues	258
GetQueryTagValues	259
Proper Names Search	260
Enable Proper Names Searches	260
Example Proper Name Searches	262
Soundex Keyword Search	263
Enable Soundex Keyword Searches	263
Perform Soundex Keyword Searches	264
Synonym Search	264
Enable Synonym Searches	264
Set up a Synonym File	265
Configure IDOL Server to Use a Synonym File	266
Perform Synonym Searches	267
Update Synonym Files	267
Set up an Additional Synonym IDOL Server	267
Install the Synonym IDOL Server	267
Create and Index a Synonym File	267
Perform Synonym Searches	268
Analytics Functions	269
Create Metafields	269
Use Metafields	272
FieldText	272
Sort	272
PrintFields	273
GetQueryTagValues FieldName	273
AgentBoolean	273
Link Queries	274
Configure IDOL Server for Linked Queries	274
Send Linked Queries	275
Example	276
Combine Different Search Types	277
Synonym and Boolean Searches	277
Synonym Search and Field Restrictions	277
Soundex and Proper Names Searches	277
Soundex and Boolean Searches	278
Soundex and Proximity Searches	278
Soundex Search and Field Restrictions	278
Exact Phrase and Boolean Searches	278
Exact Phrase and Proximity Searches	278
Exact Phrase Search and Field Restrictions	279
Boolean and Proximity Searches	279
Boolean Search and Field Restrictions	279
Proximity Searches and Field Restrictions	279
Wildcard in Queries	280
Wildcards in Query Text	280

Wildcards in FieldText Queries	281
Matches for One or More Strings	282
Wildcard Searches in Japanese, Chinese, Korean, and Thai	283
Query for Nonalphanumeric Characters	283
Text	283
FieldText	284
Examples	284
Optimize Retrieval of Tagged Documents	285
Query Syntaxes	285
Chapter 15: Customize Results	289
Change the Results Display	289
Set the Number of Results to Display	289
Change Result Sorting (Display Order)	289
Sort Options for Query, Suggest, and SuggestOnText	290
Sort for GetTagValues and GetQueryTagValues	292
Batch (Page) Results	293
Change the Field Display	293
Returned Fields	294
Display Additional Metafields	294
Display Document Fields	294
Configure IDOL Server to Always Display Specific Fields	294
Display Specific Fields for Individual Queries	295
Use XSLT Templates to Change Output Format	297
Enable the XSLT Templates	298
Apply XSLT Templates to Actions	298
Generate Summaries	299
Types of Summaries	299
Return Summaries with Query Results	299
Summarize Text or Documents	300
Cluster Results	300
Generate Hyperlinks	302
About Hyperlinks	302
Implement Hyperlinks	302
Provide Spell Correction	303
How Spell Correction Works	303
Spell Correction File	303
Automatic Query Guidance	304
About Automatic Query Guidance	304
Enable Automatic Query Guidance	305
About the QuerySummary Parameter	306
Generate Query Summaries (Dynamic Thesaurus)	307
About Query Summaries	308
Configure IDOL Server to Generate Query Summaries	308
Perform an Action with the QuerySummary Parameter	309
Generate Dynamic Clusters	310
Configure IDOL Server to Enable Dynamic Clusters	311

Perform an Action with the QuerySummary Parameter	311
Display Cluster Information	312
Display the Number of Documents a Dynamic Cluster Contains	313
Create a Cluster Map	314
Chapter 16: Manipulate Result Relevance	317
Boost Relevance	317
Use a Field Process to Boost Relevance	317
Use the BIAS Field Specifier to Boost Relevance	319
BIASDATE	321
BIASDISTCARTESIAN	321
BIASDISTSPHERICAL	322
Use Multipliers to Boost Relevance	323
Use the AutnRankType Field to Boost Relevance	324
Chapter 17: Manipulate the Results Set	325
Combine Parameter	325
Simple	325
FieldCheck	326
ReferenceTypeFields	326
Multiple Options	327
Exceptions	327
FieldCheck Parameter	328
Predict Parameter	328
Store and Retrieve the Result State	328
Store the Result State	329
Query with the State Token	329
Use a State Token with Index Actions	330
Expire State Tokens	330
Chapter 18: View Documents	333
About the View Service	333
Configure the View Service	333
Enable View to Access Documents	334
Configure View to Block Particular URLs or Hosts	334
Configure View to Use a Proxy Server	335
Configure View to Highlight Terms	335
Configure View to Embed Images	336
Configure the View Cache	337
Configure the Internal View Cache	337
Use a Memcached Server View Cache	337
Use a Shared View Cache	338
Distribute View Servers	338
View Documents	339
View the Document Directly in the Web Browser	339
Use IDOL Admin to View Documents	340
View the Latest Version of a Document	340
View an Uploaded Document	341

Highlight Terms	341
Highlight Boolean Expressions	342
Highlight Expressions in Different Languages	342
Highlight Multiple Link Terms	342
Specify Document Processing	342
View Document Information	343
View Templates	343
Apply a Template to a Document	344
Apply a Default Template to All Documents	344
Modify the HTML Output for Documents	345
Modify the HTML Output for PDF Files	345
Hide Graphics	347
Show Revised Content and Revision Information	347
Format Revised Content	347
Show Hidden Content	349
Hidden Content in Microsoft Documents	349
Part V: Administration and Maintenance	351
Chapter 19: Set up Security	353
Set up Security on Documents	353
Set up an SSL Connection	356
Set up SSL between IDOL components	357
Set up SSL for Shared Communications	359
Set up SSL for Mailer	359
Set up SSL for the View Component	360
Set up SSL for Communications to Remote Servers	360
Log SSL Settings	360
Check SSL Status	361
Chapter 20: Add Users to IDOL Server	363
Create IDOL Users	363
Flat Structure	363
Hierarchical Structure	364
Create Users in IDOL Admin	364
Manage Roles in IDOL Admin	364
Integrate with a Third-Party User Structure	365
Implement User Account Security	365
Create User PIN Codes	365
Add a PIN Code for a User	366
Authenticate Users with PIN Codes	366
Set User Name and Password Restrictions	367
Enable Password and PIN Code Time Restrictions	367
Set Maximum Login Attempts	368
Lock and Unlock User Accounts	369
Chapter 21: Mail	371
Automatically Email Agent and Channel Results	371

Send Custom Emails	373
Send Emails in Batches	374
Mailer Templates	374
Edit Templates	375
Chapter 22: Administer IDOL Server	379
Enable Configuration Changes	379
Delete and Restore Documents	380
Delete Documents by Reference	380
Delete Documents and Ranges of Documents	381
Delete Documents in IDOL Admin	381
Restore Deleted Documents	382
Locate Duplicate Documents	382
Create and Delete Databases	383
Create a New Database	383
Send a DRECREATEDBASE Index Action	384
Add a Database to the IDOL Server Configuration File	384
Create a new database in IDOL Admin	385
Modify a Database Configuration	386
Delete a Database and All its Documents	386
Delete a Database and All its Documents by Sending an Action	386
Delete a Database and All its Documents by using IDOL Admin	387
Delete All Documents from a Database	387
Delete All Documents from a Database by Sending an Action	387
Delete All Documents from a Database by using IDOL Admin	387
Expire Documents	388
Set up a Field Process for Expiration Dates	388
Expire Immediately	389
Expire at Regular Intervals	389
Change Document Metadata	390
Change Document Field Values	391
Change Document Field Values by Running an Index Action	391
Change Document Field Values by Using IDOL Admin	395
Edit the Spelling Correction Cache	399
Compact the Data Index	399
Compact the Data Index Immediately	400
Compact the Data Index at Regular Intervals	401
Initialize the Data Index	401
Validate the Data Index	402
Validate the Data Index Immediately	402
Validate Subindexes On Startup	403
Validate the Data Index Automatically	403
Repair an Index After Validation Fails	403
Regenerate with a Server Restart	404
Regenerate with an Index Action	404
Regenerate with IDOL Admin	405
Chapter 23: Back up the IDOL Server	407

Back up Content	407
Back up the Entire IDOL Server Data Index	407
Back up the Data Index Immediately	407
Back up the Data Index at Regular Intervals	408
Back up the Data Index Automatically	409
Back up the Data Index Dynamically	410
Export IDX Documents from IDOL Server	411
Export XML Documents from IDOL Server	412
Use IDOL Admin to Export Indexed Documents	414
Archive Index Actions	415
Restore Content	416
Return a List of Backup Files	416
Restore from a Backup File	416
Restore to a Time	417
Restore Exported Content	418
Export Users, Roles, Agents, and Profiles	418
Import Users, Roles, Agents, and Profiles	418
Back up Categories, Taxonomies, and Cluster Jobs	419
Restore Categories, Taxonomies, and Cluster Jobs	420
Chapter 24: Troubleshoot IDOL Server	421
IDOL Server Log Files	421
Customize Logging	423
Create a Diagnostics Package	424
IDOL Statistics Server	425
Part VI: Appendixes	427
Appendix A: IDOL Server Configuration File	429
IDOL Server Configuration File	429
Modify Configuration Parameter Values	429
Configuration File Sections	430
[ACIEncryption] Section	431
[Agent] Section	431
[AgentDRE] Section	431
[AnalysisSchedules] Section	431
[CatDRE] Section	432
[Category] Section	432
[Cluster] Section	433
[Community] Section	433
[DAHEngines] Section	433
[DAHEngineN] Section	434
[Databases] Section	434
[DataDRE] Section	434
[DIHEngines] Section	435
[DIHEngineN] Section	435
[DistributionIDOLServers] Section	435

[DistributionSettings] Section	436
[DocumentTracking] Section	436
[DRE] Section	436
[FieldProcessing] Section	437
[FlushLock] Section	439
[IDOLServerN] Section	440
[IndexCache] Section	440
[IndexNotify] Section	440
[IndexQueue] Section	440
[IndexServer] Section	441
[LanguageTypes] Section	441
[License] Section	442
[Logging] Section	442
[MemoryCache] Section	444
[MyLockServer] Sections	444
[MyProperty] Sections	445
[Paths] Section	446
[Profile] Section	447
[ProfileNamedAreas] Section	447
[Role] Section	447
[Schedule] Section	447
[SectionBreaking] Section	448
[Security] Section	448
[Server] Section	449
[Service] Section	449
[SSLOptionN] Section	450
[Summary] Section	450
[Synonym] Section	450
[Taxonomy] Section	451
[TermCache] Section	451
[User] Section	451
[UserCustom] Section	452
[UserSecurity] Section	452
[UserSecurityFields] Section	453
[Viewing] Section	455
Appendix B: Password Encryption	457
Encrypt Passwords	457
Create a Key File	457
Encrypt a Password	457
Decrypt a Password	458
Appendix C: Languages and Language Files	461
Supported Languages and Common Encodings	461
Supported Encodings	505
TermSize Parameter	507
Per-Language Sentence-Breaking Files	508
Stop Word Lists for Supported Languages	509

- Appendix D: Manually Create IDX Files 511
 - IDX Format 511
 - Section a Document 513
- Appendix E: Category XML Format 515
 - Introduction 515
 - XML Format 515
 - Example Category XML Files 523
- Appendix F: GetStatus Action Response 525
 - GetStatus Action 525
 - IDOL Server GetStatus Response 526
 - Example IDOL Server GetStatus Response 533
- Appendix G: Error Codes 539
 - Community Error Codes 539

- Glossary 541

- Send Documentation Feedback 551

Part I: Introduction

This section provides a brief introduction to IDOL Server.

For more details on installing and running IDOL Server, refer to the *IDOL Getting Started Guide*.

- ["Introduction to IDOL Server"](#)

Chapter 1: Introduction to IDOL Server

HPE Intelligent Data Operating Layer (IDOL) Server integrates unstructured, semi-structured, and structured information from multiple repositories through an understanding of the content. It delivers a real-time environment to automate operations across applications and content, removing all the manual processes involved in getting information to the right people at the right time.

- [IDOL Server Operations](#)23
- [Getting Started](#)29

IDOL Server Operations

HPE IDOL Server can perform the following intelligent operations across structured, semistructured, and unstructured data.

Agents	Expertise
Alerts	Hyperlinks
Automatic Query Guidance	Mailing
Categorization	Profiles
Channels	Retrieval
Cluster Data	Spelling Correction
Collaboration	Summarization
Dynamic Clusters	Taxonomy Generation
Dynamic Thesaurus	Viewing
Education	

Note: Your license determines which of these operations your IDOL Server installation can perform.

Agents

Users can create agents in IDOL Server to find and monitor information that is relevant to their interests. The agents collect this information from a configurable list of Internet and intranet sites, news feeds, chat streams, and internal repositories.

You can create agents in a user-friendly way using:

- Natural language descriptions
- Example content (point and click)

- Legacy keyword or Boolean expressions

IDOL Server provides the conceptual information that it needs to create agents. The server accepts a piece of content (training text, a document, or a set of documents) or reference (identifier), and returns an encoded representation of the concepts. This representation includes the specific underlying patterns of terms and associated probabilistic ratings for each concept.

Users can retrain their agents by submitting a new piece of content (training text, a document, or a set of documents). The server uses the new concepts to adapt the agent.

Alerts

IDOL Server analyzes data when it receives new documents, and compares the concepts in documents with user agents. If new data matches a user agent, it immediately notifies the user by email or a third-party system (for example by SMS or a pager).

Automatic Query Guidance

IDOL Server finds the most salient terms and phrases in query results, and automatically clusters these terms and phrases. It uses the clustered phrases to provide a hierarchical set of queries that guide users to the result area that they are looking for.

Categorization

IDOL Server can automatically categorize data. HPE categorization allows you to derive categories from the concepts found in unstructured text. This process ensures that IDOL Server accurately classifies all data in the correct context. HPE categorization is a scalable solution capable of handling high volumes of information accurately and consistently.

HPE categorization does not rely on rigid rule-based category definitions such as legacy keyword and Boolean operators. Instead, HPE infrastructure uses a pattern matching process based on concepts. It can then automatically tag data sets, route content, or alert users to relevant information pertinent to the user profile.

HPE hooks into various repositories and data formats respecting all security and access entitlements, delivering complete reliability.

Category Matching

IDOL Server accepts a category or piece of content and returns categories ranked by conceptual similarity. This ranking determines the most appropriate categories for the piece of content, so that IDOL Server can subsequently tag, route, or file the content accordingly.

Channels

IDOL Server can automatically provide users with a set of hierarchical channels with highly relevant information pertinent to the respective channel. Channels are similar to agents, aggregating information that is relevant to the channel concept. Usually, administrators set up channels that are available to all users.

Real-time information dynamically updates in the channels. This process eliminates the requirement for manual intervention or pretagging, and minimizes the effort required to maintain it. Moreover, the administrator can add and remove channels on the fly, without recategorizing all the data.

Cluster Information

IDOL Server automatically clusters information. Clustering takes a large repository of unstructured data, agents, or profiles and automatically partitions the data to cluster similar information together. Each cluster represents a concept area in the knowledge base, and contains a set of items with common properties.

Collaboration

IDOL Server automatically matches users with common explicit interest agents or similar implicit profiles. This information creates virtual expert knowledge groups.

Dynamic Clusters

When it processes queries, IDOL Server automatically clusters the query results, and then clusters the first set of clusters to produce subclusters. This process allows you to generate a hierarchy of clusters that allows users to navigate quickly to their area of interest.

Dynamic Thesaurus

When it processes queries, IDOL Server can automatically suggest alternative queries, allowing users to quickly produce a variety of relevant result sets.

Eduction

Eduction is a tool that you can use to extract an entity (a word, phrase, or block of information) from text, based on a pattern you define. The pattern can be a dictionary of names such as people or places. The pattern can also describe what the sequence of text looks like without listing it explicitly, for example, a telephone number. The entities are contained inside grammar files.

When you use Eduction with IDOL Server, Eduction extracts the entities while the document is indexed and adds them into fields for easy retrieval.

The Eduction capability of IDOL Server is described in the *Eduction User Guide*.

Expertise

IDOL Server accepts a natural language or Boolean search string and returns users who own matching agents or profiles. This process allows instant identification of experts in a subject, eliminating time-consuming searches for specialists, and unnecessary researching of subjects for which expert knowledge is already available.

Hyperlinks

You can automatically generate hyperlinks in real time. These link to contextually similar content, for example to recommend related articles, documents, affinity products or services, or media content that relates to textual content.

IDOL Server automatically inserts these links when it retrieves the document. This process means that new documents can reference older documents, and that archived documents can link to the latest news or material on the subject.

Email Users

IDOL Server matches the agents and profiles against its document content at regular intervals, and automatically notifies users of documents that match their agents or profiles by emailing them.

Profiles

IDOL Server automatically creates interest and expertise profiles for users, in real time.

You can create interest profiles by tracking the content that a user views and extracting a conceptual understanding of it. IDOL Server then uses this understanding to keep user interest profiles up to date. You can use interest profiles to:

- Target information to users.
- Recommend content to users.
- Alert users to the existence of content.
- Put users in touch with other users who have similar interests.

You can create expertise profiles by tracking the content that a user creates and extracting a conceptual understanding of it. IDOL Server uses this understanding to keep user expertise profiles up to date. You can use expertise profiles to trace users who are experts in particular subject areas.

Search and Retrieval

IDOL Server offers a range of retrieval methods, from simple legacy keyword search to sophisticated conceptual querying.

Conceptual Matches

IDOL Server accepts a piece of content (a sentence, paragraph, or page of text, the body of an email, a record containing human-readable information, or the derived contextual information of an audio or speech snippet) or reference (identifier) as input and returns references to conceptually related documents ranked by relevance, or contextual distance. IDOL Server uses this process to generate automatic hyperlinks between pieces of content.

Advanced Keyword Search

IDOL Server matches any term or phrase that appears in quotation marks in its exact form before stemming.

Boolean and Bracketed Boolean Search

IDOL Server accepts simple or complex Boolean and bracketed Boolean expressions and returns a list of matching documents. You can form Boolean expressions using a set of Boolean and proximity operators.

AND	XOR/EOR	WNEAR
NOT	NEAR	BEFORE
OR	DNEAR	AFTER

Exact Phrase Search

IDOL Server provides the ability to search for exact phrases by putting quotation marks around a string of words. For example: "world market".

Field Restrictions

Simple field restrictions in query text allow you to restrict results to documents that contain specific values in specific fields.

FieldText Search

FieldText searches provide a wide range of field specifiers that you can use to query fields, restrict query results, or bias query result scores.

Fuzzy Search

If a search string is not quite accurate (for example, if it contains spelling mistakes), a fuzzy query returns results that contain words that are similar to the entered string.

Parametric Search

IDOL Server supports real-time navigation across multiple taxonomies with no additional manual configuration necessary, including full access to intersections of diverse taxonomy definitions.

From among the complete set of field names in the corpus, you can define a subset of fields in the server configuration as parametric fields.

After indexing, IDOL Server creates and stores a structure that contains information about all tag/value pairs that occur in defined parametric fields. A tag/value pair is a particular textual or numerical value paired with a specific field name.

You can then query IDOL Server with the name of a parametric field or fields. IDOL Server returns a list of all textual values that appear in the specified field or fields in all the documents stored in the server. This underlying operation can power a user interface that enables users to refine the scope of a query from the complete corpus to a subset of highly relevant documents.

Advanced Parametric Refinement provides an improved user experience coupled with increased productivity using an advanced real-time information discovery process.

Proper Names Search

IDOL Server recognizes names and treats them as a unit.

Proximity Search

IDOL Server returns documents in which specific terms occur within a specified proximity with a higher weighting.

Soundex Keyword Search

If the spelling of a keyword is not quite accurate but is phonetically correct, a Soundex keyword search returns results that contain the keyword and phonetically similar keywords. This process uses a configurable Soundex algorithm.

Synonym Search

A synonym query returns results that are conceptually similar to the query terms or conceptually similar to the synonyms that are available for the query terms.

Spell Check

IDOL Server can automatically spell check the query text it receives and suggest correct spelling for terms that its dictionary does not contain.

Summarization

IDOL Server accepts a piece of content and returns a summary of the information. IDOL Server can generate different types of summary.

- **Conceptual Summaries.** Conceptual summaries contain the most salient concepts of the content.
- **Contextual Summaries.** Contextual summaries relate to the context of the original query. They provide the most applicable dynamic summary in the results of a particular query.
- **Quick Summaries.** Quick summaries include a few sentences of the result documents.

Taxonomy Generation

Automatic taxonomy generation can automatically understand and create deep hierarchical contextual taxonomies of information. You can use clustering, or any other conceptual operation, as a seed for the

process.

The resulting taxonomy can:

- Provide insight into specific areas of the information.
- Provide an overall information landscape.
- Act as training material for automatic categorization, which then places information into a formally dictated and controlled category hierarchy.

Automatic Taxonomy Based on Cluster Result

IDOL Server can use cluster results to build taxonomies automatically and in real time.

Automatic Taxonomy to Category Generation

After the automatic taxonomy generation process has taken place, IDOL Server contextually understands the type of data it is dealing with. It uses this understanding to generate a deep hierarchical contextual taxonomy, known as an information landscape. Much like the automatic cluster to category generation, this feature uses the taxonomy results to create categories.

View Documents

IDOL Server uses HPE KeyView filters to convert documents into HTML format for viewing in a Web browser. It can convert documents that it retrieves from a local directory, intranet, or Internet source. It can also retrieve the document in its original format.

Getting Started

The *IDOL Getting Started Guide* contains information about installing and running IDOL Server. It also provides an overview of different IDOL setups that you can use.

You perform IDOL Server operations by sending *ACI actions* to IDOL Server from your Web browser. A list of all available ACI actions, as well as configuration parameters, index actions, and service actions, is available in the *IDOL Server Reference*.

Send Actions to IDOL Server

You query IDOL Server by sending actions from your Web browser. The general syntax of these actions is:

`http://IDOLhost:port/action=action&requiredParams&optionalParams`

where:

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

<i>action</i>	is the name of the action you want IDOL Server to perform (for example, Query).
<i>requiredParams</i>	are the parameters that you must supply for the action you request. (Not all actions have required parameters.)
<i>optionalParams</i>	are parameters that you can supply for the action you request. (Not all actions have optional parameters.)

Note: Separate parameters with an ampersand (&).

Tip: As an alternative to sending ACI actions directly, you can run any IDOL action from the IDOL Admin interface. For more information, refer to the *IDOL Admin User Guide*.

Display Online Help

You can display *IDOL Expert* and the *IDOL Server Reference* by sending an action from your Web browser.

For IDOL Server to display help, the help data file (`help.dat`) must be available in the same directory as the service instance.

To display help for IDOL Server

1. Start IDOL Server.
2. Send the following action from your Web browser:

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

where:

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

For example:

`http://12.3.4.56:9000/action=Help`

3. On the help landing page, click one of the following options to open the relevant help set.

IDOL Expert	<i>IDOL Expert</i> provides conceptual overviews and expert knowledge of IDOL Server and its features and functionality.
IDOL Reference	The <i>IDOL Server Reference</i> contains details of all the actions and configuration parameters that you can set in IDOL Server.

If you are new to IDOL, you can use *IDOL Expert* to find out more about the different functions that IDOL can perform. You can use the *IDOL Server Reference* to look up specific configuration parameters and actions.

Tip: You can also view the *IDOL Server Reference* without starting IDOL Server. In the IDOL Server installation directory, open the `help` directory and click `index.html`.

The navigation panel for the *IDOL Server Reference* lists the following options to display reference information.

Tab	Description
Actions	Describes the actions that you can send to IDOL Server. Actions allow you to query IDOL Server, and to instruct it to perform a variety of operations.
Configuration Parameters	Describes the parameters that determine how the IDOL Server operates. You can set configuration parameters in the IDOL Server configuration file.
Index Actions	Describes the index actions that you can send to IDOL Server. Index actions allow you to index content into IDOL Server, and to administer the IDOL Server data index.
Service Actions	Describes service actions. Service actions allow you to return data about the IDOL Server service, and to control IDOL Server.

Part II: Store Content in IDOL Server

This section explains the concept of indexing and describes how to index document content and metadata into IDOL Server.

- ["Configure Content Storage"](#)
- ["Index Data"](#)
- ["Fields"](#)
- ["Language Support"](#)
- ["Set Up Document Tracking"](#)

Chapter 2: Configure Content Storage

IDOL Server stores the content of documents in data indexes. (The default data indexes are in the IDOL Server databases *News* and *Archive*.) The process of storing content in IDOL Server is called *indexing*. This section describes how to set up IDOL Server for indexing by editing the IDOL Server configuration file.

- [Edit the Configuration File](#)35
- [Stored Content](#)35
- [Set up the Field Index Process](#) 38
- [Configure IDOL Server for Language and to Encode](#)40
- [Optimize Index Process](#)41

Edit the Configuration File

You configure IDOL Server by manually editing the IDOL Server configuration file. The configuration file is stored at the location:

```
instalDir\IDOLServer.cfg
```

where *instalDir* is the directory in which the IDOL Server is installed.

Stored Content

Before you start to index files into IDOL Server, you must:

- decide how you want to store content.
- set up field indexing.
- configure IDOL Server to process required languages.
- optimize the indexing process according to your system.

You can also configure IDOL Server to process documents that it receives (for example from an IDOL connector) before it indexes them. You can configure IDOL Server to perform a single task on incoming documents or combine several tasks in a complex process.

Related Topics

- ["Process Data before you Index" on page 44](#)

Disable Content Storage

You can disable content storage if you do not require IDOL Server to return the content of fields or summaries with results. This option saves the memory that the storing of fields normally requires.

To disable content storage

- In the [Server] section of the IDOL Server configuration file, set `NodeTableStoreContent` to `False`.

If you disable content storage, it affects the performance of the following actions.

<code>GetContent</code>	Returns only the references and the title of results.
<code>GetTagValues</code>	Unavailable.
<code>List</code>	Returns only the references and the title of results.
<code>Query</code>	Returns only the references and the title of results. You cannot restrict by fields.
<code>Suggest</code>	Returns only the references and the title of results. You cannot restrict by fields.
<code>SuggestOnText</code>	Returns only the references and the title of results. You cannot restrict by fields.
<code>Summarize</code>	Unavailable for indexed documents. You can generate summaries only if you supply text.
<code>TermGetBest</code>	IDOL Server saves the best terms for a document on indexing. These are the only terms available.

Store IDOL Server Data Files on Multiple Disks

You can store data files across multiple partitions to gain space if your IDOL Server data becomes too big to store on one volume (when the stored terms, references, content, and so on increase in size).

For example:

```
[PATHS]
DyntermPath=C:\IDOL\dynterm
NodetablePath=D:\IDOL\nodetable
RefIndexPath=E:\IDOL\refindex
MainPath=F:\IDOL\main
TagPath=.G:\IDOL>tagindex
```

Compress the Data Index

You can configure IDOL Server to compress the `nodetable` directory to reduce the disk footprint, by using the `NodeTableCompression` configuration parameter.

Note: If you change the compression method after you have started the server, the compression applies only for operations that index new data or rewrite existing data (such as `DREREPPLACE`). To compress or change the compression of the whole index, you can run a `DRECOMPACT` operation with the `NodeTableCompactWindowKB` parameter set. For more information, refer to the *IDOL Server Reference*.

For more information about the `NodeTableCompression` option, and the compression method to use, refer to *IDOL Expert*.

To configure IDOL Server to compress the nodetable index

1. Open the IDOL Server configuration file in a text editor.
2. In the [Server] section, add the NodeTableCompression parameter and set it to the compression method that you want to use to compress the index.

You can set it to one of the following values:

- zlib
- snappy
- lz4
- lz4hc

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

Allocate Files to IDOL Server Databases

You can configure IDOL Server to use a field in each document to determine the database into which it indexes the document.

To configure IDOL Server to read the database from a document field

1. Open the IDOL Server configuration file in a text editor.
2. In the [FieldProcessing] section, list a process that identifies database fields.

For example:

```
[FieldProcessing]
0=MyFirstProcess
1=MySecondProcess
2=DatabaseFields
```

3. Create a section for the database field identifying process. For example:

```
[DatabaseFields]
```

4. Create a property for the process (you define the property later, by using one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

Optionally, use the PropertyMatch parameter to identify a specific value that fields must have to be processed.

Note: The property that you create must not have the same name as the process.

For example:

```
[MyFirstProcess]
Property=MyFirstProperty
PropertyFieldCSVs=*/MyField,*/MySecondField
PropertyMatch=*myString*
```

```
[MySecondProcess]
Property=MySecondProperty
```

```
PropertyFieldCSVs=*/MyOtherField,*/MyOtherSecondField
```

```
[DatabaseFields]  
Property=Database  
PropertyFieldCSVs=*/DREDBNAME,*/DB,*/Database
```

5. Create a section for your indexing property and set the DatabaseType parameter to True.

For example:

```
[MyFirstProperty]  
HiddenType=True
```

```
[MySecondProperty]  
Index=True
```

```
[Database]  
DatabaseType=True
```

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

Set up the Field Index Process

HPE IDOL connectors aggregate content and metadata, process it and then index it into IDOL Server in the form of fields. To improve IDOL Server performance, divide these fields into the following groups:

- **Prevent from storing.** Prevent IDOL Server from storing fields that you do not want to query by setting the CantHaveFieldsCSVs parameter in your IDOL Server configuration file. Alternatively, add the CantHaveFields parameter to your DREADD or DREADDATA index action.
- **Index fields.** Store fields that contain text that you want to query frequently as Index fields. IDOL Server processes Index fields linguistically when it stores them. IDOL Server applies stemming and stop word lists to the text, which allows IDOL Server to process queries for these fields more quickly. Typically, you set up DRETITLE and DRECONTENT as Index fields.

Do not use Index fields to store URLs or content that you are unlikely to use. Also, when you query the value only in its entirety, it is more efficient to query with a field specifier (for example, MATCH), than to store the data in an Index field.

HPE does not recommend indexing all fields in documents because it can potentially slow the indexing process, increase disk usage, and increase requirements.

- **Numeric fields.** Store fields that contain numerical values or dates as numeric fields and numeric date fields. When IDOL Server indexes these fields, it stores them in a fast look-up table in memory, which enables it to return the fields more quickly.
- **Field-check fields.** If a large number of the documents that you want to store in IDOL Server contain a field whose entire value is frequently used to restrict results, store this field as a FieldCheckType field. When IDOL Server indexes these fields, it stores them in a fast look-up table in memory, which enables it to return the fields more quickly.
- **Ordinary fields.** By default IDOL Server stores all fields that are not identified as special fields as ordinary fields.

Note: You can query all stored fields using field specifiers in FieldText queries. You can also query

Index fields using text queries.

Related Topics

- ["Index Fields" on page 83](#)
- ["NumericType Fields" on page 87](#)
- ["NumericDateType Fields" on page 86](#)
- ["FieldCheckType Fields" on page 88](#)
- ["FieldText Search" on page 222](#)

Index XML Attributes

You can index XML attributes in the same way that you index ordinary fields. However, you must refer to them using the following format for IDOL Server to be able to read them:

**/tagName/_ATTR_attributeName*

where:

<i>tagName</i>	is the name of the tag.
<i>attributeName</i>	is the name of the attribute that you want IDOL Server to read.

For example:

```
<FARM ANIMAL="sheep" COLOR="white"> Farmer Joe </FARM>
```

To identify the ANIMAL attribute to IDOL Server, refer to it as

```
*/FARM/_ATTR_ANIMAL
```

To identify the COLOR attribute to IDOL Server, refer to it as

```
*/FARM/_ATTR_COLOR
```

Example:

```
<ROOM Name="The Kitchen">  
<FURNITURE>Table</FURNITURE >  
<ITEM Type="China">Dish</ITEM>  
</ROOM>
```

To identify the Name attribute to IDOL Server, refer to it as

```
*/ROOM/_ATTR_Name
```

To identify the Type attribute to IDOL Server, refer to it as

```
*/ITEM/_ATTR_Type
```

To store XML attributes in Index fields

1. Open the IDOL Server configuration file in a text editor.
2. List an indexing process in the [FieldProcessing] section.

For example:

```
[FieldProcessing]
0=MyFirstProcess
2=IndexingFields
```

3. Create a section for the indexing process, in which you create a property for the process (you define a property later, by using one or more applicable configuration parameters). Identify the fields that you want to associate with the processes.

Note: The property that you create must not have the same name as the process.

For example:

```
[MyFirstProcess]
Property=MyFirstProperty
PropertyFieldCSVs=*/MyField,*/MySecondField
PropertyMatch=*myString*
```

```
[IndexingFields]
Property=IndexFields
PropertyFieldCSVs=*/FIELD/_ATTR_ANIMAL,*/FIELD/_ATTR_COLOR,*/ROOM/_ATTR_
Name,*/ITEM/_ATTR_Type
```

4. Create a section for your indexing property in which you set the `Index` parameter to `True`.

For example:

```
[MyFirstProperty]
HiddenType=True
```

```
[IndexFields]
Index=True
```

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

Configure IDOL Server for Language and to Encode

Before you index documents that contain different languages into IDOL Server, you must configure it to recognize the language and encoding of documents, so that it can deal with them appropriately.

IDOL Server can automatically identify the language and encoding of a document when it indexes it. You must have an IDOL Server license that includes automatic language detection. To use this feature, set `AutoDetectLanguagesAtIndex` to `True` in the `[Server]` section of the IDOL Server configuration file.

If your license does not include this functionality, you must specify the language and encoding of documents that you index into IDOL Server. Alternatively, you can configure a field process that allows IDOL Server to read the language of a document from one of its fields.

Related Topics

- ["Language Support" on page 99](#)

Optimize Index Process

The speed of the indexing process is usually less critical than the speed of the query process. However, with large amounts of data being indexed into IDOL Server, it is important to improve the efficiency of the process where possible. In addition, the way that you configure the indexing process can affect the efficiency of the query process.

Index Process

The indexing process works in two stages:

1. IDOL Server creates a representation of the new data in the index cache.
2. IDOL Server synchronizes the cache with data that it currently contains, and stores the new data on disk, removing it from the index cache.

When you schedule indexing, consider the recommendations in this chapter about IDOL Server content (particularly on selecting fields to index), and on running indexing and querying processes at different times. In addition, the delayed synchronization feature allows you to change the stage at which IDOL Server synchronizes the index cache. What you use depends on whether your priority is achieving fast query speeds or making new information available to the user as quickly as possible.

Delayed Synchronization

The delayed synchronization feature allows you to select how IDOL Server synchronizes the index cache with the IDOL Server data. This process is useful in systems where you schedule indexing tasks at times when IDOL Server is also handling queries.

By default, synchronization occurs as soon as a representation of data has been made in the index cache. New data is available to the user (as query results) quickly, so use this setting in systems where current data is the priority. However, synchronization uses resources that IDOL Server could otherwise use for querying.

Delayed synchronization reduces the impact of this effect by collecting multiple data representations in the index cache and then synchronizing them all with IDOL Server data in one go. This process is useful in systems where query speed is more important than having current data.

Note: HPE recommends using delayed synchronization if you index a lot of small files (files that are smaller than 100 MB).

The `DeLayedSync` parameter in the `[Server]` section of the IDOL Server configuration file allows you to specify whether the indexing process uses delayed synchronization.

To configure delayed synchronization

1. Set `DeLayedSync` to `True` if you want IDOL Server to delay synchronization. In this case, IDOL Server stores data on disk only when:
 - the index cache is full.
 - the index cache contains some data and the timeout specified by `MaxSyncDeLay` expires.

2. Set `De1ayedSync` to `Fa1se` if you do not want IDOL Server to delay synchronization.

Chapter 3: Index Data

IDOL Server stores the content of documents in data indexes. (The default data indexes are in the IDOL Server databases *News* and *Archive*.) The process of storing content in IDOL Server is called *indexing*. This section describes the indexing process.

- [Index Overview](#) 43
- [Process Data before you Index](#) 44
- [DREADD: Index IDX and XML Files Directly](#) 44
- [DREADDATA: Index Data over a Socket](#) 50
- [Index Stop Words](#) 54
- [Index Nonalphanumeric Characters](#) 54
- [Prevent Duplicate Documents](#) 58
- [Add Metadata to Documents](#) 65
- [Check Index Status](#) 65
- [Tag Documents into Clusters](#) 69

Index Overview

You can index only files in XML or IDX format into IDOL Server. If the data that you want to index into IDOL Server is in XML format, you can index it directly into IDOL Server, without having to first *import* it (convert its content and metadata to IDX).

IDOL connectors use the DREADD and DREADDATA index actions to index data into IDOL Server. You also can use these actions to directly index data into IDOL Server.

Note: Before you index data into IDOL Server, review the setup instructions described in "[Configure Content Storage](#)" on page 35.

If your data is not in XML format, you must first import it. You can import data using one of three methods.

- **Import with a connector.** The IDOL connectors (for example, File System Connector, HTTP Connector, Oracle Connector, and so on) retrieve documents from different repositories and import them into IDX or XML file format. Refer to the appropriate connector guide for further information on how to import documents.
- **Import manually.** You can create a text file in either XML or IDX format, which contains the information that you want to index into your IDOL Server in specific IDOL Server fields.
- **Import with IDOL Admin.** You can use the wizard on the **Index** tab on the Console page in the Control section of IDOL Admin to submit data for IDOL Server to index. For more information, refer to the *IDOL Admin User Guide*.

Related Topics

- ["Manually Create IDX Files " on page 511](#)

- ["IDX Format" on page 511](#)

After the documents are in XML or IDX file format, you can index them into IDOL Server using one of two methods.

- **Index with a connector.** The IDOL connectors index the IDX files that they create into the IDOL Server to which they connect. Refer to the appropriate connector guide for detailed information on how to index documents.
- **Index directly.** You can index XML and IDX files into an IDOL Server by issuing an HTTP request from your Web browser.

Related Topics

- ["DREADD: Index IDX and XML Files Directly" below](#)
- ["DREADDDATA: Index Data over a Socket" on page 50](#)

Depending on where the data to index is located, the indexing steps for each document take place in the following order.

Local document (accessed through the file system)	Remote document (accessed over the indexing port)
<ul style="list-style-type: none"> • IDOL Server receives a file path. • IDOL Server opens the file and reads the document data. • IDOL Server indexes the document. 	<ul style="list-style-type: none"> • IDOL Server receives a stream of data over the port. • IDOL Server saves the data locally. • IDOL Server opens the local file and reads the document data. • IDOL Server indexes the document.

Process Data before you Index

The IDOL Connector Framework Server (CFS) allows you to process documents that it receives (for example from an IDOL connector) before it sends them to the IDOL Server index. For more information about using CFS to process your documents, refer to the *Connector Framework Server Administration Guide*.

DREADD: Index IDX and XML Files Directly

The DREADD index action (case sensitive) directly indexes an IDX or XML file that is located on the same machine as the IDOL Server. For example:

`http://IDOLhost:indexPort/DREADD?requiredParams&optionalParams`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the indexing port of the IDOL Server (specified in the <code>IndexPort</code> parameter in the [Server] section of the IDOL Server configuration file).

DREADD Parameters

The following parameters are available for the DREADD index action.

Required Parameters

You must include the following parameters.

filename or path	The name or location of the IDX or XML file you want to index. The DREADD index action also accepts IDX or XML files that are compressed by gzip.
DREDDbName=database	The name of the IDOL database into which the file content indexes. You do not require this parameter if your IDX or XML files already contain a database field. By default, IDOL Server reads from this field.

Optional Parameters

You can include any of the following parameters as required. Separate parameters with an ampersand (&).

ACLFIELDS=fields	One or more document fields from which IDOL Server reads ACLs (access control lists).
CantHaveFields=fields	One or more document fields to discard (not index). By default, all fields are indexed.
DatabaseFields=fields	One or more document fields that contain the name of the database into which you want to index the document.
DateFields=fields	One or more document fields from which you want IDOL Server to read the date of the document.
Delete	IDOL Server deletes the IDX or XML file after it is indexed.
DocumentDelimiters=fields	One or more fields in a file that indicates the beginning and end of an individual document, when the file contains multiple documents. Document delimiters cannot be nested.
DocumentFormat=XML IDX	The format (XML or IDX) that IDOL Server assigns to a file when a file format is ambiguous.
ExpiryDateFields=fields	One or more fields that contain the expiration date of the document (the date when it is deleted or—if you set <code>ExpireIntoDatabase</code> in the IDOL Server configuration file—when it is moved into another database).

<code>FlattenIndexFields=fields</code>	<p>One or more fields in a hierarchically structured document whose content you want to index as one level.</p>
<code>IDXFieldPrefix=prefix</code>	<p>When you index an IDX file, IDOL Server transforms it into XML by placing it under the <code>Document</code> subtree (each of the IDX fields is given the prefix <code>Document</code>, to construct a simple XML hierarchy). If you do not want to call the subtree <code>Document</code>, use this parameter to specify a different name.</p>
<code>IndexFields=fields</code>	<p>One or more fields that you want to index explicitly into IDOL Server.</p> <p>Explicitly indexing fields optimizes the query process when you use these fields to restrict queries. Use Index fields to hold data that is particularly significant to you (for example, the title of the document), and that you are likely to use frequently to restrict queries.</p>
<code>KeepExisting=True False</code>	<p>If you set <code>KillDuplicates</code> to <code>Reference</code> or <code>FieldName</code>, you can set <code>KeepExisting</code> to <code>True</code> to discard the document received for indexing and keep the matching document that already exists in the database.</p>
<code>KillDuplicates=option</code>	<p>Specify one of the options described in "Deduplication Options—KillDuplicates" on page 59 to determine how IDOL Server handles indexing of duplicate text.</p> <p>If you do not use the <code>KillDuplicates</code> option, indexing defaults to the setting specified for the <code>KillDuplicates</code> parameter in the <code>[Server]</code> section of the IDOL Server configuration file.</p>
<code>KillDuplicatesDB=database</code>	<p>The database to which IDOL Server moves duplicate documents.</p>
<code>KillDuplicatesDBField=fields</code>	<p>The name of a field in duplicate documents that contains the name of the database to which IDOL Server moves duplicate documents. If the field does not exist in the document, it uses the value of <code>KillDuplicatesDB</code>.</p>
<code>KillDuplicatesMatchDBs=Db1+Db2+Db3</code>	<p>Lists the databases to search for duplicate matches, separated by plus signs (+).</p>
<code>KillDuplicatesMatchTargetDB=True False</code>	<p>Whether to search the database that the document is to index into for duplicate matches.</p>

	Set to True to search the database.
KillDuplicatesPreserveFields=True False	The name of IDX fields that IDOL Server must copy to a newer copy of the same document, when it performs KillDuplicates.
LanguageFields=fields	One or more fields that contain the language type of the document.
LanguageType=type	The language type to apply to a document if it has no fields that specify the language type. You define language types and how to handle them in the [LanguageTypes] section of the IDOL Server configuration file.
MustHaveFields=fields	One or more fields (in an IDX document only) that IDOL Server must store. By default, IDOL Server stores all fields. If you use this parameter, IDOL Server discards all document fields that are not specified—which means that you cannot query or print them.
SectionFields=fields	One or more fields that indicate the start of a new section in the document (for IDX only; IDOL Server automatically sections XML data).
SecurityFields=fields	One or more fields that contain the security type of the document.
SecurityType=type	The security type to apply to a document if the document has no fields that specify the security type. You define security types and how to handle them in the IDOL Server configuration file.
TitleFields=fields	One or more fields from which IDOL Server reads the document title.

Related Topics

- ["Specify Field Names" on the next page](#)
- ["Use KeepExisting to Minimize the Index Load" on page 63](#)

Note: Parameters used in the DREADD index action override the equivalent settings specified in the IDOL Server configuration file.

DREADD Examples

http://MyHost:20001/DREADD?C:\Documents and Settings\JBrown\Market.idx&DREDBNAME=Biz

```
http://MyHost:20001/DREADD?D:\Content\Price.idx&DREDBNAME=Shop&KillDuplicates=Reference
```

Specify Field Names

If you specify multiple field names, separate them with commas. There must be no space before or after any comma. You can use Wildcards.

When naming fields, use the format */FieldName* to match root-level fields, **/FieldName* to match all fields except root-level, or */Path/FieldName* to match fields with the specified path. If you just specify *FieldName*, IDOL Server automatically adds */** to it.

Related Topics

- ["Wildcards in Queries" on page 280](#)

ACLFields Example

```
&ACLFields=*/AUTONOMYMETADATA
```

IDOL Server reads ACLs from any fields called AUTONOMYMETADATA.

DatabaseFields Example

```
&DatabaseFields=Document/DREDbName,*/myDB
```

IDOL Server indexes the document into the database with the name contained in any DREDbName field below the Document level and with the name contained in any fields called myDB.

DateFields Example

```
&DateFields=Document/DREDate,*/myDocDate
```

IDOL Server extracts dates from any fields called DREDate contained below the Document level and from any fields called myDocDate.

Note: If you index documents that contain fields with partial dates into IDOL Server, the document is assigned a metadata field specifying a full date. This date is the first day of the month if the day is not specified, and the first of January if neither the day or the month are specified. The format of the partial date must be a format specified in the DateFormatCSVs configuration parameter.

DocumentDelimiters Example

```
&DocumentDelimiters=*/DOCUMENT,*/SPEECH
```

IDOL Server marks the beginning and end of individual documents in the file by opening and closing DOCUMENT and SPEECH tags.

ExpiryDateFields Example

```
&ExpiryDateFields=Document/DREExpiryDate,*/myExpiryDate
```

IDOL Server reads the expiration date from any DRExpiryDate field below the Document level and from any fields called myExpiryDate.

FlattenIndexFields Example

```
<documents>
  <article id="_21498602">
    <url>http://example.com/21490.html</url>
    <hltext_display>The history of pharmacogenetics </hltext_display>
    <source>Science Online</source>
    <media_type>text</media_type>
    <subject>
      <text>The prologue to pharmacogenetics began to play out around 1850 and
spanned some 60 years into the 1900s.</text>
      <text>In 1953, the molecular basis of heredity, the double helix of DNA,
was described.</text>
    </subject>
    <valid_time>Jul 13 2001 5:00AM</valid_time>
  </article>
</documents>
```

If you specify FlattenIndexFields=*/subject, and index the above document, IDOL Server indexes any content in a subject field or a field within a subject field as the content of the subject field.

If you then query the subject field for a particular term that is actually contained in a level below the subject field (such as the term “pharmacogenetics”), the content of both text fields returns. If you do not flatten the subject field the query does not return results, because the subject field itself does not contain the term.

IndexFields Example

```
&IndexFields=*/DRECONTENT,*/DRETITLE
```

IDOL Server explicitly indexes the DRECONTENT and DRETITLE field in documents.

LanguageFields Example

```
&LanguageFields=Document/DRELanguageType,*/myLanguageType
```

In this example, IDOL Server reads the language type of documents from any DRELanguageType field below the Document level and any myLanguageType fields.

MustHaveFields Example

```
&MustHaveFields=*/DRECONTENT,*/DRETITLE
```

In this example, IDOL Server stores only the DRECONTENT and DRETITLE fields in the document.

SectionFields Example

```
&SectionFields=Document/DRESection,*/mySection
```

In this example, any DRESection field below the Document level and any mySection fields indicate the start of a new section.

SecurityFields Example

```
&SecurityFields=Document/DRESecurity,*/mySecurity
```

In this example, IDOL Server reads the security type of documents from any DRESecurity field below the Document level and any mySecurity fields.

TitleFields Example

```
&TitleFields=*/DRETITLE
```

In this example, IDOL Server reads a document title from its DRETITLE field.

DREADDATA: Index Data over a Socket

The DREADDATA index action (case sensitive) allows you to directly index data over a socket into IDOL Server. For example:

```
DREADDATA?optionalParamsData#DREENDDATAkillDuplicatesOption\n\n
```

Note: This index action requires a POST request method.

Related Topics

- ["Send Data with a POST Method" on the next page](#)

DREADDATA Parameters

The following parameters are available for the DREADDATA index action.

<i>Data</i>	<p>The content of the IDX or XML document to index. You can use gzipped data. You must add #DREENDDATA to the end of your data. #DREENDDATA must be uncompressed, even if your data is in gzip format.</p> <p>This parameter is required.</p>
<i>optionalParams</i>	<p>The DREADDATA action accepts the same optional parameters as the DREADD action, except for KillDuplicatesOption.</p> <p>Note: The DREDBName parameter, which is required for DREADD, is an optional parameter for DREADDATA.</p>
<i>killDuplicatesOption</i>	<p>This optional parameter is equivalent to the KillDuplicates parameter for DREADD, except that it does not use the "KillDuplicates=Option" syntax.</p> <p>You append the option directly to the #DREENDDATA tag that ends the <i>Data</i> parameter (for example, #DREENDDATAREFERENCE). The following option</p>

	<p>values are available:</p> <ul style="list-style-type: none">• NOOP (available for DREADDATA only)• NONE• REFERENCE• REFERENCEMATCHW• <i>FieldName</i>• <i>ReferenceField</i>, GREATER: <i>VersionField</i> <p>For more information about these options, see "Deduplication Options—KillDuplicates" on page 59.</p>
--	--

Related Topics

- "[DREADD Parameters](#)" on page 45
- "[Deduplication Options—KillDuplicates](#)" on page 59

Note: Parameters that you set in the DREADDATA action override any equivalent settings specified in the IDOL Server configuration file.

Send Data with a POST Method

You must send the DREADDATA action using a POST request method. There are two ways to send a POST request over a socket to IDOL Server:

- use the Curl command-line tool
- use a script

Note: You can use these methods for other actions that require a POST request method, such as DREREPPLACE, but you must modify the script.

Use the cURL Command-Line Tool

Curl is an open source command-line tool for transferring files with URL syntax. If you have cURL installed on your computer, and a command prompt that allows you to use new lines, you can use a cURL request to send the action. For example:

```
curl "http://host:port/DREADDATA?" -d "  
#DREREFERENCE Test  
#DREDBNAME Default  
#DREENDDOC  
#DREENDDATA  
"
```

Alternatively, you can add the #DREENDDATA tag into the IDX or XML document, or gzipped IDX or XML file, after the data to index. The #DREENDDATA tag must be uncompressed. Then you can use a cURL command to send this document to IDOL Server.

For example:

```
curl --data-binary @filename "http://host:port/DREADDATA?"
```

where *filename* is the name of the IDX, XML, or gzipped IDX or XML file to index.

For information on cURL, refer to <http://curl.haxx.se/>.

Use a Script

Another method of sending a POST request is to use a script to open a socket and send the data. The following is an example script in the Perl programming language that sends a DREADDATA index action to a specified port.

To run the script

- Open a command prompt and type the following command:

```
PerlScript.pl HostNameIndexPortFilename [Parameters]
```

where:

<i>PerlScript</i> .pl	is the name of the file that contains the Perl script that performs the DREADDATA index action.
<i>HostName</i>	is the host name or IP address of the host on which the IDOL Server runs.
<i>IndexPort</i>	is the index port for the IDOL Server you send the data to.
<i>Filename</i>	is the name of the file that you index into IDOL Server by using the DREADDATA action. You can use an IDX, XML, or gzipped IDX or XML file.
<i>Parameters</i>	are any optional parameters that you use for the DREADDATA action.

Example Perl Script

```
# Performs a /DREADDATA index action
use IO::Socket;

if (@ARGV<3)
{
    print "Usage: doDreAddData.pl <hostname> <indexport> <filename> [parameters]\n";
    exit;
}

my $host = $ARGV[0];
my $port = $ARGV[1];
my $filename = $ARGV[2];
my $params = $ARGV[3];
my $footer = "\r\n#DREENDDATA\r\n\r\n";
my $iaddr = inet_aton($host) or die "$!";
my $paddr = sockaddr_in($port,$iaddr);
my $proto = getprotobyname('tcp');

socket(SOCK, PF_INET, SOCK_STREAM, $proto) or die "socket: $!";
connect(SOCK, $paddr) or die "connect: $!";

my $filesize = -s $filename;
```

```
$filesize += length($footer);  
open (FILE, $filename) or die "couldn't open file: $!";  
  
print SOCK "POST /DREADDATA?$params HTTP/1.1\r\n";  
print SOCK "Connection: close\r\n";  
print SOCK "Content-Length: $filesize\r\n\r\n";  
while (<FILE>)  
{  
    print SOCK $_;  
}  
print SOCK $footer;  
  
SOCK->autoflush(1);  
  
while(<SOCK>){  
    print $_;  
}  
  
close FILE;  
close SOCK;
```

DREADDATA Examples

Example 1:

```
POST /DREADDATA?LanguageType=EnglishUTF8 HTTP/1.0  
Content-Length: 604  
#DRREFERENCE 392348A0  
#DREFIELD authorname1="Brown"  
#DREFIELD authorname2="Edgar"  
#DREFIELD title="Dr."  
#DREDATE 1998/08/06  
#DRETITLE  
Jurassic Molecules  
#DRECONTENT  
Using a technique called test tube evolution, scientists created a nucleic acid  
enzyme, the first known enzyme that uses an amino acid to start chemical activity.  
Scientists hope that the creation of this molecule will lead to the elusive  
precursor. The precursor, by definition, will have to contain both the genetic code  
for replication and an enzyme to trigger self replication.  
#DREDBNAME Science  
#DREENDDOC  
#DREENDDATAREFERENCE
```

Example 2:

```
POST /DREADDATA?DREDbName=Poetry&DateFields=*/Date HTTP/1.0  
Content-Length: 515  
Content-Type: application/xml  
<DOCUMENT>
```

```
<DREREFERENCE>572801A2</DREREFERENCE>  
<AUTHOR>George Eliot</AUTHOR>  
<Date>2005/24/03</Date>  
<DRETITLE>Roses</DRETITLE>  
<DRECONTENT>  
You love the roses - so do I. I wish  
The sky would rain down roses, as they rain  
From off the shaken bush. Why will it not?  
Then all the valley would be pink and white  
And soft to tread on. They would fall as light  
As feathers, smelling sweet; and it would be  
Like sleeping and like waking, all at after!  
</DRECONTENT>  
</DOCUMENT>  
#DREENDDATANOOP
```

Index Stop Words

IDOL Server does not normally index stop words, but you can index them by using the `StopWordIndex` configuration parameter in the `[LanguageTypes]` section of the configuration file. For example:

```
[LanguageTypes]  
StopwordIndex=1
```

Queries match stop words only in a phrase, as shown in the following examples.

Query	Result
winnie the bear	IDOL Server ignores the stop word <i>the</i> , as usual.
"winnie the bear"	IDOL Server includes the stop word <i>the</i> in the exact phrase search and matches it.
copy of "winnie the bear"	IDOL Server ignores the stop word <i>of</i> , but matches <i>the</i> within the phrase.

Note: Wildcard search terms do not expand to stop words, even in phrases.

Related Topics

- ["Stop Word Lists for Supported Languages" on page 509](#)

Index Nonalphanumeric Characters

You can configure nonalphanumeric characters to index differently based on the query result that you want. By using these configurations, you can query nonalphanumeric terms in exactly the same form as in a document to match that document.

You must set the configuration parameters discussed in this section in the `[MyLanguage]` section of the IDOL Server configuration file.

Note: If you want to change these settings after you index content into IDOL Server, you must reindex the content.

Related Topics

- ["Enable Transliteration" on page 115](#)

Term Separators

IDOL Server automatically generates separators for each language to determine where one term ends and another begins. These include characters such as spaces, tabs, carriage returns, and line feeds.

To ensure that IDOL Server uses a character as a separator, specify it in the `AugmentSeparators` configuration parameter. IDOL Server replaces all separator characters with a space.

For example, the following table describes the query matching for when `AugmentSeparators=, -`.

Indexed string	Query terms matched
second-hand guitar	<ul style="list-style-type: none">• <i>second</i>• <i>hand</i>• <i>guitar</i>

Note: The hyphen is a separator only if it is not listed in `HyphenChars`, because `HyphenChars` takes precedence over separators.

To ensure that IDOL Server does not use a character as a separator, specify it in the `DiminishSeparators` configuration parameter. IDOL Server removes nonseparators at index time.

For example, the following table describes the query matching for when `DiminishSeparators=%`.

Indexed string	Query terms matched
file_name	<i>filename</i>

To ensure that IDOL Server indexes a character as its own token, specify it in the `SoftSeparators` configuration parameter.

For example, the following table describes the query matching for when `SoftSeparators=1234567890`.

Indexed string	Query terms matched
459	<ul style="list-style-type: none">• <i>4</i>• <i>5</i>• <i>9</i>• <i>45</i>• <i>59</i>• <i>459</i>

In this example, IDOL Server tokenizes all numbers as single digits, so that 459 is indexed as 4 5 9.

Related Topics

- ["Hyphenated Terms" on the next page](#)

Index Nonalphanumeric Characters for Retrieval

To ensure that a nonalphanumeric character is available for querying, specify it in the `TangibleCharacters` configuration parameter.

For example, the following table describes the query matching for when `TangibleCharacters=?!`.

Indexed string	Query terms matched
help!	<i>help!</i> Queries do not match documents that contain the same word without the exclamation mark.

Note: You cannot specify spaces, returns, and tabs as `TangibleCharacters`.

To ensure that IDOL Server indexes numbers with decimals or commas together as a single term for querying, specify both characters in the `NumberPunctuation` configuration parameter.

IDOL Server treats characters that you set as `NumberPunctuation` as `TangibleCharacters` when they occur in terms with a number on both sides.

For example, the following table describes the query matching for when `NumberPunctuation=.,`.

Indexed string	Query terms matched
815,290.50	<i>815,290.50</i>
73.8A	<i>73.8A</i>
738.	<i>738</i> (number punctuation does not apply.)

Note: These results can vary depending on your `IndexNumbers` configuration parameter setting.

When you configure a character in `NumberPunctuation` and another tokenization parameter, such as `HyphenChars`, `TangibleCharacters`, or `AugmentSeparators`, IDOL Server processes the parameters in order of precedence. For example, `TangibleCharacters` takes priority over `NumberPunctuation`, so if you configure the period in both parameters, you get the same results as if you only configured it in `TangibleCharacters`.

For more information about the order in which IDOL Server applies different tokenization configurations, refer to *IDOL Expert*.

When a configured `NumberPunctuation` character occurs in a value where it does not have a number on each side, the processing depends only on your other configuration. For example, the following table describes query matching for the value `25.R` in various configurations. This value does not activate `NumberPunctuation`, because there is a number on only one side of the period.

If the period (.) is configured as	Query terms matched
non-separator (default)	25R
HyphenChars	25, R, and 25R
TangibleCharacters	25.R
AugmentSeparators	25 and R.

Related Topics

- ["Configure the Number Index Process" on page 85](#)

Hyphenated Terms

By default, when IDOL Server indexes a hyphenated term, it stems each of its components and indexes them. It also removes the hyphen from the term, stems the resulting term, and indexes that.

For example, the following table describes the default query matching for hyphenated terms.

Indexed string	Query terms matched
second-hand guitar	<ul style="list-style-type: none"> • <i>second</i> • <i>hand</i> • <i>secondhand</i> • <i>guitar</i>

To treat other characters as hyphens, specify them in the HyphenChars configuration parameter.

For example, the following table describes the query matching for when HyphenChars=-&.

Indexed string	Query terms matched
Barnes&Noble	<ul style="list-style-type: none"> • <i>Barnes</i> • <i>Noble</i> • <i>BarnesNoble</i>

Note: To stop IDOL Server from indexing hyphenated terms this way, set HyphenChars=NONE. This means that no characters are used as HyphenChars. The default setting is HyphenChars=-&.

At query time, IDOL Server tokenizes the hyphenated term as the whole term and individual subterms, all separated by the configured DefaultQueryOperator. For example, if the DefaultQueryOperator is OR (the default value), a query for second-hand becomes secondhand OR second OR hand.

Note: If you change the DefaultQueryOperator, you might not get expected query results for hyphenated terms.

The separate parts of a hyphenated term (such as *second* and *hand*) are available only for basic keyword searches, and they do not match proximity expressions or field-restricted searches. For

example, if a document contains the phrase `second-hand car`, a keyword search for `hand` matches the document, but an exact phrase search for `hand car` does not.

In most situations, HPE recommends that you set `HyphenChars` to `NONE`, and use the hyphen as a separator (for example, set `AugmentSeparators` to `-`). For more information, refer to *IDOL Expert*.

Character Tokenization

You can tokenize characters into *N-grams* of a specified size. Set the `NGram` configuration parameter in your language configuration section to the number of characters to use in each N-gram group.

Note: You must not use `NGram` with the `SentenceBreaking` configuration parameter.

For example, if you set `NGram` to 2, IDOL Server tokenizes the word *Hello* as:

```
he el ll lo
```

To tokenize only multibyte strings, set `NGramMultiByteOnly` to `True`.

```
[Japanese]
NGram=2
NGramMultiByteOnly=True
```

For this configuration, if you have a document that contains both English and Asian (multibyte) text, IDOL Server tokenizes the Asian text according to the `NGram` parameter. It does not tokenize the English text.

To tokenize only multiple-byte strings in Chinese, Japanese, and Korean characters (and ignore multiple-byte strings in other languages), set `NGramOrientalOnly` to `True`.

```
[Japanese]
NGram=2
NGramOrientalOnly=True
```

For this configuration, if you have a document that contains multibyte text in both Japanese and Greek, IDOL Server tokenizes the Japanese text according to the `NGram` parameter. It does not tokenize the Greek multibyte text.

Prevent Duplicate Documents

You can configure IDOL Server to implement *deduplication* when indexing documents. This process prevents storage of the same document or document content. If IDOL Server determines that the document to index matches an existing document, it replaces the existing document with the new document.

IDOL uses deduplication options to determine whether documents match. See "[Deduplication Options—KillDuplicates](#)" on the next page.

You can enable deduplication in one of three ways:

- Enable deduplication for all indexing jobs by using the `KillDuplicates` configuration parameter in the `[Server]` section of the IDOL Server configuration file. See "[Enable Deduplication for all Index Jobs](#)" on page 61.

You can use the `KillDuplicatesChecksumField` configuration parameter with deduplication to prevent unnecessary updating of existing documents in IDOL Server. See ["Use KillDuplicatesChecksumField to Prevent Unnecessary Indexing"](#) on page 62.

You can also use the `KillDuplicatesPreserveFields` configuration parameter with deduplication to copy the specified IDX fields from an existing document to a newer version.

- Enable deduplication for individual indexing jobs by using the `KillDuplicates` action parameter in the DREADD and DREADDATA actions. See ["Enable Deduplication for Individual Index Jobs"](#) on page 63.

Use the `KeepExisting` action parameter with deduplication to discard the incoming document instead of replacing the existing document. This option reduces the indexing load. See ["Use KeepExisting to Minimize the Index Load"](#) on page 63.

- Enable deduplication when indexing with Connector Framework Server (CFS) by setting the `KillDuplicates` configuration parameter for the connector. See ["Enable Deduplication for Connector Index Jobs"](#) on page 63.

Some other IDOL Server parameters affect the behavior of the deduplication settings. See ["Deduplication Constraints"](#) on page 64.

You can deduplicate after indexing by using the DREDUPLICATE index action. See ["Locate Duplicate Documents"](#) on page 382.

Deduplication Options—KillDuplicates

Use the following parameters to specify deduplication options. IDOL Server uses these parameters to determine whether documents match.

- The `KillDuplicates` parameter specified in either the `[Server]` section of the IDOL Server configuration file or in the DREADD or DREADDATA index action.
- The `KillDuplicates` parameter specified in the `[Indexing]` section of the CFS configuration file.

The following options are available for the deduplication parameters.

NONE	Allows duplicate documents in IDOL Server. IDOL Server does not replace or delete documents.
REFERENCE	Replaces an existing document with the new document if the document to index has the same value in its DREREFERENCE field.
REFERENCEMATCH <i>N</i>	Replaces the existing document with the new document if the content of the document to index is more than <i>N</i> percent similar to the existing document. IDOL Server determines the similarity by comparing the content of the <code>SourceType</code> fields in the document, or the <code>Index</code> fields if no <code>SourceType</code> fields are configured. Note: This method can deduplicate only documents that are already synced in the IDOL Server index. It cannot deduplicate similar documents in the same index job.
<i>FieldName</i>	Replaces the existing document with the new document if the document to index contains a <code>ReferenceType</code> field named <i>FieldName</i> that has the same content as the <i>FieldName</i> field in the existing document.

	<p>You can specify multiple <code>ReferenceType</code> fields in this option (separated by a plus symbol or space), in which case IDOL Server deletes documents that contain any of the specified fields with identical content.</p> <p>Note: You identify fields as <code>ReferenceType</code> fields through field processes in the IDOL Server configuration file. If you list multiple fields in the same <code>PropertyFieldCSVs</code> parameter where you list the <code>FieldName</code> for deduplication, IDOL Server uses all the fields to eliminate duplicate documents. If you want to define multiple <code>ReferenceType</code> fields but do not want to use all fields for duplicate elimination, set up multiple field processes.</p>
<p><code>ReferenceField</code> <code>,GREATER:</code> <code>VersionField</code></p>	<p>Replaces the existing document with the new document if the document to index contains a <code>ReferenceType</code> field named <code>ReferenceField</code> that has the same content as the <code>ReferenceField</code> field in the existing document, and if the <code>VersionField</code> field in the document to index has a higher value than the <code>VersionField</code> in the existing document. For XML documents, you must fully qualify the path of the XML field that you want to use as the version field (you cannot use wildcard values).</p> <p><code>VersionField</code> must contain a positive integer value, but you do not need to configure it as a numeric field. If only one of the incoming and current documents has a valid value in the <code>VersionField</code>, IDOL Server keeps the version with a valid <code>VersionField</code>. When both documents have the same <code>VersionField</code>, IDOL Server keeps the existing document.</p> <p>Note: When you index IDX documents, for the version comparison to work correctly, the value in the field that you use as the <code>VersionField</code> must be listed in quotation marks (""). That is, the field must have the following format in the IDX:</p> <pre>#DREFIELD MyField=""</pre> <p>IDOL Server treats existing documents with a missing or non-numeric value in the <code>VersionField</code> as having a version number of negative infinity. It treats a new document with a missing or non-numeric value in the <code>VersionField</code> as having a version number of 0.</p>
<p>NOOP (DREADDDATA only)</p>	<p>Use the <code>KillDuplicates</code> parameter in the <code>[Server]</code> section of the IDOL Server configuration file to determine how to treat duplicate documents.</p> <p>Note: This option is available only for the DREADDDATA action.</p>

Related Topics

- ["ReferenceType Fields" on page 90](#)
- ["Enable Deduplication for all Index Jobs" on the next page](#)

When you specify a deduplication option, note that:

- If you postfix any of these options with =2, IDOL Server applies the KillDuplicates process to all databases, rather than just the database into which the current IDX or XML file indexes. For example:

```
KillDuplicates=REFERENCE=2
```

- The setting in the KillDuplicates option in either the DREADD or DREADDATA index action overrides the setting in the KillDuplicates configuration parameter.

Enable Deduplication for all Index Jobs

To enable deduplication for all indexing jobs—in other words, to set deduplication by default for the DREADD and DREADDATA actions—use the KillDuplicates configuration parameter in the [Server] section of the configuration file. Note that you must enable deduplication before you start indexing documents into IDOL Server.

You can use the KillDuplicatesChecksumField parameter to configure IDOL to reverse normal deduplication and retain the existing document instead of the incoming document, based on the value of a specified field in the incoming document.

You can use the KillDuplicatesPreserveFields parameter to configure one or more IDX fields that IDOL Server copies to a newer version of a duplicate document.

Related Topics

- ["Use KillDuplicatesChecksumField to Prevent Unnecessary Indexing" on the next page](#)

To enable deduplication as the default for all indexing jobs

1. Open the IDOL Server configuration file in a text editor.
2. In the [Server] section, set the KillDuplicates parameter to REFERENCE, REFERENCEMATCHW, the names of the ReferenceType fields to use to determine which documents are duplicates, or a combination of ReferenceType field and a field that contains a document version number. For more information about these options, see ["Deduplication Options—KillDuplicates" on page 59](#), or refer to the *IDOL Server Reference*.

You can identify fields that contain document references by setting up an appropriate field process. When you index a document that has the same value in the same ReferenceType field as an existing document in IDOL Server, IDOL Server detects the duplicate. It deletes the existing document and replaces it with the new one.

3. Save and close the configuration file. Restart IDOL Server for your changes to take effect. You can now index documents into IDOL Server.

Related Topics

- ["Deduplication Options—KillDuplicates" on page 59](#)
- ["Set up ReferenceType Fields" on page 90](#)

Limit ReferenceType Fields used for Deduplication

You identify fields as ReferenceType fields through field processes. If you list multiple fields in the same PropertyFieldCSVs parameter where you list the FieldName for deduplication, IDOL Server uses all the fields to eliminate duplicate documents. For example:

```
[SetReferenceFields]  
Property=Reference  
PropertyFieldCSVs=*/DRREFERENCE,*/URL
```

In this example, IDOL Server uses both the DRREFERENCE field and URL field to eliminate duplicate copies if you set KillDuplicales to DRREFERENCE.

If you want to define multiple ReferenceType fields but do not want to use them all for duplicate elimination, set up multiple field processes. For example:

```
[SetReferenceFields]  
Property=Reference  
PropertyFieldCSVs=*/DRREFERENCE
```

```
[SetMoreReferenceFields]  
Property=Reference  
PropertyFieldCSVs=*/URL
```

In this example, IDOL Server uses only the DRREFERENCE field to eliminate duplicate copies if you set KillDuplicales to DRREFERENCE. It does not use the URL field.

Related Topics

- ["Set up ReferenceType Fields" on page 90](#)

Use KillDuplicalesChecksumField to Prevent Unnecessary Indexing

By default, when IDOL Server detects that a new document is a duplicate of an existing one, it replaces the existing document with the new one.

For either of these two KillDuplicales options, you can also use the KillDuplicalesChecksumField configuration parameter to specify a checksum field. IDOL Server then checks the value of this field in both documents. If the value is the same, IDOL Server keeps the existing document rather than replacing it with the new document.

This process prevents unnecessary updates. For example, when refetching a Web site, use KillDuplicalesChecksumField to configure IDOL to update the index for this site only if the site has changed.

Note: The KillDuplicalesChecksumField must be a ReferenceType field.

Use KillDuplicalesPreserveFields to Preserve a Field

If there is a field that you want to keep in all versions of a document, regardless of whether it is later deleted or changed, you can use the KillDuplicalesPreserveFields configuration parameter.

To preserve fields, set KillDuplicalesPreserveFields to a comma-separated list of fields that you want to save.

When IDOL Server receives a duplicate document, it copies this field from the existing version of the document to the newer version when it performs KillDuplicales.

Note: If there is more than one copy of the document in the IDOL Server index when a new version

arrives, IDOL Server copies the preserve field from the existing duplicate with the highest document ID.

Enable Deduplication for Individual Index Jobs

To enable deduplication for individual indexing jobs, use the `KillDuplicates` action parameter in the `DREADD` or `DREADDDATA` index actions.

You can use the `KeepExisting` action parameter when directly indexing data into IDOL with deduplication to reduce the indexing load.

You can use the following action parameters to move duplicates to a specified database.

- `KillDuplicatesDB`
- `KillDuplicatesDBField`
- `KillDuplicatesMatchDBs`
- `KillDuplicatesMatchTargetDB`
- `KillDuplicatesPreserveFields`

You can use any of the deduplication options with `DREADD` and `DREADDDATA` actions. When you use either of these actions:

- The `KillDuplicates` setting specified in either action overrides the same setting in the `KillDuplicates` configuration parameter.
- If you do not specify the `KillDuplicates` action parameter with either of the actions, the setting in the `KillDuplicates` configuration parameter is used.

Related Topics

- ["DREADD: Index IDX and XML Files Directly" on page 44](#)
- ["DREADDDATA: Index Data over a Socket" on page 50](#)
- ["Use KeepExisting to Minimize the Index Load" below](#)
- ["DREADDDATA Parameters" on page 50](#)
- ["Deduplication Options—KillDuplicates" on page 59](#)

Use KeepExisting to Minimize the Index Load

If you set `KillDuplicates` to `Reference` or `FieldName`, you can use the `KeepExisting` action parameter to minimize the indexing load on IDOL when deduplicating.

Set `KeepExisting` to `True` to reverse normal deduplication and discard the document it has received for indexing and keep the existing matching document that it already contains instead.

Enable Deduplication for Connector Index Jobs

If you use a connector to retrieve documents from a remote repository for indexing into IDOL Server, you can configure the `KillDuplicates` configuration parameter for the CFS to set deduplication.

- The options available for the `KillDuplicates` IDOL Server configuration parameter are also available for the CFS configuration parameter.

- The same constraints for deduplication apply when you configure for deduplication using a `KillDuplicates` option for the CFS.

For more information on connector deduplication, refer to the *Connector Framework Server Reference*.

Related Topics

- ["Deduplication Options—KillDuplicates" on page 59](#)
- ["Deduplication Constraints" below](#)

Deduplication Constraints

There are some constraints on deduplication when using other IDOL parameters.

Use the Combine Operation

IDOL Server cannot use the same `ReferenceType` field for deduplication as it uses for the `Combine` action parameter. The `Combine` operation occurs at query time and clashes with deduplication. If you intend to deduplicate when indexing and use the `Combine` action parameter, you must set up separate `ReferenceType` fields for these processes.

Related Topics

- ["Use KillDuplicates and Combine on ReferenceType Fields" on page 91](#)

Use Deduplication with DIH Reference-Based Indexing

You can enable the DIH for reference-based indexing. Refer to the *DIH Administration Guide*.

If you index documents into IDOL with the DIH enabled for reference-based indexing, it might prevent deduplication of documents with different references. In this case, use only one of the following deduplication options:

- `KillDuplicates=REFERENCE`
- `KillDuplicates=NONE`

Use Deduplication with DIH Field-Based Indexing

You can use field-based indexing in the DIH to ensure correct deduplication in a distributed system. For more information on configuring the DIH for field-based indexing, refer to the *DIH Administration Guide*.

If you set `KeepExisting` to `False`, or use `KillDuplicatesDB` options, it might prevent correct deduplication. To deduplicate correctly, you can distribute data by the `DeDupeHash` field (MD5 hash) of the documents. In this way, DIH sends all duplicates to the same child server. Setting `KillDuplicates` to `DeDupeHash` during the indexing action then ensures accurate deduplication.

To use a field for deduplication, you must configure it as a `ReferenceType` field. You do not need to configure it as `ReferenceType` in the DIH configuration file.

Deduplication of content occurs for all reference fields specified in a single `PropertyFieldCSVs` list in the IDOL Server configuration file. To use only the `DeDupeHash` field to deduplicate, and not also the `DRREFERENCE`, you must set these reference fields in separate field processing sections in the IDOL Server configuration file.

Add Metadata to Documents

When you index a document into IDOL Server, IDOL Server automatically stores all its metadata as fields. You can add additional fields to a document after you index it by running a DREREPLACE index action.

Related Topics

- ["Set up the Field Index Process" on page 38](#)
- ["Change Document Field Values" on page 391](#)

Check Index Status

You can check whether the indexing of data into IDOL Server is successful by entering the following URL into your Web browser:

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

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>port</i>	is the IDOL Server ACI port (the Port value specified in the [Server] section of IDOL Server configuration file).

The IndexerGetStatus action displays the status of the IDOL Server index queue.

Tip: As an alternative to submitting the IndexerGetStatus action in your Web browser, you can use the **Indexer Status** page in the Monitor section of the IDOL Admin interface to view the current queue of indexing jobs. You can also view information about the index queue on the **Index Queue** tab in the Status section of IDOL Admin. For more information, refer to the *IDOL Admin User Guide*.

IndexerGetStatus Example

An IndexerGetStatus action is sent to IDOL Server following a DREADD index action. IDOL Server returns the following output:

```
<autnresponse>
<action>INDEXERGETSTATUS</action>
<response>SUCCESS</response>
<responsedata>
<timeformat>YYYY/MM/DD HH:NN:SS</timeformat>
<state>Inactive (no jobs)</state>
<item>
  <id>1</id>
  <origin_ip>127.0.0.1</origin_ip>
  <received_time>2015/03/31 16:14:43</received_time>
```

```

    <start_time>2015/03/31 16:14:44</start_time>
    <end_time>2015/03/31 16:16:44</end_time>
    <duration_secs>120</duration_secs>
    <percentage_processed>100</percentage_processed>
    <documents_processed>44710</documents_processed>
    <documents_deleted>0</documents_deleted>
    <status>-1</status>
    <description>Finished</description>
    <docidrange>1-44710</docidrange>
    <index_command>
    /DREADD?myfile.idx&KILLDUPLICATES=REFERENCE&DREDBNAME=Archive
    </index_command>
  </item>
</responsedata>
</autnresponse>

```

where:

Tag Name	Description
<timeformat>	The time and date format that the response uses.
<state>	The status of the Content component indexing thread.
<id>	The ID number of the index action.
<origin_ip>	The IP address of the machine that sent the index action to IDOL Server.
<received_time>	The time that IDOL Server received the action.
<start_time>	The time that IDOL Server started processing the index action.
<end_time>	The time that IDOL Server finished processing the index action.
<duration_secs>	The total amount of time in seconds that IDOL spent processing the index action.
<documents_processed>	The number of documents that IDOL Server processed during the indexing job.
<documents_deleted>	The number of documents deleted during the indexing process.
<status>	The status code of the current status of the index action in the IDOL Server index queue.
<description>	The description of the <status> number.
<docidrange>	The range of DocIDs of documents that were processed during the index job.
<index_command>	The index action for the index job.

For the DRECOMPACT index action, IndexerGetStatus also returns a <drecompact_status> section to show the time that each stage of the compaction takes. This section contains <stage> tags, with the following attributes.

name	The name of the compaction stage.
description	The description of the compaction stage.
percentage_complete	The percentage of the compaction stage that is complete.
time_seconds	The time that the compaction stage has taken.
paused	If you pause the DRECOMPACT index action, this attribute is set to True for the paused stage.

The Content component returns the following status messages in the <state> tag to show whether it is currently processing jobs, or if it is paused:

Running
Paused
Paused (DREFLUSHANDPAUSE)
Paused (out of disk space)
Paused (waiting for flush lock)
Inactive (no jobs)

Related Topics

- ["IndexerGetStatus Status Codes" below](#)

IndexerGetStatus Status Codes

Codes in **bold** are status messages. All other codes indicate that there is a problem with the indexing process.

Code	Message	Explanation
-1	Finished	The indexing process is complete.
-2	Out of disk space	IDOL Server ran out of disk space before it completed the the indexing process.
-3	File not found	The index file could not be found.
-4	Database not found	The database into which you are trying to index could not be found.
-5	Bad parameter	The indexing action syntax is incorrect.

Code	Message	Explanation
-6	Database exists	The database that you are trying to create already exists.
-7	Queued	The indexing action is queued and it is run when all preceding indexing actions are complete.
-8	Unavailable	IDOL Server is about to shut down or indexing is paused.
-9	Out of Memory	IDOL Server ran out of memory before the indexing process could be completed.
-10	Interrupted	The indexing action was interrupted.
-11	XML is not well formed	Indexing failed because the XML is not well formed.
-12	Retrying interrupted command	IDOL Server is processing an index action that was previously interrupted.
-13	Backup in progress	IDOL Server is performing a backup.
-14	Max index size reached	The indexing job exceeds the maximum indexing size (your license determines the maximum indexing size).
-15	Max number of sections reached	The indexing job exceeds the maximum number of sections that you can index. Your license determines the number of sections that you can index.
-16	Indexing Paused	The indexing process was paused.
-17	Indexing Resumed	The indexing process was restarted.
-18	Indexing Cancelled	The indexing process was cancelled.
-19	Out of file descriptors	IDOL Server ran out of file descriptors.
-20	LanguageType not found	The language type of the index data could not be found.
-21	SecurityType not found	The security type of the index data could not be found.
-22	Child engines returned differing messages	The child servers returned different messages to the DIH. This code is reported by DIH only.
-23	Badly formatted index command	The indexing action was rejected by a child server because the syntax is not valid.
-25	To be sent to DRE	The index action is queued to be sent to the IDOL Server. This code is reported by DIH only.
-26	DREADDATA: Data received did not include #DREENDDATA	The data in the DREADDATA action did not contain a #DREENDDATA statement indicating the end of the data.

Code	Message	Explanation
-27	Command failed more times than the configured retry limit	The indexing action exceeded the maximum number of retries specified by the <code>MaximumRetries</code> parameter in the DIH configuration. This code is reported by DIH only.
-28	The index ID specified is invalid	The index ID returned by the child server is not valid. This code is reported by DIH only.
-29	Command was redistributed to sibling engines as this engine was either unavailable or not accepting index jobs	The indexing action was sent to sibling servers because the child server was either unavailable or not accepting indexing jobs. This code is reported by DIH only.
-30	Database name too long	The name of the database in which you are indexing documents is too long. The length is defined internally as 63 characters.
-31	Command ignored due to id match	The <code>DREINITIAL</code> action was ignored because it did not match the ID specified in the <code>InitialID</code> parameter.
-33		The database cannot be created because the maximum number of databases was exceeded. The maximum is defined internally as 32,767.
-34	Pending commit	The indexing job is complete and the documents become available for searching after the next delayed sync cycle, which is specified in the <code>DelayedSync</code> parameter.
-35	Initializing	The indexing job is being started. This code is reported by DIH only.
-36	Reading IDX	The IDX file is being read from disk, prior to being sent to the DRE. This code is reported by DIH only.
-38	Processing in remote engine	The target engine of a <code>DREEXPORTREMOTE</code> operation is processing the exported documents.

Note: If the `IndexerGetStatus` action returns a positive number, this number indicates the percentage of the indexing queue that has been completed.

Tag Documents into Clusters

After indexing, you can tag documents into clusters of similar documents. Tagging can be useful for grouping duplicate documents together.

Use the index action `DRETAGDOCCLUSTERS`. This action takes the following parameters.

TagField	The full field name that contains document tags.
MinScore	The matching threshold to determine whether a document belongs to a cluster.
TagSourceField	The full field name to use as the source of the TagField value.
MinID	The first document ID to tag.
MaxID	The last document ID to tag.
CheckSumField	A reference field to use to determine whether a document is an exact match of another document.
TaggedDBName	The database which IDOL Server moves tagged documents to and retrieves tags from.
RelevanceField	The full field name that holds the relevance score of the document to its cluster.
DatabaseMatch	The names of databases that contain documents that you want to tag.
CheckSumDBs	The names of databases that you can checksum match against.
ClusterDBs	The names of databases that you can cluster against. This list includes TaggedDBName if specified.

DRETAGDOCCLUSTERS Example

IDOL Server indexes three documents:

```
#DREREFERENCE A
#DREDBNAME Default
#DREFIELD CHECKSUM="ABCD1234"
#DRECONTENT
apple banana cheese
#DREENDDOC
```

```
#DREREFERENCE B
#DREDBNAME Default
#DREFIELD CHECKSUM="ABCD1234"
#DRECONTENT
apple banana cheese
#DREENDDOC
```

```
#DREREFERENCE C
#DREDBNAME Default
#DREFIELD CHECKSUM="XYZ9876"
#DRECONTENT
apple banana data
#DREENDDOC
```

After indexing, you send the following action:

```
[...]  
/DRETAGDOCCLUSTERS?TagField=DOCUMENT/CLUSTERID&MinScore=60&TagSourceField=DOCUMENT/  
DREREFERENCE&MinID=1&MaxID=3&ChecksumField=DOCUMENT/CHECKSUM&TaggedDBName=tagged&Re  
levanceField=DOCUMENT/CLUSTERSCORE
```

IDOL Server modifies the data:

```
#DREREFERENCE A  
#DREDBNAME Tagged  
#DREFIELD CHECKSUM="ABCD1234"  
#DREFIELD CLUSTERID="A"  
#DREFIELD CLUSTERSCORE="100.00"  
#DRECONTENT  
apple banana cheese  
#DREENDDOC
```

```
#DREREFERENCE B  
#DREDBNAME Tagged  
#DREFIELD CHECKSUM="ABCD1234"  
#DREFIELD CLUSTERID="A"  
#DREFIELD CLUSTERSCORE="100.00"  
#DRECONTENT  
apple banana cheese  
#DREENDDOC
```

```
#DREREFERENCE C  
#DREDBNAME Tagged  
#DREFIELD CHECKSUM="XYZ9876"  
#DREFIELD CLUSTERID="A"  
#DREFIELD CLUSTERSCORE="70.00"  
#DRECONTENT  
apple banana data  
#DREENDDOC
```

A is tagged as A because it does not match any existing clusters.

B is tagged as A because its CHECKSUM field matches A's.

C is tagged as A because it is similar to A and has a score higher than the specified MinScore (60).

Chapter 4: Fields

Both document content and metadata are stored in IDOL Server as fields. Retrieving data means retrieving the values of one or more fields. This section describes how to set up and use fields.

- [About Fields](#) 73
- [Configure a Field Process](#) 76
- [Update Field Configuration](#) 79
- [Index Fields](#) 83
- [Configure the Number Index Process](#) 85
- [NumericDateType Fields](#) 86
- [NumericType Fields](#) 87
- [FieldCheckType Fields](#) 88
- [ReferenceType Fields](#) 90
- [Highlight Fields](#) 93
- [BitFieldType Fields](#) 94
- [Metadata Fields](#) 96
- [Change Field Values](#) 97

About Fields

Data passes to IDOL Server (for example, from IDOL connectors) in the form of IDX or XML fields. IDOL Server stores all the fields that it receives so that you can search any field by using `FieldText` queries. To optimize IDOL Server performance, you can specify how it processes and stores the fields it receives.

You can associate fields with special properties in the IDOL Server configuration file. For example, you can instruct IDOL Server to treat these fields (or documents that contain them) in a specific way or read specific information from them.

You can associate a field with more than one property, as long as the properties do not clash.

You can assign the following properties to fields.

<code>ACLType</code>	Fields that hold access control lists (ACLs).
<code>AlwaysMatchType</code>	Fields that queries always match when they are present with a non-empty value.
<code>AutnRankType</code>	Fields that hold the document rank.
<code>BitFieldCompressed</code>	The index for <code>BitFieldType</code> fields is compressed.
<code>BitFieldMaxMemoryKB</code>	The maximum memory (in KB) to allocate for each associated <code>PropertyFieldCSVs</code> field that has the <code>BitFieldType</code> property.

BitFieldType	Fields that hold information on document sets. See "BitFieldType Fields" on page 94.
CountType	The number of occurrences of the associated fields are stored in a fast look-up table in memory to optimize matching of the fields when you use the MATCHCOVER and EQUALCOVER FieldText specifiers.
DatabaseType	Fields that hold the database that documents belongs to.
DateType	Fields that hold the date of documents.
DocumentTrackingType	Fields that hold the tracking IDs of documents.
ExpireAfterDelay	An offset, in hours, to add to the expiration date in the associated ExpireDateType field.
ExpireDateType	Fields that hold the expiration dates of documents.
FieldCheckType	A field that occurs in a large number of documents and holds a value that is frequently used to restrict query results.
FlattenIndexType	Fields that originate from hierarchically structured documents and whose content is stored as one level.
HiddenType	Fields whose content is hidden.
HighlightType	If fields contain terms that match a query, these terms are highlighted. See "Highlight Fields" on page 93.
Index	Fields that are stored as Index fields. See "Index Fields" on page 83.
IndexNumbers	Restrict the numbers to index for IndexNumbersType fields.
IndexNumbers1MaxLength	Restrict the length of pure numeric terms for IndexNumbersType fields.
IndexNumbers2MaxLength	Restrict the length of alphanumeric terms for IndexNumbersType fields.
IndexNumbersType	Fields to index as numeric or mixed-alphanumeric fields.
InvertedAgentType	Fields that are contained within inverted agents.
LangDetectType	Fields to use for automatic language detection when AutoDetectLanguagesAtIndex is set to True.
LanguageType	Fields that hold the language type of documents.
MatchType	Fields to store in a fast look-up table in memory to optimize matching of the fields when you use the following FieldText specifiers: BIASVAL, EQUALCOVER, MATCH, MATCHALL, MATCHCOVER, STRING, WILD.

MemCachedType	Fields to store in a memory cache.
NonReversibleType	Fields whose content is not line-reversed on index, even if the document is detected as being right-to-left Arabic or Hebrew.
NumericDateType	Fields that contain numeric dates and are stored in a fast look-up table in memory to optimize matching of the fields when you use <code>FieldText</code> specifiers. See "NumericDateType Fields" on page 86 .
NumericIntegerOnly	Fields with the <code>NumericType</code> property store signed 64-bit integer-only values, rather than doubles.
NumericNormalMaxMem	The maximum memory (in KB) to allocate for each associated <code>PropertyFieldCSVs</code> field that has one of these properties: <code>NumericDateType</code> , <code>NumericType</code> , <code>ParametricType</code> .
NumericType	Fields that contain numeric data and are stored in a fast look-up table in memory to optimize matching of the fields when you use <code>FieldText</code> specifiers. See "NumericType Fields" on page 87 .
OcrFilterType	Fields to evaluate by the OCR filter, and not to index or store if its quality is unsatisfactory (the field is stored, but is empty).
ParametricType	Fields that hold parametric values.
ParametricRangeType	Fields that contain numeric values to use to generate numeric ranges for parametric searches.
PrintType	Fields whose content is displayed with results when you set the query action <code>Print</code> parameter to <code>Fields</code> .
Ranges	The number of ranges to use for <code>ParametricRangeType</code> fields.
ReferenceMemoryMappedType	Fields that hold a value that is an existing value in a different <code>ReferenceType</code> field. This is then used in combination with the <code>FieldRecurse</code> action parameter and the field specifier <code>MATCHRECURSE</code> .
ReferenceType	Fields that hold document references. See "ReferenceType Fields" on page 90 .
SectionBreakType	Fields that hold the section number of documents that were split by the <code>Import</code> module.
SecurityType	The security type of documents that contain associated fields.
SortType	Fields to store for fast sorting when you use the <code>ARANGE FieldText</code> specifier or an alphabetical <code>Sort</code> option when querying IDOL Server.
SourceType	Fields to use to generate summaries and to suggest conceptually similar documents.
SynonymType	Fields that hold the name of the synonym job whose settings apply

	to documents that contain associated fields.
TextParseIndexType	Fields to treat as Index fields when sending a query using the TextParse and AgentBooleanField parameters.
TitleType	Fields that hold document titles.
TrimSpaces	Fields from which to remove multiple, leading, or trailing spaces before storing in IDOL Server.
Weight	The factor by which the weight of terms in associated fields is increased if they match query terms.

Note: You cannot configure a field with more than one numeric-based type concurrently. Numeric-based types include `NumericType`, `NumericDateType`, `MatchType`, `ParametricRangeType`, `ReferenceMemoryMappedType`, and `ParametricType` when the `ParametricNumericMapping` configuration parameter is set to `True`.

If your configuration file contains a field with a conflicting configuration, the server uses an internal precedence to set the field property. It flags the configuration conflict in the logs, and in configuration validation.

You can view information about the indexing fields that are defined for the documents in the data index on the **Field Types** page in the Monitor section of the IDOL Admin interface. For more information, refer to the *IDOL Admin User Guide*.

Related Topics

- ["Display Online Help" on page 30](#)
- ["Set up the Field Index Process" on page 38](#)
- ["Configure a Field Process" below.](#)

Configure a Field Process

The `[FieldProcessing]` section in the IDOL Server configuration file allows you to identify particular fields in documents. You can then apply any type of processing to them or the document that contains them during the indexing process, depending on the field value.

In this way you can apply multiple processes to documents without needing to set up a configuration section for each process combination.

Note: When identifying fields, use the following formats:

- `/FieldName` to match root-level fields.
- `*/FieldName` to match all fields except root-level.
- `/Path/FieldName` to match fields that the specified path points to.

Field names must not contain spaces, accents, or multibyte characters, and they must not start with a number. For IDX documents, IDOL Server converts these text elements to underscores (`_`) when it indexes the fields. You must also change any queries that reference these field names to use the modified field name.

To apply processes to fields or documents that contain specific fields

1. In the [FieldProcessing] section, list the processes to apply to fields. For example:

```
[FieldProcessing]
0=MyFirstProcess
1=IndexFields
2=MyCombinedProcess
3=IndexAndWeightHigher
```

2. Create a section for each process that you listed. In each section, declare a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields to associate with the processes.

You can use the PropertyMatch parameter to identify a specific value that fields must have to be processed. (This is useful if you set up a process that identifies security or language fields.)

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]
Property=MyFirstProperty
PropertyFieldCSVs=*/MyField,*/MySecondField
PropertyMatch=*myString*
```

```
[IndexFields]
Property=MySecondProperty
PropertyFieldCSVs=*/DRECONTENT,*/DRETITLE
```

```
[MyCombinedProcess]
Property=MyCombinedProperty
PropertyFieldCSVs=*/MyDateField,*/MyIndexField
```

```
[IndexAndWeightHigher]
Property=IndexHigherWeight
PropertyFieldCSVs=*/SUMMARIES
```

3. Create a section for each of the properties and specify appropriate configuration parameters for each. These configuration parameters define the processes that are applied to all the fields (or all documents that contain the fields) that you previously associated with the processes.

For example:

```
[MyFirstProperty]
HiddenType=True
```

```
[MySecondProperty]
Index=True
```

```
[MyCombinedProperty]
DateType=True
Index=True
```

```
[IndexHigherWeight]
Index=True
Weight=2
```

Related Topics

- ["Display Online Help" on page 30](#)

Example:

```
[FieldProcessing]
0=IndexFields
1=IndexAndWeightHigher
2=SectionBreakFields
3=DateFields
4=DatabaseFields
5=SetReferenceFields

[IndexFields]
// Controls which fields are indexed
Property=Index
PropertyFieldCSVs=*/DRECONTENT,*/DRETITLE

[IndexAndWeightHigher]
// Fields to index with a weight
Property=IndexWeight
PropertyFieldCSVs=*/SUMMARIES

[SectionBreakFields]
// Field containing document section number
Property=Section
PropertyFieldCSVs=*/DRESECTION

[DateFields]
// Fields containing the document date
Property=Date
PropertyFieldCSVs=*/DREDATE,*/harvest_time

[DatabaseFields]
// CSV of field names that define the document database
Property=Database
PropertyFieldCSVs=*/DREDBNAME

[SetReferenceFields]
// CSV of fields that define the document URL
Property=Reference
PropertyFieldCSVs=*/DREREFERENCE,*/DRETITLE

//-----Properties-----//

[Index]
```

Index=True

[IndexWeight]
Index=True
Weight=2

[Section]
SectionBreakType=True

[Date]
DateType=True

[Database]
DatabaseType=True

[Reference]
ReferenceType=True
TrimSpaces=True

Update Field Configuration

The field processes that you configure affect how IDOL Server processes data at index time. As a result, if you want to change the configuration after you have indexed data, there are often additional steps required to update your index.

There are three methods that you can use to update your field configuration:

- Manually update the configuration file, and restart the server. In this case, you can use the `Regenerate*Index` configuration parameters, where applicable, to automatically regenerate the modified indexes when you restart the server.
- Use the `DREREGENERATE` index action to modify the field configuration, and run any index regeneration at the same time.
- Use the IDOL Admin interface.

These methods are the same as the methods that you can use to regenerate an index after a validation failure. For more information, see ["Repair an Index After Validation Fails" on page 403](#).

You can make many field configuration changes using any of these methods. The `DREREGENERATE` index action allows you to make these configuration updates without restarting your server. However, for some changes you must reindex your content to make configuration changes. The following table describes how to update the field configuration for a particular property type.

Field type	Update method
ACLType	Requires reindex
AlwaysMatchType	Requires reindex
AutnRankType	Requires reindex

Field type	Update method
BitFieldType (and BitFieldCompressed, BitFieldMaxMemoryKB)	RegenerateBitFieldIndex or DREREGENERATE with Type=BitField. You can use IDOL Admin to update this field type.
CountType	RegenerateCountIndex or DREREGENERATE with Type=Count. You can use IDOL Admin to update this field type.
DatabaseType	Requires reindex
DateType	Requires reindex
DocumentTrackingType	Requires reindex
ExpireDateType (and ExpireAfterDelay)	Requires reindex
FieldCheckType	Requires reindex
FlattenIndexType	Requires reindex
HiddenType	No additional regeneration required
HighlightType	No additional regeneration required
Index (and Weight)	Requires reindex
IndexNumbers (and IndexNumbersMaxLength, IndexNumbersType)	Requires reindex
InvertedAgentType	Requires reindex
LangDetectType	Requires reindex
LanguageType	Requires reindex
MatchType	RegenerateMatchIndex or DREREGENERATE with Type=Match. You can use IDOL Admin to update this field type.
MemCachedType	You must restart the server to change the configuration
NonReversibleType	No additional regeneration required
NumericDateType	RegenerateNumericDateIndex or DREREGENERATE with Type=NumericDate. You can use IDOL Admin to update this field type.
NumericType (and NumericIntegerOnly, NumericNormalMaxMem)	RegenerateNumericIndex or DREREGENERATE with Type=Numeric. You can use IDOL Admin to update this field type.

Field type	Update method
OcrFilterType	Requires reindex
ParametricType	RegenerateParametricIndex or DREREGENERATE with Type=Parametric. You can use IDOL Admin to update this field type.
ParametricRangeType (and Ranges)	RegenerateParametricIndex or DREREGENERATE with Type=Parametric
PrintType	No additional regeneration required
ReferenceMemoryMappedType	Requires reindex
ReferenceType	Requires reindex
SectionBreakType	Requires reindex
SecurityType	Requires reindex. You can use IDOL Admin to update this field type.
SortType	RegenerateSortIndex or DREREGENERATE with Type=Sort. You can use IDOL Admin to update this field type.
SourceType	Requires reindex
SynonymType	Requires reindex
TextParseIndexType	Requires reindex
TitleType	Requires reindex
TrimSpaces	Requires reindex

Note: If you attempt to use the DREREGENERATE index action to change a property that requires reindexing, the DREREGENERATE index action returns an error response and does not make the change.

When you use the DREREGENERATE index action to update the field configuration, you can set the Type parameter to **Auto** to automatically regenerate the indexes for the fields that you have changed. You can also set Type to **None** if you do not want to regenerate the indexes immediately, for example so that you can make a series of field configuration changes and then regenerate the indexes in an additional index action.

Related Topics

- ["Repair an Index After Validation Fails" on page 403](#)

Update Fields in the Configuration File

Use the following procedure to update fields in the configuration file.

Note: You must use this method for changes where you must reindex content for the changes to take effect.

To update field configurations in the configuration file

1. Open the IDOL Server configuration file in a text editor.
2. Find the field configuration section that you want to modify.
3. Modify any of the configuration parameters that you want to change. For details of the configuration parameters, refer to the *IDOL Server Reference*.
4. (Optional) If you are modifying a field type that can be regenerated, find the [Server] section, and set the appropriate `Regenerate*Index` configuration parameter to `True`. For example, to update the configuration for `MatchType` fields, set the `RegenerateMatchIndex` parameter to `True`.

You can alternatively skip this step and run a `DREREGENERATE` index action after you restart the server.

5. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
6. Update your content:
 - If you have modified a field type where the change requires you to reindex your data, reindex your data.
 - If you need to regenerate the index, run a `DREREGENERATE` index action with `Type` set to the appropriate index. For example:

```
http://idolhost:9001/DREREGENERATE?Type=Match
```

For more information, refer to the *IDOL Server Reference*.

7. If you used the `Regenerate*Index` configuration parameters, set the parameters to `False` again in the configuration file. This step means that the server does not waste time by regenerating the index every time you restart the server.

Update Field Configuration with an Index Action

Use the following procedure to make field configuration changes for fields where you can regenerate the content. This method allows you to update the field configuration without restarting the server, which avoids downtime.

Note: You can update the lists of fields associated with a particular property, but if you want to add a new field process or add an additional property to an existing process, you must update the configuration file manually.

To update field configurations with an index action

1. Open the IDOL Server configuration file in a text editor.
2. Find the field configuration section that you want to modify.
3. Send a `DREREGENERATE` index action to IDOL Server, with the `FieldProcessingSection` parameter set to the name of the configuration section that you want to modify. Set any of the following parameters to update the lists of fields:

- `AugmentFieldCSVs`. A list of fields that you want to add to the `PropertyFieldCSVs` parameter.
- `DiminishFieldCSVs`. A list of fields that you want to remove from the `PropertyFieldCSVs` parameter.
- `AugmentNegativeFieldCSVs`. A list of fields that you want to add to the `PropertyNegativeFieldCSVs` parameter.
- `DiminishNegativeFieldCSVs`. A list of fields that you want to remove from the `PropertyNegativeFieldCSVs` parameter.
- `Type`. Set this parameter to `None` if you do not want to regenerate the index immediately, for example if you want to modify several field configuration sections and then run a single regenerate operation. The default value is `Auto`, which automatically regenerates the index for the field configuration that you modify.

Note: If you set `Type` to `None`, you must run a `DREREGENERATE` index action manually to regenerate the indexes that you have modified. The `DREREGENERATE` index action does not automatically check all field configurations that might need regeneration.

For example:

```
DREREGENERATE?FieldProcessingSection=SetMatchFields&AugmentFieldCSVs=*/NewMatchField,*/*SpecialMatch&DiminishFieldCSVs=*/ExistingMatchField
```

This example updates the `[SetMatchFields]` configuration section, adding `NewMatchField` and `SpecialMatch` to the `PropertyFieldCSVs`, and removing `ExistingMatchField`. The `DREREGENERATE` index action automatically regenerates the `Match` index to make the configuration changes available.

Update Field Configuration with IDOL Admin

Use the following procedure to update your field configuration by using the IDOL Admin interface.

To update your field configuration by using IDOL Admin

1. In the **Service Control** tab in the Console page, click **Regenerate**.
The Regenerate dialog box opens.
2. In the **Type** list, click the field type that you want to update.
3. Select a priority for the index action. This determines how IDOL Server queues the action.
4. Click **Regenerate**.

You can monitor the progress of the `DREREGENERATE` action in the Recent Tasks panel.

Index Fields

Store fields that contain text which you want to query frequently as Index fields. Index fields are processed linguistically when they are stored in IDOL Server. This means that stemming and stop word lists are applied to text in Index fields before they are stored, 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.

Do not use the Index field type for fields that contain:

- URLs or content that you are unlikely to use.
- content that you query frequently, but whose values you query only in their entirety. It is more efficient to use field specifiers (for example, MATCH) to query these values.
- numeric values or dates as Index fields. Instead, store these fields as numeric fields and numeric date type fields.

Related Topics

- ["NumericType Fields" on page 87](#)
- ["NumericDateType Fields" on page 86](#)

To set up Index fields

1. Open the IDOL Server configuration file in a text editor.
2. List an indexing process in the [FieldProcessing] section. For example:

```
[FieldProcessing]
0=MyFirstProcess
1=MySecondProcess
2=IndexingFields
```

3. Create a section for the indexing process, and in each section, create a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

You can use the PropertyMatch parameter to identify a specific value that fields must have to be processed.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]
Property=MyFirstProperty
PropertyFieldCSVs=*/MyField,*/MySecondField
PropertyMatch=*myString*
```

```
[MySecondProcess]
Property=MySecondProperty
PropertyFieldCSVs=*/MyOtherField,*/MyOtherSecondField
```

```
[IndexingFields]
Property=IndexFields
PropertyFieldCSVs=*/DRECONTENT,*/DRETITLE
```

4. Create a section for your indexing property in which you set the Index parameter to True. For example:

```
[MyFirstProperty]
HiddenType=True
```

```
[MySecondProperty]  
Index=True
```

```
[IndexFields]  
Index=True
```

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

Configure the Number Index Process

You can configure IDOL Server to index numeric and alphanumeric fields in several ways by using the configuration parameter `IndexNumbers`. Set `IndexNumbers` to one of the following values to specify how IDOL Server treats numbers:

0	Numbers are not indexed.
1	All numbers are indexed (irrespective of whether they appear on their own or as part of a word).
2	Numbers are indexed only if they are part of a word (for example DRE4, Y2K and so on).

To restrict this to a narrower set of data, use the field processing property `IndexNumbersType`. Create an `IndexNumbersFields` section and specify which fields qualify. You can limit only terms that are indexed within the field property.

For example:

```
[English]  
IndexNumbers=2
```

```
[IndexNumbersFields]  
PropertyFieldCSVs=*/MYFIELD  
Property=IndexNumbers
```

```
[IndexNumbers]  
IndexNumbersType=True  
IndexNumbers=2
```

This means that IDOL indexes the numeric terms in `*/MYFIELD` that satisfy `IndexNumbers=2` (non-numeric and mixed-alphanumeric), whereas all other fields index with `IndexNumbers=1` (all numeric terms).

Note: If the `IndexNumbers` configuration parameter is not specified in a property section, its default is 0.

You can also limit the indexing of numeric or mixed-alphanumeric terms by the length of the term.

For example:

```
[IndexNumbers1]  
IndexNumbersType=True  
IndexNumbers=1
```

```
IndexNumbers1MaxLength=5  
IndexNumbers2MaxLength=6
```

This means that fields with this property have all numbers indexed, assuming the language has `IndexNumbers=1` configured, except for pure numeric terms longer than five characters, which are not indexed. Alphanumeric terms longer than six characters are also not indexed.

Note: You cannot set the length to more than 255.

NumericDateType Fields

You can configure IDOL Server to identify fields that contain dates. When these fields are indexed, IDOL Server stores them in a fast look-up table in memory, so that it can quickly return the fields.

Note: You cannot configure a field with two numeric-based types concurrently. Numeric-based types include `NumericType`, `NumericDateType`, `MatchType`, `ParametricRangeType`, `ReferenceMemoryMappedType`, and `ParametricType` when the `ParametricNumericMapping` configuration parameter is set to `True`.

IDOL Server converts dates to numerical values (epoch seconds) and identifies the fields that contain the numerical date values.

To set up memory mapping for NumericDateType fields

1. Open the IDOL Server configuration file in a text editor.
2. List a process that identifies numeric date fields in the `[FieldProcessing]` section. For example:

```
[FieldProcessing]  
0=MyFirstProcess  
1=NumericDateFields
```

3. Create a section for each process that you listed, and in each section, create a property for the process (you define the property by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]  
Property=MyProperty  
PropertyFieldCSVs=*/MyField,*/MyOtherField
```

```
[NumericDateFields]  
Property=NumDate  
PropertyFieldCSVs=*/BIRTHDAY,*/STARTDATE
```

4. Create a section for the property in which you set the `NumericDateType` parameter to `True`. This enables IDOL Server to memory map the associated `PropertyFieldCSVs` fields, and identify them as fields that contain date values. For example:

```
[NumDate]  
NumericDateType=True
```

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

If you now send a query for a specific value that is stored in the BIRTHDAY field, IDOL Server memory maps the range that this value is in, so that it can return results more quickly next time a value that lies in this range is queried.

For example:

```
http://12.3.4.56:4000/action=Query&FieldText=RANGE{01/01/1980,31/12/1980}:BIRTHDAY
```

The BIRTHDAY field must contain a numeric date value that is between 01/01/1980 and 31/12/1980 for this document to be returned.

NumericType Fields

You can configure IDOL Server to identify fields that contain numerical values. When these fields are indexed, IDOL Server stores them in a fast look-up table in memory, so that it can quickly return the field. A numeric field can contain a comma-separated list of numbers. IDOL Server stores each value as a numeric value for this field, for this document.

Note: You cannot configure a field with two numeric-based types concurrently. Numeric-based types include `NumericType`, `NumericDateType`, `MatchType`, `ParametricRangeType`, `ReferenceMemoryMappedType`, and `ParametricType` when the `ParametricNumericMapping` configuration parameter is set to `True`.

To set up `NumericType` fields to speed numeric queries

1. Open the IDOL Server configuration file in a text editor.
2. List a process that identifies numeric fields in the `[FieldProcessing]` section. For example:

```
[FieldProcessing]  
0=MyFirstProcess  
1=PriceFields
```

3. Create a section for each process that you listed, and in each section, declare a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]  
Property=MyProperty  
PropertyFieldCSVs=*/MyField,*/MyOtherField
```

```
[PriceFields]  
Property=Price  
PropertyFieldCSVs=*/PRICE
```

4. Create a section for the property in which you set the `NumericType` parameter to `True`. This enables IDOL Server to memory map the associated `PropertyFieldCSVs` fields. For example:

```
[Price]
NumericType=True
```

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

If you now send a query for a specific value that is stored in the `PRICE` field, IDOL Server memory maps the range that this value is in, so that it can return results more quickly next time a value that lies in this range is queried.

Examples:

```
http://12.3.4.56:4000/action=Query&FieldText=NRANGE{50,100}:PRICE
```

The `PRICE` field must contain a number between 50 and 100 (including decimal numbers) for this document to return.

```
http://12.3.4.56:4000/action=Query&Text=computer&Sort=PRICE:numberincreasing
```

IDOL Server sorts the results that it returns for the query according to the values that their `PRICE` fields contain. The results whose `PRICE` field contains the smallest value is listed first, followed by results with increasing values in the `PRICE` field.

FieldCheckType Fields

You can configure IDOL Server to identify a field contained in a large number of documents whose entire value is frequently used to restrict results (for example, a field that stores category names). When this field is indexed, IDOL Server stores it in a fast look-up table in memory, so that it can quickly return the field.

Note: If you set `URLAnalysis` to `True` in the `[Server]` section of the IDOL Server configuration file, you cannot identify a field as a `FieldCheckType` field, because IDOL Server automatically uses the domain it finds in `ReferenceType` fields as the `FieldCheck` value.

To set up `FieldCheckType` fields

1. Open the IDOL Server configuration file in a text editor.
2. List a process that identifies numeric fields in the `[FieldProcessing]` section. For example:

```
[FieldProcessing]
0=MyFirstProcess
1=FieldCheckTypeIdentification
```

3. Create a section for each process that you listed, and in each section, create a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]
Property=MyProperty
PropertyFieldCSVs=*/MyField,*/MyOtherField
```

```
[FieldCheckTypeIdentification]
Property=FieldCheck
PropertyFieldCSVs=*/CATEGORY
```

4. Create a section for the property in which you set the `FieldCheckType` parameter to `True`. This enables IDOL Server to memory map the associated `PropertyFieldCSVs` fields. For example:

```
[FieldCheck]
FieldCheckType=True
```

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

When you now use a `Query`, `Suggest`, or `SuggestOnText` action to query for results, you can:

- Use the `Combine` action parameter to restrict the result output to the most relevant result for each available `FieldCheckType` field value (by setting it to `FieldCheck`).
- Use the `FieldCheck` action parameter to restrict the result output to documents whose `FieldCheckType` field matches a specific value (this is also available for the `GetQueryTagValues` action).

Combine Parameter Example

In this example, IDOL Server is configured to store the `Category` field as a `FieldCheckType` field.

The following query is sent to IDOL Server.

```
http://12.3.4.56:4000/action=Query&Text=The best thing to do in your spare
time&Combine=FieldCheck
```

If IDOL Server contains 50 documents that match the query text, of which eight contain a `Category` field with the value `Sport`, five contain a `Category` field with the value `Gardening`, and one contains a `Category` field with the value `Cooking`, the above query returns only three results:

- The most relevant of the documents whose `Category` contains the value `Sport`.
- The most relevant of the documents whose `Category` contains the value `Gardening`.
- The document whose `Category` contains the value `Cooking`.

FieldCheck Parameter Example

In this example, IDOL Server is configured to store the `Color` field as a `FieldCheckType` field.

The following query is sent to IDOL Server.

```
http://12.3.4.56:4000/action=Query&Text=A fast sports car&FieldCheck=Red
```

This query returns only results whose content matches the specified `Text` and whose `FieldCheckType` field has the value `Red`.

ReferenceType Fields

ReferenceType fields are used to identify documents. Before you index a document into IDOL Server, you have to set up a field process that determines which of the fields in a document are used as its ReferenceType field (note that a document can have multiple ReferenceType fields).

At index time, you can use ReferenceType fields to eliminate duplicate copies of documents. At query time, you can use ReferenceType fields to filter results (for example, by using the Combine action parameter or by specifying references that results must or must not match).

Note that if you want to eliminate duplicate document copies and use the Combine action parameter, you must set up separate ReferenceType fields for these processes.

Related Topics

- ["Prevent Duplicate Documents" on page 58](#)
- ["Combine Parameter" on page 325](#)
- ["Use KillDuplicates and Combine on ReferenceType Fields" on the next page](#)

Set up ReferenceType Fields

You must set up a field process to identify ReferenceType fields before you start indexing documents into IDOL Server.

To set up ReferenceType fields

1. Open the IDOL Server configuration file in a text editor.
2. In the [FieldProcessing] section, add a process that identifies ReferenceType fields. For example:

```
[FieldProcessing]
0=MyFirstProcess
1=MySecondProcess
2=SetReferenceFields
```

3. Create a section for the process that you added, and in each section create a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]
Property=MyFirstProperty
PropertyFieldCSVs=*/MyField,*/MySecondField

[MySecondProcess]
Property=MySecondProperty
PropertyFieldCSVs=*/MyThirdField
```

```
[SetReferenceFields]
Property=Reference
PropertyFieldCSVs=*/DRREFERENCE,*/URL
```

4. Create a section for each of the properties and specify appropriate configuration settings for each. These configuration parameters define the processes that are applied to all the fields (or all documents that contain the fields) that you previously associated with the processes. For example:

```
[MyFirstProperty]
HiddenType=True
```

```
[MySecondProperty]
Index=True
```

```
[Reference]
ReferenceType=True
TrimSpaces=True
```

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

You can now index documents into IDOL Server.

Note: If you do not set up a field process that identifies `ReferenceType` fields, IDOL Server automatically allocates a unique number to each document that is indexed. IDOL Server uses this number as the reference for the document.

Use KillDuplicates and Combine on ReferenceType Fields

When you instruct IDOL Server to eliminate duplicate document copies at index time using a specific `ReferenceType` field (by setting the `KillDuplicates` parameter in the IDOL Server configuration file), it automatically uses any field listed for `PropertyFieldCSVs` alongside this `ReferenceType` field in the IDOL Server configuration to eliminate duplicate document copies as well.

However, IDOL Server cannot use the same field for deduplication as for the `Combine` action parameter, because the `Combine` operation clashes (carried out at query time) with IDOL Server eliminating duplicate fields. This clash means that, if you want to eliminate duplicate document copies and use the `Combine` action parameter, you must set up separate `ReferenceType` fields for these processes.

Related Topics

- ["Prevent Duplicate Documents" on page 58](#)

To use KillDuplicates and Combine on ReferenceType fields

1. Open the IDOL Server configuration file in a text editor.
2. In the `[FieldProcessing]` section, add two processes that identify `ReferenceType` fields (note that you must set up a field process to identify `ReferenceType` fields before you start indexing documents into IDOL Server). One of them is used to eliminate duplicate copies of documents, and the other one is used for the `Combine` operation.

For example:

```
[FieldProcessing]
0=MyFirstProcess
1=MySecondProcess
2=SetUpReferenceFields
3=SetUpMoreReferenceFields
```

3. Create a section for the processes that you added, and in each section, create a property for the respective process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with each process.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]
Property=MyFirstProperty
PropertyFieldCSVs=*/MyField,*/MySecondField
```

```
[MySecondProcess]
Property=MySecondProperty
PropertyFieldCSVs=*/MyThirdField
```

```
[SetUpReferenceFields]
Property=ReferenceFields
PropertyFieldCSVs=*/DRREFERENCE,*/URL
```

```
[SetUpMoreReferenceFields]
Property=MoreReferenceFields
PropertyFieldCSVs=*/DRETITLE
```

4. Create a section for each of the properties and specify appropriate configuration settings for each. These configuration parameters define the processes that are applied to all the fields (or all documents that contain the fields) that you previously associated with the processes. For example:

```
[MyFirstProperty]
HiddenType=True
```

```
[MySecondProperty]
Index=True
```

```
[ReferenceFields]
ReferenceType=True
TrimSpaces=True
```

```
[MoreReferenceFields]
ReferenceType=True
TrimSpaces=True
```

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

After you index documents into IDOL Server, you can use, for example, the `*/DREREFERENCE` field to eliminate duplicate copies of documents. (IDOL Server then automatically also uses the `*/URL` field for deduplication because it is listed alongside `*/DREREFERENCE` for `PropertyFieldCSVs`.) This leaves you free to use the `*/DRETITLE` field for the `Combine` operation.

Highlight Fields

When you run a `Query`, `Suggest`, or `SuggestOnText` action, you can highlight sentences or words in the results that are related to the terms in the query (or the terms in the text or document that you are suggesting on).

IDOL Server checks which fields highlighting applies to and then highlights all sentences or words that are based on the terms in the results that it returns.

To set up highlight fields

1. Open the IDOL Server configuration file in a text editor.
2. List a highlighting process in the `[FieldProcessing]` section. For example:

```
[FieldProcessing]
0=MyFirstProcess
1=HighlightFields
```

3. Create a section for each process that you listed, and in each section, create a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]
Property=MyProperty
PropertyFieldCSVs=*/MyField,*/MyOtherField
```

```
[HighlightFields]
Property=Highlight
PropertyFieldCSVs=*/DRETITLE,*/DRECONTENT
```

4. Create a section for the property in which you set the `HighlightingType` parameter to `True`. This enables the highlighting of all matched terms that are contained in the associated `PropertyFieldCSVs` fields. For example:

```
[Highlight]
HighlightType=True
```

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

Note: In a stand-alone configuration where you run the `View` server separately, you must specify the `Content` host and port in the `[Server]` section of the `View` configuration file.

BitFieldType Fields

If you have documents that can be part of several different document sets, you can use `BitFieldType` fields to efficiently store information on which sets the documents belong to.

The value in a `BitFieldType` field is a hexadecimal number, which in turn represents a binary number. The binary number is a representation of the sets that a document belongs to, with each binary digit representing a particular set of documents. If a document is part of a set, the bit corresponding to that set is a 1. If a document is not part of that set, the bit is a 0.

For example, if a document is present in sets 0, 5, 9, 11, 12, and 13, it has the following binary representation:

```
11101000100001
```

where the digit at the furthest right position represents set 0, the digit to the left of set 0 represents set 1 and so on. Set numbers increase from right to left.

This number is the binary representation of the decimal number 14881, and the hexadecimal number 3A21. Therefore, the `BitField` contains the value 3A21 to indicate that the document is part of these sets:

```
#DREFIELD BitField="003A21"
```

In this way, information on sets can be stored in a single field per document, for an arbitrarily large number of sets.

To set up `BitFieldType` fields

1. Open the IDOL Server configuration file in a text editor.
2. List a Bit Field process in the `[FieldProcessing]` section. For example:

```
[FieldProcessing]
0=MyFirstProcess
1=BitFields
```

3. Create a section for each process that you listed, and in each section, create a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[MyFirstProcess]
Property=MyProperty
PropertyFieldCSVs=*/MyField,*/MyOtherField

[BitFields]
Property=BitFieldSetFields
PropertyFieldCSVs=*/WORKBOOK,*/BITFIELD
```

4. Create a section for the property in which you set the `BitFieldType` parameter to `True`. This enables IDOL Server to store the contents of the `PropertyFieldCSVs` fields as bit fields. For example:

```
[BitFieldSetFields]
BitFieldType=True
```

5. To compress the `BitField` index, set `BitFieldCompressed` to `True` in the property section. For example:

```
[BitFieldSetFields]
BitFieldType=True
BitFieldCompressed=True
```

6. Set the `BitFieldMaxMemoryKB` parameter to the maximum memory (in KB) that can be used by `BitFieldType` fields. If this is zero (the default) there is no limit to the memory.

```
[BitFieldSetFields]
BitFieldType=True
BitFieldMaxMemoryKB=True
```

7. If you want to define `BitFieldType` fields or add extra `BitFieldType` fields, but have already indexed content into IDOL, you can set `RegenerateBitFieldIndex` to `True` in the `[Server]` section. This allows IDOL Server to generate the files it requires to internally identify `BitFieldType` fields on startup, so that you need only to restart IDOL Server to be able to use `BitFieldType` fields, rather than having to reindex all your data.

```
[Server]
(...)
RegenerateBitFieldIndex=True
```

You can also use the `DREREGENERATE` index action to regenerate the `BitFieldType` index while the server is running.

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

Note: Each document that you store in IDOL Server must contain only one instance of any particular `BitFieldType` field.

Edit Set Information after Indexing

After a document is indexed into IDOL, you can edit the information in `BitFieldType` fields by using the `DREREPPLACE` action with the `#DREFIELDBITOR`, `#DREFIELDBITXOR`, and `#DREFIELDBITAND` operators.

Related Topics

- ["Change Document Field Values" on page 391](#)

Find Documents within a Set

You can query IDOL Server using `FieldText` with the `BITSET` field specifier to find only documents that are part of a particular set. In this case, you use a decimal value to identify the set number. For

example:

```
http://localhost:9000/action=Query&FieldText=BITSET{4,18}:BitField
```

This query returns documents that are part of set 4 or set 18 for sets defined by the `BitField` field.

Related Topics

- ["BITSET" on page 237](#)

Metadata Fields

Metadata fields are fields that IDOL Server creates for documents at index time to display information about the documents when they are returned as results for a query. Some document metadata fields are always displayed when IDOL Server returns a document as a query result. You can display all document metadata fields by adding `XMLMeta=True` to your query.

IDOL Server displays the following metadata fields for results:

- **<autn:baseid>**. If the document has multiple sections, this is the ID of the first section of the document. If the document is not sectioned, this value is the same as the document ID.
- **<autn:content>**. The text content of the document.
- **<autn:database>**. The IDOL Server database in which the document is stored.
- **<autn:date>**. The date (in epoch seconds) when the document was created. This date is read from the field that has been identified by the `DateType` parameter in the IDOL Server configuration file. If no field has been identified, the date the document was indexed is used instead.
- **<autn:expiredate>**. The date (in epoch seconds) when the document expires. This date is read from the field that has been identified by the `ExpireDateType` parameter in the IDOL Server configuration file. If you have set an offset in the `ExpireAfterDelay` parameter, the `<autn:expiredate>` field includes this offset to calculate the expiration date. When a document expires, it is deleted from IDOL Server or moved to a different database (depending on what you set `ExpireIntoDatabase` to in the IDOL Server configuration file).
- **<autn:id>**. The document ID. This ID is assigned to the document at index time. If IDOL Server is compacted, the IDs of documents change.
- **<autn:language>**, **<autn:languageencoding>**, **<autn:language-type>**. The language, encoding, and language type associated with the document. The language type is read from the field that you identified by the `LanguageType` property in the IDOL Server configuration file. The language and encoding of the document are read from the `Language` and `Encoding` parameters set for this language type in the configuration file.

If no field from which the language type can be read has been identified, the `DefaultLanguageType` that you set in the configuration file is used instead, unless automatic language detection is enabled, or the document has been submitted to IDOL Server with an index action that sets a specific language type for the document.
- **<autn:links>**. A list of stemmed terms that are contained both in the query and in the result document.
- **<autn:reference>**. The document reference. This is read from the field that has been identified by the `ReferenceType` parameter in the IDOL Server configuration file. If no field has been identified, IDOL Server automatically generates a reference for the document at index time.

- **<autn:section>**. The number of sections the document has been split up into at index time. The first section is section 0.
- **<autn:title>**. The document title. This is read from the field that has been identified by the `TitleType` parameter in the IDOL Server configuration file. If no field has been identified, the document is not given a title.
- **<autn:weight>**. The percentage relevance that the document has to the query.

When you set `XMLMeta` to `True`, IDOL Server also returns the following fields for your results set. These metadata fields are used by the DAH for sorting.

- **<autn:numericdatesort>**. The index (starting from zero) of the elements of your defined sort order that refer to a `NumericDateType` field.
- **<autn:numeric sort>**. The index (starting from zero) of the elements of your defined sort order that refer to a `NumericType` field.

Change Field Values

You can use the `DREREPLACE` index action to change the values of fields or add fields to a document after you index content into IDOL Server.

Related Topics

- ["Change Document Field Values" on page 391](#)

Chapter 5: Language Support

This section describes how IDOL Server supports processing in many different languages and how you configure that support.

- IDOL Language Support Concepts 99
- Run IDOL Server in Multiple Languages 101
- Determine the Languages that are Enabled 102
- Define Language Types 103
- Associate Language Types with Documents 104
- Add LanguageType Fields to Documents 107
- Define a Default Language Type 108
- Define a General Language 108
- Enable Automatic Language Detection 109
- Specify the Language Type of a Query 110
- Convert Results to a Specific Encoding 110
- Return Documents in Multiple Languages 111
- Return Documents in a Specific Language 113
- Create a Custom Stem File for a Language 113
- Decompose Compound Words 114
- Enable Transliteration 115

IDOL Language Support Concepts

IDOL Server uses probabilistic modeling and therefore does not require any form of language-dependent parsing, dictionaries, or translation modules.

Treating words as abstract symbols of meaning allows IDOL technology to derive understanding through the context in which symbols occur rather than a rigid definition of grammar. Slang and other variations in language do not affect the software analysis.

IDOL Server can build up a statistical understanding of the patterns in any language. The more information IDOL Server has about a particular type of information (for example, legal terms, pharmaceutical developments, technology, and so on), the more understanding it gains of those topics.

You can think of a new language as simply another type of information, for which IDOL Server needs enough material to learn from. Therefore, it is possible to mix more than one language in IDOL Server as long as you have sufficient amounts of each language to build its understanding.

The choice of language does not compromise the accuracy of the concepts extracted by IDOL Server. The underlying algorithm is the same regardless of the language used.

IDOL internationalization functionality enables:

- **automatic language detection.** IDOL Server can automatically detect the language and encoding of documents that it processes. This feature allows you to set up processes that IDOL Server automatically applies to documents or document metadata if they are in a specific language. For example, if IDOL Server identifies a document as Chinese, it automatically applies the appropriate preliminary linguistic tools.

Note: If a document contains multiple languages, IDOL Server determines which language it contains most, and processes the document according to the settings for this language.

- **cross-lingual systems.** You can set up cross-lingual systems in IDOL Server. This feature allows you to produce multilingual results for queries, or to restrict results to documents in a specific language or encoding. For example, an English query can return information both in English and Spanish.

Although IDOL technology is language independent, it can be beneficial to use language-dependent features to optimize the ability of IDOL Server to match concepts irrespective of their appearance in text. IDOL therefore provides the following features:

- **stop word lists.** Every language has words that do not carry much significant meaning. In grammatical terms these are normally prepositions, conjunctions, auxiliary verbs, and so on (for example, words such as *the*, *a*, and *to* in English). These words can be safely ignored when processing content.

IDOL provides as standard a set of stop word lists for the most commonly used languages.

- **stemming.** In languages, some words have a common morphological root. IDOL provides stemming algorithms that reduce words to this form. This process allows you to match concepts regardless of the grammatical use of words. In English for example, the words *help*, *helpful*, *helping*, and *helped* can all be stripped to their stem *help* without significant loss of meaning.

IDOL provides as standard a set of stemming algorithms for the most commonly used languages. IDOL applies stemming after it discards stop words, both at index time (when content is stored in IDOL Server) and at query time (IDOL removes stop words and stems query text before matching).

Note: IDOL Server also supports per-language use of a stemming file, which you can use in conjunction with the stemming algorithms to specify stems for individual words.

- **multiple encodings.** IDOL supports multiple encodings for languages such as Greek and Russian. You can use different encodings interchangeably, which means that it does not matter which encoding a language is given in. For example, it is possible to query in one recognized encoding for a language and receive results that are in other encodings.
- **transliteration schemes.** Transliteration is the ability to represent letters that do not belong to the Latin alphabet or words that contain accented letters with the corresponding characters of another alphabet. This makes familiarity with the accents and special characters of different languages unnecessary.
- **canonicalization of characters.** Some encodings have more than one way to represent a character. For example, the Japanese katakana script can have full width or half width characters. Regardless of its width the character in itself carries the same meaning.

The IDOL software infrastructure uses canonicalization to ensure that it treats all character forms equally. It automatically converts to an internationally recognized canonical form.

Related Topics

- ["Create a Custom Stem File for a Language" on page 113](#)

- ["Enable Transliteration" on page 115](#)

Run IDOL Server in Multiple Languages

You can combine multiple languages in one IDOL Server. Use the outline below to determine what you have to do.

To combine multiple languages in one IDOL Server

1. Before you index your documents, ensure that the IDOL Server configuration file contains the languages that you want to use. See ["Determine the Languages that are Enabled" on the next page](#).
2. If the configuration file does not contain all the languages that you want to use, add the missing languages. Set up a field process that enables IDOL Server to associate these languages with documents. See ["Define Language Types" on page 103](#) and ["Associate Language Types with Documents" on page 104](#).
3. Check the documents that you want to index into IDOL Server:
 - IDOL Server can read the language type (that is, language and encoding) from a document field. If some of your documents do not contain these fields, IDOL Server applies the default language type. See ["Add LanguageType Fields to Documents" on page 107](#) and ["Define a Default Language Type" on page 108](#).
If you do not want to associate the default language type with your documents, enable automatic language detection. See ["Enable Automatic Language Detection" on page 109](#).
Alternatively, you can manually index your documents into IDOL Server, adding the language type of the documents to each index action. In this case, you must index documents in batches, where each batch must have the same language type. See ["Index Data" on page 43](#).
 - IDOL Server automatically processes documents that contain fields that specify the language type. You must add any missing languages to the IDOL Server configuration file.

Related Topics

- ["Languages and Language Files" on page 461](#)

When you query IDOL Server, by default IDOL Server returns only documents that have the same language as the language type (that is, language and encoding) of the query. You can change this behavior in the following ways:

- To use a language type in the query text that is not the default language type, add the `LanguageType` parameter to your query.
- To return results in a specific encoding, add the `OutputEncoding` parameter to your query. You can return only encodings that are compatible with the query language.
- To return documents in multiple languages, add the `AnyLanguage` parameter to your query.
- To return documents in a specific language, add the `AnyLanguage` and `MatchLanguage` parameters to your query.

Related Topics

- ["Specify the Language Type of a Query" on page 110](#)

- ["Convert Results to a Specific Encoding" on page 110](#)
- ["Return Documents in Multiple Languages" on page 111](#)
- ["Return Documents in a Specific Language" on page 113](#)

Determine the Languages that are Enabled

You can determine the languages that IDOL Server can process by looking at the IDOL Server configuration file.

To check which languages are enabled in IDOL Server

1. Open the IDOL Server configuration file in a text editor.
2. Find the [LanguageTypes] section.

This section lists the languages that IDOL Server can process. For example:

```
[LanguageTypes]
DefaultLanguageType=englishUTF8
DefaultEncoding=UTF8
LanguageDirectory=C:\IDOLServer\IDOL\langfiles
0=Afrikaans
1=Albanian
2=Arabic
3=Armenian
4=Azeri
5=Basque
6=general
```

- The `DefaultLanguageType` parameter specifies the language type to apply when:
 - IDOL Server cannot read the language type nor encoding of a document from a specified field.
 - the action does not include a `LanguageType` parameter.
 - automatic language detection is not enabled.
- The `general` language category is for documents whose encoding is identified, but whose language is not.
- The `LanguageDirectory` parameter specifies the directory that contains resource files (such as stop word lists) that IDOL Server uses to process languages.

Tip: You can also view information on licensed and configured languages and encodings on the **Languages** tab on the Status page in IDOL Admin. For more information, refer to the *IDOL Admin User Guide*.

Related Topics

- ["Define a Default Language Type" on page 108](#)
- ["Define a General Language" on page 108](#)

Define Language Types

To run IDOL Server in multiple languages, specify the language types that you want IDOL Server to process. A language type is a combination of the language and encoding.

Note: You must specify languages and language types before you index data into IDOL Server.

To specify language types

1. Open the IDOL Server configuration file in a text editor.
2. Find the [LanguageTypes] section and list the languages that you want IDOL Server to process. You must use UTF-8 characters when specifying a language.

For example:

```
[LanguageTypes]
0=English
1=Afrikaans
2=Albanian
3=Arabic
4=Armenian
5=Azeri
```

3. In the [LanguageTypes] section, set any configuration parameters that you want to apply to all languages. For details of the configuration parameters you can use, refer to the *IDOL Server Reference*.

Note: As well as the general language configuration parameters, you can set any of the individual language configuration parameters in the [LanguageTypes] section. The value in this section sets the default value for all languages, which you can override in the individual language configuration sections.

For example:

```
[LanguageTypes]
DefaultLanguageType=englishUTF8
DefaultEncoding=UTF8
LanguageDirectory=C:\IDOLserver\IDOL\langfiles
GenericTransliteration=True
StopWordIndex=1
ProperNames=3
TangibleCharacters=!?
```

4. For each language that you use, create a section using the name of the language.
In this section, specify appropriate settings that determine how IDOL Server handles this language. For details on the configuration parameters you can use, refer to the *IDOL Server Reference*.
5. For each section, add the Encodings parameter and define the encodings and corresponding language types used by the language.

For example:

```
[english]
Encodings=UTF8:englishUTF8
Stoplist=english.dat
IndexNumbers=1
```

```
[afrikaans]
Encodings=UTF8:afrikaansUTF8
IndexNumbers=1
```

```
[albanian]
Encodings=UTF8:albanianUTF8
IndexNumbers=1
```

```
[arabic]
Encodings=ARABIC_ISO:arabicARABIC_ISO,ARABIC:arabicARABIC,UTF8:arabicUTF8
IndexNumbers=1
```

```
[armenian]
Encodings=UTF8:armenianUTF8
IndexNumbers=1
```

```
[azeri]
Encodings=UTF8:azeriUTF8
IndexNumbers=1
```

```
[general]
Encodings=UTF8:generalUTF8,CYRILLIC:generalCYRILLIC
IndexNumbers=1
```

6. Save the configuration file.
7. You can now configure IDOL Server to associate the language types that you defined with documents.

Related Topics

- ["Supported Languages and Common Encodings" on page 461](#)
- ["Display Online Help" on page 30](#)
- ["Associate Language Types with Documents" below](#)

Associate Language Types with Documents

After you define all the language types you want IDOL Server to process, set up a field process that enables IDOL Server to associate these language types with documents.

The way that you configure this field process depends on the documents that you want to index into IDOL Server:

- ["Documents that Contain a Language Type Field"](#)
- ["Documents that Contain Field Data that can Identify Language"](#)

Related Topics

- ["Define Language Types" on page 103](#)

Documents that Contain a Language Type Field

Use the following procedure to set up languages if all the documents that you want to index into IDOL Server contain a field that specifies the language type.

To configure IDOL Server to read language type from a field

1. Set up a process for looking up the language of a document in the [FieldProcessing] section.

For example:

```
[FieldProcessing]
0=LookForLanguage
```

2. Create a section for the process. Create a Property for the process and identify the field that you want the process to apply to.

For example:

```
[LookForLanguage]
Property=SetLanguage
PropertyFieldCSVs=*/DRELANGUAGE,*/myLanguageType
```

3. Create a section for this property. Set the LanguageType parameter to True to map the values of the */DRELANGUAGE fields to the equivalent language type in the [LanguageTypes] section.

For example:

```
[SetLanguage]
LanguageType=True
```

```
[LanguageTypes]
0=russian
```

```
[russian]
Encodings=CYRILLIC:russianCYRILLIC,CYRILLIC_ISO:russianCYRILLIC_ISO,CYRILLIC_
KOI8:russianCYRILLIC_KOI8,UTF8:russianUTF8
Stoplist=russian.dat
IndexNumbers=1
```

4. Save the configuration file and start IDOL Server.
5. You can now index documents into IDOL Server.

Related Topics

- ["Index Data" on page 43](#)

Documents that Contain Field Data that can Identify Language

Use the following procedure to set up languages if all the documents that you want to index into IDOL Server contain a field that contains data that you can use to identify the language type.

To configure IDOL Server to identify languages from field data

1. Use the [FieldProcessing] section of the IDOL Server configuration file to define each language property that you want IDOL Server to be able to detect.

For example:

```
[FieldProcessing]
0=DetectArabic
1=DetectEnglish
2=DetectChSimplified
3=DetectChTraditional
4=DetectFrench
```

2. Define a section with the name of the respective language type for each of the languages that you defined in the [FieldProcessing] section. In this section, specify the fields that IDOL Server must look for and the values that those fields must have to recognize the document as a particular language type.

For example:

```
[DetectArabic]
Property=SetArabicProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=arabic
```

```
[DetectEnglish]
Property=SetEnglishProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=*eng*,uk,*british
```

```
[DetectChSimplified]
Property=SetChSimplifiedProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=*ChSimp*,ChineseSimp*
```

```
[DetectChTraditional]
Property=SetChTraditionalProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=*ChTrad*,ChineseTrad*
```

```
[DetectFrench]
Property=SetFrenchProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/DRELANGUAGETYPE,*/LANG
PropertyMatch=*fre*,fran*
```

3. Define a section with the same value as the respective property for each Property that you

defined in the [FieldProcessing] subsections. In this section, you can then specify the language type (which you must also list in the [LanguageTypes] section where you define how you want IDOL Server to handle the individual languages).

For example:

```
[SetArabicProperty]
LanguageType=Arabic
HiddenType=True
```

```
[SetEnglishProperty]
LanguageType=English
HiddenType=True
```

```
[SetChSimplifiedProperty]
LanguageType=ChSimplified
HiddenType=True
```

```
[SetChTraditionalProperty]
LanguageType=ChTraditional
HiddenType=True
```

```
[SetFrenchProperty]
LanguageType=French
HiddenType=True
```

4. Save the configuration file and start IDOL Server.
5. You can now index documents into IDOL Server.

Related Topics

- ["Index Data" on page 43](#)

Add LanguageType Fields to Documents

You can configure any of the IDOL connectors to add fields to documents from which IDOL Server can read the language type of the documents.

To add a language type field to documents in the connector

1. Open the configuration file of your IDOL connector in a text editor.
2. In the [TaskName] or [Ingestion] section, set the IngestActions parameter to `META:FieldName=FieldVal`.

For example:

```
IngestActions=META:DRELanguage=englishUTF8
```

Note: If you add this setting to a [TaskName] section, it applies only to the connector task defined in that section. If you add the setting to the [Ingestion] section, it applies to all connector tasks.

3. Save the configuration file.

Alternatively, you can use a Lua script to add a field in a Connector Framework Server (CFS) task. In a setup with a single CFS for multiple connectors, and documents in a single language, this method allows you to process all documents in one task, rather than setting up the ingest action for each connector.

Define a Default Language Type

You can specify a default language type (language and encoding) that IDOL uses for documents of unspecified language. It uses the default language type when:

- IDOL Server cannot read the language type of a document from a specified field.
- the action does not include a `LanguageType` parameter.
- automatic language detection is not enabled.

The default language type is also the default for the `LanguageType` parameter in actions such as `Query` and `Summarize`.

To specify a default language type

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[LanguageTypes]` section and enter the language type to associate with any document that does not contain a language type field.

If you use automatic language detection, IDOL Server uses the detected language and encoding to determine the language type of documents, and not the default language type.

For example:

```
[LanguageTypes]
DefaultLanguageType=englishUTF8
LanguageDirectory=C:\HewlettPackardEnterprise\IDOLServer\common\langfiles
```

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

Define a General Language

You can specify a General language, which IDOL uses for documents with an unconfigured language, but whose encoding is identified. IDOL categorizes documents as the General language when:

- IDOL Server cannot read the language type nor the encoding of a document from a specified field.
- automatic language detection is enabled.

If IDOL Server detects an unconfigured language type, it indexes to the equivalent `General` language type for that encoding, if it exists. It also logs a warning message in the index log so that you can add an appropriate language type to the configuration file. IDOL also indexes unknown languages to the `General` language type for the encoding, if it exists. If the encoding is unknown, IDOL indexes the document to the default language.

If you set `DiscardUnconfiguredLanguagesAtIndex` to `True`, IDOL does not index documents with unconfigured languages, even if a `General` language exists for that encoding.

To specify a General language category

1. Open the IDOL Server configuration file in a text editor.
2. Add the appropriate encodings to the [General] section.
3. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

If the document has no identified language or encoding, IDOL indexes it to the `DefaultLanguageType`.

Related Topics

- ["Define Language Types" on page 103](#)
- ["Define a Default Language Type" on the previous page](#)

Enable Automatic Language Detection

If your IDOL Server license includes automatic language detection, IDOL Server can automatically identify the language and encoding of a document when it is indexed. IDOL Server analyzes a certain amount of text in the document content fields (fields for which `SourceType` is set to `True` in the IDOL Server configuration file).

To enable automatic language detection

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Server] section and add this setting:

```
AutoDetectLanguagesAtIndex=True
```

3. Set `DiscardUnconfiguredLanguagesAtIndex` to `True` if you do not want to index documents with a language type that is not configured.

Set `DiscardUnknownLanguagesAtIndex` to `True` if you do not want to index documents whose language IDOL cannot recognize. For example, it might not recognize the language because the document does not contain language, or it might not have enough text for IDOL Server to determine the language.

By default IDOL Server indexes the document using the default language type. It also logs a warning message in the index log, so that you can add an appropriate language type.

4. You can change the amount of text that IDOL Server analyzes to detect the language of a document. By default, IDOL Server uses only a few sentences. In some situations, increasing the amount of text to analyze can give more accurate results, such as when significant amounts of a minor second language are present.

Add the `MaxLanguageDetectTerms` setting to the [Server] section, specifying the number of terms (words) that IDOL Server uses for detection. For example:

```
MaxLanguageDetectTerms=1000
```

5. By default, IDOL Server detects any 7-bit ASCII characters as UTF-8. If you instead want to group these documents with documents using 8-bit ASCII, disable the `LangDetectUTF8` parameter by setting it to `False`.

Ensure that the encoding option you want is present in the language type configuration (see ["Define Language Types" on page 103](#)). If there are no compatible encodings configured for the detected language, IDOL assigns the default language type.

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

Note: If you enable automatic language detection and set up a field process that reads the language of a document from one of its fields, IDOL Server uses the field process rather than autodetection to determine the document language and encoding.

Specify the Language Type of a Query

When you send a query to IDOL Server, by default it reads the query language type as the `DefaultLanguageType` defined in the IDOL Server configuration file.

You must correctly set the language type of any query text so that IDOL Server can handle the query text correctly (for example, stemming correctly). If you want to send a query that does not use the default language type, you must add the `LanguageType` parameter to your query action. This parameter specifies that the query uses the language and encoding set in the IDOL Server configuration file for the specified `LanguageType`.

For example:

The following query uses the language and encoding specified for the `DefaultLanguageType`, so you can send it to IDOL Server without adding the `LanguageType` parameter:

```
http://12.3.4.56:4000/action=Query&Text=The Bayes theory of probability
```

The following query uses the language and encoding specified for the German language type:

```
http://12.3.4.56:4000/action=Query&Text=Einsteins  
Relativitätstheorie&LanguageType=German
```

Related Topics

- ["Define a Default Language Type" on page 108](#)

Convert Results to a Specific Encoding

You can send the following types of query to IDOL Server:

- **Text queries.** Queries that contain some form of query text (for example, `Query`, `SuggestOnText`, `Summarize`, and so on).
- **Text-free queries.** Queries that do not contain any query text (for example, `Suggest`, `List`, `GetContent`, and so on).

Text Queries

When you send a query action to IDOL Server, by default it returns results that use the same language and encoding as the query text. This language type can be:

- the language and encoding specified in the `LanguageType` parameter sent with the query.
- the language type specified in the `DefaultLanguageType` configuration parameter if the query does not include a `LanguageType` parameter.

To return query results in a specific encoding, add the `OutputEncoding` parameter to your query action. This parameter converts the results of a query to any type of encoding that is compatible with the query language. If you specify an encoding that is not compatible with the query language, IDOL Server returns an appropriate message.

You can specify a default value for the `OutputEncoding` parameter in the `DefaultEncoding` configuration parameter.

For example:

```
http://12.3.4.56:4000/action=Query&Text=Neurologia i  
Neurochirurgia&LanguageType=PolishEASTERNEUROPEAN&OutputEncoding=EASTERNEUROPEAN_  
ISO
```

In this example, IDOL Server converts all query results to `EASTERNEUROPEAN_ISO`.

Related Topics

- ["Supported Languages and Common Encodings" on page 461](#)

Text-Free Queries

By default, query actions that do not contain any query text return results in the encoding specified in the `DefaultLanguageType` parameter. If any of the query results are not compatible with this encoding, IDOL Server indicates this in the results.

If you want a query action to return results in a specific encoding, you can add the `OutputEncoding` parameter to your query. IDOL Server converts all results to this encoding, as long as they are compatible with it. If any of the query results are not compatible with this encoding, IDOL Server returns an appropriate message. You can specify a default value for the `OutputEncoding` parameter in the `DefaultEncoding` configuration parameter.

For example:

```
http://12.3.4.56:4000/action=Suggest&ID=9016&OutputEncoding=EASTERNEUROPEAN_ISO
```

In this example, IDOL Server converts all query results to `EASTERNEUROPEAN_ISO`.

Return Documents in Multiple Languages

When you send a query action to IDOL Server, by default it returns results that use the same language and encoding as the query text. This language type can be:

- the language and encoding specified in the `LanguageType` parameter sent with the query.
- the language type specified in the `DefaultLanguageType` configuration parameter if the query does not include a `LanguageType` parameter.

To return documents in any language for your query rather than in the query language, set the `AnyLanguage` parameter to `True` to your query.

When IDOL Server receives the query, it applies the stemming algorithm and stop word list that is appropriate for the query language type. It returns only documents that contain words which match the stopped and stemmed terms in the query (that is, the words in result documents must stem to the same root as the words in the query text).

For example:

```
http://12.3.4.56:4000/action=Query&Text=Innovative internet marketing solutions in  
Baghdad&AnyLanguage=True
```

In this example, IDOL Server returns documents in multiple languages that contain terms that match terms in the specified `Text`.

The query returns documents in multiple languages only if they contain terms that match terms in the query (for example, query text that contains the term *Baghdad* might return documents in English, French, German, and so on).

International Stop List

When you use the `AnyLanguage` parameter in all or most of your queries, you might want to use the international stop list. This stop list contains common stop words for all languages, but does not include terms that are stop words in one language, but useful search terms in another language.

For multilingual searching, this stop list ensures that the query includes all useful terms, while still removing all stop words that are never useful terms.

The `international.dat` stop list is included in the `langfiles` directory of a default IDOL Server installation. You must configure each of the languages in your configuration file to use this stop list. If you have existing data in IDOL Server, you must also reindex the data.

To configure the international stop list

1. Open the IDOL Server configuration file in a text editor.
2. In the `[Languages]` configuration section, find the configuration sections for all the languages that you want to return in `AnyLanguage` queries, including the language specified in the `DefaultLanguageType` parameter.
3. In each configuration section, set `Stoplist` to `international.dat`. For example:

```
[english]  
Encodings=UTF8:generalUTF8  
Stoplist=international.dat
```

```
[italian]  
Encodings=UTF8:italianUTF8  
Stoplist=international.dat
```

```
[spanish]  
Encodings=UTF8:spanishUTF8  
Stoplist=international.dat
```

4. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
5. Reindex all your data into IDOL Server to apply the changes to all your documents.

Related Topics

- ["Stop Word Lists for Supported Languages" on page 509](#)

Return Documents in a Specific Language

When you send a query action to IDOL Server, by default it returns results that use the same language and encoding as the query text. This language type can be:

- the language and encoding specified in the `LanguageType` parameter sent with the query.
- the language type specified in the `DefaultLanguageType` configuration parameter if the query does not include a `LanguageType` parameter.

To return documents in one more specific languages for the query, you can set the `MatchLanguage` parameter to a list of the languages that you want to match. When you set this parameter, IDOL Server applies the stemming algorithm and stop word list that is appropriate for the query language type as usual (it does not translate the query). The results include only documents in the specified languages that match the stopped and stemmed terms in the original query (that is, the words in result documents must stem to the same root as the words in the query text).

For example:

```
action=Query&Text=Birmingham&LanguageType=EnglishUTF8&MatchLanguage=Dutch+German
```

In this example IDOL Server stems the query term *Birmingham* according to the stemming rules for the `EnglishUTF8` language type. It returns any documents in Dutch or German that contain a term that has the same stem as the English stem of *Birmingham*.

Note: If you specify `MatchLanguage`, you cannot specify `MatchLanguageType` or `MatchEncoding` in your query.

To return documents that match the query terms, regardless of the document language, you can add the `AnyLanguage` parameter to your query. See ["Return Documents in Multiple Languages" on page 111](#).

Tip: You can use the **Languages** tab in the **Status** page of the IDOL Admin interface to search for all documents in a particular language. You can also modify that query in IDOL Admin. For more information, refer to the *IDOL Admin User Guide*.

Create a Custom Stem File for a Language

You can override the default stemming rules for certain words in a particular language by creating a language-specific *stemming file*. This file is a list of words and their stems. If a stemming file exists, IDOL Server uses it to stem the terms that it contains. Terms that are not in the file stem according to the default stemming rules.

HPE recommends use of a stemming file only for unusual or specialized terms where the default rules do not generate a stem. A stemming file is not intended to be a complete replacement for the IDOL stemming algorithms.

To set up a stemming file

1. Create a text file.
2. Format the file as a stop word list. The first line is an encoding designation. Subsequent lines

contain individual word pairs; a term followed by its stem. For example:

```
[UTF8]
mice mouse
mouse mouse
children child
```

Note: To ensure that two words stem to the same value, you must add both words to the stemming file, with the appropriate stem.

3. Save the file with a name of your choice (for example, `english_stem.dat`) in the directory `installDir/common/langfiles`.
4. Open the IDOL Server configuration file. In the `[MyLanguage]` section for the stemming file language, set the `StemmingFile` configuration parameter to the name of your stemming file. For example:

```
[english]
Encodings=UTF8:englishUTF8
Stoplist=english.dat
StemmingFile=english_stem.dat
```

5. Ensure that this `[MyLanguage]` section does not set `Stemming` to `False`. The default value for `Stemming` in a language is `True`.

If you disable stemming for a language, but provide a stemming file, IDOL Server stems terms in the file, but does not stem other terms.

Decompose Compound Words

You can configure IDOL to automatically separate a compound word into root words at both index and query time. For example, the German word for *bicycle pump* is a single word *Fahrradpumpe* that can be divided into *Fahrrad* and *Pumpe*.

To specify decomposition

1. Create a text file. Use the following format:

```
[UTF8]
rollercoaster roller coaster
hemidemisemiquaver hemi demi semi quaver
```

Each line defines the decomposition for one word. The first word on a line is broken into the remaining words on the line.

2. Store the text file in the `langfiles` directory of your IDOL installation.
3. Open the IDOL Server configuration file in a text editor. For each language that the decomposition file applies to, specify the file name in the `DecompositionFile` configuration parameter. For example:

```
[German]
DecompositionFile=german_decomp.txt
```

Note: Each of the terms in the output from the decomposition is also stemmed.

Enable Transliteration

Transliteration is the process of converting accented characters such as øù into equivalent non accented characters. This process is useful in environments where accented keyboards are not available.

Enable Generic Transliteration

The default IDOL Server configuration file uses generic transliteration. HPE recommends that you use generic transliteration because it is the best way to ensure that cross-lingual search can happen.

To enable transliteration for all languages

- In the [LanguageTypes] configuration section, set the `GenericTransliteration` parameter to `True`.

Generic transliteration performs transliteration as described in the following table.

Language or character type	Transliteration
Symbols	All dashes and hyphens to a hyphen character.
Latin	Accented characters to non-accented characters
Spanish	Accented vowels áéíóúü to non-accented vowels
Portuguese	Accented vowels àáâãçéêíóôõü to non-accented vowels
Greek	Accented Greek characters to non-accented characters
Cyrillic (including Serbian extensions)	All characters mapped to A–Z
Arabic	Arabic character normalization
Japanese	Half width katakana to full width katakana Full width 0–9, A–Z, a–z to single byte 0–9, A–Z, a–z
Chinese	Full width 0–9, A–Z, a–z to single byte 0–9, A–Z, a–z

For all other languages, transliteration does not apply, except for hyphen normalization.

Note: Languages with a sentence-breaking library might be transliterated as part of the sentence-breaking process.

When you set `GenericTransliteration` to `True`, it applies to all languages, unless you specifically disable transliteration for a language.

You can disable transliteration for an individual language by setting the `Transliteration` parameter to `False` in the individual language configuration section. This option completely disables transliteration for that language.

Enable Transliteration for Individual Languages

In cases where you need a particular transliteration, you can set `GenericTransliteration` to `False` and use the per-language transliteration schemes.

To turn transliteration on or off for an individual language

- In the `[MyLanguage]` individual language configuration sections, set the `Transliteration` parameter to `True` or `False`.

When you set `GenericTransliteration` to `False`, IDOL Server always performs transliteration for the languages in the following table, even if you also set `Transliteration` to `False`.

Language	Transliteration
Japanese	Half width katakana to full width katakana Full width 0–9, A–Z, a–z to single byte 0–9, A–Z, a–z
Chinese	Full width 0–9, A–Z, a–z to single byte 0–9, A–Z, a–z
Greek	Accented Greek characters to non-accented characters
Spanish	Accented vowels áéíóúü to non-accented vowels
Portuguese	Accented vowels àáâãçêêíîóôõúü to non-accented vowels
Russian	Some removal of characters
Arabic	Arabic character normalization
Persian	
Sindhi	
Urdu	
Malay	
Malayalam	
Pushto	

Transliteration is optional for the language groups in the following table.

Language group	Transliteration
Western European	àáâãä=á â=aa ç=c èéêë=e íîï=i òóôõö=o ø=oe ùúûü=u œ(oe)=oe æ=ae ß=ss ñ=nh ý=y ð=d þ=th

Language group	Transliteration
German	Same as Western European apart from: ä=ae ö=oe ü=ue
Scandinavian	Same as Western European apart from: ä=ae ö=oe ü=ue
Catalan	Same as Western European apart from: ç=sz
Cyrillic	All characters mapped to A–Z Transliteration scheme uses British Standard 2979:1958
South Slavic	For transliteration scheme, refer to <i>A Handbook of Bosnian, Serbian and Croatian</i> by Brown & Alt.

The following table describes the languages that each of these language groups contain.

Western European	Czech Dutch English French Hungarian	Italian Maori Mirandese Polish Portuguese	Romanian Slovakian Spanish Turkish
German	German		
Scandinavian	Danish Finnish Icelandic	Norwegian Swedish	
Catalan	Catalan		
Cyrillic	Russian Tajik		
South Slavic	Bosnian Serbian Croatian		

For all other languages, transliteration does not apply, except for hyphen normalization.

For full details of the transliteration of different characters in different transliteration modes, refer to *IDOL Expert*.

Related Topics

- ["Index Nonalphanumeric Characters" on page 54](#)

Chapter 6: Set Up Document Tracking

Document tracking is a feature present in IDOL components that are involved in indexing. It reports upon the progress of documents as they pass through an index chain. Every time a document reaches a certain stage in the indexing process, the component commits event data to a back end, which stores the events.

The back end can be a log file, or an SQL database (this option requires an additional library, which is available in the IDOL Server installer). You retrieve the events by using an appropriate interface for your chosen back end, for example an SQL client for a database.

This section describes how to set up document tracking using either SQL or IDOL Server log files as a back end.

- [Set up Document Tracking with an SQL Back End](#) 119
- [Set up Document Tracking with a Log Back End](#) 133
- [Configure Event Storage](#) 134

Set up Document Tracking with an SQL Back End

This section contains instructions for setting up document tracking in IDOL with an SQL back end. It includes instructions for setting up a PostgreSQL or Microsoft SQL Server database. PostgreSQL is the recommended and most fully tested database, but you can also use Microsoft SQL Server (2008 (SP3) or 2012), or Oracle.

- ["Set up Document Tracking with PostgreSQL"](#)
- ["Set up Document Tracking with Microsoft SQL Server"](#)

Set up Document Tracking with PostgreSQL

This section describes how to set up document tracking with PostgreSQL. This process includes the following steps:

- Install the PostgreSQL database.
- Create the database structure.
- Install SQL drivers on the IDOL host machines.
- Configure IDOL components.

Set up PostgreSQL to Store Tracking Information

This section describes how to set up the PostgreSQL back end.

The example actions and commands are for an IDOL Server running on a Microsoft Windows platform, and a PostgreSQL server running on a Linux platform. Other combinations of platforms are possible.

Install the SQL Database

The following Linux command example installs the latest stable PostgreSQL server, using the default port (5432). You can use any recent, stable version (on any port).

```
sudo apt-get install postgresql
```

You can test that your server is up with `psql`.

For Windows, you can install PostgreSQL from the following Web site:

<http://www.postgresql.org/download/windows/>

For more detailed installation instructions, refer to the PostgreSQL wiki.

https://wiki.postgresql.org/wiki/Main_Page

Set up the Database and Table

This section describes how to set up the database in PostgreSQL on Linux. On Windows, you can complete the tasks by using Pgadmin.

The PostgreSQL installation creates a user for you.

To set up the database and table on Linux

1. Create a database with an arbitrary name. For example:

```
sudo -u postgres createdb mydoctrackdb
```

You can test it by using the following example command:

```
sudo -u postgres psql -d mydoctrackdb
```

2. Create the tables to store document tracking events, by running the following commands from `psql` or your SQL client:

```
CREATE TABLE type(  
    typeid serial PRIMARY KEY,  
    type varchar(64) NOT NULL UNIQUE,  
    is_error smallint,  
    is_terminal smallint  
);
```

```
-- Broken into several INSERTs here for clarity only  
INSERT INTO type(typeid, type, is_error, is_terminal)  
VALUES(0, 'Unknown', 0, 0);
```

```
INSERT INTO type(typeid, type, is_error, is_terminal)  
VALUES(10, 'Committed', 0, 1),  
      (20, 'Deleted', 0, 1),  
      (30, 'Indexed', 0, 0),  
      (40, 'Received', 0, 0);
```

```
INSERT INTO type(typeid, type, is_error, is_terminal)
```

```
VALUES(50, 'Updated', 0, 1),
      (-10, 'Warning', 1, 0),
      (-20, 'Error', 1, 1),
      (-30, 'Rejected', 1, 1);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES(90, 'Added', 0, 0),
      (100, 'Delete received.', 0, 0),
      (110, 'Update received.', 0, 0);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES(120, 'Non-importing add received.', 0, 0),
      (130, 'Import:Queue', 0, 0),
      (140, 'Import:Importing', 0, 0);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES(150, 'Import:Pre', 0, 0),
      (160, 'Import:Post', 0, 0),
      (170, 'Import:Finished', 0, 0);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES (180, 'Import:Cancel', 0, 1);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES (190, 'Import:Extracting metadata', 0, 0);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES (200, 'Import:Extracting metadata finished', 0, 0);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES (210, 'Import:ExtractMetaAbort', 1, 1);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES (220, 'Import:Abort', 1, 1);

INSERT INTO type(typeid, type, is_error, is_terminal)
VALUES (230, 'Replaced', 0, 0);

CREATE TABLE source(
  sourceid serial PRIMARY KEY,
  source varchar(128) NOT NULL UNIQUE
);

CREATE TABLE event(
  eventid serial PRIMARY KEY,
  docuid varchar(128) NOT NULL,
  typeid int NOT NULL,
  sourceid int NOT NULL,
```

```
timestamp bigint NOT NULL,  
  
CONSTRAINT type_fk FOREIGN KEY(typeid)  
  REFERENCES type(typeid)  
  ON DELETE CASCADE ON UPDATE CASCADE,  
  
CONSTRAINT source_fk FOREIGN KEY(sourceid)  
  REFERENCES source(sourceid)  
  ON DELETE CASCADE ON UPDATE CASCADE  
);  
  
CREATE TABLE metadata(  
  metadataid serial PRIMARY KEY,  
  key varchar(32) NOT NULL,  
  value varchar(1024) NOT NULL  
);  
  
CREATE TABLE event_metadata(  
  eventid int NOT NULL,  
  metadataid int NOT NULL,  
  
  CONSTRAINT event_fk FOREIGN KEY(eventid)  
    REFERENCES event(eventid)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
  
  CONSTRAINT metadata_fk FOREIGN KEY(metadataid)  
    REFERENCES metadata(metadataid)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);  
  
CREATE TABLE docuid_reference(  
  docuid varchar(128) NOT NULL,  
  ref varchar(4096) NOT NULL,  
  
  UNIQUE (docuid, ref)  
);  
  
CREATE TABLE doctrack_schema_version(  
  key varchar(64) NOT NULL,  
  value varchar(128) NOT NULL  
);  
  
INSERT INTO doctrack_schema_version(key,value)  
  VALUES('major_version', '1'),  
        ('minor_version', '1'),  
        ('IDOL_version', '10.9');  
  
CREATE RULE no_schema_version_insert AS
```

```
ON INSERT TO doctrack_schema_version  
DO INSTEAD NOTHING;
```

```
CREATE RULE no_schema_version_delete AS  
ON DELETE TO doctrack_schema_version  
DO INSTEAD NOTHING;
```

Note: If you want to set up document tracking in an existing IDOL Server installation that uses the deprecated IndexTasks component, you must also add the following statements for the type table:

```
INSERT INTO type(typeid, type, is_error, is_terminal)  
VALUES(60, 'IndexTasksCompleted', 0, 0),  
      (70, 'IndexTasksStarted', 0, 0),  
      (80, 'IndexTask', 0, 0);
```

To set up the database and table on Windows

1. Use Pgadmin to run the SQL **Create Table** command. Right-click **databases/mydoctrackdb** (or the name of the database that you created) and select **Create script**.

Database Access Permissions

On Windows, you can modify the configuration in Pgadmin by selecting the appropriate item on the left pane, and then clicking **Tools/Server Configuration**.

To modify the database access permissions

1. Find the `pg_hba.conf` host-based authentication file by inspecting the PostgreSQL configuration file. The following lines in the `postgresql.conf` file identify the location:

```
hba_file = '/etc/postgresql/9.1/main/pg_hba.conf' # host-based authentication  
file
```

Note: The location of the `postgresql.conf` file can vary, depending on your version and operating system. On Linux, you can run the following command to find the path to the configuration file:

```
ps -ef | grep postgres
```

2. Modify the `pg_hba.conf` configuration file, which you located in [Step 1](#), to allow your IDOL Servers to access the database. Find the following section, and add appropriate lines for your client IP addresses.

```
# TYPE  DATABASE  USER          ADDRESS          METHOD  
host    all       my.user.name  10.2.123.123/32  trust
```

For simplicity, you can set the `USER` field to `all`. HPE recommends that you use a secure `METHOD`, such as `md5`, after you have tested the system.

3. Allow PostgreSQL to accept connections. In the `postgresql.conf` file, find the `listen_addresses` parameter and uncomment or modify it:

```
listen_addresses = '*'
```

4. Save and close the `pg_hba.conf` and `postgresql.conf` files.

5. Restart PostgreSQL, by using the following command (on Linux):

```
sudo -u postgres /etc/init.d/postgresql restart
```

Set up the IDOL Host Machines

To communicate with the SQL server, you must install an SQL driver (for your server type), and you must install an ODBC driver manager on the IDOL host machines.

Note: To use document tracking with a PostgreSQL back end, you must have PostgreSQL ODBC client driver version 9.1.0 or later.

Install the SQL Driver and Manager for PostgreSQL

Microsoft Windows has a driver manager by default. If you require a driver manager for Linux, try UnixODBC, for example by using the following command:

```
sudo apt-get install unixodbc
```

You can download drivers for both Windows and Linux on the PostgreSQL Web site:

<http://www.postgresql.org/ftp/odbc/versions/>

Make sure that you install the correct version for your platform (for example, 64-bit).

On Linux, it might be easier to use your package manager. For PostgreSQL:

```
sudo apt-get install odbc-postgresql
```

Note: Make a note of the name of the SQL driver that you install, because you must reference it in a configuration parameter, or a data source name (DSN).

Check the Installed Drivers

Use the following procedures to check the installed drivers.

Note: You can use the Windows user interface to find the installed drivers for the Microsoft SQL Server back end.

To check the installed drivers on Linux and UNIX ODBC

- In your terminal (not in `psql`), type the following command to list the available drivers:

```
odbcinst -d -q
```

To find a list of drivers on Windows

1. Run the following command to open the driver manager:

```
%windir%\system32\odbcad32.exe
```

2. In the driver manager, review the information on the **Drivers** tab. The following driver is required:
PostgreSQL ANSI or PostgreSQL ANSI (x64)

Configure IDOL Components

To use document tracking with an SQL back end, you must use the HPE document tracking library, which is included in the IDOL Server installer. You can store this library in any accessible location, and then configure the location in IDOL Server.

You must then configure your IDOL Server components to use the SQL document tracking back end. The DIH, Content component, Connector Framework Server (CFS), and CFS Connectors support document tracking.

After configuration, an IDOL component automatically adds itself to the `Source` table on startup.

Note: If you retire an IDOL component, HPE recommends that you leave the entry in the `Source` table, because existing records might refer to the source of the retired component.

To configure IDOL Server for document tracking

1. Open the IDOL Server configuration file in a text editor.
2. If you are using DIH in a unified IDOL configuration, turn on document tracking for the DIH by setting the `DocumentTracking` parameter to `True` in the `[DistributionSettings]` section.

```
[DistributionSettings]
DocumentTracking=True
```

Note: In a stand-alone DIH, set the `DocumentTracking` parameter to `True` in the `[Server]` section of the DIH configuration file.

3. Create a `[DocumentTracking]` configuration section.
4. In this `[DocumentTracking]` section, set the `Backend` parameter to `Library`.
5. Set `LibraryPath` to the absolute path to your document tracking library. In the default IDOL Server installation, the library is located in `InstallPath/IDOL/modules/`, and the library name is `dt_odbc.dll` (on Windows), or `libdt_odbc.so` (on UNIX).
6. Set `ConnectionString` to the connection string to use, with subparameters set for your setup. For example:

```
ConnectionString=Driver=PostgreSQL ANSI(x64); Server=sql-host.mycompany.com;
Port=5432; Database=mydoctrackdb; UID=postgres; Password=password;
```

You can also use a DSN instead of subparameters. For example:

```
ConnectionString=DSN=MyDSN
```

Set the parameters consistent with your environment. On Linux, the connection string cannot contain spaces.

On Linux, you can set the `Driver` subparameter to the path to your SQL ODBC driver shared object (for example, `/usr/lib/x86_64-linux-gnu/odbc/psqlodbc.so` or `/usr/lib/odbc/psqlodbc.so`). HPE recommends that you use the ANSI version of libraries.

Tip: On Linux, if you use a DSN in your connection string, and you see `File not Found` errors in your IDOL logs, try specifying the driver explicitly in the `ConnectionString` configuration parameter:

```
ConnectionString=Driver=/usr/lib/psqlodbc.so;DSN=mydsn;
```

7. Set any other configuration parameters for document tracking. For more information, refer to the *IDOL Server Reference*. For example:

```
[DocumentTracking]  
MaxEventsPerFile=500  
TimeoutSeconds=20  
UIDFieldName=UID
```

Note: Connectors generate the document ID strings, and add them to the `UIDFieldName` field, so you must not use a field that already exists for another purpose.

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

Set up Document Tracking with Microsoft SQL Server

This section describes how to set up document tracking with Microsoft SQL Server 2008 (SP3). This process includes the following steps:

- Install Microsoft SQL Server.
- Create the database structure.
- Install SQL drivers on the IDOL host machines.
- Configure IDOL components.

Set up Microsoft SQL Server to Store Tracking Information

This section describes how to use Microsoft SQL Server as your SQL back end.

Configuration Example for Microsoft SQL Server

For the Microsoft SQL Server back end, you must set the `ConnectionString` configuration parameter in the `[DocumentTracking]` configuration section. This parameter must contain a valid connection string, as understood by SQL Server and your ODBC driver manager.

HPE recommends that you use a DSN if you are running IDOL on a Windows operating system. This option lets the operating system save your settings. If you use Windows integrated authentication, use a connection string of the following form:

```
ConnectionString=DSN=dt;database=odbc_test_db
```

If you use SQL Authentication `[user+password]`, add `TRUSTED_CONNECTION=yes;`. For example:

```
ConnectionString=DSN=dt;TRUSTED_CONNECTION=yes;database=odbc_test_db
```

Note: You can set the database that you want to connect to in the DSN configuration GUI, or Microsoft SQL server. However, for SQL Server 2005, SQL Server 2008, and SQL Server 2012, HPE recommends that you explicitly specify it in the connection string connection options.

You can also use a DSN on UNIX operating systems, but you might find configuration easier if you include all parameters in the `ConnectionString` parameter and omit the DSNs.

Troubleshoot Connection and Authentication Problems

The following table describes some common connection and authentication problems that you can identify in the SQL Management Studio, on the server side.

Issue	Suggestion
SQL Server is not configured to allow SQL Authentication.	In Server Properties, review the settings on the Security tab.
SQL Server is not configured to allow remote connections.	In Server Properties, review the settings on the Connections tab.
SQL Server is not mapping the user to the correct permissions for your database.	In the properties for the user, review the settings on the User Mapping tab.

For the client, check the following in the DSN configuration GUI:

- The driver must be SQL Server Native Client 10.0.
- The DSN must have **Use ANSI quoted identifiers**, **Use ANSI nulls**, **padding**, **warnings**, and **Perform translation for character data** selected.

Initialization Commands

You can run the following commands from an SQL command-line interface, or from the GUI in Microsoft SQL Server Management Studio.

```
CREATE TABLE type(
    typeid integer identity PRIMARY KEY,
    type varchar(64) NOT NULL UNIQUE,
    is_error smallint,
    is_terminal smallint
);
SET IDENTITY_INSERT type ON
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES(0, 'Unknown', 0, 0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (10, 'Committed', 0, 1);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (20, 'Deleted', 0, 1);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (30, 'Indexed', 0, 0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (40, 'Received', 0, 0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (50, 'Updated', 0, 1);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (-10, 'Warning', 1, 0)
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (-20, 'Error', 1, 1);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (-30, 'Rejected', 1, 1);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (90, 'Added', 0, 0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (100, 'Delete
received.', 0, 0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (110, 'Update
received.', 0, 0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (120, 'Non-importing
```

```
add received.',0,0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(130,'Import:Queue',0,0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(140,'Import:Importing',0,0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(150,'Import:Pre',0,0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(160,'Import:Post',0,0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(170,'Import:Finished',0,0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(180,'Import:Cancel',0,1);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(190,'Import:Extracting metadata',0,0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(200,'Import:Extracting metadata finished',0,0);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(210,'Import:ExtractMetaAbort',1,1);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES
(220,'Import:Abort',1,1);
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES (230,'Replaced',0,0);

CREATE TABLE source(
    sourceid integer identity PRIMARY KEY,
    source varchar(128) NOT NULL UNIQUE
);

CREATE TABLE event(
    eventid integer identity PRIMARY KEY,
    docuid varchar(128) NOT NULL,
    typeid int NOT NULL,
    sourceid int NOT NULL,
    timestamp bigint NOT NULL
);

CREATE TABLE metadata(
    metadataid integer identity PRIMARY KEY,
    [key] varchar(32) NOT NULL,
    value varchar(1024) NOT NULL
);

CREATE TABLE event_metadata(
    eventid int NOT NULL,
    metadataid int NOT NULL,
);

CREATE TABLE docuid_reference(
    docuid varchar(128) NOT NULL,
```

```
    ref varchar(900) NOT NULL,  
);  
  
CREATE TABLE doctrack_schema_version(  
    [key] varchar(64) NOT NULL,  
    value varchar(128) NOT NULL  
);  
  
INSERT INTO doctrack_schema_version([key],value) VALUES ('major_version', '1');  
INSERT INTO doctrack_schema_version([key],value) VALUES ('minor_version', '1');  
INSERT INTO doctrack_schema_version([key],value) VALUES ('IDOL_version', '10.9');
```

Note: If you want to set up document tracking in an existing IDOL Server installation that uses the deprecated IndexTasks component, you must also add the following statements for the type table:

```
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES  
(60, 'IndexTasksCompleted', 0, 0);  
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES  
(70, 'IndexTasksStarted', 0, 0);  
INSERT INTO type(typeid, type, is_error, is_terminal) VALUES  
(80, 'IndexTask', 0, 0);
```

Set up the IDOL Host Machines

To communicate with the SQL server, you must install an SQL driver (for your server type), and you must install an ODBC driver manager on the IDOL host machines.

Install the SQL Driver and Manager for Microsoft SQL Server

Microsoft Windows comes with SQL Server drivers by default. You can use `odbcad32.exe` (in `WindowsDirectory\system32`) to verify that SQL Server Native Client 10.0 is installed.

For SQL Server 2008 SP3, you can download the driver from the Microsoft Web site:

<http://www.microsoft.com/en-us/download/details.aspx?id=27596>

Note: You might need a different version of the ODBC drivers if you are using a different edition of SQL Server.

UNIX machines can use the open-source FreeTDS driver to connect to Microsoft SQL Server. A driver developed by Microsoft also exists, targeted at Red Hat Enterprise Linux. This example uses FreeTDS, which can be obtained from the following Web site:

<http://www.freetds.org/>

Download the driver distribution, and then read, edit, and run the following commands.

1. Unzip the package:

```
tar -zxf freetds-stable.tgz
```

2. Use the following commands to build the driver:

```
cd freetds-0.91
```

```
./configure --prefix=YourFileDirectory --with-tdsver=8.0  
make  
make install
```

These commands copy the drivers to your prefix directory. The driver itself is copied to

Prefix/lib/libtdsodbc.so

You can reference this driver in your connection string when configuring IDOL. For more information, see ["Configure IDOL Components" below](#).

For example:

```
ConnectionString=Driver=/freetds/inst/lib/libtdsodbc.so;TDS_  
Version=7.0;Server=myserverhostname\SQLEXPRESS;Port=56841;UID=idol;PWD=password;
```

Check the Installed Drivers

Use the following procedures to check the installed drivers.

Note: You can use the Windows user interface to find the installed drivers for the Microsoft SQL Server back end.

To check the installed drivers on Linux and UNIX ODBC

- In your terminal (not in an SQL client), type the following command to list the available drivers:

```
odbcinst -d -q
```

To find a list of drivers on Windows

1. Run the following command to open the driver manager:

```
%windir%\system32\odbcad32.exe
```

2. In the driver manager, review the information on the **Drivers** tab. The following driver is required:
SQL Server Native Client 10.0

Configure IDOL Components

To use document tracking with an SQL back end, you must use the HPE document tracking library, which is included in the IDOL Server installer. You can store this library in any accessible location, and then configure the location in IDOL Server.

You must then configure your IDOL Server components to use the SQL document tracking back end. The DIH, Content component, Connector Framework Server (CFS), and CFS Connectors support document tracking.

After configuration, an IDOL component automatically adds itself to the *Source* table on startup.

Note: If you retire an IDOL component, HPE recommends that you leave the entry in the *Source* table, because existing records might refer to the source of the retired component.

To configure IDOL Server for document tracking

1. Open the IDOL Server configuration file in a text editor.
2. If you are using DIH in a unified IDOL configuration, turn on document tracking for the DIH by setting the `DocumentTracking` parameter to `True` in the `[DistributionSettings]` section.

```
[DistributionSettings]
DocumentTracking=True
```

Note: In a stand-alone DIH, set the `DocumentTracking` parameter to `True` in the `[Server]` section of the DIH configuration file.

3. Create a `[DocumentTracking]` configuration section.
4. In this `[DocumentTracking]` section, set the `Backend` parameter to `Library`.
5. Set `LibraryPath` to the absolute path to your document tracking library. In the default IDOL Server installation, the library is located in `InstallPath/IDOL/modules/`, and the library name is `dt_odbc.dll` (on Windows), or `libdt_odbc.so` (on UNIX).
6. Set `ConnectionString` to the connection string to use, with subparameters set for your setup. For example:

```
ConnectionString=DSN=MyDSN
```

Set the parameters consistent with your environment. On Linux, the connection string cannot contain spaces.

On Linux, you can set the `Driver` subparameter to the path to your SQL ODBC driver shared object. HPE recommends that you use the ANSI version of libraries.

Tip: On Linux, if you use a DSN in your connection string, and you see `File not Found` errors in your IDOL logs, try specifying the driver explicitly in the `ConnectionString` configuration parameter:

```
ConnectionString=Driver=/freetds/inst/lib/libtdsodbc.so;DSN=mydsn
```

7. Set any other configuration parameters for document tracking. For more information, refer to the *IDOL Server Reference*. For example:

```
[DocumentTracking]
MaxEventsPerFile=500
TimeoutSeconds=20
UIDFieldName=UID
```

Note: Connectors generate the document ID strings, and add them to the `UIDFieldName` field, so you must not use a field that already exists for another purpose.

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

Verify the Setup

The following section describes how to troubleshoot and verify your setup.

Check IDOL Configuration

When you start the Content component, it logs the following message in the application log when the document tracking module loads correctly:

```
Starting document tracker
```

IDOL Server logs messages about document tracking to the Index log stream. To see these messages, set the `LogLevel` parameter to `Full` for this log stream.

Index Content

To test the document tracking setup, you can create an IDX document to add to IDOL Server. Make sure that you include the configured `UIDFieldName` in your IDX. Alternatively, you can use a connector to generate the IDX and index it.

The following example IDX has the UID field configured as the `UIDFieldName`.

```
dtsample.idx
#DRREFERENCE sample_idx_001
#DRETITLE A document to test document tracking
#DREDBNAME Default
#DREFIELD UID="001"
#DREFIELD Myfield1="field1"
#DREFIELD Myfield2="field2"
#DRECONTENT Did it work?
#DREENDDOC
#DREENDDATANOOP
```

Index this document into your IDOL Server, for example by using a `DREADD` index action:

```
http://idol-server:9000/DREADD?dtsample.idx&IndexUID=test
```

Query Your Document

Run a select statement to query for the document in your SQL back end. For example:

```
test:~$ sudo -u postgres psql -d odbc_test
/usr/lib/postgresql/8.4/bin/psql: /usr/local/lib/libldap_r-2.4.so.2: no versi
on information available (required by /usr/lib/libpq.so.5)
psql (8.4.15)
Type "help" for help.

odbc_test=# select * from event limit 5;
 eventid | docuid | typeid | sourceid | timestamp
-----+-----+-----+-----+-----
  114187 |  1678 |     10 |         25 | 1371487651
  114188 |  1679 |     10 |         25 | 1371487651
  114189 |  1680 |     10 |         25 | 1371487651
  114190 |  1681 |     10 |         25 | 1371487651
  114191 |  1682 |     10 |         25 | 1371487651
(5 rows)

odbc_test=#
```

You might need to wait some time (30-40 seconds) for IDOL Server to send records to your back end. After this time, if the document is not present, check the index log for SQL errors.

Clean Results

At the end of a test run, you might want to clean the database of tracking events. It is safe to delete entries from the Event table.

You must not clear the Source table while IDOL components are running. Stop the components before clearing the Source table or removing any entries from it.

Note: If you retire an IDOL component, HPE recommends that you leave the entry in the Source table, because existing records might refer to the source of the retired component.

Set up Document Tracking with a Log Back End

This section describes how to set up document tracking to store information in an IDOL Server log file.

To configure document tracking in logs

1. Open the IDOL Server configuration file in a text editor.
2. If you are using DIH in a unified IDOL configuration, turn on document tracking for the DIH by setting the DocumentTracking parameter to True in the [DistributionSettings] section.

```
[DistributionSettings]
DocumentTracking=True
```

Note: In a stand-alone DIH, set the `DocumentTracking` parameter to `True` in the `[Server]` section of the DIH configuration file.

3. Create a `[DocumentTracking]` configuration section.
4. In this `[DocumentTracking]` section, set the `Backend` parameter to `Log`.
5. Set any other configuration parameters for document tracking. For more information, refer to the *IDOL Server Reference*. For example:

```
[DocumentTracking]
MaxEventsPerFile=500
TimeoutSeconds=20
UIDFieldName=UID
```

6. Find the `[Logging]` configuration section.
7. Configure a new log stream for document tracking, with `LogTypeCSVs` set to `doctrack`. For more information, see "[Customize Logging](#)" on page 423.

For example:

```
[Logging]
LogLevel=FULL
LogDirectory=Logs
0=ApplicationLogStream
1=ActionLogStream
2=DocumentTrackingLogStream
...
```

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

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

Configure Event Storage

You can configure how document tracking stores the events by modifying the `[DocumentTracking]` configuration in your components. By default, document tracking stores all events, but you can use the `PositiveEventFilter` and `NegativeEventFilter` configuration parameters to reduce the number of events to store.

- `PositiveEventFilter` allows you to configure a list of events that you explicitly want to store.
- `NegativeEventFilter` allows you to configure a list of events that you do not want to store.

If you configure both parameters, `PositiveEventFilter` takes precedence. For more information, refer to the *IDOL Server Reference*.

For a full list of event types that you can use in the filter parameters, see "[Document Tracking Event Definitions](#)" below.

Document Tracking Event Definitions

The following table lists the event definitions that are used for document tracking.

Event type	Generated by	Description
Warning	General	An event that suggests some user attention might be required for this document.
Error	General	An event that suggests that further processing of the document might stop because of errors.
Committed	Content	A document is indexed and available to search.
Deleted	Content	A document has been deleted.
Indexed	Content	A document has been processed for indexing (but might not be searchable until flushed).
Updated	Content	A document has been updated by DREREPPLACE.
Rejected	Content	This document has been rejected and will definitely not be indexed.
Received	DIH	A document has been received.
Added	Connectors	A document add command has been added to the connector ingest queue.
Deleted	Connectors	A document delete command has been added to the connector ingest queue.
Replaced	Connectors	A document replace command (delete and add) has been added to the connector ingest queue.
Updated	Connectors	A document update command has been added to the connector ingest queue.
Delete received.	CFS	CFS has received a document delete command.
Update received.	CFS	CFS has received a document update command.
Non-importing add received.	CFS	CFS has received a document that does not require import (for example, because it is metadata only).
Import:Queue	CFS	A document has been placed in the import queue.
Import:Importing	CFS	A document is being imported (that is, processed by KeyView).

Import:Pre	CFS	A document is being processed by the pre-import tasks.
Import:Post	CFS	A document is being processed by the post-import tasks.
Import:Finished	CFS	A document has been successfully imported.
Import:Cancel	CFS	A document import process has been cancelled.
Import:Extracting metadata	CFS	The import process is extracting metadata from a document.
Import:Extracting metadata finished	CFS	The import process has completed extracting metadata from a document.
Import:ExtractMetaAbort	CFS	The metadata extraction process was stopped for an import.
Import:Abort	CFS	The import process was stopped.

The following event definitions are generated by the deprecated IndexTasks component:

Event type	Generated by	Description
IndexTasksStarted	IndexTasks	The start of all tasks in IndexTasks.
IndexTask	IndexTasks	A task has been started in IndexTasks. The task name is available as a custom value for this event.
IndexTasksCompleted	IndexTasks	The end of all tasks in IndexTasks.

Part III: IDOL Server Operations

This section shows how you can make best use of the many information-retrieval, analysis, classification, and management capabilities of IDOL Server.

- ["Agents"](#)
- ["Categorization"](#)
- ["Document Classification"](#)
- ["Binary Categories"](#)
- ["Agent Boolean Agents and Categories"](#)
- ["Cluster Information"](#)
- ["Profiles"](#)

Chapter 7: Agents

This section describes how to set up and use agents.

- [About Agents](#) 139
- [Manipulate Agents](#) 139
- [Query with Agents](#) 141
- [Collaboration and Expertise with Agents](#) 142

About Agents

Agents automatically find documents for you that match your interests. A user who is interested in football and gardening could, for example, create a Real Madrid agent and a Pest Control agent.

When you create an agent, you give it training text. This training provides an example of the type of text that the agent must look for, so that an agent returns only documents, profiles, categories, or other agents that conceptually match its training.

For example, you could create a Mortgage agent and train it with text that is similar to the type of results you expect the agent to return. You can train the agent with text that you type yourself, or with documents. After you train the agent and specify details for it (such as the maximum number of results the agent returns, the minimum conceptual similarity of results and so on), you can run the agent. You can edit or retrain the agent at any time to fine-tune it.

Note: By default agents match against all IDOL Server databases. However, you can restrict the matching to one or more databases.

Related Topics

- ["AgentBoolean Agents and Categories" on page 169](#)

Manipulate Agents

This section describes how to create, modify, view, and delete agents.

Tip: As an alternative to using actions to manipulate agents, you can create, modify, view, and delete agents by using the IDOL Admin interface. Refer to the *IDOL Admin User Guide* for more details.

Create an Agent

You can create agents by using the AgentAdd action. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:4000/action=AgentAdd&UserName=Administrator&AgentName=Global+Warming&Training=Factors+affecting+global+warming&FieldMinScore=60
```

This action uses port 4000 to create an agent called Global Warming for the Administrator user. The agent is stored in the IDOL Server agent index, which is on a machine with the IP address 12.3.4.56. The agent is trained to find documents whose concept matches the concept of the text Factors affecting global warming. Only documents that have a conceptual relevance of at least 60 percent to this text can return as results.

Related Topics

- ["Create AgentBoolean Agents and Categories" on page 172](#)
- ["Optimize AgentBoolean Matching" on page 174](#)

Edit an Agent

You can edit agents by using the AgentEdit action. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:4000/action=AgentEdit&UserName=Administrator&AgentName=Global+Warming&FieldMinScore=75
```

Retrain an Agent

You can retrain agents by using the AgentRetrain action. For details on this action, refer to the *IDOL Server Reference*. Retraining the agent modifies the concepts of its training with the concepts of the text that you use for retraining. For example:

```
http://12.3.4.56:4000/action=AgentRetrain&UserName=Administrator&AgentName=Global+Warming&PositiveDocs=534+352+4534
```

This action uses port 4000 to retrain the Global Warming agent for the Administrator user with the documents that have the IDs 534, 352, and 4534.

Copy an Agent

You can copy an agent by using the AgentCopy action. For details on this action, refer to the *IDOL Server Reference*. You can copy an agent to use it as a template. You can copy the agent and then modify the copy. For example:

```
http://12.3.4.56:4000/action=AgentCopy&UserName=Administrator&AgentName=Global+Warming&DestinationUserName=JSmith&DestinationAgentName=Environment
```

This action uses port 4000 to copy the Global Warming agent details from the Administrator user to the Environment agent for the user JSmith.

View Agent Details

You can view the details of an agent by using the AgentRead action. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:4000/action=AgentRead&UserName=Administrator&AgentName=Global+Warming
```

This action requests the details of the Global Warming agent for the Administrator user from IDOL Server.

Delete an Agent

You can delete an agent from the IDOL Server agent index by using the AgentDelete action. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:4000/action=AgentDelete&UserName=Administrator&AgentName=Global+Warming
```

This action deletes the Global Warming agent for the Administrator user from IDOL Server.

Related Topics

- ["Display Online Help" on page 30](#)

Query with Agents

You can query with an agent by using the AgentGetResults action. For details on this action, refer to the *IDOL Server Reference*.

Note: When you match an agent against the IDOL Server databases, all the agent terms are internally postfixed with a tilde (~) to indicate that the terms are stemmed and must not be stemmed again.

For example:

```
http://12.3.4.56:4000/action=AgentGetResults&UserName=Administrator&AgentName=Global+Warming&DREDatabaseMatch=News,Archive
```

This action uses port 4000 to request the results of the Global Warming agent for the Administrator user from IDOL Server, which is located on a machine with the IP address 12.3.4.56. It matches the Global Warming agent against the IDOL Server News and Archive databases.

Modify Document References for an Agent

You can specify a set of documents to mark as read or unread for an agent by using the AgentChangeDocsReadStatus action. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:9000/action=AgentChangeDocsReadStatus&UserName=Administrator&AgentName=Global+Warming&DocRefs=1234,5678,2953
```

This action modifies the agent called Global Warming for the Administrator user, and marks the documents with the IDs 1234, 5678, and 2953 as read, so that they do not return in the agent results.

Related Topics

- ["Display Online Help" on page 30](#)

Collaboration and Expertise with Agents

You can use agents in IDOL Server to collaborate with other users or to locate experts in your field of interest.

Collaboration

IDOL Server automatically matches users with common explicit interest agents or similar implicit profiles. You can use this information to create virtual expert knowledge groups.

You can use the `Community` action to find agents or profiles in the community that match the agents or the profiles of a specific user. For example:

```
http://IDOLhost:port  
/action=Community&UserName=JSmith&Agents=True&Profiles=True&AgentsFindProfiles=True  
&ProfilesFindAgents=True
```

This action instructs IDOL Server to find agents and profiles in the user community that match both the agents and the profiles of the user `JSmith`.

Expertise

IDOL Server accepts a natural language or Boolean search string and returns users who own matching agents or profiles. This action allows instant identification of experts in any subjects at hand, eliminating time-consuming searches for specialists, and unnecessary researching of subjects for which expert knowledge is already available

You can use the `Community` action to find agents or profiles in the community that match a natural language or Boolean search string. For example:

```
http://IDOLhost:port/action=Community&Text=how does the cost of funds, such as the  
costs of performing a credit evaluation on the business requesting a loan,  
determine the spread between the federal funds rate and the prime  
rate&AgentsFindProfiles=True&ProfilesFindAgents=True
```

This action instructs IDOL Server to find agents and profiles in the user community that match the specified text.

Chapter 8: Categorization

The IDOL Server categorization capability allows you to create and administer categories, and to use them for categorization, suggesting, sentiment analysis, and matching.

- [Introduction to Categorization](#) 143
- [Create a Hierarchical Category Structure](#) 144
- [View and Administer Categories](#) 148
- [Categorize Data](#) 152
- [Suggest Categories](#) 152
- [Match Categories](#) 153
- [Create Taxonomies](#) 153
- [Categorization Example](#) 155

Introduction to Categorization

The IDOL Server Category component automatically organizes text documents of any type into predefined categories. These sections describe how to adapt the categorization process to obtain the best possible performance.

For example, start with a data set of 1,000 news stories and a list of categories such as Sports, Politics, Entertainment, Science, and Business. The categorization process has two principal stages: training and testing. Divide the data set into training data, which might consist of 800 of the stories, and test data, which contains the remaining 200.

In the training stage, Category uses the training data to build the agents that it uses later to categorize the test data. A human expert sorts the data into the categories, by reading each news story and deciding which category it belongs to. More than one category might be appropriate for some stories. For example, you could place a story about patenting the human genome in both Science and Business. Other stories might not have an appropriate category, so they are discarded. After the training data has been sorted manually, you train the agents by running the training sections of the Categorizer.

After training, you are ready to categorize additional documents. You enter the test documents into Category, which automatically places them into the category or categories that its mathematical rules decide are most appropriate. Similar to the expert sorting the training data, Category might place a particular document in more than one category, or in no category at all.

The human expert must also sort the 200 test documents. You can then examine the performance of Category and determine how well it categorizes. If it is categorizing optimally, you can then add any future news stories into Category to categorize automatically. If required, you can add the original 200 test documents to the training set.

Related Topics

- ["AgentBoolean Agents and Categories" on page 169](#)

Create a Hierarchical Category Structure

IDOL Server provides a single category, the *root* category, which you cannot delete or modify. The root category serves as a base for the hierarchical category structure that you create. You can create categories under the root category:

- from scratch
- from clusters
- from legacy topic sets
- by copying categories
- by generating a taxonomy
- from XML

All categories are stored on disk, and become available for querying only if they are indexed into the IDOL Server category index.

After you create categories, you can:

- train the categories
- retrain the categories
- move the categories

Create Categories from Scratch

You can create categories from scratch.

To create categories from scratch

1. Create the category by using the `CategoryCreate` action. For example:

```
action=CategoryCreate&Name=Botanics&Category=1
```

From this action, IDOL Server creates the `Botanics` category with an ID of 1. The new category is a child of the `root` category, which has the ID 0. Categories that you create from scratch are by default stored as child categories of the `root` category. However, you can specify an alternative parent category when you create a category. The `Category` parameter is optional. If you do not specify an ID, the system automatically generates a random ID.

2. Create a child category.

IDOL Server returns an ID for the category that it creates. You can use this ID to identify the category, for example, to add a child category to it:

```
action=CategoryCreate&Name=Perennials&Parent=1
```

In this example, IDOL Server creates the `Perennials` category. The new category is a child of the category with the ID 1 (in this case, the `Botanics` category).

3. Train the new category. You can set training action parameters for the `CategoryCreate` action to train a category when you create it.
4. Optionally, move categories to create a hierarchical structure or to modify their position in the category hierarchy.

Related Topics

- ["Train Categories" on page 147](#)
- ["Move Categories" on page 148](#)
- ["Create AgentBoolean Agents and Categories" on page 172](#)

Create Categories from Clusters

You can use the `CategoryImportFromCluster` action to create categories from clusters that IDOL Server has previously created. It imports these categories with training that is generated from the cluster concepts.

IDOL Server stores the categories that it imports from clusters in the `root` category, unless you specify a parent category for them. For example:

```
action=CategoryImportFromCluster&SourceJobName=Job1&BuildNow=True
```

In this example, IDOL Server imports all the clusters in the `Job1` cluster source job to categories in the `root` category. The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

Related Topics

- ["Cluster Information" on page 179](#)
- ["Build Categories" on page 151](#)

Create Categories from Legacy Topic Sets

You can use the `CategoryImportFromTopic` action to import categories from existing legacy topic sets. IDOL Server creates one category for each topic set. When you import a topic set, you can specify whether you want to maintain the original Boolean rules of the topic, or import the topic as an IDOL concept matching agent.

All categories that you import from legacy topic sets are child categories of the `root` category. You can move them to create a hierarchical structure. For example:

```
action=CategoryImportFromTopic&Topic=MyTopicFile.otl&BuildNow=True
```

In this example, IDOL Server imports the topic sets that are stored in the `MyTopicFile.otl` to categories in the `root` category. The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

Related Topics

- ["Build Categories" on page 151](#)

Create Categories by Copying Categories

You can create a category by copying an existing category and retraining or editing it. IDOL Server stores the new category in the same position in the `root` category as the original category, unless you

specify a parent category for it. For example:

```
action=CategoryCopy&Category=123456789012345&Name=BotanicsCopy&Parent=98765432109876&BuildNow=True
```

In this example, IDOL Server copies the category with the ID 123456789012345. It calls the new category `BotanicsCopy` and stores it as a child category of the category with the ID 98765432109876. The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

Related Topics

- ["Build Categories" on page 151](#)

Create Categories when you Generate a Taxonomy

You can use the `TaxonomyGenerate` action to generate a taxonomy to build categories from clusters or query results. IDOL Server stores the imported categories in the root category, in a hierarchical structure that reflects the hierarchical structure of the taxonomy.

Related Topics

- ["Generate a Taxonomy from Clusters" on page 154](#)
- ["Generate a Taxonomy from Query Results" on page 155](#)

Create Categories from XML

The `CategoryImportFromXML` action allows you to create categories by importing category information from an XML file. This file can be a third-party category XML hierarchy provided that it follows the IDOL category XML format. The categories are imported with the training set in the XML file.

Related Topics

- ["Category XML Format" on page 515](#)

Create Categories from Partitions

The `CategoryPartition` action divides up a specified set of documents into several partitions. IDOL Server assigns each document to a partition, and generates a title for each partition. This option is similar to clustering, but it places all documents in a query set into partitions. For example:

```
action=CategoryPartition&DREQuery=child+benefit+cuts&NumResults=500&NumPartitions=6
```

In this example, IDOL Server retrieves all the documents that match the specified query terms up to a maximum of 500 results, splits them into six partitions, and returns the results as an ACI response.

To create categories from the partitions, add the `CreateCategories` action parameter to the `CategoryPartition` action.

For more information, refer to the *IDOL Server Reference*.

Related Topics

- ["Cluster Information" on page 179](#)

Create Categories for Sentiment Analysis

The `CategorySetupSentimentAnalysisCats` action allows you to set up and train positive, negative, and neutral sentiment categories to use to perform sentiment analysis.

After you run the `CategorySetupSentimentAnalysisCats` action, you can perform sentiment analysis without having to perform any other actions.

For example:

```
action=CategorySetupSentimentAnalysisCats&PositiveCatName=PositiveFeedback&Positive  
CatTraining=the new procedure for routing service calls has really improved our  
customer satisfaction&Parent=71056
```

In this example, IDOL Server sets up a category, `PositiveFeedback`, as a child of the category with the ID 71056, and trains that category with the specified text.

Related Topics

- ["Suggest Categories with Confidence Values" on page 153](#)

Train Categories

You can use the `CategorySetTraining` action to train a category. Category training can consist of text, documents, a Boolean expression, category content, or a combination of these. These elements identify text, documents, agents, profiles, and other categories that match the category.

For example:

```
action=CategorySetTraining&Category=323499876022105571056&DocID=238,785,9912&BuildN  
ow=True
```

In this example, IDOL Server trains the category with the ID 323499876022105571056 using the content of the documents with the IDs 238, 785, and 9912. The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

You can also train categories when you create them by assigning training action parameters to the `CategoryCreate` action.

Note: Categories that you create or import by using an action other than `CategoryCreate` are already trained. However, you can retrain them.

Related Topics

- ["Build Categories" on page 151](#)

Retrain Categories

Use the `CategorySetTraining` action to retrain a category. You can use text, documents, a Boolean expression, and category content or a combination of these to retrain a category. When you retrain a category, its original training merges with the new training. For example:

```
action=CategorySetTraining&Category=323499876022105571056&Boolean=dog AND NOT  
cat&BuildNow=True
```

In this example, IDOL Server retrains the category with the ID 323499876022105571056 using the Boolean expression `dog AND NOT cat`. The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

Related Topics

- ["Build Categories" on page 151](#)

Move Categories

You can use the `CategoryMove` action to move individual categories in the category hierarchy. For example:

```
action=CategoryMove&Category=124365780934532&Parent=12398234987345876
```

In this example, IDOL Server moves the category that has the ID 124365780934532 to the category with the ID 12398234987345876 (to make category 12398234987345876 the new parent of category 124365780934532).

View and Administer Categories

IDOL Server allows you to do these tasks to maintain your category hierarchy:

- view category details
- view category hierarchy details
- view category terms and weights
- view category training
- change category fields
- change category term weights
- replace categories
- activate categories
- build categories
- delete categories
- delete category training
- export categories to XML
- sync the IDOL Server category index with the categories stored on disk

View Category Details

Use the `CategoryGetDetails` action to view category fields. For example:

```
action=CategoryGetDetails&Category=124365780934532
```

In this example, IDOL Server returns all fields in the category with the ID 124365780934532.

View Category Hierarchy Details

Use the `CategoryGetHierDetails` action to view hierarchy details for a category. For example:

```
action=CategoryGetHierDetails&Category=124365780934532
```

In this example, IDOL Server returns the hierarchy details for the category with the ID 124365780934532.

View Category Terms and Weights

Use the `CategoryGetTNW` action to view category stemmed terms and their weights. For example:

```
action=CategoryGetTNW&Category=124365780934532
```

In this example, IDOL Server returns the terms and weights of the category with the ID 124365780934532.

View Category Training

Use the `CategoryGetTraining` action to view category training. For example:

```
action=CategoryGetTraining&Category=124365780934532
```

In this example, IDOL Server returns the training of the category with the ID 124365780934532.

Change Category Fields

Use the `CategorySetDetails` action to set the value of one or more category fields, or to create new fields in a category.

By default each category has a threshold of 0 and is set to return six results. Use the `CategorySetDetails` action fields `THRESHOLD` and `NUMRESULTS` to set the threshold and the number of results that a category can return. For example:

```
action=CategorySetDetails&Category=124365780934532&Fields=THRESHOLD,NUMRESULTS&Values=60,10&BuildNow=True
```

In this example, IDOL Server sets the `THRESHOLD` field of the category with the ID 124365780934532 to 60 and its `NUMRESULTS` to 10. The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

Related Topics

- ["Build Categories" on page 151](#)

Reset Category Fields

Use the `CategoryResetDetails` action to reset the value of the category fields to their default values.

Use the `CategoryResetDetails` action to reset the values of any of these fields: `DATABASES`, `FIELDTEXT`, `NUMRESULTS`, `THRESHOLD`, `TAXONOMYROOT`, `SIMPLECATDEFAULTCAT`, `SIMPLECATPARAM`, and `FIELD`. The action also removes values associated with the fields. For example:

```
action=CategoryResetDetails&Category=324987602&Params=NUMRESULTS,THRESHOLD
```

In this example, IDOL Server resets the `NUMRESULTS` and `THRESHOLD` fields of the category with ID 324987602 to their default values of 6 and 0.

Change Category Term Weights

You can use the `CategorySetTNW` action to change the weights of terms in the category that you believe are weighted inappropriately. For example:

```
action=CategorySetTNW&Category=124365780934532&Terms=tax,monei,budget&Weights=2353,1223,1023&BuildNow=True
```

In this example, IDOL Server sets the weight of the term `tax` to 2353, the weight of the term `monei` to 1223 and the weight of the term `budget` to 1023 (`tax`, `monei` and `budget` are what IDOL Server stems the words *Tax*, *Money* and *Budget* to). The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

Related Topics

- ["Build Categories" on the next page](#)

Remove Category Term Weights

You can use the `CategorySetTNW` action to remove any modifications that you made to the weights of terms in a category. For example:

```
action=CategorySetTNW&Category=124365780934532&BuildNow=True
```

In this example, IDOL Server removes all previous changes to the weights of all terms in the category with the ID 124365780934532. The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

Related Topics

- ["Build Categories" on the next page](#)

Replace Categories

You can use the `CategoryReplace` action to replace a category with another category. For example:

```
action=CategoryReplace&FromCategory=123456789012345&ToCategory=98765432109876&BuildNow=True
```

In this example, IDOL Server replaces the 98765432109876 category with the 123456789012345 category. The `BuildNow` parameter instructs IDOL Server to build the categories immediately, so that they become active. You can also activate the category at a later point by using the `CategoryBuild` action.

Related Topics

- ["Build Categories" below](#)

Activate or Deactivate Categories

You can use the `CategoryActivate` action to activate or deactivate a category. You cannot query inactive categories or return them as results. By default IDOL Server activates a category when it builds it. For example:

```
action=CategoryActivate&Category=3234998760221&Active=True
```

In this example, IDOL Server activates the category with the ID 3234998760221.

Use the `CategoryGetHierDetails` action to find out whether categories are active.

Related Topics

- ["View Category Hierarchy Details" on page 149](#)

Build Categories

You can use the `CategoryBuild` action to build a category. You must build a category after you create it and train it, as well as every time that you retrain it. Building a category identifies the concepts of the training for the category and indexes the category into the IDOL Server category index. For example:

```
action=CategoryBuild&Category=32349987602210557106
```

In this example, IDOL Server builds the category with the ID 32349987602210557106.

Note: If you train or retrain a category by using the `CategorySetTraining` action with `BuildNow` set to `True`, you do not have to run a `CategoryBuild` action, because the category was built immediately after it was trained.

IDOL Server also builds categories immediately if you assign training parameters to the `CategoryCreate` action to train a category when you create it.

Delete Categories

You can use the `CategoryDelete` action to delete a category. Deleting a category removes the category from disk and from the IDOL Server category index. For example:

```
action=CategoryDelete&Category=32349987602210557106
```

In this example, IDOL Server deletes the category with the ID 32349987602210557106.

Delete Category Training

You can use the `CategoryDeleteTraining` action to delete all or part of the training for a category. Deleting a category removes the training from disk and from the IDOL Server category index. For example:

```
action=CategoryDeleteTraining&Category=32349987602210557106
```

In this example, IDOL Server deletes the training for the category with the ID 32349987602210557106.

Export Categories to XML

You can use the `CategoryExportToXML` action to export a category to XML format, including its descendants, training documents, and terms and weights. For example:

```
action=CategoryExportToXML
```

In this example, IDOL Server exports the entire category structure to XML.

Synchronize IDOL Server with Stored Categories

You can use the `CategorySyncCatDRE` action to synchronize the IDOL Server category index with the categories stored on disk. `CategorySyncCatDRE` deletes the current contents of the category index, and overwrites it with the category information stored on disk. For example:

```
action=CategorySyncCatDRE
```

In this example, IDOL Server synchronizes its category index with the categories stored on disk.

Categorize Data

You can configure IDOL Server to automatically categorize data and index it.

To automatically categorize documents before storing them in IDOL Server, set up a preprocessing task in Connector Framework Server (CFS). IDOL Server matches incoming documents against categories that its category index contains and returns matching categories. CFS then tags the incoming documents according to the categories that they match.

For more information, refer to the *Connector Framework Server Administration Guide*.

Related Topics

- ["Perform AgentBoolean Queries" on page 172](#)

Suggest Categories

IDOL Server can suggest conceptually similar categories for:

- documents
- text
- categories

Suggest Categories for Documents

You can use the `CategorySuggestFromDocument` action to suggest categories from the IDOL Server category index that are conceptually similar to a specified document. For example:

```
action=CategorySuggestFromDocument&DocID=125
```

In this example, IDOL Server returns categories that are conceptually similar to the document with the ID 125.

Suggest Categories for Text

You can use the `CategorySuggestFromText` action to suggest categories from the IDOL Server category index that are conceptually similar to specified text. For example:

```
action=CategorySuggestFromText&QueryText=Caring for passiflora incarnata
```

In this example, IDOL Server returns categories that are conceptually similar to the text `Caring for passiflora incarnata`.

Suggest Categories for Categories

You can use the `CategorySuggestFromCategory` action to suggest categories from the IDOL Server category index that are conceptually similar to a specified category. For example:

```
action=CategorySuggestFromCategory&Category=32349987602210557106
```

In this example, IDOL Server returns categories that are conceptually similar to the category with the ID `3234998760221055`.

Suggest Categories with Confidence Values

You can use the `CategorySimpleCategorize` action to suggest categories from the IDOL Server category index that are conceptually similar to a specified document, piece of text, or file content, and to return terms and weights for each category.

You can also specify cluster job names and cluster numbers to import into categories when you use the `CategorySimpleCategorize` action. This enables you to perform sentiment analysis on clusters.

Related Topics

- ["Create Categories for Sentiment Analysis" on page 147](#)

Match Categories

You can use the `CategoryQuery` action to match categories against data, agents, profiles, and other categories. For example:

```
action=CategoryQuery&Category=32349987602210557106
```

In this example, IDOL Server matches the category with the ID `32349987602210557106` against all its databases and returns conceptually similar data, agents, profiles, and categories.

Create Taxonomies

The IDOL Server taxonomy generation feature allows you to automatically create hierarchical contextual taxonomies of clusters or other information. It provides you with an overview of the

information landscape and an insight into specific areas of the information.

You can also create a taxonomy manually and name it by the category that is its root node.

Generate Taxonomies Automatically

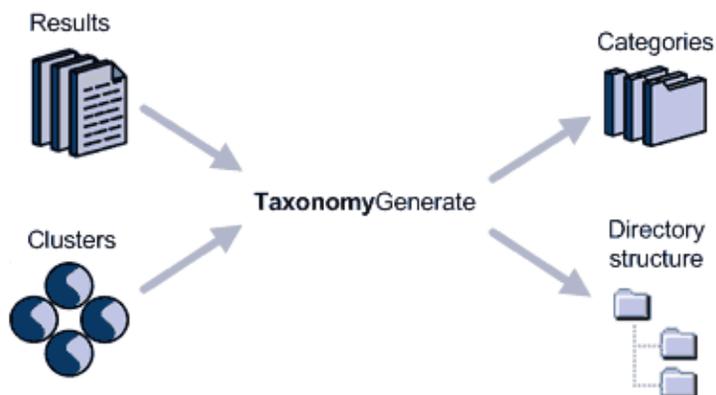
The `TaxonomyGenerate` action allows you to generate a hierarchical taxonomy from one or more clusters or query results.

The taxonomy generator adapts the Bayesian and information theoretic methods to concept selection. It applies Bayesian algorithms to identify statistical relationships between concepts and sets of concepts (at the document and document set level). It then filters them to form the hierarchical structure of the final taxonomy.

You can write the taxonomy to disk as a directory structure, or import the taxonomy into the category hierarchy.

Note: Before you create a taxonomy from an IDOL Server, ensure that IDOL Server does not contain duplicate documents or text that is repeated in multiple documents (for example, document headers). Ensure that these are stripped out at the import stage to gain optimal results.

You can set up a schedule that runs the `TaxonomyGenerate` action at regular intervals.



Related Topics

- ["Cluster Information" on page 179](#)

Generate a Taxonomy from Clusters

Use the `TaxonomyGenerate` action with the `SourceJobName` and `Cluster` parameters to generate a taxonomy from one or more clusters. For example:

```
action=TaxonomyGenerate&SourceJobName=Taxonomy1&Cluster=0,1
```

In this example, IDOL Server generates a taxonomy from the **Taxonomy1** cluster.

Generate a Taxonomy from Query Results

Use the `TaxonomyGenerate` action with the `DREQuery` parameter to generate a taxonomy from a query. For example:

```
action=TaxonomyGenerate&DREQuery=new+tax+cuts
```

In this example, IDOL Server generates a taxonomy from the results that it returns from its data index for the query `new tax cuts`.

Schedule Taxonomy Generation

You can set up a schedule to run the `TaxonomyGenerate` action at regular intervals.

Related Topics

- ["Set up Schedules" on page 187](#)

Create Named Taxonomies

You can take advantage of the hierarchical nature of IDOL Server categorization to manually create and store named taxonomies in IDOL Server. You can then use those taxonomies to investigate searches or suggestions.

If you set up a hierarchy of categories using the `CategoryCreate` or `CategoryImportFromXML` actions, you can then run `CategorySetDetails` for the topmost category with `TaxonomyRoot` set to `True`. That category becomes the root of the new taxonomy and its name is the new taxonomy name.

You can also generate a named taxonomy automatically by running the `TaxonomyGenerate` action and setting `TaxonomyRoot` to `True`.

You can subsequently use the taxonomy name as a parameter in actions such as `CategoryFind` and `CategorySuggestFromDocument`, to restrict the scope of the action results.

Categorization Example

This section contains an example of how you might set up categorization. Each step provides several examples of possible actions for that step.

To use Categorization

1. Create a few categories.

```
action=CategoryCreate&Name=Animals&Category=10
```

```
action=CategoryCreate&Name=Cats&Parent=10&Category=11
```

```
action=CategoryCreate&Name=Dogs&Parent=10&Category=12
```

2. Train the new categories (from a selection of indexed documents, free text, and Boolean rules).

```
action=CategorySetTraining&Category=10&DocRef=http://foo.com/animals&DocID=109,  
178
```

```
action=CategorySetTraining&Category=11&Training=Cats and kittens can be a  
variety of colours, such as tabby or tortoiseshell
```

```
action=CategorySetTraining&Category=12&Boolean=dog OR hound
```

3. Build the categories (this processes the training into terms and weights, enabling CategoryQuery and CategorySuggest).

```
action=CategoryBuild&Category=10&Recurse=True
```

4. Manually adjust the terms and weights for one of the categories, and rebuild it.

```
action=CategorySetTnW&Category=11&Terms=cat,kitten&Weights=500,400
```

```
action=CategoryBuild&Category=11
```

5. Query for documents matching a particular category, with optional parameters.

```
action=CategoryQuery&Category=11
```

```
action=CategoryQuery&Category=12&DatabaseMatch=News&NumResults=10
```

6. Categorize already-indexed documents or free text.

```
action=categorySuggestfromDocument&DocID=1465
```

```
action=CategorySuggestFromText&QueryText=the cat in the hat
```

Chapter 9: Document Classification

IDOL Server document classification allows you to create classifiers, which use the random forest algorithm to classify documents into a set of classes that you define.

- [Introduction to Document Classification](#)157
- [Use Document Classification](#) 157
- [View and Administer Classifiers](#) 162

Introduction to Document Classification

Document classification allows you to automatically assign documents to classes according to values that occur in a set of fields that you specify. Classification uses the random forest algorithm, and you can use it as an alternative to the IDOL Server conceptual categorization (see "[Categorization](#)" on [page 143](#)).

Classification works by analyzing the contents of various feature fields in the documents. You choose feature fields that contain useful information for classifying the documents. Typically, feature fields contain small snippets of information, rather than large portions of text. For example, you might use a name, or color field, rather than the document content. The feature fields that you use depend on the classifier and classes that you want to create.

To use classification, you create one or more *classifiers*. A classifier contains a set of *classes* (similar to categories), which represent the topics that you want to assign documents to. You train each class with a set of documents that represent the kind of content that you want the class to match.

After you create and train a classifier, you can query the classifier with new documents, and the classifier returns the details of the class that each new document belongs to.

Use Document Classification

The following sections describe how to create and train a classifier, and query the classifier with new documents.

For more information about the classification actions, refer to the *IDOL Server Reference*.

Choose Feature Fields

Before you create a classifier, you must choose the fields in your documents that you want to use to classify documents. These are the feature fields for the classifier.

Feature fields generally contain short pieces of information, such as a name or a very brief description. A good choice of feature field is similar to a good choice of `ParametricType` field. For example, if you want to create a food classifier, you might use a field that stores ingredients, or a meal name, rather than a field that contains a recipe procedure or a detailed description of a type of food.

The feature fields must contain information that describes features of the different classes that you want to create for your classifier. For example, to classify meals as vegetarian or meat-based, you must find feature fields that describe features of vegetarian or meat-based meals.

The exact choice of feature field also depends on the contents of your documents.

For example, the following IDX document describes part of a recipe for soup:

```
#DREFERENCE Food/Carrot and Coriander Soup
#DRETITLE Carrot and Coriander Soup
#DRESECTION 0
#DREFIELD Ingredient="carrots"
#DREFIELD Ingredient="onion"
#DREFIELD Ingredient="potato"
#DREFIELD Herbs="coriander"
#DREFIELD Seasoning="vegetable stock"
#DREFIELD Meal="soup"
#DREFIELD Equipment="food processor"
#DREFIELD PreparationTime="20 minutes"
#DREFIELD CookingTime="1 hour"
#DREFIELD Description"This easy recipe makes a tasty carrot and coriander soup"
#DRECONTENT
Example soup recipe
#DREENDDOC
```

- For a classifier that distinguishes between vegetarian meals and meat-based meals, you might choose the `Ingredient` field.
- For a classifier that distinguishes between savory recipes and desserts, you might choose the `Meal` and `Ingredient` fields.
- For a classifier that distinguishes between quick recipes and meals that take longer, you might choose the `PreparationTime` and `CookingTime` fields.

You can choose more than one feature field for a classifier. The classifier does not distinguish between data from different feature fields. It extracts the content from all the available feature fields from a document, and uses all the content to train the classifier (or classify a document).

For example, if your document had the fields:

```
#DREFIELD Ingredient1="carrots"
#DREFIELD Ingredient2="onion"
#DREFIELD Ingredient3="potato"
```

You can set `Ingredient1`, `Ingredient2`, and `Ingredient3` as feature fields. If you use this document for classification, it gives the same results as if you used a document with the following fields:

```
#DREFIELD Ingredient1="onion"
#DREFIELD Ingredient2="potato"
#DREFIELD Ingredient3="carrots"
```

Create a Classifier

You create a classifier with a unique name and a set of feature fields.

To create a classifier

- Send a `ClassifierCreate` action to IDOL Server, with the following parameters:
 - `ClassifierName` set to the name of the new classifier. This name must be unique in IDOL Server.
 - `ClassifierType` set to `RandomForest`.
 - `FeatureFields` set to a comma-separated list of the feature fields that you want to use for the classifier.

For example:

```
action=ClassifierCreate&ClassifierName=food&FeatureFields=Ingredient,Herbs,Seasoning
```

This action creates a `food` classifier, which uses the `Ingredient`, `Herbs`, and `Seasoning` fields to classify documents.

Create and Train Classes

After you create the classifier, you create and assign training to the classes. You can either create the classes and assign training in a single action, or you can create the classes and train them later.

The documents that you use to train the class must exist in the IDOL Server data index. You provide training in the form of a state token, which you create by using the `Query` action with the `StoreState` parameter set to `True`. See ["Choose Training Documents for Classes" on the next page](#).

To create a class

- Send a `ClassifierAddClass` action to IDOL Server, with the following parameters:
 - `ClassifierName` set to the name of the classifier.
 - `ClassName` set to the name of the new class.
 - (Optional) `StateID` set to a state token that lists the documents that you want to use to train the class.

For example:

```
action=ClassifierAddClass&ClassifierName=food&ClassName=vegetarian&StateID=B8UGIK95FKJG-23
```

This action creates a `vegetarian` class in the `food` classifier. It assigns the documents from the state token `B8UGIK95FKJG-23` as training for the new class.

If you do not train the class when you create it, you can add training by using the `ClassifierSetClassTraining` action. You can also use this action to retrain a class. For more information, see ["Retrain a Class" on page 162](#).

You must run the `ClassifierAddClass` action for each class that you want to create in the classifier.

Choose Training Documents for Classes

When you create a classifier, you must train each of the classes with content that represents the classes that you want to define. The content must exist in your IDOL Server data index, and the content must contain the feature fields that you have defined for the classifier.

You provide training to the classes as a state token. You create state tokens by sending the `Query` action with the `StoreState` parameter set to `True`. Therefore, to train a class, you must have a single query that returns the documents that define that class.

For some classifications, you might be able to perform a complex query that returns enough documents to train your classifier. However, the best way to find training is usually to manually categorize a set of documents, and add a field that labels the document with its class. You can then use a simple `FieldText` query to find all documents with a particular label.

For example, if you label a set of documents with a `MealType` field, with a value of *savory* or *dessert*, you can use the following query to find and save the results to use as training for the *savory* class:

```
action=Query&FieldText=MATCH{savory}:MealType&MaxResults=1000&StoreState=True
```

You can use the resulting state token that this query returns to train the class. You can also create similar queries to train your other classes.

After you have trained the classifier, you can classify any new documents, and automatically add the label field to those documents.

Note: To get the best results out of your classifiers, use as many training documents as possible. HPE recommends that you use a minimum of 200 to 300 training documents for each class.

Train the Classifier

You must train the classifier before you can use it to classify documents. During this stage, IDOL Server retrieves all the training documents from the index, and extracts the feature fields. It uses the content to train each class in the classifier.

For IDOL Server to successfully train the classifier, it must have at least two classes, each of which must have training assigned.

Note: When IDOL Server trains the classifier, it ignores any very rare features.

To train a classifier

- Send a `ClassifierTrain` action to IDOL Server, with the `ClassifierName` parameter set to the name of the classifier.

For example:

```
action=ClassifierTrain&ClassifierName=food
```

This action trains the `food` classifier.

Note: The action returns an error if IDOL Server could not extract any features from the training

documents (for example, because none of the training documents contained the feature fields for the classifier).

Classify Documents

You can use a trained classifier to classify documents, by using the `ClassifierQuery` action.

The document can either be:

- the document reference for a document that exists in the IDOL Server index.
- a percent-encoded IDX or XML document.

In both cases, IDOL Server extracts the classifier feature fields from the query document, and compares the values in these feature fields against the trained classes in the classifier. The action returns the class that the document matches most closely.

To classify a document that exists in the index

- Send the `ClassifierQuery` action with the following parameters.
 - `ClassifierName` set to the name of the classifier to use to classify the document.
 - `DocRef` set to the IDOL Server reference of the document to classify.

For example:

```
action=ClassifierQuery&ClassifierName=food&DocRef=http://www.example.com/documents/carrots
```

To classify a document that does not exist in the index

- Send the `ClassifierQuery` action with the following parameters.
 - `ClassifierName` set to the name of the classifier to use to classify the document.
 - `QueryText` set to the percent-encoded IDX or XML document (IDOL Server detects the correct format automatically).

For example:

```
action=ClassifierQuery&ClassifierName=food&QueryText=%23DREFERENCE%20Food%20Leek%20and%20Potato%20Pie%0D%0A%23DRETITLE%20Leek%20and%20Potato%20Pie%0D%0A%23DRESECTION%20%0D%0A%23DREFIELD%20Ingredient%3D%22leeks%22%0D%0A%23DREFIELD%20Ingredient%3D%22potatoes%22%0D%0A%23DREFIELD%20Ingredient%3D%22cheese%22%0D%0A%23DREFIELD%20Ingredient%3D%22pastry%22%0D%0A%23DREFIELD%20Ingredient%3D%22butter%22%0D%0A%23DREFIELD%20Ingredient%3D%22egg%22%0D%0A%23DREFIELD%20Herbs%3D%22rosemary%22%0D%0A%23DREFIELD%20Herbs%3D%22thyme%22%0D%0A%23DREFIELD%20Meal%3D%22pie%22%0D%0A%23DREFIELD%20Equipment%3D%22pie%20dish%22%0D%0A%23DREFIELD%20PreparationTime%3D%2210%20minutes%22%0D%0A%23DREFIELD%20CookingTime%3D%221%20hour%22%0D%0A%23DREFIELD%20Description%22This%20easy%20recipe%20makes%20a%20tasty%20leek%20and%20potato%20pie%22%0D%0A%23DRECONTENT%0D%0A%23DREENDDOC
```

This action classifies the following document:

```
#DREREERENCE Food/Leek and Potato Pie
#DRETITLE Leek and Potato Pie
#DRESECTION 0
#DREFIELD Ingredient="leek"
#DREFIELD Ingredient="potato"
#DREFIELD Ingredient="cheese"
#DREFIELD Ingredient="shortcrust pastry"
#DREFIELD Ingredient="butter"
#DREFIELD Ingredient="egg"
#DREFIELD Herbs="rosemary"
#DREFIELD Herbs="thyme"
#DREFIELD Meal="pie"
#DREFIELD Equipment="pie dish"
#DREFIELD PreparationTime="10 minutes"
#DREFIELD CookingTime="1 hour"
#DREFIELD Description"This easy recipe makes a tasty leek and potato pie"
#DRECONTENT
Pie recipe
#DREENDDOC
```

View and Administer Classifiers

After you have set up classification, you can list and view classifiers, retrain classes, and delete classes and classifiers.

List and View Classifiers

The `ClassifierList` and `ClassifierGetInfo` actions allow you to view information about the classifiers you have created. `ClassifierList` returns information for all classifiers, and `ClassifierGetInfo` returns information for a single classifier that you specify. Both actions return the number of classes in the classifier, the feature fields that the classifier uses, and whether the classifier has been trained.

For example:

```
action=ClassifierList
```

This action returns the names of all your classifiers, and the information for each classifier.

```
action=ClassifierGetInfo&ClassifierName=food
```

This action returns information for the `food` classifier.

Retrain a Class

You can change the training documents associated with a class in a classifier by using the `ClassifierSetClassTraining` action. This action overwrites any existing training for the class with the new training. After you retrain a class, you must retrain the classifier.

To retrain a classifier

1. Send the `ClassifierSetClassTraining` action with the following parameters:
 - `ClassifierName` set to the name of the classifier.
 - `ClassName` set to the name of the class to retrain.
 - `StateID` set to a state token that lists the documents that you want to use to train the class.

For example:

```
action=ClassifierSetClassTraining&ClassifierName=food&ClassName=vegetarian&StateID=G7KPID13APWM-15
```

This action updates the training for the `vegetarian` class in the `food` classifier to use the documents listed in the state token `G7KPID13APWM-15`.

2. Send a `ClassifierTrain` action to IDOL Server, with the `ClassifierName` parameter set to the name of the classifier.

For example:

```
action=ClassifierTrain&ClassifierName=food
```

This action trains the `food` classifier, and updates the training for the retrained classes.

Delete a Class

You can delete a class from a classifier by using the `ClassifierDeleteClass` action. After you send this action, IDOL Server automatically retrains the classifier.

For example:

```
action=ClassifierDeleteClass&ClassifierName=food&ClassName=vegetarian
```

This action deletes the `vegetarian` class from the `food` classifier and retrains the classifier.

Delete a Classifier

You can delete a classifier that you no longer need by using the `ClassifierDelete` action. This action deletes the classifier and all associated classes.

For example:

```
action=ClassifierDelete&ClassifierName=food
```

This action deletes the `food` classifier.

Chapter 10: Binary Categories

This section describes how to set up and use binary categories.

- [About Binary Categories](#) 165
- [Create and Administer Binary Categories](#) 165
- [Query with Binary Categories](#) 167
- [Binary Category Example](#) 167

About Binary Categories

A binary category is a special kind of category, designed to answer yes/no questions about documents, files, and text, like *Is this document spam?*, *Does this violate company policy?*, or *Is this work-related?*.

After you create a binary category, you train it. Unlike regular categories, which receive only positive training, binary categories can receive both positive and negative training. You provide the binary category with documents or text that result in a yes (POSITIVE) answer when querying with the binary category. You also provide documents or text that result in a no (NEGATIVE) answer to the same query.

After training, the binary category determines whether a piece of text, a file on disk, or a document in IDOL results in a *POSITIVE* or a *NEGATIVE* result. IDOL also provides a score (0–1) for the result, which indicates the confidence.

Create and Administer Binary Categories

The following section describes how to create, train, delete, change, and view binary categories.

Related Topics

- ["Display Online Help" on page 30](#)

Create a Binary Category

You can use the `BinaryCatCreate` action to create binary categories. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:9000/action=BinaryCatCreate&Name=spam_binarycat
```

This action uses port 9000 to instruct IDOL Server, which is located on a machine with the IP address 12.3.4.56, to create a new binary category named `spam_binarycat`.

Train a Binary Category

You can use the `BinaryCatTrain` action to train a binary category. Unlike normal categories, which have only positive training, binary categories can have both positive and negative training.

If the binary category has existing training, `BinaryCatTrain` adds the new training to it. If you want to replace the training for a binary category, you must first use the `BinaryCatDeleteTraining` action to delete the existing training. For example:

```
http://12.3.4.56:9000/action=BinaryCatTrain&Name=spam_
binarycat&PositiveDocID=123,456&NegativeDocID=789,890
```

This action uses port 9000 to instruct IDOL Server, which is located on a machine with the IP address 12.3.4.56, to train the binary category named `spam_binarycat`. It uses the documents with IDs 123 and 456 for positive training, and the documents with IDs 789 and 890 for negative training.

Delete Training From a Binary Category

You can use the `BinaryCatDeleteTraining` action to remove the existing training from a binary category. Using the `BinaryCatTrain` action on a binary category with existing training adds the new training to the existing training, unless you use the `BinaryCatDeleteTraining` action first. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:9000/action=BinaryCatDeleteTraining&Name=spam_binarycat
```

This action uses port 9000 to instruct IDOL Server, which is located on a machine with the IP address 12.3.4.56, to delete the training of the binary category named `spam_binarycat`.

Change Binary Category Details

You can use the `BinaryCatSetDetails` action to change the fields for a binary category from their default values. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:9000/action=BinaryCatSetDetails&Name=spam_
binarycat&MinDocOccs=12&TestTermsPerDoc=20
```

This action uses port 9000 to instruct IDOL Server, which is located on a machine with the IP address 12.3.4.56, to change the value of the parameter `MinDocOccs` to 12, and the value of the parameter `TestTermsPerDoc` to 20, for the binary category named `spam_binarycat`.

View Binary Category Details

You can use the `BinaryCatGetDetails` action to view the details of a binary category. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:9000/action=BinaryCatGetDetails&Name=spam_binarycat
```

This action uses port 9000 to instruct IDOL Server, which is located on a machine with the IP address 12.3.4.56, to display parameter values of the binary category named `spam_binarycat`.

List Binary Categories

You can use the `BinaryCatList` action to view a list of all the binary categories in the system. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:9000/action=BinaryCatList
```

This action uses port 9000 to instruct IDOL Server, which is located on a machine with the IP address 12.3.4.56, to list all the binary categories in the system.

Delete a Binary Category

You can use the `BinaryCatDelete` action to delete a binary category from IDOL Server. For details on this action, refer to the *IDOL Server Reference*. For example:

```
http://12.3.4.56:9000/action=BinaryCatDelete&Name=spam_binarycat
```

This action uses port 9000 to instruct IDOL Server, which is located on a machine with the IP address 12.3.4.56, to delete the binary category named `spam_binarycat`.

Query with Binary Categories

After you train the binary category, you can use the `BinaryCatQuery` action to determine whether a piece of text, a file on disk, or a document in IDOL results in a “yes” (POSITIVE) or a “no” (NEGATIVE) answer for the question that the binary category asks. For example, your binary category might ask the question “Is this spam?” and could filter emails to ignore spam emails. For details on the `BinaryCatQuery` action, refer to the *IDOL Server Reference*.

For example:

```
http://12.3.4.56:9000/action=BinaryCatQuery&Name=spam_binarycat&QueryText=How to become an instant millionaire
```

This action uses port 9000 to instruct IDOL Server, which is located on a machine with the IP address 12.3.4.56, to check whether the text `How to become an instant millionaire` results in a “POSITIVE” or a “NEGATIVE” result for the question posed by the binary category `spam_binarycat`.

Related Topics

- ["Display Online Help" on page 30](#)

Binary Category Example

This section contains a step-by-step scenario for how you might use a binary category. After the initial creation of the binary category, each step provides more than one example of possible actions.

1. Create a binary category.

```
action=BinaryCatCreate&Name=spam_binarycat
```

2. Train the new binary category (from a selection of indexed documents, free text, and files).

```
action=BinaryCatTrain&Name=spam_  
binarycat&PositiveDocID=123,456&NegativeDocID=789,890
```

```
action=BinaryCatCreate&Name=spam_binarycat&PositiveTraining=Get rich quick,  
join today&NegativeTraining=We should discuss the Jones file
```

```
action=BinaryCatCreate&Name=spam_  
binarycat&PositiveDirectory=C:\spampositive&NegativeDirectory=C:\spamnegative
```

3. Query some data using the binary category.

```
action=BinaryCatQuery&Name=spam_binarycat&QueryFile=C:\unknown_email.txt
```

```
action=BinaryCatQuery&Name=spam_binarycat&QueryText=Limited space available,  
apply today
```

The following sample shows a POSITIVE result, with a score of 0.99664 from a binary category query:

```
<autnresponse xmlns:autn=""http://schemas.autonomy.com/aci/"">  
  <action>  
    BINARYCATQUERY  
  </action>  
  <response>  
    SUCCESS  
  </response>  
  <responsedata>  
    <autn:queryresult>  
      <autn:result>  
        POSITIVE  
      </autn:result>  
      <autn:score>  
        0.99664  
      </autn:score>  
    </autn:queryresult>  
  </responsedata>  
</autnresponse>
```

Chapter 11: AgentBoolean Agents and Categories

This section describes how to set up and use AgentBoolean agents and categories in IDOL Server.

- [AgentBoolean Agents and Categories](#) 169
- [Configure IDOL Server for Text Parse Queries](#) 171
- [Create AgentBoolean Agents and Categories](#) 172
- [Perform AgentBoolean Queries](#) 172
- [Optimize AgentBoolean Matching](#) 174

AgentBoolean Agents and Categories

You can create agents and categories that use keywords, conceptual information, a Boolean or proximity expression, or a FieldText expression to match documents. The fields in the agents or categories that contain these expressions are called *AgentBoolean* fields.

The following sections describe how to set up and use AgentBoolean agents and categories.

To use AgentBoolean and FieldText fields

1. Configure the fields that IDOL Server uses to match against AgentBoolean agents and categories.
2. Create agents and categories that contain Boolean or FieldText matching expressions.
3. Send queries to IDOL Server to match against the categories and agents.

After you set up AgentBoolean matching, you can optimize the system to provide the most efficient matching.

Related Topics

- ["Configure IDOL Server for Text Parse Queries" on page 171](#)
- ["Create AgentBoolean Agents and Categories" on page 172](#)
- ["Perform AgentBoolean Queries" on page 172](#)
- ["Optimize AgentBoolean Matching" on page 174](#)

Examples

IDOL Server stores agents and categories in the Agentstore component in the same way as IDOL Server stores documents. You can create agents and categories by using IDOL Server actions, or you can index an IDX or XML agent or category into the Agentstore component. For example:

```
#DRREFERENCE 947344A0
#DRETITLE
My Cat and Dog Agent
#DREFIELD MyABfield="cat AND dog"
```

```
#DREFIELD FieldTextField="MATCH{cat}:Animal"  
#DRECONTENT  
cat  
#DREENDDOC
```

Similarly, you can search for agents and categories in the Agentstore component in the same way that you search for documents in IDOL Server.

For example, you can find agents and categories that match a document. This process allows you to categorize documents, or alert users when a new document matches their agent.

In this case, AgentBoolean expressions can improve the performance and accuracy for matching documents. It also provides extra functionality that you cannot easily achieve with simple conceptual agents.

Match Specific Concepts

If you have an Apollo category, it matches documents that contain the concepts *Apollo space program* and *Greek god Apollo*. You can use a Boolean expression to specifically restrict results to one or other of the concepts. For example:

```
"Space Program" NOT "Greek god"
```

Use Field Restrictions

You can use field restrictions in your AgentBoolean expressions, to match only the most relevant documents. For example:

```
"New Zealand":COUNTRY AND wine.
```

This expression matches documents that contain the phrase *New Zealand* in the *COUNTRY* field, and contain the term *wine*.

Related Topics

- ["Simple Field Restricted Search" on page 221](#)

Use Term Occurrence Restrictions

You can use term occurrence restrictions in your agents to ensure that only the most relevant documents return. For example:

```
"Gene Therapy"[10:]
```

This expression matches documents in which the phrase *Gene Therapy* occurs ten or more times.

Categorize Documents before Indexing

You can use the Connector Framework Server (CFS) to match documents against categories before you index them, and to tag the document with the appropriate category. You can use AgentBoolean categories for more specific categorization. This method speeds up future searches for documents that match a category, because the document is already tagged.

You can also use this method to prevent documents from being indexed, based on the category data. For example, you can automatically prevent IDOL Server from indexing a document that contains sensitive or restricted material.

Related Topics

- ["Categorization" on page 143](#)

Alert Users to Documents that Match Their Agents

Connector Framework Server (CFS) allows you to alert users to documents that match their agents before you index the documents. In this case, CFS can send a `TextParse` query to IDOL Server to find all agents that match the document, and then email the users who own those agents.

`TextParse` queries allow you to send a whole IDX or XML document to IDOL Server in a query. IDOL Server extracts fields that you configure as `TextParseIndexType` from the document and uses the contents of these fields as the query text.

AgentBoolean rules improve the speed and accuracy of this agent matching procedure.

Related Topics

- ["Agents" on page 139](#)

Configure IDOL Server for Text Parse Queries

IDOL Server can categorize documents or alert users to new content that matches their agents before it indexes the documents. This process includes matching against AgentBoolean rules.

You can configure your pre-indexing tasks to send the percent-encoded IDX or XML document as query text, with the `TextParse` action parameter set to `True`.

When you use an IDX or an XML document as query text, you must configure as `TextParseIndexType` all the fields that you use to match agents and categories.

IDOL Server stores agents and categories in the Agentstore component. The Agentstore component has its own configuration file, which by default is stored in the following location:

```
installDir\agentstore
```

To configure `TextParse` fields

1. Open the IDX or XML document that you want to match against the AgentBoolean categories in a text editor.
2. Decide which fields in the document contains the content that you want to match against the AgentBoolean categories (for example, the `DRECONTENT` and the `DRETITLE` field).
3. Open the Agentstore configuration file in a text editor.
4. In the `[FieldProcessing]` section, add a new process to the list of processes. This process identifies the fields in the IDX or XML document that you want to match against the AgentBoolean categories. For example:

```
[FieldProcessing]
0=SetIndexFields
```

```
...  
15=SetAgentBooleanFields
```

5. Create a new section for the process that you added. In this section, create a property for the process (you define the property later, by setting one or more applicable configuration parameters). Set `PropertyFieldCSVs` to a list of fields to associate with the process. If you are not sure which fields to use, type `/*` to use all fields. For example:

```
[SetAgentBooleanFields]  
Property=AgentBooleanFields  
PropertyFieldCSVs=*/DRECONTENT,*DRETITLE
```

6. Create a new section for the property that you added, in which you set `TextParseIndexType` to `True`. This property indicates that IDOL Server must use the associated fields as query text in `TextParse` queries. For example:

```
[AgentBooleanFields]  
TextParseIndexType=True
```

7. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Related Topics

- ["Fields" on page 73](#)

Create AgentBoolean Agents and Categories

You can create `AgentBoolean` or `FieldText` agents and categories in IDOL Server in the same way as you create other agents. For example, you can send an `AgentAdd` action, and add the `BooleanRestriction` or `FieldTextRestriction` parameter.

Alternatively, you can manually create an `IDX` document that contains agents or categories, and index it into the `Agentstore` component. For example:

```
#DREREFERENCE 947344A0  
DRETITLE  
Cat  
#DREFIELD MyABField="(cat AND mat) AND ("furry kitten")"  
#DREFIELD FieldTextField="MATCH{cat}:Animal"  
#DRECONTENT  
cat  
#DREENDDOC
```

Related Topics

- ["Manually Create IDX Files " on page 511](#)

Perform AgentBoolean Queries

After you create or index `AgentBoolean` categories and agents, you can start querying them.

To match documents that already exist in IDOL Server against the categories and agents, use the following actions:

- `AgentGetResults`. Returns results for a particular agent.
- `CategorySuggestFromDocument`. Returns categories that match a particular document in IDOL Server.

To match IDX or XML documents that do not exist in IDOL Server, you can use the following actions with the `TextParse` parameter:

- `CategorySuggestOnText`. Returns categories that match the text you provide.
- `Query`. Returns documents that match the text you provide.

Note: You must send the `Query` action to the `Agentstore` component to return agents or categories.

To perform a `TextParse` query

1. Percent-encode the content of the IDX or XML document that you want to match against the `AgentBoolean` agents or categories. For example:

```
#DREREFERENCE http://www.catdog.com
#DRETITLE Cats and Dogs
#DREFIELD Animal10="dog"
#DREFIELD Animal11="cat"
#DRECONTENT
The organisation takes care of homeless cats and dogs
#DREENDDOC
```

Percent-encoding turns this IDX into this string:

```
%23DREREFERENCE%20http%3A%2F%2Fwww%2Ecatdog%2Ecom%0D%0A
%23DRETITLE%20Cats%20and%20Dogs%0D%0A
%23DREFIELD%20Animal10%3D%22dog%22%0D%0A
%23DREFIELD%20Animal11%3D%22cat%22%0D%0A
%23DRECONTENT%0D%0A
The%20organisation%20takes%20care%20of%20homeless%20cats%20and%20dogs%0D%0A
%23DREENDDOC
```

2. Copy the percent-encoded content string.
3. Send a `Query` action with the following parameters:
 - `Text`. Paste the percent-encoded content of the IDX or XML document to match against the `AgentBoolean` categories.
 - `TextParse`. Set this parameter to `True` to indicate that the specified `Text` is a percent-encoded document in IDX or XML format (it automatically detects the correct format).
 - `AgentBooleanField`. Set this parameter to the name of the `AgentBoolean` field to match against.
 - `DatabaseMatch`. Set this parameter to the database that contains agents or categories in the `Agentstore` component. By default, `Agentstore` databases are internal, so you must specify them explicitly.

For example:

```
action=Query&TextParse=True&AgentBooleanField=myABfield&DatabaseMatch=Activated
&Text=%23DREFERENCE%20http%3A%2F%2Fwww%2Ecatdog%2Ecom%0D%0A
%23DRETITLE%20Cats%20and%20Dogs%0D%0A
%23DREFIELD%20Animal10%3D%22dog%22%0D%0A
%23DREFIELD%20Animal11%3D%22cat%22%0D%0A
%23DRECONTENT%0D%0A
The%20organisation%20takes%20care%20of%20homeless%20cats%20and%20dogs%0D%0A
%23DREENDDOC
```

This query finds the categories that conceptually match the query text in the `Activated Agentstore` database. It then checks which of these categories contain a Boolean expression in their `myABfield` that the fields in the percent-encoded document match.

Note: IDOL Server also returns agents and categories that match the query text and do not contain the `AgentBoolean` or `FieldText` field.

IDOL Server returns only categories that match the document conceptually and contain a Boolean expression that matches the document fields. For example:

- IDOL Server returns a category that conceptually matches the document if its `myABfield` contains, for example, one of these Boolean expressions:

```
cat AND dog
```

```
cat:DRETITLE AND dog
```

- IDOL Server does not return a category that conceptually matches the document if its `myABfield` contains, for example, one of these Boolean expressions:

```
cat AND mat
```

```
cat AND dog:Animal10
```

(because `Animal10` is not configured as a `TextParseIndexType` field).

Related Topics

- ["Search and Retrieve" on page 197](#)

Optimize AgentBoolean Matching

After you set up `AgentBoolean` and `FieldText` matching, you can optimize the performance.

Configure AgentBoolean Cache Fields

When you create categories and agents in IDOL Server, the `Category` and `Community` components send these agents and categories to the `Agentstore` component.

If you manually create agents and categories and want to use a custom field for Boolean and `FieldText` restrictions, you can configure these fields in the `Agentstore` component. This configuration optimizes `AgentBoolean` and `FieldText` matching for those fields.

To configure AgentBoolean cache fields

1. Open the IDX or XML agents or categories that you want to index into IDOL Server.
2. Note the names of fields that contain the Boolean or FieldText restrictions.
3. Open the Agentstore component configuration file in a text editor.
4. In the [Server] section, set AgentBooleanCacheField to the name of the field that contains the Boolean restrictions.
5. Set FieldTextCacheField to the name of the field that contains the FieldText restrictions.

Note: You can have only one field in the AgentBooleanCacheField and FieldTextCacheField parameters.

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

Index a Dummy IDX

If your AgentBoolean expressions contain field restrictions, you must index a dummy IDX document to ensure that IDOL Server contains all the fields that you use for field restrictions in your agents and categories.

For example, if your agents and categories contain the FieldText expression `dog:Animal10`, ensure that IDOL Server contains the `Animal10` field to optimize restrictions.

A dummy IDX document contains all fields that you use in field restrictions, with empty values. For example, create a copy of your alert documents, and remove the values from each field. For example:

```
#DREREFERENCE dummy document
#DRETITLE DummyDocument
#DREFIELD Animal10=""
#DREFIELD Animal11=""
#DRECONTENT
#DRENDDOC
```

To ensure that IDOL Server does not subsequently return the dummy document in queries, index it into the Deactivated Agentstore database. You can delete the document after indexing.

Note: You must index the dummy IDX before you index or create the agents.

Related Topics

- ["Manually Create IDX Files " on page 511](#)

Customize Agent Content

In large systems that experience heavy load, you can optimize the performance of AgentBoolean queries by creating optimized agents or categories. This optimization is most useful when you use IDOL Server to:

- Return agents that match a non-indexed document, for example to alert users to new content that matches their agents.

- Return categories that match a non-indexed document, for example to categorize a document and add the category information to a field before indexing.

When IDOL Server matches query text against agents or categories, it uses the following matching order:

1. It matches the query text against the `Index` fields in the agents or categories (for example the `TRAINING` or `DRECONTENT` fields).
2. For agents that match in Step 1, it matches the query text against any Boolean restrictions in the agents or categories.
3. For agents that match in Step 2, it matches the query text against any `FieldText` restrictions in the agents or categories.

IDOL Server checks the Boolean restriction only if the agent or category content matches the document.

To optimize performance, choose your category or agent content carefully to ensure that it matches only query text that is also likely to match the Boolean expression.

Use a Minimal List of Rare Terms

For many Boolean restrictions, you can select one or two terms that queries must contain to match the Boolean expression. If you select the rarest of these combinations, fewer documents match the agent content. This process reduces the number of Boolean expressions that IDOL Server must match.

For example, if your Boolean restriction is:

```
"New Zealand" AND "French Champagne"
```

you can set the agent content to:

```
Zealand
```

In this Boolean expression, the query text must contain all four terms, so you can choose any of them as the agent content. `Zealand` is the rarest term, so fewer documents match this agent.

For other expressions, you can often similarly choose a minimal list of terms to reduce the number of documents that match against the Boolean restriction.

To find the minimal list of terms

- Send the `DocumentStats` action with the `QueryStats` parameter set to `True`. For example:

```
http://localhost:9000/action=DocumentStats&Text="New Zealand" AND "French  
Champagne"&QueryStats=True
```

This action returns a minimal list of terms that text must contain to match the Boolean expression. It provides the rarest terms where possible. The action returns the following information:

```
<autnresponse>  
  <action>DOCUMENTSTATS</action>  
  <response>SUCCESS</response>  
  <responsedata>  
    <autn:wildrequired>>false</autn:wildrequired>  
    <autn:numberrequired>1</autn:numberrequired>  
    <autn:required>ZEALAND</autn:required>
```

```
</responsedata>  
</autnresponse>
```

Using this method to choose the agent or category content can improve performance, especially in systems that receive a large number of queries.

Use AlwaysMatchType Fields

For some Boolean expressions that include Wildcards and DREFUZZY or SOUNDEX expressions, the minimal list of terms must include a Wildcard value.

For these expressions, the DocumentStats action with QueryStats set to True returns the following line:

```
<autn:wildrequired>true</autn:wildrequired>
```

The agent or category content cannot contain a Wildcard value, so you must use a different approach to ensure that IDOL Server checks the Boolean expression.

You can ensure that IDOL Server checks any documents that contain one of these expressions by configuring an AlwaysMatchType field in the Agentstore component and adding it to the agents.

IDOL Server always matches the content of an agent or a category that contains an AlwaysMatchType field. It then attempts to match the query text against the Boolean restriction, and returns any agents or categories that match at this stage.

To configure AlwaysMatchType fields

1. Open the IDX or XML files that contain your AgentBoolean documents.
2. Decide which fields you want to always match. For a document to always match, the AlwaysMatchType field must have a non-empty string.

Note: For IDOL Server categories, you must use the NOCONTENTCAT field.

3. Open the Agentstore configuration file in a text editor.
4. In the [FieldProcessing] section, add a new process to the list of processes. This process identifies the fields in the IDX or XML document that you want to always match. For example:

```
[FieldProcessing]  
0=SetIndexFields  
...  
15=SetAlwaysMatchFields
```

5. Create a new section for the process that you added. In this section, create a property for the process (you define the property, later by setting one or more applicable configuration parameters). Set PropertyFieldCSVs to a list of fields to associate with the process. For example:

```
[SetAgentBooleanFields]  
Property=AlwaysMatchFields  
PropertyFieldCSVs=*/MARKERFIELD,*/NOCONTENTCAT
```

6. Create a new section for the property that you added, in which you set AlwaysMatchType to True. This property indicates that IDOL Server must always match the associated fields in queries. For example:

```
[AlwaysMatchFields]  
AlwaysMatchType=True
```

7. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

After you configure the `AlwaysMatchType` field, you must manually create agents as IDX or XML documents.

For each agent, send the Boolean restriction in a `DocumentStats` action to IDOL Server to return the minimal list of terms.

- If the result does not contain a Wildcard value, you can create agents in the normal way.
- If the result contains a Wildcard, add the `AlwaysMatchType` field to the agent IDX. For example:

```
#DREREFERENCE WineAgent  
#DRETITLE My Wine Agent  
#DREFIELD BOOLEANRESTRICTION=""Champ*" AND "Merlot""  
#DREFIELD MARKERFIELD="1"  
#DRECONTENT  
#DREENDDOC
```

You can index your agent IDX or XML document into the `Agentstore` component agent database by using a `DREADD` or `DREADDATA` index action. For example:

```
http://localhost:9051/DREADD?C:\data\Agents.idx&DREDBName=Agent
```

Related Topics

- ["Fields" on page 73](#)

Chapter 12: Cluster Information

IDOL Server can automatically cluster information to make trends and developments in this information visible. Clustering is the process of taking a large repository of unstructured data and automatically partitioning it, so that similar information is clustered together. Each cluster represents a concept area in the knowledge base, and contains a set of items with common properties.

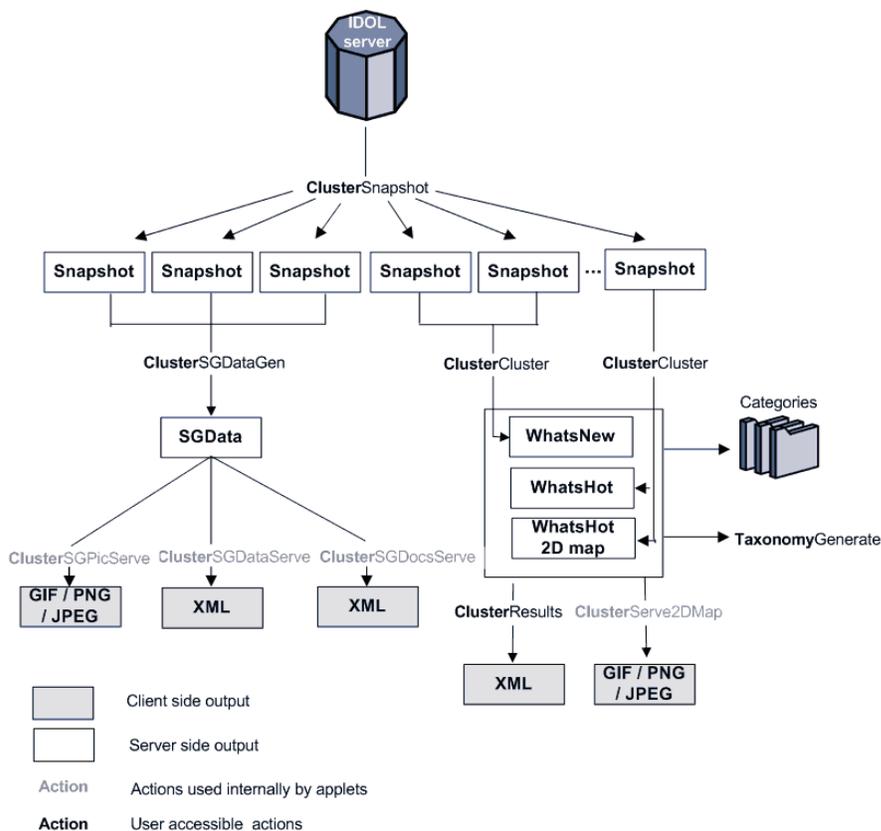
Note: IDOL Server provides several methods of clustering, including dynamic clustering and index tagging. See ["Cluster Results" on page 300](#), and ["Tag Documents into Clusters" on page 69](#).

For details of all the available types of clustering, and information about which one to choose, refer to *IDOL Expert*.

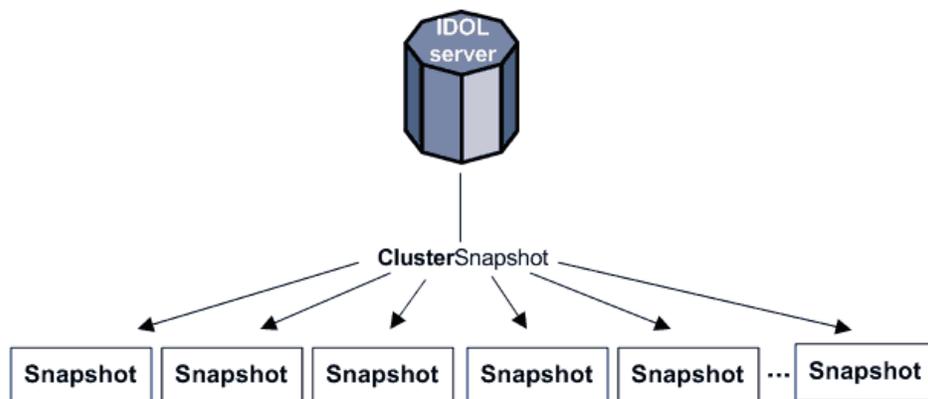
- [Generate Snapshots](#) 179
- [Generate Spectrograph Data](#) 181
- [Generate WhatsNew and WhatsHot Information](#) 182
- [Configure Clusters](#) 184
- [Set up Schedules](#) 187
- [Partition Documents into Clusters](#) 189

Generate Snapshots

To cluster information, take a snapshot of data that IDOL Server stores. You can then automatically cluster the data in this snapshot (you do not need to set up an initial taxonomy).



IDOL Server takes a snapshot of the data that it stores and, based on these snapshots, clusters related information together. Each cluster represents a concept area that contains a set of items, which share common properties.



The ClusterSnapshot action allows you to take a snapshot of the data stored in the IDOL Server data index. By default this includes data for the IDOL Server databases News and Archive. A snapshot represents the content of the data index at a particular time, and enables you to generate cluster information and spectrographs at a later point, even if the data index has changed. You can use a single snapshot to generate both cluster information and spectrograph data to save processing time.

The action adds a timestamp to each snapshot (with the AUTNDATE) and stores it in binary .cls format in the Snapshots subdirectory of the Cluster directory in your IDOL Server installation directory. This

process allows you to have several snapshots with the same name (for example, of one particular IDOL Server) and snapshots with different names (for example, of different data sets).

The results of `ClusterSnapshot` are saved as a named *snapshot job*. You can specify that job name when taking other actions on the snapshot data. You can also set up a schedule that runs the `ClusterSnapshot` action at regular intervals.

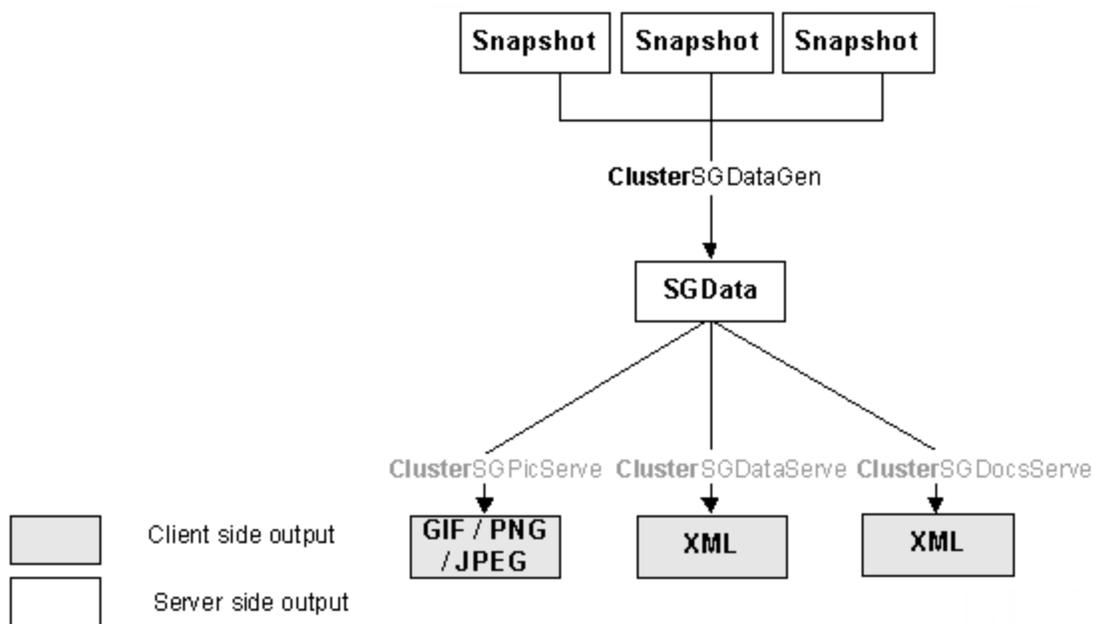
Note: The IDOL Server data index that you take a snapshot of must ideally contain at least several thousand documents with good quality content (that is, relevant text for various topics).

Related Topics

- ["Set up Schedules" on page 187](#)

Generate Spectrograph Data

The `ClusterSGDataGen` action allows you to generate spectrograph data from a set of snapshots that you took using the `ClusterSnapshot` action.



Each spectrograph dataset takes a succession of clusters from different time periods, calculates cluster similarity measures across days, and applies a graph theoretic matching algorithm. IDOL Server makes calculations about the conceptual spread of a cluster and its general quality. The spectrograph uses lines to represent the size (number of documents in a cluster) and quality of a cluster. The brighter a spectrograph line is, the more documents the cluster contains, and the thicker the line is, the higher the cluster quality.

All spectrograph data sets that you generate are stored in the `Sgdata` subdirectory of the `Cluster` directory in your IDOL Server installation directory.

You can set up a schedule that runs the `ClusterSGDataGen` action at regular intervals.

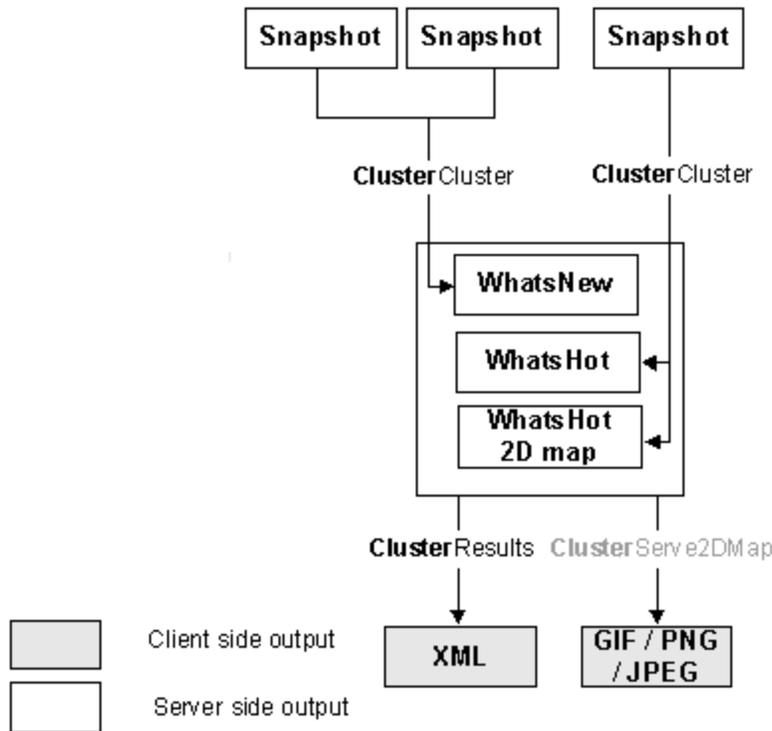
You can retrieve the spectrograph image, data, or documents by using the `ClusterSGPicServe`, `ClusterSGDataServe`, and `ClusterSGDocsServe` actions, which the Spectrograph applet runs.

Related Topics

- ["Set up Schedules" on page 187](#)

Generate WhatsNew and WhatsHot Information

The `ClusterCluster` action allows you to analyze clusters in a snapshot that you took using the `ClusterSnapshot` action.



Clustering is a multistage hybrid algorithm. After the IDOL Server Adaptive Probabilistic Concept Modelling (APCM) technology identifies similar documents, a hierarchical agglomerative clustering algorithm groups documents into conceptually similar areas. Dynamic binding and fixating produces the required clusters. The title is generated automatically by cross-correlating important concepts in a cluster with concepts in the titles of documents in that cluster.

IDOL Server saves the results of clustering as a named *cluster.job*. You can specify that job name when taking other actions on the clustered data. You can also set up a schedule that runs the `ClusterCluster` action at regular intervals.

Depending on which parameters you combine the action with, you can generate WhatsNew or WhatsHot information.

Related Topics

- ["Set up Schedules" on page 187](#)

WhatsNew Information

WhatsNew information is the latest information that is available for the clusters that IDOL Server identifies in your snapshot.

You can generate WhatsNew information by comparing two snapshots (that have the same name or different names).

IDOL Server saves the results of the `ClusterCluster` action in configuration files (`.cfg`) in the `Clusters` subdirectory of the `Cluster` directory of your IDOL Server installation. You can use the `ClusterResults` action to retrieve them in XML format.

If you configured the `ClusterCluster` action to generate a 2-D map of WhatsHot cluster information, you can use the `ClusterServe2DMap` action to return this map in one of the supported image formats (that is, GIF, PNG or JPEG).

WhatsHot Information

WhatsHot information is the most relevant information that is available for the clusters that IDOL Server identifies in your snapshot. Unlike WhatsNew information, this is not restricted to new information, which means that it can follow the progress of particular news items over time.

You can cluster WhatsHot information from a snapshot and use the IDOL HotNews portlet to display this information in a portal. You can also generate a 2-D map from WhatsHot information and display it in a portal using the IDOL 2DMap portlet.

The 2-D map gives a visualization of the similarities and difference between clusters. IDOL Server uses a dimensionality reduction algorithm to maintain intercluster similarity measures so that similar clusters are close together and very different clusters are not close together. IDOL Server uses the distribution of documents throughout the space, along with nonlinear remapping, to create the landscape.

Generate a Cluster Map after You Cluster

Typically, to create a 2-D map from your clustered data, you send a `ClusterCluster` action with the `DoMapping` parameter set to `True`. You then send the `ClusterServe2DMap` action.

If you clustered your data with `DoMapping` set to `False`, you can still generate a cluster map of the data by using the `ClusterMapFromResults` action.

You pass the job name from your `ClusterCluster` action as the `SourceJobName` parameter to `ClusterMapFromResults`, and it returns a 2-D cluster map as binary image data.

Related Topics

- ["Create a Cluster Map" on page 314](#)

Configure Clusters

You can take a snapshot of the data content that IDOL Server stores. This snapshot identifies clusters of conceptually similar documents, which enables you to generate a view of trends in the data. You do not need to generate an initial taxonomy to take a snapshot.

A set of data can contain a few large clusters or many small clusters, as well as several outliers that are not part of any cluster. Clusters can consist of highly similar documents or of less closely related ones. What constitutes optimal clustering depends on how you intend to use your clusters. However, the aim of clustering is always to generate an accurate characterization of the data content in your IDOL Server.

By default IDOL Server uses internal settings to produce clusters. You do not usually need to change these default settings. However, in some cases you might require more or less detail in your clusters, or the amount and nature of your data might mean that default clustering is not satisfactory.

You can adjust the size of the units on which to base clusters, the degree of conceptual similarity that documents within clusters must have, or the number of clusters to create.

Change the Number and Size of Clusters

There are two main stages to the clustering process:

- "Build Seeds"
- "Group Seeds into Clusters"

Build Seeds

IDOL Server builds seeds when you send the `ClusterSnapshot` action. IDOL Server takes a sample of the documents that it stores, and tries to associate individual documents with each other, based on the similarity of the concepts that the documents contain. Each group produced at this stage, containing a sample document and similar documents, is a seed.

IDOL Server stops trying to build a seed when the seed meets the requirements that `SeedSize` specifies or when there are no more documents that meet the similarity requirement that `SeedBindLevel` specifies (whichever condition is reached first). IDOL Server discards any seeds that do not reach the required size.

The number of clusters that you specify with `NumClusters` affects the number of sample documents that IDOL Server tries to create seeds from. You can adjust the relationship between the number you specify here and the size of the sample used by changing the value of `StartingSuggestOverrideFactor`.

Group Seeds into Clusters

IDOL Server groups seeds into clusters when you send the `ClusterSGDataGen` or `ClusterCluster` actions. IDOL Server tries to create clusters by grouping seeds together. The grouping is based on the similarity of the concepts that the seeds or clusters contain.

Clustering is complete when one of the following conditions is met:

- IDOL Server creates the number of clusters specified by `NumClusters`.
- IDOL Server cannot create any more clusters that meet the similarity requirement specified by `BindLevel`.

IDOL Server discards clusters that do not meet the quality requirement set by `BindLevel` or the size requirement set by `MinClusterDocs`.

For details of the clustering actions, and the settings you can make to generate the clusters from your data, refer to the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)

Configuration Parameters

The ideal values for the parameters that affect clustering depend on the nature and amount of data in your IDOL Server. You can use the `SentientClustering` parameter for the `ClusterSnapshot` action to automatically determine the correct values for `SeedSize` and `SeedBindLevel`.

This section makes general recommendations about how to manually alter these parameters according to your data. Parameters are closely interdependent, so make these changes in combination with each other (rather than just changing one of the settings). Change values in small steps.

Although you can make many changes to clustering, the number and size of clusters that IDOL Server can identify depends ultimately on the data content that it contains. You can:

- cluster a small amount of data.
- cluster a large amount of data.
- cluster very similar data.
- cluster very different data.
- change the data view.

Cluster a Small Amount of Data

If your IDOL Server has a small amount of data, it is likely to identify fewer clusters, because it is less likely that your data contains a lot of similar documents for several different topics. You can edit the following parameters to change clustering in this situation.

Note: Ideally, your IDOL Server must contain at least 500 documents.

<code>SeedSize</code>	Decrease <code>SeedSize</code> (by three to four points at a time). This option reduces the size that seeds must reach, which means that more seeds are likely to be successfully created.
<code>MinClusterDocs</code>	Decrease <code>MinClusterDocs</code> so that clusters that contain fewer documents are not discarded.
<code>StartingSuggestOverrideFactor</code>	Increase <code>StartingSuggestOverrideFactor</code> (by one or two points only). This increases the number of sample documents from which IDOL Server creates seeds, which in some cases increases the possibility of finding clusters in the data.

SeedBindLevel	Decrease <code>SeedBindLevel</code> (by one point at a time) to reduce the similarity threshold for clusters. Do not change this value until you try changing <code>SeedSize</code> , because lowering <code>SeedBindLevel</code> is more likely to allow less-relevant documents into clusters.
---------------	--

Cluster a Large Amount of Data

If your IDOL Server has a large amount of data, you probably do not need to edit any clustering parameters, because this is the situation in which clustering is most successful. In some cases (for example, if your IDOL Server contains more than a million documents), it can be beneficial to alter the following parameter.

StartingSuggestOverrideFactor	Increase the value of this parameter to increase the number of sample documents from which IDOL Server creates seeds. This is sometimes necessary to allow a broader section of the data content to be represented by the clusters that are created.
-------------------------------	--

Cluster Very Similar Data

If the documents in your IDOL Server contain highly similar concepts, IDOL Server might identify a small number of large clusters. For example, if your IDOL Server contains mostly documents about sports, then you might get one large *sports* cluster. This situation is a realistic characterization of the data in your IDOL Server, but in many circumstances is not useful. You can edit the following parameters to generate smaller, more specific clusters (for example, breaking *sports* into *football*, *tennis*, *golf*, and so on).

SeedBindLevel	<p>Increase <code>SeedBindLevel</code> to require greater similarity between the documents that form a seed, which can reduce the breadth of topics covered by the concepts in the seed documents.</p> <p>Note: Increase <code>SeedBindLevel</code> one point at a time. Increasing by too much can result in seeds being discarded because they do not contain enough documents.</p>
BindLevel	Increase <code>BindLevel</code> to require greater similarity between the concepts in seeds or clusters that merge to create a cluster. This change can decrease the size of clusters, as well as increase the number of clusters identified, because merging seeds and clusters together stops at an earlier stage.

Cluster Very Different Data

If the documents in your IDOL Server contain a wide variety of concepts, there might not be enough similar documents for IDOL Server to create seeds or clusters that characterize the data that it stores. You can lower the similarity requirement with the following parameters.

SeedBindLevel	<p>Decrease <code>SeedBindLevel</code> to reduce the similarity requirement between the documents that form a seed, which can increase the breadth of topics covered by the concepts in the seed documents.</p> <p>Note: Decrease <code>SeedBindLevel</code> one point at a time. Decreasing by too much</p>
---------------	---

	can result in seeds and clusters that contain documents that are less relevant, because the similarity requirement is too low.
BindLevel	Decrease BindLevel to reduce the similarity requirement between the concepts in seeds or clusters that merge to create a cluster. This change can increase the size of clusters, as well as increase the number of clusters identified (because fewer are discarded for not meeting the quality requirement).

Change the Data View

It might be the case that although IDOL Server identifies clusters that characterize your data successfully, you want to change the view of the data that clustering creates. The following parameters enable you to change the data view that clusters generate.

NumClusters	Increase NumClusters to obtain a more low-level view of your data by identifying more clusters. Decrease NumClusters to obtain a more high-level view by identifying fewer clusters.
MinClusterDocs	Decrease MinClusterDocs to reduce the number of clusters that are discarded. This option allows you to identify smaller clusters. Increase MinClusterDocs to increase the number of clusters that it discards. Only larger clusters are kept.
BindLevel	Decrease BindLevel to reduce the similarity requirement between the concepts in seeds or clusters that merge to create a cluster. This option can increase the size of clusters, as well as increase the number of clusters identified (because it discards fewer clusters for not meeting the quality requirement). Increase BindLevel to increase the similarity requirement between the concepts in seeds or clusters that merge to create a cluster. This can decrease the size of clusters, as well as increase the number of clusters identified, because merging seeds and clusters together stops at an earlier stage.

Set up Schedules

You can set up to 1,024 schedules, which allow you to run the following actions at regular intervals.

- ClusterSnapshot
- ClusterCluster
- ClusterSGDataGen
- TaxonomyGenerate

For details on the settings that each [AnalysisSchedule] section can contain and on how you can configure them, refer to the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)

To set up schedules

1. Open the IDOL Server configuration file in a text editor.

2. Create an [AnalysisScheduleN] section for each schedule that you want to run. Start numbering the [AnalysisScheduleN] sections from zero (so that the first schedule section is [AnalysisSchedule0]). For example:

```
[AnalysisSchedule0]
[AnalysisSchedule1]
[AnalysisSchedule2]
[AnalysisSchedule3]
[AnalysisSchedule4]
[AnalysisSchedule5]
```

In this example six schedules were created. Note that the schedules are listed in consecutive order, starting from zero.

3. Specify the settings for each schedule in the appropriate section. You can specify the action to schedule, the interval in which each schedule runs, the number of times each schedule runs, what job name to give the action, and so on. For example:

```
[AnalysisSchedule0]
ScheduleStartTime=now
ScheduleInterval=1 day
ScheduleCycles=1
ScheduleAction=ClusterSnapshot
TargetJobName=myjob
```

```
[AnalysisSchedule1]
ScheduleStartTime=now
ScheduleInterval=1 day
ScheduleCycles=1
ScheduleAction=ClusterCluster
SourceJobName=myjob
TargetJobName=myjob_clusters
DoMapping=True
```

```
[AnalysisSchedule2]
ScheduleStartTime=now
ScheduleInterval=1 day
ScheduleCycles=1
ScheduleAction=ClusterCluster
SourceJobName=myjob
TargetJobName=myjob_clusters_new
WhatsNew=True
Interval=86400
```

```
[AnalysisSchedule3]
ScheduleStartTime=now
ScheduleInterval=1 day
ScheduleCycles=1
ScheduleAction=ClusterSGDataGen
Interval=604800
SourceJobName=myjob
```

```
TargetJobName=myjob_sg

[AnalysisSchedule4]
ScheduleStartTime=now
ScheduleInterval=1 day
ScheduleCycles=1
ScheduleAction=ClusterSGDataGen
Interval=86400
SourceJobName=myjob_content
TargetJobName=compare_snapshots_sg

[AnalysisSchedule5]
ScheduleStartTime=now
ScheduleInterval=1 day
ScheduleCycles=1
ScheduleAction=TaxonomyGenerate
Cluster=0,1,2,3,4,5,6,7,8,9
SourceJobName=myjob_clusters
TargetJobName=myjob_taxonomy
WriteTaxonomy=true
NumResults=25
```

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

Tip: As an alternative to setting up schedules in the configuration file, you can also use the **Category Queue** page in the Control section of IDOL Admin to schedule `ClusterSnapshot`, `ClusterCluster`, `ClusterSGDataGen`, and `TaxonomyGenerate` actions. For more information, refer to the *IDOL Admin User Guide*.

Partition Documents into Clusters

Partition clustering is a different method of clustering that allows you to create a specified number of clusters, and to place all documents into a cluster. This method is faster than using `ClusterSnapshot`.

This method adds all documents into a cluster, which means that the clusters generated are relatively broad and might represent several topics. You might want to use this method if you want to cluster the results of a query into a predefined number of clusters.

To partition documents into clusters

- Send the `ClusterPartition` action to IDOL Server. Set the following action parameters:

DREQuery	The query that you want to cluster the results for.
NumResults	The number of results from the specified query that you want to return and cluster.
NumClusters	The number of clusters to create.

For more information about the `ClusterPartition` action, refer to the *IDOL Server Reference*.

Chapter 13: Profiles

This section describes how to create interest and expertise profiles for users for collaboration and locating experts.

- [About Profiles](#) 191
- [Profile a User](#) 191
- [Manipulate Profiles](#) 192
- [Collaboration and Expertise with Profiles](#) 193

About Profiles

IDOL Server automatically creates interest and expertise profiles for users in real time. You can configure IDOL Server to create up to four different profile types. By default, IDOL Server creates an interest and an expertise profile for each user.

You can create an interest profile by tracking the content that a user views and extracting a conceptual understanding of it. IDOL Server then uses this understanding to keep the user interest profile up to date. You can use interest profiles to target information at users, recommend content to users, alert users to the existence of content, and put users in touch with other users with similar interests.

You can create an expertise profile by tracking the content that a user creates and extracting a conceptual understanding of it. IDOL Server then uses this understanding to keep the expertise profile for that user current. You can use expertise profiles to trace users who are experts in particular subject areas.

Profile a User

You can use the `ProfileUser` action to profile a user. For details on this action, refer to the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)

Create an Interest Profile for a User

Run the `ProfileUser` action when a user views a document. IDOL Server analyzes the document that the user is viewing, and determines if it is similar to any of the concepts in their interest profile (using `MatchThreshold`).

If the content of the viewed document is similar to an existing interest profile concept, IDOL Server updates the existing concept with the new document. If several concepts are similar, only the most similar one updates. If the document content is not similar to an existing interest profile concept, IDOL Server creates a new concept in the interest profile.

Note: IDOL Server uses only the five strongest concepts in a user interest profile for recommendations, alerting, and similar user matching.

For example:

```
http://12.3.4.56:4000/action=ProfileUser&UserName=Administrator&Document=3422+5776&MatchThreshold=60&NamedArea=Interest
```

This action instructs IDOL Server to analyze the content in the 3422 and 5776 documents. If the content has a conceptual relevance of at least 60 percent to a concept in the interest profile of the Administrator user, IDOL Server uses it to update the matching interest profile concept (if several concepts are similar, only the most similar one updates). If the document content does not have a conceptual relevance of at least 60 percent to an existing interest profile concept, IDOL Server creates a new interest profile concept from it.

Create an Expertise Profile for a User

Run the ProfileUser action when a user creates text (for example, a document in IDOL Server that was authored by a user, or text that a user enters in a help desk environment). IDOL Server analyzes the text that the user created, and determines if it is similar to any concepts in the existing expertise profile (using MatchThreshold).

If the content of the viewed text is similar to an existing expertise profile concept, IDOL Server updates the existing concept with the text (if several concepts are similar, only the most similar one updates). If the text is not similar to an existing expertise profile concept, IDOL Server creates a new concept in the expertise profile.

Note: IDOL Server uses only the five strongest concepts in a user expertise profile for expertise matching.

For example:

```
http://12.3.4.56:4000/action=ProfileUser&UserName=Administrator&Document=The  
chemical structure of everyone's DNA is the same. The only difference between  
people (or any animal) is the order of the base pairs&  
MatchThreshold=60&NamedArea=Expertise
```

This action instructs IDOL Server to analyze the specified text. If the text has a conceptual relevance of at least 60 percent to any concept in the Administrator user expertise profiles, IDOL Server uses it to update the matching expertise profile concept (if several concepts are similar, only the most similar one updates). If the text does not have a conceptual relevance of at least 60 percent to an existing expertise profile concept, IDOL Server creates a new expertise profile concept from it.

Manipulate Profiles

Manipulating profiles consists of editing, querying, viewing, and deleting them.

Related Topics

- ["Display Online Help" on page 30](#)

Edit a Profile

IDOL Server stores interest and expertise profiles in the form of terms and weights. You can use the `ProfileEdit` action to edit profile terms and weights. For details on this action, refer to the *IDOL Server Reference*. For example:

```
action=ProfileEdit&PID=1-P2.3&TermCOLOR=2322
```

This action changes the weight of the `COLOR` term in the `1-P2.3` profile to `2322`.

Query with a Profile

You can use the `ProfileGetResults` action to query with a profile. For details on this action, refer to the *IDOL Server Reference*. When you query with a profile, by default it matches against the IDOL Server data index.

View Profile Details

You can use the `ProfileRead` action to view profile details. For details on this action, refer to the *IDOL Server Reference*. For example:

```
action=ProfileRead&UserName=Administrator&PID=3422
```

This action requests the details of the `3422` profile for the `Administrator` user.

Delete a Profile

You can use the `ProfileClear` action to delete a profile from the IDOL Server profile index. For details on this action, refer to the *IDOL Server Reference*. For example:

```
action=ProfileClear&UserName=Administrator&PID=450-P0.1
```

This action deletes the `450-P0.1` profile for the `Administrator` user.

Collaboration and Expertise with Profiles

You can use IDOL Server profiles to collaborate with other users or to locate experts in your field of interest.

Collaboration

IDOL Server automatically matches users with common explicit interest agents or similar implicit profiles. You can use this information to create virtual expert knowledge groups.

You can use the `Community` action to find agents or profiles in the community that match the agents or the profiles of a specific user. For example:

```
action=Community&UserName=JSmith&Agents=True&Profiles=True&AgentsFindProfiles=True&ProfilesFindAgents=True
```

This action finds agents and profiles in the user community that match both the agents and the profiles of the user JSmith.

Expertise

IDOL Server accepts a natural language or Boolean search string and returns users who own matching agents or profiles. This process allows you to identify experts in any subject, eliminating time-consuming searches for specialists and unnecessary researching of subjects for which expert knowledge is already available.

You can use the `Community` action to find agents or profiles in the community that match a natural language or Boolean search string.

For example:

```
action=Community&Text=how does the cost of funds, such as the costs of performing a credit evaluation on the business requesting a loan, determine the spread between the federal funds rate and the prime rate&AgentsFindProfiles=True&ProfilesFindAgents=True
```

This action finds agents and profiles in the user community that match the specified text.

Part IV: Results

This section describes the processes of querying IDOL Server, retrieving results, and displaying those results to users.

- "Search and Retrieve"
- "Customize Results"
- "Manipulate Result Relevance"
- "Manipulate the Results Set"
- "View Documents"

Chapter 14: Search and Retrieve

You can search IDOL Server with actions by using a Web browser, an HPE interface application (for example, IDOL Admin), or a third-party portal that uses HPE portlets.

- [Actions](#) 197
- [Conceptual Matches](#) 200
- [Keyword Search](#) 202
- [Phrase Search](#) 208
- [Boolean and Proximity Search](#) 212
- [Simple Field Restricted Search](#) 221
- [FieldText Search](#) 222
- [Fuzzy Search](#) 256
- [Parametric Search](#) 257
- [Proper Names Search](#) 260
- [Soundex Keyword Search](#) 263
- [Synonym Search](#) 264
- [Analytics Functions](#) 269
- [Link Queries](#) 274
- [Combine Different Search Types](#) 277
- [Wildcards in Queries](#) 280
- [Query for Nonalphanumeric Characters](#) 283
- [Optimize Retrieval of Tagged Documents](#) 285

Actions

You query IDOL Server by using actions. The following actions are available to all clients that have permission to query IDOL Server (set by `QueryClients` in the `[Server]` section of the IDOL Server configuration file).

Tip: You can view information on the permissions that are available on the **Overview** tab in the Status section of IDOL Admin.

GetContent	Displays the content of one or more specified documents.
GetQueryTagValues	Returns the values of parametric fields in query results.
GetTagNames	Returns all fields of a specified type.
GetTagValues	Performs a parametric search.
Highlight	Highlights link terms in text.

Query	Submits different query types to IDOL Server.
Suggest	Retrieves documents that are conceptually similar to one or more specified documents.
SuggestOnText	Retrieves documents that are conceptually similar to the terms with the highest weighting in specified text.
Summarize	Generates a summary for documents or text.
TermGetBest	Lists the conceptually most important terms in one or more specified documents.
TermGetInfo	Returns the weight and other available information for specified terms.

Note: You can check whether you have permission to query IDOL Server by using the ShowPermissions ACI action.

In addition, the following actions are available to administrative clients of IDOL Server (set by AdminClients in the [Server] section of the IDOL Server configuration file).

DetectLanguage	Determines the language of a piece of text.
GetStatus	Displays configured details about the IDOL Server setup.
IndexerGetStatus	Displays the status of index actions in the IDOL Server index queue.
List	Lists all documents that are stored in IDOL Server or any of its databases.
TermGetAll	Lists all terms that are stored in IDOL Server.

Related Topics

- ["Display Online Help" on page 30](#)
- ["Send Actions to IDOL Server" on page 29](#)

Asynchronous Actions

By default, IDOL Server processes all actions synchronously. In this case, IDOL Server does not respond to the action until it has completed the request. The result of the action is in the response to the request.

With additional configuration, you can also process query actions asynchronously. In this case, IDOL Server responds to the action immediately. The request is added to a queue of actions. The response to the request contains a token. You can use this token to determine whether the request has finished, and obtain the results of the action. To do this, use the QueueInfo action.

You might want to use asynchronous actions to run operations that take a long time, and that do not require an immediate response. For example, you might have synchronous queries to return results to users, and then use an asynchronous action to run a query summary.

Configure Asynchronous Actions

You must configure IDOL Server to allow asynchronous actions.

To configure asynchronous functionality

1. Open the IDOL Server configuration file in a text editor.
2. Create an [Actions] configuration section.
3. In the [Actions] section, add the Async parameter, and set it to AsyncActions.
4. Create an [AsyncActions] section.
5. In the [AsyncActions] section, set the Threads parameter to the number of threads that you want to use for asynchronous actions.
6. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

For example:

```
[Actions]  
Async=AsyncActions
```

```
[AsyncActions]  
Threads=2
```

Send Asynchronous Actions

To send an asynchronous action, you use the same action syntax as normal, but add the Synchronous parameter, set to False.

You can send the following actions asynchronously:

DetectLanguage	GetTagValues	Summarize
DocumentStats	Highlight	TermExpand
GetAllRefs	List	TermGetAll
GetContent	Query	TermGetBest
GetQueryTagValues	Suggest	TermGetInfo
GetTagNames	SuggestOnText	

Retrieve Results for Asynchronous Actions

The QueueInfo action allows you to retrieve the status and results of asynchronous actions.

To return the results of all asynchronous actions, send a QueueInfo action, with the QueueName parameter set to Async, and the QueueAction parameter set to GetStatus. You can set the Token parameter to the token for a specific action to retrieve results for only that action.

Note: If you do not specify the Token parameter, all results for the server return. HPE does not

recommend that you return all results for a server that has been running for a long time, because the response might time out.

Note: There are no user restrictions for viewing the queue list, so all users can see all results that the original action returns. HPE does not recommend that you use asynchronous actions if you have strict document security requirements.

Tip: You can use the `QueueInfo` action to return information about the queue, and to cancel or change the priority of actions. For more information, refer to the *IDOL Server Reference*.

You can use the Asynchronous Queues page in the Monitor section of IDOL Admin to view details of the asynchronous actions that the server is processing, view the results of those actions, and cancel actions. For more information, refer to the *IDOL Admin User Guide*.

Conceptual Matches

You can use `Query`, `Suggest`, and `SuggestOnText` actions to perform conceptual searches. IDOL Server uses advanced pattern-matching technology to conceptually match the data that you query with (using actions) against the content it holds.

Types of Matches

- **Content searches.** You can submit natural language text or a piece of content to IDOL Server. It returns references to conceptually related documents ranked by relevance or contextual distance. Natural language queries make it possible for users to find results without having to be familiar with search algorithms or syntax. Online shoppers, for example, can find specific items without knowing the exact product or brand name.
- **Community searches.** You can create agents from natural language and then match them conceptually. You can also submit profiles or natural language text to IDOL Server. It returns agents ranked by conceptual similarity. This process determines which users have similar interests, which promotes collaboration, and identifies experts in a field.
- **Category searches.** You can submit a piece of content to IDOL Server, for which it returns categories ranked by conceptual similarity. This process determines which categories the piece of content is most appropriate for, so that IDOL Server can tag, route, or file the piece of content accordingly.
- **Clusters.** You can use IDOL Server to organize large volumes of content or large numbers of profiles into self-consistent clusters. Clustering is an automatic agglomerative technique that allows IDOL Server to partition data by grouping together information that contains similar concepts.

Example Queries

This section shows some examples of the different types of queries.

Agent or Category Query

You can add agent (or category) weights to terms in your query. IDOL Server returns documents that contain highly weighted terms with a higher relevance than documents that contain only terms with lower weights. Use the `TermGetInfo` action to find out the weight of the term. Specify weights as a number in brackets next to the term. The maximum weight is 4095, but HPE recommends that you use a maximum weight of 511.

For example:

```
http://localhost:9000/action=Query&Text=Cat[50] OR Dog
```

This query returns documents that contain the term *cat* or the term *dog*. If the term *dog* has a higher weight than 50 in IDOL Server, documents that contain the term *dog* return with higher relevance than documents that contain only the term *cat*. If the term *dog* has a lower weight than 50, documents that contain the term *cat* return with higher relevance.

```
http://localhost:9000/action=Query&Text=Cat[30] OR Dog[10]
```

This query returns documents that contain the term *cat* or the term *dog*. Documents that contain only the term *cat* return with three times the relevance than documents that contain only the term *dog*.

You can apply term weights to phrases. For example:

```
http://localhost:9000/action=Query&Text="cats and dogs"[100]
```

This query returns documents that contain the phrase "cats and dogs" and applies a weight of 100 to this phrase.

You can apply weights to terms in parentheses. In this case, the weight applies to all terms within the parentheses. For example:

```
http://localhost:9000/action=Query&Text=(cat OR dog)[100] AND (fish OR dolphin)  
[150]
```

This query returns documents that contain at least one of the terms *cat* or *dog*, and at least one of the terms *fish* or *dolphin*. It assigns the terms *cat* and *dog* a weight of 100, and it assigns the terms *fish* and *dolphin* a weight of 150.

You can use multipliers to multiply the original term weight. Enter a term as **N*, where *N* is the amount to multiply by. You can use up to two decimal places. For example:

```
http://localhost:9000/action=Query&Text=cat[*2.25] OR dog[*0.5]
```

This query returns documents that contain the term *cat* or the term *dog*. The term *cat* has 2.25 times its normal relevance. The term *dog* has 0.5 times its normal relevance.

You can also use any combination of these examples to apply term weights to complex phrases or expressions. For example:

```
http://localhost:9000/action=Query&Text="cats and dogs"[*3] OR (fish[100]+dolphin  
[150])[*1.5]
```

This query applies 3 times the normal weight to the phrase "cats and dogs". It assigns a weight of 100 to the term *fish* and a weight of 150 to the term *dolphin*. It then multiplies the weights for *fish* and *dolphin* by 1.5.

Profile Query

```
http://localhost:5552/action=Query&Text= CHAMPIONLEAGU~[551] EVERTON~[407] BAYERN~[402] UEFA~[391] PREMIERSHIP~[383] FIFA~[257] STRIKER~[226] WORLDCUP~[215] EURO~[124] SOCCER~[114] CUP~[66]
```

This action sends a profile query to IDOL Server. The query contains the terms that the profile training contains, and the weight of each of the terms. IDOL Server can return agents, profiles, categories, or documents that conceptually match the terms of the query.

Text Query

```
http://localhost:5552/action=Query&Text=Gene analysis discovered methods to determine the exact sequence of nucleotides that compose a specific gene.
```

This action sends a text query to IDOL Server. IDOL Server can return agents, profiles, categories, or documents that conceptually match the query text.

Suggest Query

```
http://localhost:5552/action=Suggest&ID=10
```

This action sends a Suggest query to IDOL Server. IDOL Server can return agents, profiles, categories, or documents that conceptually match the specified document (that is, the document with the ID 10).

SuggestOnText Query

```
http://localhost:5552/action=SuggestOnText&Text=Gene analysis discovered methods to determine the exact sequence of nucleotides that compose a specific gene
```

This action sends a SuggestOnText query to IDOL Server. IDOL Server can return agents, profiles, categories, or documents that conceptually match the terms with the highest weighting in the query text.

Keyword Search

By default, IDOL Server conceptually matches queries that consist of a single keyword. It stems the keyword, and then it finds documents that contain words that have the same stem as the keyword.

For example, if you query IDOL Server with the word *lovely*, it stems the word to *love* and finds documents that contain words that also stem to *love*, for example, *lovely*, *love*, *loved*, *loving*, and so on.

Keyword Occurrence Search

You can restrict the documents that match a query term to documents in which the number of occurrences of the term or phrase is within a specified range. Specify the range with a colon-separated

pair of numbers. The maximum value that you can specify in the range is 32000.

For example:

```
action=Query&Text=Gene[3:7]
```

This query returns only documents in which *Gene* appears between three and seven times (inclusively).

```
action=Query&Text="Gene Therapy"[5:8]
```

This query returns only documents in which the phrase *Gene therapy* appears between five and eight times (inclusively).

```
action=Query&Text=Gen*[2:4]
```

This query returns only documents in which a term that starts with *Gen* (such as *Gene* or *Generation*) appears between two and four times, inclusively.

```
action=Query&Text=(Gene OR DNA)[4:8] AND Therapy
```

This query returns only documents in which the term *Gene* or the term *DNA* appear between 4 and 8 times (inclusively), and the term *Therapy* occurs.

You can specify open-ended ranges. For example:

```
action=Query&Text=Gene[10:]
```

This query returns only documents in which *Gene* appears 10 or more times.

You can also specify term weights and field restrictions. For example:

```
action=Query&Text=cat[2:4]:DRETITLE + AND + cat[100][2:4]
```

Exact Keyword Search

To find documents that contain exact matches of a keyword, enable the `AdvancedSearch` setting before you index content into IDOL Server.

If you set `AdvancedSearch` to `True` in the `[Server]` section of the IDOL Server configuration file before you index content, you can query for exact matches of keywords by putting the keyword in quotation marks when you perform the query (this setting also switches IDOL Server to an advanced weighting algorithm which improves conceptual querying).

For example, if you query IDOL Server with `"love1y"`, it does not stem the word, and finds only documents that contain *lovely*.

If you do not put the word in quotation marks, IDOL Server matches it conceptually; that is, the query *lovely* matches documents that contain, *lovely*, *love*, *loved*, *loving*, and so on.

You can also suffix a keyword with a tilde (~) to indicate that the term is already stemmed. In this case, IDOL Server does not stem the query term. However, unlike an exact keyword search using quotation marks, this query might return documents that contain other terms that stem to your query term. For example, searching for `Love~` returns documents that contain the term *love*, but also other terms that stem to *love*, for example, *lovely*, *love*, *loved*, and *loving*.

Note: Using the tilde suffix to search for a term that is not its own stem does not return documents

that contain the original term. For example, searching for `loving~` matches terms that stem to *loving*. The term *loving* stems to *love*, so this search does not return documents that contain the term *loving*.

You can use the `TermGetInfo` action to find the stem of a term.

Case-Sensitive Exact Keyword Search

To find documents that contain case-sensitive exact matches of a keyword, enable the `AdvancedCaseSearch` setting before you index content into IDOL Server:

If you set `AdvancedCaseSearch` to `True` in the IDOL Server configuration file `[Server]` section before you index content, you can query for case-sensitive exact matches of keywords by prefixing the keyword with a tilde (`~`) and putting it in quotation marks when you perform the query.

For example, if you query IDOL Server with `"~Love1y"`, it does not stem the word, and finds only documents that contain *Lovely*.

If you put a word into quotation marks but do not prefix it with a tilde (`~`), IDOL Server matches it exactly but not case-sensitively (that is, the query `"Love1y"` matches documents that contain, for example, *Lovely*, *lovely*, *lOveLy*, and so on).

If you prefix a word with a tilde (`~`) and do not put it into quotation marks, IDOL Server matches it conceptually and case-sensitively (that is, the query `Love1y` matches documents that contain, for example, *Lovely*, *Love*, *Loved*, *Loving*, and so on).

For more information about case-sensitive searching, refer to *IDOL Expert*.

Paragraph and Sentence Keyword Search

To find documents that contain groups of keywords within the same sentence or paragraph, enable the `AdvancedPlus` setting before you index content into IDOL Server. This parameter also enables the functionality of `AdvancedSearch` and `AdvancedCaseSearch`.

To query for keywords in the same paragraph or sentence by using the `PARAGRAPH` and `SENTENCE` operators, set `AdvancedPlus` to `True` in the `[Server]` section of the IDOL Server configuration file before you index content.

For example, if you query IDOL Server with `cat SENTENCE dog`, it returns only documents that contain the word *cat* and the word *dog* in the same sentence. If you query IDOL Server with `cat PARAGRAPH dog`, it returns only documents that contain the word *cat* and the word *dog* in the same paragraph.

Keyword Search Examples

The word *lovely* stems to *love*. The following words also stem to *love*.

Tip: To determine the stem of a word, use the `TermGetInfo` action. To determine the words to which a stem expands, use the `TermExpand` action.

LOVE	LOVELY	LOVING
------	--------	--------

LOVED	LOVES	LOVINGLY
LOVELIES	LOVEST	LOVINGS

Example 1 (Conceptual)

action=Query&Text=Lovely

- Default matching
- AdvancedSearch=True
- AdvancedCaseSearch=True

IDOL Server matches this query conceptually, in the default configuration or when you set `AdvancedSearch` or `AdvancedCaseSearch` to `True`. The query word is neither in quotation marks nor prefixed with a tilde (~), so IDOL Server finds documents that contain words that have the same stem as *lovely* (ignoring their case).

Example matching words	Example nonmatching words
lovely	lover
love	lovelorn
loved	loveless
loving	

Example 2 (Conceptual or Exact)

action=Query&Text="Lovely"

- Default matching

In the IDOL Server default configuration, IDOL Server ignores the quotation marks and matches the word conceptually. It finds documents that contain words that have the same stem as *lovely* (ignoring their case).

Example matching words	Example nonmatching words
lovely	lover
love	lovelorn
loved	loveless
loving	

- `AdvancedSearch=True` or `AdvancedCaseSearch=True`

If you set `AdvancedSearch` or `AdvancedCaseSearch` to `True`, IDOL Server matches the term exactly without stemming, because the query word is in quotation marks. It finds only documents that contain the word *lovely*. Because the word is not prefixed with a tilde (~), IDOL Server ignores its case.

Example matching words	Example nonmatching words
lovely	love
	loved
	loving
	lover

Example 3 (Conceptual)

action=Query&Text=~Lovely

- Default matching

In the default configuration, IDOL Server ignores the tilde (~) and matches the word conceptually. It finds documents that contain words that have the same stem as *lovely* (ignoring their case).

Example matching words	Example nonmatching words
lovely	lover
love	lovelom
loved	loveless
loving	

- AdvancedSearch=True

If you set *AdvancedSearch* to *True*, IDOL Server ignores the tilde (~) and matches the word conceptually, because the query word is not in quotation marks. It finds documents that contain words that have the same stem as *lovely* (ignoring their case).

Example matching words	Example nonmatching words
lovely	lover
love	lovelom
loved	loveless
loving	

- AdvancedCaseSearch=True

If you set *AdvancedCaseSearch* to *True*, IDOL Server matches conceptually and case-sensitively, because the query word is prefixed with a tilde (~) but not in quotation marks. It finds documents that contain words that have the same stem as *Lovely*.

Example matching words	Example nonmatching words
Lovely	lovely
Love	love
Loved	loved
Loving	loving

Example 4 (Conceptual or Exact)

action=Query&Text="~Lovely"

- Default matching

In the default configuration, IDOL Server ignores the tilde (~) and the quotation marks, and matches the word conceptually. It finds documents that contain words that have the same stem as *lovely* (ignoring their case).

Example matching words	Example nonmatching words
lovely	lover
love	lovelom
loved	loveless
loving	

- AdvancedSearch=True

If you set `AdvancedSearch` to `True`, IDOL Server ignores the tilde (~) and matches the word exactly, without stemming. It finds only documents that contain the word *lovely* (ignoring its case).

Example matching words	Example nonmatching words
lovely	love
Lovely	loved
	loving

- AdvancedCaseSearch=True

If you set `AdvancedCaseSearch` to `True`, IDOL Server matches it exactly and case-sensitively because the query word is prefixed with a tilde (~) and in quotation marks. It finds only documents that contain the word *Lovely*.

Example matching words	Example nonmatching words
Lovely	lovely
	love

Example matching words	Example nonmatching words
	loved
	loving

Phrase Search

IDOL Server provides these phrase searching options:

- **Phrase occurrence search.** Use a phrase occurrence search to find documents that contain a certain number of occurrences of a phrase.
- **Default phrase search.** Put your search string in quotation marks to treat the string as a phrase and return only documents in which a matching phrase occurs (a phrase in a document qualifies as a match if it stems the same way as the query phrase).
- **Exact phrase search.** Enable *AdvancedSearch* before you index content into IDOL Server to find exact matches for terms or phrases. In this case, if you query IDOL Server with a term or a phrase in quotation marks, it matches them in their exact prestemmed form.
- **Case-sensitive exact phrase search.** Enable *AdvancedCaseSearch* before you index content into IDOL Server to find case-sensitive exact matches for terms or phrases. You can then query IDOL Server with a term or a phrase in quotation marks, to match them in their exact prestemmed form, or prefix a term with a tilde (~) to find only case-sensitive matches. For more information about case-sensitive searching, refer to *IDOL Expert*.

Phrase Occurrence Search

You can restrict the documents that match a query phrase to documents in which the number of occurrences of the phrase is within a specified range. Specify the range with a colon-separated pair of numbers. You can use a maximum value of 32000. For example:

```
action=Query&Text="Gene therapy"[5:8]
```

This query returns only documents in which the phrase *Gene therapy* appears between five and eight times (inclusive). This also applies for Wildcard terms.

You can specify open-ended ranges. For example:

```
action=Query&Text="Gene therapy"[10:]
```

This query returns only documents in which the phrase *Gene therapy* appears 10 or more times.

Default Phrase Search

If you query IDOL Server with a phrase in quotation marks, by default it returns only documents in which a matching phrase occurs (a phrase in a document qualifies as a match if it stems the same way as the query phrase).

For example:

```
action=Query&Text="fresh and lovely"
```

IDOL Server removes any stop words that the query contains (the example query above contains the stop word *and*) and applies stemming. It is as if the query were this:

```
action=Query&Text="fresh love"
```

When it matches the query, IDOL Server returns only documents that contain a phrase that stems the same way as the phrase in the query string. The query "fresh and lovely" can return only documents that contain a phrase that stems to *fresh love* (for example, *fresh love*, *freshest loving*, *fresh and lovely*, *freshly loved*, and so on).

Exact Phrase Search

If you enable `AdvancedSearch` before you index content into IDOL Server, and submit a phrase in quotation marks to IDOL Server, IDOL Server matches the phrase in its exact prestemmed form (this setting also switches IDOL Server to an advanced weighting algorithm that improves conceptual querying).

For example:

```
action=Query&Text="fresh and lovely"
```

IDOL Server removes any stop words that the query contains (the example query above contains the stop word *and*) but does not apply stemming. It is as if the query were:

```
action=Query&Text="fresh lovely"
```

When it matches the query, IDOL Server returns only documents that contain a phrase that matches the phrase in the query string. The query "fresh and lovely" returns only documents that contain a phrase that matches the phrase *fresh lovely* (for example, *fresh lovely*, *fresh and lovely*, *fresh or lovely*, and so on).

Case-Sensitive Exact Phrase Search

If you enable `AdvancedCaseSearch` before you index content into IDOL Server, and submit a phrase in quotation marks to IDOL Server, IDOL Server matches the phrase in its prestemmed form (the same way it matches if `AdvancedSearch` is enabled). If you prefix a term with a tilde (~), IDOL Server matches the term case-sensitively.

For example:

```
action=Query&Text="fresh and ~Lovely"
```

IDOL Server removes any stop words that the query contains (the example query above contains the stop word *and*) but does not apply stemming. It is as if the query were:

```
action=Query&Text="fresh ~Lovely"
```

When it matches the query, IDOL Server returns only documents that contain a phrase that matches the phrase in the query string. The query "fresh and ~Lovely" can return only documents that contain a phrase which matches the phrase *fresh Lovely* (for example, *fresh Lovely*, *fresh and Lovely*, *Fresh or Lovely*, and so on).

For more information about case-sensitive searching, refer to *IDOL Expert*.

Phrase Search Examples

The word *fresh* stems to *fresh*, and the word *lovely* stems to *love*. The following words also stem to *fresh* and *love*.

Tip: To determine the stem of a word, use the `TermGetInfo` action. To determine the words to which a stem expands, use the `TermExpand` action.

FRESH	FRESHEST	FRESHLY
FRESHES	FRESHING	FRESHNESS
LOVE	LOVELY	LOVING
LOVED	LOVES	LOVINGLY
LOVELIES	LOVEST	LOVINGS

Example 1 (Conceptual)

`action=Query&Text=fresh and Lovely`

- Default matching
- `AdvancedSearch=True`
- `AdvancedCaseSearch=True`

IDOL Server matches this query conceptually in all configurations, because the phrase is not in quotation marks and none of its words are prefixed with a tilde (~).

IDOL Server first finds documents that contain both words that have the same stem as *fresh* and words that have the same stem as *lovely* (ignoring their case), and then documents that contain either words that have the same stem as *fresh* or words that have the same stem as *lovely* (ignoring their case).

Example matching words	Example nonmatching words	Example matching words	Example nonmatching words
fresh	fresher	lovely	lover
freshness	freshman	love	lovelorn
freshly	freshwater	loved	loveless
freshest		loving	

Example 2 (Conceptual or Exact)

`action=Query&Text="fresh and Lovely"`

- Default matching

In the default configuration, the quotation marks indicate that this is a phrase search. IDOL Server removes any stop words in the phrase, and applies stemming. It finds only documents in which a word whose stem matches the stem of *fresh* occurs immediately before a word that matches the stem of *lovely* (ignoring their case).

Example matching phrases	Example nonmatching phrases
fresh lovely	freshman lover
freshest love	fresher lovelorn
freshly loved	lovely and fresh
fresh and lovingly	

- AdvancedSearch=True or AdvancedCaseSearch=True

If you set `AdvancedSearch` or `AdvancedCaseSearch` to `True`, IDOL Server removes any stop words that the phrase contains, and then matches it exactly without stemming, because the phrase is in quotation marks. Because none of the words are prefixed with a tilde (~), it finds only documents that contain the phrase *fresh lovely* (ignoring its case).

Example matching phrases	Example nonmatching phrases
fresh lovely	freshest love
fresh and lovely	freshly loved
fresh or lovely	fresh and lovingly
	lovely and fresh

Example 3 (Conceptual or Exact)

`action=Query&Text="fresh and ~Lovely"`

- Default matching

In the default configuration, the quotation marks indicate that this is a phrase search. IDOL Server ignores the tilde (~). It removes any stop words that the phrase contains, and applies stemming. It finds only documents in which a word whose stem matches the stem of *fresh* occurs immediately before a word that matches the stem of *lovely* (ignoring their case).

Example matching phrases	Example nonmatching phrases
fresh lovely	freshman lover
freshest love	fresher lovelorn
freshly loved	lovely and fresh
fresh and lovingly	

- `AdvancedSearch=True`

If you set `AdvancedSearch` to `True`, IDOL Server removes any stop words that the phrase contains and then matches it exactly without stemming, because the phrase is in quotation marks. It ignores the tilde (~). IDOL Server finds only documents that contain the phrase *fresh lovely* (ignoring its case).

Example matching phrases	Example nonmatching phrases
fresh lovely	freshest love
fresh and lovely	freshly loved
fresh or lovely	fresh and lovingly
	lovely and fresh

- `AdvancedCaseSearch=true`

If you set `AdvancedCaseSearch` to `True`, IDOL Server removes any stop words that the phrase contains and then matches it exactly without stemming, because the phrase is in quotation marks. Because *Lovely* is prefixed with a tilde (~), it is matched case-sensitively. IDOL Server finds only documents that contain the phrase *fresh Lovely*.

Example matching phrases	Example nonmatching phrases
fresh Lovely	fresh lovely
fresh and Lovely	fresh and lovely
Fresh and Lovely	fresh or lovely
Fresh or Lovely	Lovely fresh

Boolean and Proximity Search

You can use the `Query` action to submit standard Boolean searches to IDOL Server, and to submit proximity searches, which allow you to give words that appear close together in the search string a higher weighting.

Note: You must specify all operators in capital letters.

Boolean Search Operators

The following operators allow you to manipulate a query by applying them to words, exact phrases, or other Boolean expressions. IDOL Server uses APCM (Adaptive Probabilistic Concept Modeling) to rank the results that match the Boolean query.

Boolean search operators

Operator	Explanation
AND	<p>Binary operator. Ensures that every document that returns contains both terms. For example:</p> <pre>action=Query&Text=cat+AND+dog</pre> <p>This query returns only documents that contain both <i>cat</i> and <i>dog</i>.</p>
NOT	<p>Unary operator. Ensures that the term following NOT is excluded from all the returned documents. For example:</p> <pre>action=Query&Text=cat+NOT+dog</pre> <p>This query returns only documents that contain <i>cat</i> and not <i>dog</i>.</p> <p>Note: NOT applies only to the term that immediately follows it. To exclude multiple terms, place them in brackets. To exclude a phrase, put the phrase in quotation marks and in brackets. For example:</p> <p>Doc 1: <i>I went to the city for the New Year</i></p> <p>Doc 2: <i>I went to New York City for the New Year</i></p> <p>The following query does not match either of these documents:</p> <pre>action=Query&Text=city NOT (New York)</pre> <p>The following query matches the first document but not the second:</p> <pre>action=Query&Text=city NOT ("New York")</pre>
OR	<p>Binary operator. One or both terms must appear for the document to return. This is the default behavior if no explicit operator is given between two terms. For example:</p> <pre>action=Query&Text=cat+OR+dog</pre> <p>This query returns only documents that contain either <i>cat</i>, <i>dog</i>, or both terms.</p>
EOR or XOR	<p>Binary operator. Logical exclusive OR. Only one of the terms is permitted to appear for the document to return. This operator is rarely used. For example:</p> <pre>action=Query&Text=cat+XOR+dog</pre> <p>This query returns only documents that contain either the term <i>cat</i> or the term <i>dog</i>. Documents that contain both <i>cat</i> and <i>dog</i> do not return.</p>
()	<p>Bracketed expressions. These expressions are evaluated left to right and can be nested. They dictate the precedence and behavior of combined operator statements. For example:</p> <pre>action=Query&Text=(fish EOR pie) AND (chips EOR mash)</pre> <p>This query returns only documents that contain one of these combinations:</p> <p><i>fish</i> and <i>chips</i></p> <p><i>fish</i> and <i>mash</i></p>

Boolean search operators, continued

Operator	Explanation
	<i>pie and chips</i> <i>pie and mash</i>

Proximity Search Operators

You can apply proximity operators to words, exact phrases, or Boolean expressions to perform a proximity search. Note the following details:

- If the two specified words are adjacent to each other, their proximity is 1. If one word separates them, their distance is 2, and so on.
- Proximity operators do not count stop words. For example, because *and* is a stop word, the terms *cat* and *dog* have the proximity 1 in the text:
 - *catdog*
 - *cat and dog*.
- IDOL Server uses APCM (Adaptive Probabilistic Concept Modeling) to rank results.
- Proximity operators work recursively so that nested Boolean queries can have proximity operators apply to brackets or phrases. For example, in the expression

```
(term1) NEAR10 ((term2) DNEAR2 (term3))
```

the NEAR10 operator ensures that *term1* is in proximity to an occurrence of *term2* within two of *term3*.

Proximity search operators

Operator	Explanation
NEAR <i>N</i>	Returns only documents in which the second term is within <i>N</i> words of the first term—that is, the terms are <i>N</i> or fewer words apart. If you do not specify <i>N</i> , NEAR defaults to 5. For example: action=Query&Text=red+NEAR1+green This query returns only documents in which the term <i>red</i> is adjacent to the term <i>green</i> . For example, documents that contain <i>red green</i> or <i>green red</i> return. Documents that contain <i>red orange green</i> do not return (because the terms are not close enough to each other).
DNEAR <i>N</i>	Directed NEAR. Returns only documents in which the second term is within <i>N</i> words of the first term, in the specified order. If you do not specify <i>N</i> , DNEAR defaults to 5. For example: action=Query&Text=red+DNEAR2+green This query returns only documents in which the term <i>green</i> follows the term <i>red</i> , and is no more than two words away from the term <i>red</i> . For example, documents that contain <i>red orange green</i> return, but documents that contain <i>green orange red</i> or <i>red orange blue green</i> do not return.

Proximity search operators, continued

Operator	Explanation
WNEAR/N	<p>Weighted NEAR (with OR operation). This proximity operator returns documents that contain either of the two terms. It promotes relevance when the terms are <i>N</i> or fewer words apart (closer together implies higher relevance). If you do not specify <i>N</i>, WNEAR defaults to 5. For example:</p> <pre>action=Query&Text=dog+WNEAR7+cat</pre> <p>This query returns documents that contain either <i>dog</i> or <i>cat</i>. It gives extra relevance to documents in which <i>dog</i> and <i>cat</i> appear seven or fewer words apart in a piece of text. This weight increases as the terms get closer to each other. Documents in which the terms occur more than seven words apart, or in which only one term occurs, return with normal relevance.</p>
YNEAR/N	<p>Weighted NEAR (with AND operation). This proximity operator returns documents that contain both of the terms. It promotes relevance when the terms are <i>N</i> or fewer words apart (closer together implies higher relevance). If you do not specify <i>N</i>, YNEAR defaults to 5. For example:</p> <pre>action=Query&Text=dog+YNEAR7+cat</pre> <p>This query returns documents that contain both <i>dog</i> and <i>cat</i>. It gives extra relevance to documents in which <i>dog</i> and <i>cat</i> appear seven or fewer words apart in a piece of text. This weight increases as the terms get closer to each other. Documents in which the terms occur more than seven words apart return with the normal relevance.</p>
BEFORE	<p>Returns only documents in which the first term precedes the second one. For example:</p> <pre>action=Query&Text=red+BEFORE+green</pre> <p>This query returns only documents in which the term <i>green</i> appears later than the term <i>red</i>.</p> <p>You can also use BEFORE for <code>FieldText</code> queries. For a <code>FieldText</code> query to successfully compare two occurrences of the same field, you must set the <code>XMLFullStructure</code> configuration parameter to <code>True</code> in the <code>[Server]</code> section of the IDOL Server configuration file.</p>
AFTER	<p>Returns only documents in which the first term appears later than the second one. For example:</p> <pre>action=Query&Text=red+AFTER+green</pre> <p>This query returns only documents in which the term <i>red</i> appears later than the term <i>green</i>.</p> <p>You can also use AFTER for <code>FieldText</code> queries. For a <code>FieldText</code> query to successfully compare two occurrences of the same field, you must set the <code>XMLFullStructure</code> configuration parameter to <code>True</code> in the <code>[Server]</code> section of the IDOL Server configuration file.</p>
XNEAR	<p>Returns only documents in which the second term is exactly <i>N</i> words from the first term.</p>

Proximity search operators, continued

Operator	Explanation
	<p>For example:</p> <pre>action=Query&Text=cats+XNEAR2+dogs</pre> <p>This query returns only documents in which the term <i>dogs</i> follows the term <i>cats</i> and is exactly two words away from the term <i>cats</i>. This means that documents which contain <i>cats and dogs</i> return, but documents that contain <i>dogs and cats</i> or <i>cats, dogs</i> do not return.</p>
SENTENCE	<p>Returns only documents in which the second term is in the same sentence as the first term.</p> <p>For example:</p> <pre>action=Query&Text=cats+SENTENCE+dogs</pre> <p>This query returns only documents in which the term <i>dogs</i> appears in the same sentence as the word <i>cats</i>.</p>
SENTENCE <i>NN</i>	<p>Returns only documents in which the second term is in the same sentence as the first term, and they are within <i>NN</i> words of each other. For example:</p> <pre>action=Query&Text=cats+SENTENCE10+dogs</pre> <p>This query returns only documents in which the term <i>dogs</i> occurs in the same sentence as, and within 10 words of, the word <i>cats</i>.</p> <p>Note: SENTENCE0 has the same behavior as SENTENCE.</p>
DSENTENCE	<p>Returns only documents in which the second term occurs later than the first term, in the same sentence. For example:</p> <pre>action=Query&Text=cats+DSENTENCE+dogs</pre> <p>This query returns only documents in which the term <i>dogs</i> occurs later than the word <i>cats</i>, in the same sentence.</p>
DSENTENCE <i>NN</i>	<p>Returns only documents in which the second term occurs later than the first term, and within <i>NN</i> words in the same sentence. For example:</p> <pre>action=Query&Text=cats+DSENTENCE10+dogs</pre> <p>This query returns only documents in which the term <i>dogs</i> occurs later in the same sentence than, and within 10 words of, the word <i>cats</i>.</p> <p>Note: DSENTENCE0 has the same behavior as DSENTENCE.</p>
PARAGRAPH	<p>Returns only documents in which the second term is in the same paragraph as the first term.</p> <p>For example:</p> <pre>action=Query&Text=red+PARAGRAPH+green</pre>

Proximity search operators, continued

Operator	Explanation
	This query returns only documents in which the term <i>green</i> appears in the same paragraph as the word <i>red</i> . The words do not have to be in the same sentence in the paragraph.
PARAGRAPH <i>NN</i>	Returns only documents in which the second term is in the same paragraph as the first term, and they are within <i>NN</i> words of each other. For example: action=Query&Text=cats+PARAGRAPH20+dogs This query returns only documents in which the term <i>dogs</i> occurs in the same paragraph as, and within 20 words of, the word <i>cats</i> . <div style="background-color: #e0e0e0; padding: 5px;">Note: PARAGRAPH0 has the same behavior as PARAGRAPH.</div>

WHEN and NOTWHEN Search Operators

If you set `XMLFullStructure` to `True` in the configuration file, you can use the `WHEN` and `NOTWHEN` search operators to return documents in which XML fields occur with a common parent field. When you set `XMLFullStructure` to `True`, IDOL Server gives each occurrence of the same XML field name a different field code, so that it can identify each one uniquely.

To return only XML documents in which fields or attributes that have a specific value occur together, apply the `WHEN` operator to words or phrases. You can use the `WHEN` operator to return only those XML documents in which:

- two fields with the same parent field contain specified terms or phrases. See ["Example 1" below](#).
- two attributes that occur in the same field contain specified terms or phrases. See ["Example 2" on the next page](#).
- a field contains a specified term or phrase, and has an attribute with a specific value. See ["Example 3" on page 219](#).

To return only XML documents in which one field or attribute that has a specific value does not occur in the same parent field as another specified field or attribute and value, use the `NOTWHEN` operator. You can use the `NOTWHEN` operator in the same way as the `WHEN` operator. See ["Example 4" on page 219](#).

Example 1

You can use `WHEN` to return only XML documents in which two fields with the same parent field contain specified terms or phrases.

For example:

```
action=Query&Text=audi:make+WHEN+red:color
```

This query returns only XML documents in which the `make` and `color` fields are children of the same parent field, and contain the values *audi* and *red* respectively.

For example, this query returns the following document:

```
<XML>
  <car>
    <make>audi</make>
    <color>red</color>
  </car>
  <car>
>    <make>mercedes</make>
    <color>silver</color>
  </car>
</DOC>
</XML>
```

It does not return the following document:

```
<XML>
  <DOC>
    <car>
      <make>audi</make>
      <color>silver</color>
    </car>
    <car>
      <make>mercedes</make>
      <color>red</color>
    </car>
  </DOC>
</XML>
```

You can also use complex nested expressions with the WHEN operator in query text. For example:

```
action=Query&Text=(London:CITY WHEN English:LANG) WHEN ("United Kingdom":COUNTRY)
```

Example 2

You can use WHEN to return only XML documents in which two attributes that occur in the same field contain specified terms or phrases. For example:

```
action=Query&Text=English:_ATTR_LANG WHEN "Cape Town":_ATTR_CAPITAL
```

This query returns only XML documents in which the LANG and CAPITAL attributes occur in the same field, and contain the values *English* (in the LANG attribute) and *Cape Town* (in the CAPITAL attribute).

For example, this query returns the following document:

```
<XML>
  <DOC>
    <COUNTRY CAPITAL="Cape Town" LANG="English" POP="44">South Africa</COUNTRY>
    <COUNTRY CAPITAL="Berlin" LANG="German" POP="80">Germany</COUNTRY>
  </DOC>
</XML>
```

It does not return the following document:

```
<XML>
  <DOC>
    <COUNTRY CAPITAL="Cape Town" LANG="Afrikaans" POP="10">South Africa</COUNTRY>
    <COUNTRY CAPITAL="London" LANG="English" POP="80">England</COUNTRY>
  </DOC>
</XML>
```

Example 3

You can also use `WHEN` to return only XML documents in which a field contains a specified term or phrase, and has an attribute that has a specific value. For example:

```
action=Query&Text=Fr.html:_ATTR_HREF WHEN France:A
```

This query returns only XML documents in which an `A` field contains the value *France*, and has an `HREF` attributes with the value *Fr.html*.

For example, this query returns the following document:

```
<XML>
  <DOC>
    <A HREF="Fr.html">France</A>
  </DOC>
</XML>
```

Example 4

You can use `NOTWHEN` to return XML documents that contain one field with a specified term and phrase, but that do not contain another field and value in the same parent field.

```
action=Query&Text=audi:make+NOTWHEN+red:color
```

This query returns only XML documents in which the `make` field contains the value `audi`, under a parent field that does not also have a `color` field with the value `red`.

For example, this query returns the following document:

```
<XML>
  <DOC>
    <car>
      <make>audi</make>
      <color>silver</color>
    </car>
    <car>
      <make>mercedes</make>
      <color>red</color>
    </car>
  </DOC>
</XML>
```

It does not return the following document:

```
<XML>
  <DOC>
    <car>
      <make>audi</make>
      <color>red</color>
    </car>
    <car>
      <make>mercedes</make>
      <color>silver</color>
    </car>
  </DOC>
</XML>
```

Specify the Number of Levels from the XML Root

You can add a numeric value to the WHEN operator to indicate the number of hierarchical levels from the XML root from which to match the terms or phrases.

For example, if you have the following XML document:

```
<?xml version="1.0"?>
<XML>
  <DOCUMENT>
    <DREREFERENCE>Reference_1</DREREFERENCE>
    <car>
      <make>ford</make>
      <color>
        <exterior>blue</exterior>
        <interior>black</interior>
      </color>
    </car>
  </DOCUMENT>
  <DOCUMENT>
    <DREREFERENCE>Reference_2</DREREFERENCE>
    <car>
      <make>ferrari</make>
      <color>
        <exterior>blue</exterior>
        <interior>brown</interior>
      </color>
    </car>
    <car>
      <make>ford</make>
      <color>
        <exterior>yellow</exterior>
        <interior>blue</interior>
      </color>
    </car>
```

```
</DOCUMENT>  
</XML>
```

The following query returns both documents, because the `exterior` and `make` fields share the same parent field (`<DOCUMENT>`), which is two levels from the root, and contain *blue* and *ford* respectively.

```
action=Query&Text=blue:EXTERIOR+WHEN2+ford:MAKE
```

The following query returns only `Reference_1`, because the `exterior` and `make` fields share the same parent field (`<car>`), which is three levels from the root, and contain *blue* and *ford* respectively.

```
action=Query&Text=blue:EXTERIOR+WHEN3+ford:MAKE
```

The following query does not return either document, because the `exterior` and `make` fields do not share the same parent field four levels from the root.

```
action=Query&Text=blue:EXTERIOR+WHEN4+ford:MAKE
```

Precedence of Search Operators

Boolean and proximity operators have the following precedence:

```
Highest:      NOT  
              NEAR; DNEAR; XNEAR; YNEAR  
              AND; BEFORE; AFTER; WHEN; SENTENCE; DSENTENCE; PARAGRAPH  
Lowest:      OR; WNEAR; EOR
```

Operators that have the same level of precedence have neither left or right associativity. You can use brackets to bind terms together as appropriate. Proximity operators must have terms on either side, and cannot be adjacent to brackets.

Simple Field Restricted Search

You can use simple field restrictions within a `Query` action `Text` parameter to return results that contain specific values in specific fields. You can also combine query text with a field restriction to increase the relevance of results that contain specific values in specific fields. For these field restrictions:

- you can use only fields that IDOL Server stores as Index fields.
- you can use Wildcards.
- you cannot match more than one value, or a value that contains spaces or punctuation.
- each individual field restriction must contain fewer than 512 characters.

Related Topics

- ["Set up the Field Index Process" on page 38](#)

Example Queries

```
action=Query&Text=cat:DRETITLE
```

This query returns only documents that contain the value *cat* in their `DRETITLE` field.

```
action=Query&Text=cat dog:DRETITLE
```

This query returns documents that contain the term *cat* in any field and the term *dog* in their DRETITLE field. Documents that contain either *cat* (in any field) or *dog* in their DRETITLE field also return, but with a lower relevance.

```
action=Query&Text=cat:CREATURE:FAUNA dog:ANIMAL
```

This query returns only documents that contain the value *cat* in their CREATURE or FAUNA field and the value *dog* in their ANIMAL field. Documents that contain either *cat* in their CREATURE or FAUNA field or *dog* in their ANIMAL field also return, but with a lower relevance.

```
action=Query&Text=engin*:Title
```

This query returns only documents whose title field contains the specified string (for example, *engineer*, *engineering* and so on). Note that IDOL Server expands Wildcards before it stems terms.

FieldText Search

FieldText queries are Query, Suggest, or SuggestOnText actions that include the FieldText parameter. You can combine this parameter with the Text parameter to restrict the results that your query returns to documents that contain a specific value in a specified field. You can also use the FieldText parameter on its own to query for documents that contain specific values in specific fields. HPE does not recommend this type of query because it slows the IDOL Server processing speed.

To specify how documents must match fields and field values to return as results, the FieldText parameter uses field specifiers which identify the pattern of values in fields. These field specifiers fall into three groups, described in these sections:

- ["Field Specifiers for Common Restrictions" on page 224](#)
- ["Field Specifiers for Advanced Restrictions" on page 232](#)
- ["Field Specifiers to Bias Result Scores" on page 255](#)

Note: FieldText queries that include commas and braces within the query have specific percent-encoding requirements. For information about percent-encoding, see ["FieldText" on page 284](#).

FieldText Query Guidelines

- In addition to document fields, you can also apply FieldText restrictions to metafields, such as autn_date, autn_database, autn_section, autn_langtype, and autn_language. For example:

```
FieldText=STRING{Archiv}:autn_database
```

In this example, the autn_database metafield must contain the substring *Archiv* for this document to return. For example, if the autn_database metafield has the value *Archive* or *Archives*, this document returns.

```
FieldText=WILD{eng*}:autn_langtype
```

In this example, the autn_langtype metafield must contain a value that starts with *eng* (for example, *englishASCII* or *English_UTF8*) for this document to return as a result.

- You can use multiple field specifiers simultaneously by combining them with the Boolean operators

AND, NOT, and OR. For example:

```
FieldText=GREATER{6.95}:PRICE:PREIS+AND+MATCH{Brown}:AUTHOR:AUTOR
```

In this example, the PRICE or PREIS field must contain a number greater than 6.95, and the AUTHOR or AUTOR field must have the value *Brown*, for the document to return as a result.

```
FieldText=(LESS{10}:PRICE+AND+MATCH{Penguin}:PUBLISHER)+OR+(NRANGE  
{20,30}:PRICE+AND+MATCH{George Orwell}:AUTHOR)
```

In this example, the documents must meet one of these two criteria:

- the PRICE field contains a number smaller than 10, and its PUBLISHER field has the value *Penguin*.
 - the PRICE field contains a number between 20 and 30, and its AUTHOR field has the value *George Orwell*.
- You can use the proximity operators BEFORE and AFTER to specify the order in which certain fields must occur in a document. For example:

```
FieldText=FieldText=MATCH{Penguin}:PUBLISHER BEFORE MATCH{George Orwell}:AUTHOR
```

This example returns only documents where the PUBLISHER field contains the value *Penguin* earlier in the document than the AUTHOR field that contains the value *George Orwell*.

Note: For a FieldText query to successfully compare two occurrences of the same field, you must set the XMLFullStructure configuration parameter to True in the [Server] section of the configuration file.

```
FieldText=MATCH{George Orwell}:AUTHOR AFTER MATCH{Thomas Brown}:AUTHOR
```

This example returns only documents where one occurrence of the AUTHOR field contains the value *George Orwell* later in the document than another occurrence of the AUTHOR field that contains the value *Thomas Brown*. This query returns results only if you have set XMLFullStructure to True.

- When specifying fields, use the format /FieldName to match root-level fields, FieldName to match all fields except root-level, or /Path/FieldName to match fields that the specified path points to. To identify XML attributes, use the format <tag name>/_ATTR_<attribute name>, for example, FARM/_ATTR_ANIMAL. You can also use Wildcards when identifying fields, for example, /Fi*d1 or /Field*.
- All string matching is case insensitive, unless you set the CaseSensitive parameter to True (this parameter does not apply for TERM, TERMPHRASE, and TERMALL, which are always case insensitive). Strings can contain punctuation, except braces ({}). You must percent-encode ampersands (&) in strings as %26. To distinguish commas within strings from separator commas, percent-encode commas twice within strings (%252C).

You must not percent-encode commas that separate multiple items, and braces that start and end lists.

For example, to match the two strings hello,world and goodbye,again:

```
FieldText=MATCH{hello%252Cworld,goodbye%252Cagain}:FIELD
```

Related Topics

- ["Metadata Fields" on page 96](#)

Field Specifiers for Common Restrictions

Specifier	Finds documents in which a specified field contains...
MATCH	a value that exactly matches one or more specified strings
EQUAL, GREATER, LESS, NOTEQUAL, NRANGE	a number
GTNOW, LTNOW, or RANGE	a date
WILD	a value that matches a specified Wildcard string

Fields whose Value Exactly Matches One or More Strings

You can use the following field specifier to return documents with fields that contain a specified string.

MATCH

The MATCH field specifier (case sensitive) finds documents in which a specified field contains a value that exactly matches a specified string.

`FieldText=MATCH{yourStrings}:yourFields`

where:

<i>yourStrings</i>	is one or more strings. A document returns only if one of these strings is matched by one of <i>yourFields</i> exactly. The matching is case insensitive. FieldText queries which include commas and braces within the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284 .
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if the value in this field exactly matches one of <i>yourStrings</i> . To specify multiple fields, separate them with colons (there must be no space before or after a colon).

Examples:

`FieldText=MATCH{Archive,Web,docs}:DB:DATABASE`

The DB or DATABASE field must have the value *Archive*, *Web*, or *docs* for the document to return as a result.

`FieldText=MATCH{Premier league}:DB`

The DB field must have the value *Premier League* for the document to return as a result.

```
FieldText=MATCH{0-226-10389-7}:ISBN
```

The ISBN field must have the value *0-226-10389-7* for the document to return as a result.

Fields that Contain a Number

You can use the following field specifiers (case sensitive) to return documents with fields that contain numbers. To optimize the processing time of queries for fields that contain numbers, store them as numeric fields in IDOL Server during the indexing process.

Related Topics

- ["NumericType Fields" on page 87](#)

EQUAL

The EQUAL field specifier (case sensitive) allows you to find documents in which a specified field contains a number that matches one of the numbers specified by you.

```
FieldText=EQUAL{yourNumbers}:yourFields
```

where:

<i>yourNumbers</i>	is one or more numbers. A document returns only if one of <i>yourFields</i> contains one of these numbers.
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains one of <i>yourNumbers</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

Examples:

```
FieldText=EQUAL{1234567890123}:ACCOUNT:KONTO
```

The ACCOUNT or KONTO field must contain the number *1234567890123* for the document to return.

```
FieldText=EQUAL{3.9,4.9,7}:ID
```

The ID field must contain the number *3.9*, *3.90*, *4.9*, *4.90*, *7*, or *7.0* for the document to return.

GREATER

The GREATER field specifier (case sensitive) allows you to find documents in which a specified field contains a number that is greater than a number you specify.

```
FieldText=GREATER{yourNumber}:yourFields
```

where:

<i>yourNumber</i>	is a number. A document returns only if one of <i>yourFields</i> contains a number that is greater than this number. By default, the range is exclusive. You can add an equals sign (=) to include the number that you specify.
-------------------	--

<i>yourFields</i>	<p>is one or more fields. A document returns only if it contains one of these fields, and if the number in this field is greater than <i>yourNumber</i>.</p> <p>To specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>
-------------------	---

Examples:

FieldText=GREATER{66}:ID

The ID field must contain a number greater than 66 for the document to return.

FieldText=GREATER{5.59}:PRICE:PREIS

The PRICE or PREIS field must contain a number greater than 5.59 for the document to return.

FieldText=GREATER{=6}:QUANTITY

The QUANTITY field must contain a value of 6 or greater for the document to return.

LESS

The LESS field specifier (case sensitive) allows you to find documents in which a specified field contains a number that is smaller than a number that you specify.

FieldText=LESS{*yourNumber*}:*yourFields*

where:

<i>yourNumber</i>	<p>is a number. A document returns only if one of <i>yourFields</i> contains a number that is smaller than this number.</p> <p>By default, the range is exclusive. You can add an equals sign (=) to include the number that you specify.</p>
<i>yourFields</i>	<p>is one or more fields. A document returns only if it contains one of these fields, and if the number in this field is smaller than <i>yourNumber</i>.</p> <p>If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>

Examples:

FieldText=LESS{66}:ID

The ID field must contain a smaller number than 66 for the document to return.

FieldText=LESS{5.59}:PRICE:PREIS

The PRICE or PREIS field must contain a smaller number than 5.59 for the document to return.

FieldText=LESS{=6}:QUANTITY

The QUANTITY field must contain a value of 6 or lower for the document to return.

NOTEQUAL

The NOTEQUAL field specifier (case sensitive) allows you to find documents in which a specified field contains a number that does not match a number that you specify.

`FieldText=NOTEQUAL{yourNumber}:yourFields`

where:

<i>yourNumber</i>	is a number. A document returns only if one of <i>yourFields</i> does not contain this number.
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field does not contain <i>yourNumber</i> . To specify multiple fields, separate them with colons (there must be no space before or after a colon).

Examples:

`FieldText=NOTEQUAL{1234567890123}:ACCOUNT:KONTO`

The ACCOUNT or KONTO field must not contain the number *1234567890123* for the document to return.

`FieldText=NOTEQUAL{3.9}:ID`

The ID field must not contain the number *3.9* for this document to return.

NRANGE

The NRANGE field specifier (case sensitive) allows you to find documents in which a specified field contains a number that falls in the inclusive range of two numbers that you specify.

`FieldText=NRANGE{yourNumbers}:yourFields`

where:

<i>yourNumbers</i>	is two numbers, separated by a comma (there must be no space before or after the comma). A document returns only if one of <i>yourFields</i> contains a number that falls within the inclusive range of the specified numbers (including decimal numbers). By default, the range is inclusive. You can use the greater than (<) and less than (>) symbols to specify an exclusive range. You can make either or both ends of the range exclusive.
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains a number that falls within the inclusive range of <i>yourNumbers</i> . To specify multiple fields, separate them with colons (there must be no space before or after a colon).

Examples:

`FieldText=NRANGE{1,99}:CODE`

The CODE field must contain a number between *1* and *99* (inclusive) for the document to return.

```
FieldText=NRANGE{1234567890123,2345678901234}:ACCOUNT:KONTO
```

The ACCOUNT or KONTO field must contain a number between 1234567890123 and 2345678901234 (inclusive) for the document to return.

```
FieldText=NRANGE{36.5,42.3}:CODE
```

The CODE field must contain a number between 36.5 and 42.3 (inclusive) for the document to return.

```
FieldText=NRANGE{>1,5}:CODE
```

The CODE field must contain a value that is greater than 1, up to and including 5 for the document to return.

Fields that Contain a Date

You can use the following field specifiers (case sensitive) to return documents with fields that contain dates.

Note: To optimize the processing time of queries for fields that contain dates, store them as numeric date fields in IDOL Server during the indexing process.

Related Topics

- ["NumericDateType Fields" on page 86](#)

GTNOW

The GTNOW field specifier (case sensitive) allows you to find documents in which a specified field contains a date that is greater than the AUTNDATE (that is, all documents that were indexed with dates after the current time).

```
FieldText=GTNOW{yourFields}
```

where:

<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains a date that is greater than the AUTNDATE (that is, all documents that were indexed with dates after the current time). To specify multiple fields, separate them with colons (there must be no space before or after a colon).
-------------------	---

Examples:

```
FieldText=GTNOW{TIME}
```

The TIME field must contain a date that is greater than the AUTNDATE (that is, all documents that were indexed with dates after the current time) for the document to return.

```
FieldText=GTNOW{TIME:DATE}
```

The TIME or DATE field must contain a date that is greater than the current time (that is, all documents that were indexed with dates after the current time) for the document to return.

LTNOW

The LTNOW field specifier (case sensitive) allows you to find documents in which a specified field contains a date that is smaller than the AUTNDATE (that is, all documents that were indexed with dates before the current time).

FieldText=LTNOW{*yourFields*}

where:

<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains a date that is smaller than the current time. To specify multiple fields, separate them with colons (there must be no space before or after a colon).
-------------------	--

Examples:

FieldText=LTNOW{*}/TIME

The TIME field must contain a date that is smaller than the AUTNDATE (that is, all documents that were indexed with dates before the current time) for the document to return.

FieldText=LTNOW{*:*TIME:DATE

The TIME or DATE field must contain a date that is smaller than the AUTNDATE (that is, all documents that were indexed with dates before the current time) for the document to return.

RANGE

The RANGE field specifier (case sensitive) allows you to find documents in which a specified field contains a date that falls within the inclusive range of two dates that you specify.

FieldText=RANGE{*yourDates*}:*yourFields*

where:

<i>yourDates</i>	is two dates separated by a comma (there must be no space before or after the comma). A document returns only if one of <i>yourFields</i> contains a date that falls within the inclusive time span of the specified dates. You can use the formats in the table below to specify each date. By default, the range is inclusive. You can use the greater than (<) and less than (>) symbols to specify an exclusive range. You can make either or both ends of the range exclusive.
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains a date that falls within the inclusive range of <i>yourDates</i> . To specify multiple fields, separate them with colons (there must be no space before or after a colon).

Date formats

Format	Explanation
D+/M+/#YY+	A date. For example, 1/3/05, 23/12/1999, 10/07/40, or 8/5/2012. If the year is a number less than 40, it is read as a year in the 2000s. If the year is a number between 40 and 99, it is read as a year in the 1900s. For example, 01/02/01 is read as 1 February 2001 , and 01/03/40 is read as 1 March 1940 .
HH:NN:SS D+/M+/#YY+	A time and date. For example, 10:30:45 1/3/05, 18:55:00 23/12/99, 01:23:45 10/07/1940, or 07:15:00 8/5/2012. If the year is a number less than 40, it is read as a year in the 2000s. If the year is a number between 40 and 99, it is read as a year in the 1900s. For example, 01/02/01 is read as 1 February 2001 , and 01/03/40 is read as 1 March 1940 .
HH:NN:SS D+/M+/#YY+ #ADBC	A time and date with a time period. For example, 10:30:45 1/3/05 AD, 18:55:00 23/12/99 CE. For the time period, you can use AD, CE, BC, BCE or any predefined list of EPOCH indicators.
N	A positive or negative number of days from the current date. For example, -1 specifies yesterday's date, 0 specifies today's date, 1 specifies tomorrow's date, 2 specifies two days from now (the current date plus two), and so on.
Ns	A positive or negative number of seconds from now. For example, -60s specifies one minute ago, -900s specifies 15 minutes ago, -3600s specifies one hour ago and so on. 60s specifies one minute from now, 900s specifies 15 minutes from now, 3600s specifies one hour from now, and so on.
Ne	Epoch seconds (seconds since 1 January 1970). For example, 1012345000e specifies 22:56:40 on 29 January 2002. Note: IDOL Server uses the local time zone to match epoch values. For example, if your local time zone is GMT-6, 1012345000e is 03:56:40 on 29 January 2002. If you use this value as the beginning of a range, IDOL Server retrieves documents after 03:56:40 on 29 January 2002.

For RANGE, you can specify an open-ended range by using a period (.):

- If you type a period for the first point in time, the beginning of the time period is not restricted (the period ranges up to the specified date, including any earlier date).
- If you type a period for the second point in time, the end of the time period is not restricted (the period ranges from the specified date, including any later date).

Examples:

FieldText=RANGE{01/01/90,1/1/01}:DATE

The DATE field must contain a date between 01/01/1990 and 1/1/2001 for the document to return.

FieldText=RANGE{01/01/02,01/01/2003}:DATE:DATUM

The DATE or DATUM field must contain a date between 01/01/2002 and 01/01/2003 for the document to return.

```
FieldText=RANGE{-14,-7}:DATE
```

The DATE field must contain a date 14 to 7 days before the current date for the document to return.

```
FieldText=RANGE{0,1}:DATE
```

The DATE field must contain today's or tomorrow's date (which is possible, for example, if the document originates from a different time zone or if the field contains an expiration date) for the document to return.

```
FieldText=RANGE{01/01/99,.}:DATE:FECHA
```

The DATE or FECHA field can contain any date after 01/01/1999 for the document to return.

```
FieldText=RANGE{.,10/10/04}:DATE
```

The DATE field can contain any date before 10/10/2004 for the document to return.

```
FieldText=RANGE{-172800s,-1}:DATE
```

The DATE field must contain a time between 48 and 24 hours ago.

```
FieldText=RANGE{198765e,.}:DATE
```

The DATE field must contain a date between 198765 seconds after the epoch and the current time.

```
FieldText=RANGE{>05/06/2013,05/06/2014}:DATE
```

The DATE field must contain a date after 05/06/2013, up to and including 05/06/2014.

Fields whose Value Matches Wildcard Strings

WILD

The WILD field specifier (case sensitive) allows you to find documents in which a specified field contains a string that matches a specified Wildcard string.

If the query does not contain any Wildcard characters (? or *), the WILD field specifier acts in the same way as the MATCH field specifier.

```
FieldText=WILD{yourStrings}:yourFields
```

where:

<i>yourStrings</i>	is one or more strings that contain Wildcards. A document returns only if one of <i>yourFields</i> matches one of these strings. FieldText queries which include commas and braces within the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284 .
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains one of <i>yourStrings</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

Examples:

FieldText=WILD{*.html,*.htm}:URL

The URL field value must end with *.html* or *.htm* for this document to return as a result.

FieldText=WILD{passi*incarnata}:Climbers:Plants

The Climbers or Plants field value must contain a phrase that begins with *passi* and ends with *incarnata* (for example, *passionflower incarnata* or *passiflora incarnata*) for this document to return as a result.

FieldText=WILD{*www.example.com*.txt}:PATH

The PATH field value must contain a path that contains *www.example.com* and ends with *.txt* (for example, *http://www.example.com/files/doc.txt*) for the document to return as a result.

FieldText=WILD{wom?n }:Clothes

The Clothes field value must contain a word that matches the specified Wildcard string (for example, *woman* or *women*) for this document to return as a result.

Field Specifiers for Advanced Restrictions

Field specifiers

Specifier	Use to find documents in which a specified field...
ARANGE	contains a value that falls within a specific alphabetical range
BITAND, BITANDHEX, BITANDOFFHEX	contains a value that results in a nonzero value when a bitwise AND operation is carried out against it
BITSET	contains a value in a BitFieldType field where the specified bit is set
BOOLEANFIELD	contains a Boolean agent
DISTCARTESIAN	contains Cartesian (x/y) coordinates values within a specified distance from a specified point
DISTSPHERICAL	contains latitude and longitude values within a specified distance from a specified point
EMPTY	does not exist or does not contain a value
EXISTS	exists, irrespective of its value
FUZZY	contains a value that is similar to a specified string
MATCHALL or EQUALALL	contains multiple instances, whose values include at least one match for each of the specified strings or numeric values
MATCHCOVER or EQUALCOVER	contains multiple instances, all of whose values are

Field specifiers, continued

Specifier	Use to find documents in which a specified field...
	matched in the specified strings or numeric values
MATCHRECURSE	contains a specified reference in a ReferenceMemoryMappedType field recursively to a maximum number of times
NOTMATCH, NOTSTRING, NOTWILD	contains multiple instances, at least one of whose values does not match the specified string
POLYGON	contains Cartesian (x/y) coordinate values within a specified polygonal shape
STRING, STRINGALL, SUBSTRING	contains a specified string
TERM, TERMALL, TERMEXACT, TERMEXACTALL, TERMEXACTPHRASE, TERMPHRASE	whose value match specific terms or phrases

Fields whose Value Falls within a Specific Alphabetical Range

ARANGE

The ARANGE field specifier (case sensitive) allows you to find documents in which a specified field contains a term that falls within the inclusive alphabetical range of two terms that you specify.

FieldText=ARANGE{*yourTerms*}:*yourFields*

where:

<i>yourTerms</i>	<p>is two terms separated by a comma (there must be no space before or after the comma). A document returns only if one of <i>yourFields</i> contains a term that falls within the inclusive alphabetical range of the specified terms.</p> <p>You can use a period (.) in place of one of the terms to represent an unrestricted value.</p> <ul style="list-style-type: none"> • If you use the period in place of the first term, it includes all values up to the second term. • If you use the period in place of the second term, it includes all values after the first term. <p>It uses unicode tables to determine alphabetical order. This means that non-7-bit ASCII characters (α, ä, å, ç, ð, ê, ë, ø, ö, ü, û, ß, ÿ, and so on) come after z in the alphabet.</p>
<i>yourFields</i>	<p>is one or more fields. A document returns only if it contains one of these fields, and if this field contains a term that falls within the inclusive alphabetical range of <i>yourTerms</i>.</p> <p>To specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>

Examples:

```
FieldText=ARANGE{aardvark,alligator}:ANIMAL
```

The ANIMAL field must contain a value that alphabetically falls between *aardvark* and *alligator*. If the ANIMAL field contains the value *aardvark*, *ant*, *anteater*, *antelope*, or *alligator*, the document returns. If the ANIMAL field contains the value *armadillo*, it does not return.

```
FieldText=ARANGE{bear,buffalo}:ANIMAL:TIER
```

The ANIMAL or TIER field must contain a value that alphabetically falls between *bear* and *buffalo*. If the field contains the value *bear*, *bee*, *Biene*, *bird*, or *buffalo*, the document returns. If the ANIMAL field contains the value *Büffel* or *chipmunk*, it does not return.

```
FieldText=ARANGE{dog,.}:ANIMAL AND ARANGE{.,cat}:PET
```

The ANIMAL field must contain a value that alphabetically falls after *dog*, and the PET field must contain a value that alphabetically falls before *cat*.

Fields with a Nonzero Value for Bitwise AND

You can use the following field specifiers (case sensitive) to return documents with fields whose value result in a nonzero value when a bitwise AND operation is carried out against a specified value.

BITAND

The BITAND field specifier (case sensitive) allows you to find documents with a field whose integer value does not result in 0 when a bitwise AND operation is carried out between this value and an integer value that you specify.

```
FieldText=BITAND{yourInteger}:yourBitFields
```

where:

<i>yourInteger</i>	is an integer. A document returns only if one of <i>yourBitFields</i> contains a value that results in a nonzero value when a bitwise AND operation is carried out between this value and the specified integer.
<i>yourBitFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains an integer that results in a nonzero value when a bitwise AND operation is carried out between it and <i>yourInteger</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

Example:

```
FieldText=BITAND{128}:BitField
```

The binary representation of the integer value 128 is compared with the binary representations of the integer values that BitField fields in IDOL Server contain. Only documents whose BitField values result in a nonzero value when they are compared to the binary representation of 128 return.

For example, if the BitField of a document contains the integer value 129, it returns, but a document whose BitField contains the value 127 does not return.

The following tables show the field value comparison.

Integer	Binary	
128	1000 0000	
129	1000 0001	
	1000 0000	this evaluates to True

Integer	Binary	
128	1000 0000	
127	0111 1111	
	0000 0000	this evaluates to False

BITANDHEX

The BITANDHEX field specifier (case sensitive) allows you to find documents with a field whose hexadecimal string value does not result in zero when a bitwise AND operation is carried out between this value and a hexadecimal string that you specify.

`FieldText=BITANDHEX{yourHexString}:yourBitFields`

where:

<i>yourHexString</i>	is a hexadecimal string. A document returns only if one of <i>yourBitFields</i> contains a value that results in a nonzero value when a bitwise AND operation is carried out between this value and the specified hexadecimal string.
<i>yourBitFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains a hexadecimal string that results in a nonzero value when a bitwise AND operation is carried out between it and <i>yourHexString</i> . To specify multiple fields, separate them with colons (there must be no space before or after a colon).

Example:

`FieldText=BITANDHEX{7F}:BitField`

The binary representation of the hexadecimal value 7F is compared with the binary representations of the hexadecimal values that BitField fields in IDOL Server contain. Only documents whose BitField values result in a nonzero value when they are compared to the binary representation of 7F return.

For example, if the BitField of a document contains the hexadecimal value C0, it returns, but a document whose BitField contains the hexadecimal value 80 does not return.

The following tables show the field value comparison.

Hex	Binary	
-----	--------	--

7F	0111 1111	
C0	1100 0000	
	0100 0000	this evaluates to True

Hex	Binary	
7F	0111 1111	
80	1000 0000	
	0000 0000	this evaluates to False

BITANDOFFHEX

The BITANDOFFHEX field specifier (case sensitive) allows you to find documents with a field whose hexadecimal string value does not result in zero when a bitwise AND operation is carried out between this value and an offset hexadecimal string you specify.

`FieldText=BITANDOFFHEX{NN,yourHexString}:yourBitFields`

where:

<i>NN</i>	is the number of 16-bit chunks by which the value in <i>yourHexString</i> and in <i>yourBitFields</i> is shifted before the bitwise AND operation is carried out (this allows you to store sparse bit masks more efficiently).
<i>yourHexString</i>	is a hexadecimal string. A document returns only if one of <i>yourBitFields</i> contains a value that results in a nonzero value when a bitwise AND operation is carried out between this value and the specified hexadecimal string.
<i>yourBitFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains a hexadecimal string that results in a nonzero value when a bitwise AND operation is carried out between it and <i>yourHexString</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

Example:

`FieldText=BITANDOFFHEX{01,0a001}:BitOffField`

The binary representation of the hexadecimal value 01,0a001 is compared with the binary representations of the hexadecimal values that BitOffField fields in IDOL Server contain. Only documents whose BitOffField values result in a nonzero value when they are compared to the binary representation of 01,0a001 (after they have been shifted left by one 16-bit chunk) return.

For example, if the BitOffField contains the value 1,bc01, the document returns, but a document whose BitOffField contains the value 0,5ffeffff does not return.

The following tables show the field value comparison.

nn,hexstring	Hex	Binary
--------------	-----	--------

01,0a001	A0010000	1010 0000 0000 0001 0000 0000 0000 0000	
1,bc01	BC010000	1011 1100 0000 0001 0000 0000 0000 0000	
		1010 0000 0000 0001 0000 0000 0000 0000	this evaluates to True

nn,hexstring	Hex	Binary	
01,0a001	A0010000	1010 0000 0000 0001 0000 0000 0000 0000	
0,5ffeffff	5FFEFFFF	0101 1111 1111 1110 1111 1111 1111 1111	
		0000 0000 0000 0000 0000 0000 0000 0000	this evaluates to False

Fields that Contain BitFieldType Information

BITSET

The BITSET field specifier (case sensitive) allows you to find documents with a BitFieldType field that contains a value where the specified bit is set. This allows you to search for documents that belong to a particular set of documents.

Note: You can use the BITSET field specifier only for BitFieldType fields.

FieldText=BITSET{*yourBitSets*}:*yourFields*

where:

<i>yourBitSets</i>	are one or more set numbers, separated by commas. These set numbers must be in normal decimal form.
<i>yourFields</i>	are one or more BitFieldType fields. A document returns only if it contains one of these fields, and if this field contains a value where the bit corresponding to <i>yourBitSets</i> is set. To specify multiple fields, you must separate them with colons (there must be no space before or after a colon).

Examples:

FieldText=BITSET{0,4}:BitField

This example matches any document where the binary representation of the hexadecimal value contained in BitField has the bit set (the binary digit is 1) for set 0 or set 4.

For example, if a document has a BitField that contains the hexadecimal value A4, it does not match the query. The binary representation is 10100100, where the bits for set 0 and 4 are both 0, so the document is not part of these sets.

If the document has a BitField that contains the hexadecimal value A8, it does match the query. The binary representation is 10101000, where the bit for set 4 is a 1, so the document is part of set 4.

FieldText=BITSET{2}:BitField AND BITSET{15}:BitField

The binary representation of the hexadecimal value contained in `BitField` must have the bits switched on for both sets 2 and 15.

For example, if a document has a `BitField` that contains the hexadecimal value `B110`, it does not match the query. The binary representation is `1011000100010000`, where the bit is set to 1 for set 15, but not for set 2.

If the document has a `BitField` that contains the hexadecimal value `B114` it matches the query. The binary representation is `1011000100010100`, where the bit is set to 1 for both set 15 and set 2.

Fields whose Values are Boolean Agents

BOOLEANFIELD

The `BOOLEANFIELD` field specifier (case sensitive) allows you to find documents in which a specified Boolean agent field contains an expression that matches text you specify. A Boolean agent is a Boolean or proximity expression that legacy technologies use to categorize documents.

Note: If you are using a `Query` action, HPE recommends that you use the `AgentBooleanField` action parameter rather than the `BOOLEANFIELD` field specifier. However, if you want to match more than one Boolean agent field, you must use the `BOOLEANFIELD` field specifier.

`FieldText=BOOLEANFIELD{yourText}:yourFields`

where:

<i>yourText</i>	is text. A document returns only if one of <i>yourFields</i> contains a Boolean or proximity expression that matches the specified text.
<i>yourFields</i>	is one or more Boolean agent fields. A document returns only if it contains one of these fields, and if this field contains a Boolean or proximity expression that matches <i>yourText</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

For example:

`BOOLEANFIELD{The cat sat on the mat}:MyFirstBooleanField:MySecondBooleanField`

Any document that has a `MyFirstBooleanField` or `MySecondBooleanField` field that contains a Boolean or proximity expression that matches the specified text returns. For example, the Boolean and proximity expressions `cat AND mat`, `cat OR mat`, `cat BEFORE mat`, and `cat DNEAR1 sat` could match *The cat sat on the mat*. Therefore, documents that contain any of these Boolean and proximity expressions are returned.

Documents whose `MyFirstBooleanField` or `MySecondBooleanField` fields contain, for example, `cat AND mat AND dog` or `mat BEFORE cat` are not returned.

Fields that are within a Specified Distance from a Specified Point

DISTCARTESIAN

The DISTCARTESIAN field specifier allows you to find documents whose X/Y position is within a specified distance from a specified point.

FieldText=DISTCARTESIAN{*coordX*,*coordY*,*dist*}:X:Y

where:

<i>coordX</i>	is the specified X coordinate.
<i>coordY</i>	is the specified Y coordinate.
<i>dist</i>	is the distance in kilometers from the specified coordinates.
X	is the document field that contains the X coordinate.
Y	is the document field that contains the Y coordinate.

Note: You must specify two fields in the order X:Y.

Example:

FieldText=DISTCARTESIAN{10,11,5}:X:Y

This example matches all documents whose (X,Y) position is within a distance of 5 units of the point (10,11).

DISTSPHERICAL

FieldText=DISTSPHERICAL{*lat*,*long*,*dist*}:LATFIELD:LONGFIELD

where:

<i>lat</i>	is the latitude. Specify latitude positions south of the equator as negative.
<i>long</i>	is the longitude. Specify longitude positions west of the Greenwich Meridian as negative.
<i>dist</i>	is the distance in kilometers from the specified latitude and longitude.
LATFIELD	is the document field that contains the latitude.
LONGFIELD	is the document field that contains the longitude.

Note: You must specify two fields in the order latitude:longitude.

Example:

FieldText=DISTSPHERICAL{37.75,-122.4,20}:LAT:LONG

This example matches all documents whose position is within a 20 kilometer radius of San Francisco (37.75N, 122.4W). The latitude and longitude position of a document in this example is contained in the fields LAT and LONG, respectively.

Fields that Do Not Exist or Contain No Value

EMPTY

The EMPTY field specifier (case sensitive) allows you to find documents in which a specified field does not exist or contains no value.

`FieldText=EMPTY{}:yourFields`

where:

<i>yourFields</i>	is one or more fields. A document returns only if it does not contain any of these fields or if these fields are empty. To specify multiple fields, separate them with colons (there must be no space before or after a colon).
-------------------	--

Examples:

`FieldText=EMPTY{}:ID`

A document must not contain an ID field, or must hold no value in its ID field, to return.

`FieldText=EMPTY{}:ID:Name`

A document must not contain an ID or Name field, or must hold no value in its ID or Name field, to return.

Specific Fields, Irrespective of their Value

EXISTS

The EXISTS field specifier (case sensitive) allows you to find documents that contain a specified field even if this field contains no value.

`FieldText=EXISTS{}:yourFields`

where:

<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields (even if the field is empty). To specify multiple fields, separate them with colons (there must be no space before or after a colon).
-------------------	--

Examples:

`FieldText=EXISTS{}:ID`

A document must contain an ID field to return.

`FieldText=EXISTS{}:ID:NAME`

A document must contain an ID or NAME field (or both) to return.

Fields whose Values are Similar to a Specified String

FUZZY

The FUZZY field specifier (case sensitive) allows you to find documents in which a specified field contains a term that is similar to a specified term or phrase.

```
FieldText=FUZZY{yourTerms}:yourFields
```

where:

<i>yourTerms</i>	is one or more terms (or phrases). A document returns only if one of these terms (or phrases) is similar to a string in one of <i>yourFields</i> . FieldText queries which include commas and braces within the query have specific percent-encoding requirements. For information about percent-encoding, see " FieldText " on page 284.
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if the value in this field is similar to one of <i>yourTerms</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

Example:

```
FieldText=FUZZY{Business News,Arkive}:DRETITLE
```

The DRETITLE field value must be similar to the term *Business News*, or *Arkive* for the document to return. For example, a document whose DRETITLE field contains *Business News* returns, but a document whose DRETITLE field contains *Document Arkive* does not.

At Least One Field Instance Matches a Specified String or Number

You can use the following field specifiers (case sensitive) to return documents with multiple instances of the same fields and at least one field instance contains a specified string or number.

MATCHALL

The MATCHALL field specifier (case sensitive) allows you to find documents in which a specified field occurs in multiple instances, and in which there is at least one match among those instances for each of a set of strings that you specify.

Note: You can optimize the field specifier speed by restricting the field to the MatchType property type.

```
FieldText=MATCHALL{yourStrings}:yourField
```

where:

<i>yourStrings</i>	is one or more strings. A document returns only if all these strings have exact
--------------------	---

	<p>matches among the instances of <i>yourField</i>. Separate the strings with commas (there must be no space before or after a comma).</p> <p>FieldText queries which include commas and braces within the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284.</p>
<i>yourField</i>	<p>is the name of the field to match against. A document returns only if it contains the field and only if all <i>yourStrings</i> are matched at least once in various instances of the field.</p>

Examples:

```
FieldText=MATCHALL{Archive,Web,docs}:DIRECTORY
```

The DIRECTORY fields must include at least the values *Archive* and *Web* and *docs* for the document to return as a result.

```
FieldText=MATCHALL{Smith,Garcia,Lee}:SURNAME
```

The values *Smith*, *Garcia*, and *Lee* must all have matches in the SURNAME fields for the document to return as a result.

```
FieldText=MATCHALL{Smith%5C, John, Garcia%5C, Joaquin}:FULLNAME
```

The values *Smith*, *John* and *Garcia*, *Joaquin* must both have matches in the FULLNAME fields for the document to return as a result.

EQUALALL

The EQUALALL field specifier (case sensitive) allows you to find documents in which a specified field occurs in multiple instances, and in which there is at least one value among those instances that is equal to each of a set of numeric values that you specify.

Note: You can optimize the field specifier speed by restricting the field to the NumericType property type.

```
FieldText=EQUALALL{yourValues}:yourNumericField
```

where:

<i>yourValues</i>	<p>is one or more numeric values. A document returns only if all these values occur among the instances of <i>yourNumericField</i>. Separate the numbers with commas (there must be no space before or after a comma).</p>
<i>yourNumericField</i>	<p>is the name of the field to match against. A document returns only if it contains the field and only if all <i>yourValues</i> are matched at least once in various instances of the field.</p>

Example:

```
FieldText=EQUALALL{32,98.6,212}:FAHRENHEIT
```

The FAHRENHEIT fields must include at least the values 32, 98.6, and 212 for the document to return as a result.

```
FieldText=EQUALALL{1999,2000,2001}:YEAR
```

The values *1999*, *2000*, and *2001* must all appear in YEAR fields for the document to return as a result.

All Field Instances Match a Specified String or Number

You can use the following field specifiers (case sensitive) to return documents with multiple instances of the same fields and all field instances contain a specified string or number.

MATCHCOVER

The MATCHCOVER field specifier (case sensitive) allows you to find documents in which the values in all instances of a specified field have matches in the set of values provided in the specifier. In other words, the specifier must *cover* all instances of the field. A search that uses MATCHCOVER is slower than one that uses MATCH.

Note: You can optimize the field specifier speed by restricting the field to the MatchType and CountType property types. You must specify both property types.

```
FieldText=MATCHCOVER{yourStrings}:yourField
```

where:

<i>yourStrings</i>	is one or more strings. A document returns only if the value in each of its instances of <i>yourField</i> matches one of the strings in <i>yourStrings</i> . Separate the strings with commas (there must be no space before or after a comma). FieldText queries which include commas and braces within the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284 .
<i>yourField</i>	is the name of the field to match against. A document returns only if: <ul style="list-style-type: none">• it contains one or more instances of the field and the value of each instance is found in <i>yourStrings</i>.• it does not contain the field at all.

Example:

```
FieldText=MATCHCOVER{Confidential,Secret,TopSecret,FBI}:SECURITYLEVEL
```

For a document to return as a result, its SECURITYLEVEL fields must not contain any values that are not in the specified list. For example, if a document includes a SECURITYLEVEL field with the value *MI5*, it does not return. (If a document has no SECURITYLEVEL field at all, it returns.)

EQUALCOVER

The EQUALCOVER field specifier (case sensitive) allows you to find documents in which the values in all instances of a specified field are found in the set of values provided in the specifier. In other words, the specifier must *cover* all instances of the field.

Note: You can optimize the field specifier speed by restricting the field to the NumericType and CountType property types. You must specify both property types.

```
FieldText=EQUALCOVER{yourValues}:yourField
```

where:

<i>yourValues</i>	is one or more numeric values. A document returns only if the value in each of its instances of <i>yourField</i> equals one of the values in <i>yourValues</i> . Separate the numbers with commas (there must be no space before or after a comma). FieldText queries which include commas and braces within the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284 .
<i>yourField</i>	is the name of the field to match against. A document returns only if: <ul style="list-style-type: none"> • it contains one or more instances of the field and the value of each instance equals a value in <i>yourValues</i>. • it does not contain the field at all.

Example:

```
FieldText=EQUALCOVER{9,10,11,12}:GRADELEVEL
```

For a document to return as a result, its GRADELEVEL fields must have no values that are not in the specified list. For example, if a document includes a GRADELEVEL field with the value 8, it does not return. (If a document has no GRADELEVEL field, it returns.)

Fields that Contain a Specified ReferenceMemoryMappedType Field

MATCHRECURSE

The MATCHRECURSE field specifier matches documents that contain a specified reference in a ReferenceMemoryMappedType field recursively to a maximum number of times. You must restrict this field specifier to a single ReferenceMemoryMappedType field. It has the following syntax:

```
action=Query&FieldText=MATCHRECURSE{Ref,RecurseNumber}:yourField
```

where:

<i>Ref</i>	is the initial reference.
<i>RecurseNumber</i>	is the maximum number of times to recursively return references by using the value of the ReferenceMemoryMappedType field.
<i>yourField</i>	is the name of the ReferenceMemoryMappedType field.

For example, if you define PARENT as a ReferenceMemoryMapped field, the query

```
action=Query&FieldText=MATCHRECURSE{MyRef,1}:PARENT
```

matches the document with the reference *MyRef* (parent) and documents whose PARENT field contains *MyRef* (children). The query

```
action=Query&FieldText=MATCHRECURSE{MyRef,2}:PARENT
```

matches the document with the reference *MyRef* (parent), documents whose PARENT field contains *MyRef* (children), and documents whose PARENT field contains the references in the returned child documents (grandchildren).

Fields that Do Not Contain a Specified Value

NOTMATCH

The NOTMATCH field specifier (case sensitive) allows you to find documents in which at least one instance of the specified fields contains a value that does not match the specified string.

If there are one or more instances of a particular field in the document, the document returns as long as at least one instance does not contain any of the specified strings, even if another instance of the field does match. The document does not return if all instances of the specified fields contain an exact match of one of the specified strings.

`FieldText=NOTMATCH{yourStrings}:yourFields`

where:

<i>yourStrings</i>	is one or more strings. A document returns only if at least one instance of one of <i>yourFields</i> contains a value that is not an exact match for these strings. The matching is case insensitive. FieldText queries which include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284 .
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if the value in at least one instance of the field does not exactly match any of <i>yourStrings</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

For example:

`FieldText=NOTMATCH{cat}:ANIMAL`

At least one instance of the ANIMAL field must have a value other than `cat` for the document to return as a result.

For example, if a document contains only:

```
#DREFIELD ANIMAL="cat"
```

it does not return as a result.

However, if the document contains:

```
#DREFIELD ANIMAL="cat"
```

```
#DREFIELD ANIMAL="dog"
```

it returns as a result, because one of the ANIMAL fields does not contain the value `cat`.

If you want to find documents in which the specified string is not present in any instance of the specified field, use the MATCH specifier with the Boolean operator NOT. For example, `FieldText=NOT+MATCH{cat}:ANIMAL` does not return any documents that have an ANIMAL field with the value `cat`, even if there are other ANIMAL fields with different values.

Related Topics

- ["MATCH" on page 224](#)

NOTSTRING

The NOTSTRING field specifier (case sensitive) allows you to find documents in which at least one instance of the specified fields contains a value that does not contain any of the specified strings as a substring.

If there are one or more instances of a particular field in the document, the document returns as long as at least one instance does not contain any of the specified strings, even if another instance of the field does contain the string. The document does not return if all instances of the specified fields contain one of the specified strings as a substring.

FieldText=STRING{*yourStrings*}:*yourFields*

where:

<i>yourStrings</i>	is one or more strings. A document returns only if at least one instance of one of <i>yourFields</i> contains a value that is not a substring of any of these strings. FieldText queries which include commas and braces within the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284 .
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if the value in at least one instance of the field does not contain any of <i>yourStrings</i> as a substring. If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

For example

```
FieldText=NOTSTRING{cat,dog}:ANIMAL:TOPIC
```

At least one instance of the ANIMAL or TOPIC field value must contain a value that does not contain the substring *cat* or *dog* for the document to return.

For example, if a document contains only:

```
#DREFIELD ANIMAL="old cat"
```

or

```
#DREFIELD TOPIC="dogs like playing catch "  
#DREFIELD ANIMAL="dog"
```

it does not return as a result.

However, if the document contains:

```
#DREFIELD TOPIC="dogs have trouble catching mice"  
#DREFIELD ANIMAL="dog"  
#DREFIELD ANIMAL="mouse"
```

it returns as a result, because one of the ANIMAL fields does not contain the substring *cat* or *dog*.

If you want to find documents in which the specified string is not present in any instance of the specified field, use the `STRING` specifier with the Boolean operator `NOT`. For example, `FieldText=NOT+STRING{cat,dog}:ANIMAL` does not return any documents that have an `ANIMAL` field with the substring `cat` or `dog`, even if there are other `ANIMAL` fields with different values.

Related Topics

- ["STRING" on page 249](#)

NOTWILD

The `NOTWILD` field specifier (case sensitive) allows you to find documents in which at least one instance of the specified field contains a value that does not match the specified Wildcard string.

If there are one or more instances of a particular field in the document, the document returns as long as at least one instance does not contain the specified string, even if another instance of the field does match. The document does not return if all instances of the specified fields contain the specified string.

If the query does not contain any Wildcard characters (`?` or `*`), the `NOTWILD` field specifier acts in the same way as the `NOTMATCH` field specifier.

```
FieldText=NOTWILD{yourStrings}:yourFields
```

where:

<i>yourStrings</i>	is one or more strings that contain Wildcards. A document returns only if at least one instance of one of <i>yourFields</i> does not contain any of these strings. FieldText queries that include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284 .
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if the value in at least one instance of the field does not contain any of <i>yourStrings</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

For example:

```
FieldText=NOTWILD{passi*incarnata}:Climbers:Plants
```

At least one instance of the `Climbers` or `Plants` field must contain a value which does not contain a phrase that begins with `passi` and ends with `incarnata` (for example, `passionflower incarnata` or `passiflora incarnata`) for this document to return as a result.

For example, if a document contains only:

```
#DREFIELD Climbers="passiflora incarnata"
```

or

```
#DREFIELD Climbers="passiflora incarnata"  
#DREFIELD Plants="passionflower incarnata"
```

it does not return as a result.

However, if the document contains:

```
#DREFIELD Climbers="passiflora incarnata"  
#DREFIELD Plants="passionflower incarnata"  
#DREFIELD Climbers="bindweed"
```

it returns, because one of the `Climbers` fields contains a value that does not match the Wildcard string `passi*incarnata`.

If you want to find documents in which the specified string is not present in any instance of the specified field, use the `WILD` specifier with the Boolean operator `NOT`. For example, `FieldText=NOT+WILD{passi*incarnata}:Climbers` does not return any documents that have a `Climbers` field with a phrase that begins with `passi` and ends with `incarnata`, even if there are other `Climbers` fields with different values.

Related Topics

- ["WILD" on page 231](#)
- ["MATCH" on page 224](#)

Fields that Contain Coordinates within a Specified Area

POLYGON

The `POLYGON` field specifier (case sensitive) allows you to find documents whose `X/Y` position is within a specified polygonal shape.

```
FieldText=POLYGON{coordX, coordY, coordX, coordY, ...}:X:Y
```

where:

<i>coordX, coordY</i>	are the coordinates for one of the vertices. Specify an <i>x,y</i> pair of coordinates for each vertex of the polygon, working either clockwise or counterclockwise around the polygon. You can specify a concave polygon, but the edges must not cross themselves. You can specify coordinates with decimal numbers.
<i>X</i>	is the document field that contains the X coordinate.
<i>Y</i>	is the document field that contains the Y coordinate.

Note: You must always specify two fields, in the order `X:Y`. You can specify more than one pair of fields in the form `X1:Y1:X2:Y2` and so on.

For example:

```
FieldText=POLYGON{1,1,-1,1,0,-2,1,-1}:XPOS:YPOS
```

This example matches all documents whose (`X,Y`) position is within the quadrilateral with vertices at `(1,1)`, `(-1,1)`, `(0,-2)`, `(1,-1)`.

Fields that Contain a Specified String

You can use the following field specifiers (case sensitive) to return documents with fields that contain a specified string.

STRING

The `STRING` field specifier (case sensitive) allows you to specify one or more strings, of which one must be contained as a substring in a specified field.

`FieldText=STRING{yourStrings}:yourFields`

where:

<i>yourStrings</i>	is one or more strings. A document returns only if one of these strings is a substring of the value in one of <i>yourFields</i> . FieldText queries that include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284 .
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if the value in this field contains one of <i>yourStrings</i> as a substring. If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

Examples:

`FieldText=STRING{cat,dog}:ANIMAL`

The `ANIMAL` field value must contain the substring *cat* or *dog* for the document to return. For example, if the `ANIMAL` field has the value *scattering*, the document returns.

`FieldText=STRING{old cat}:ANIMAL:TOPIC`

The `ANIMAL` or `TOPIC` field value must contain the substring *old cat* for the document to return. For example, if the `ANIMAL` field has the value *old cat*, *old caterpillar*, or *bold cats*, the document returns.

`FieldText=STRING{example.com}:COMPANY`

The `COMPANY` field value must contain the substring *example.com* for the document to return. For example, if the `COMPANY` field has the value *example.com* or *http://www.example.com/content/home*, the document returns.

`FieldText=STRING{a,b}:MISC`

The `MISC` field value must contain the substring *a,b* for the document to return. For example, if the `MISC` field has the value *a,b* or *a,b,c*, the document returns.

STRINGALL

The `STRINGALL` field specifier (case sensitive) allows you to specify one or more strings, which must all be contained as a substring in a specified field.

`FieldText=STRINGALL{yourStrings}:yourFields`

where:

<i>yourStrings</i>	<p>is one or more strings. A document returns only if all these strings are substrings of the value in one of <i>yourFields</i>.</p> <p>If you want to specify multiple strings, separate them with commas (there must be no space before or after a comma). FieldText queries which include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284.</p>
<i>yourFields</i>	<p>is one or more fields. A document returns only if it contains one of these fields, and if the value in this field contains <i>yourStrings</i> as substrings.</p> <p>If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>

Examples:

```
FieldText=STRINGALL{cat,dog}:ANIMAL
```

The ANIMAL field value must contain the substrings *cat* and *dog* for the document to return. For example, if the ANIMAL field has the value *grooming cats and dogs* or *doggedly scattering seeds*, the document returns.

```
FieldText=STRINGALL{old cat,young dog}:ANIMAL:TOPIC
```

The ANIMAL or TOPIC field value must contain the substrings *old cat* and *young dog* for this document to return. For example, if the ANIMAL field has the value *old cat chases young dog*, or *young doggedly chasing bold cats*, the document returns.

```
FieldText=STRINGALL{a\b,e\f}:MISC
```

The MISC field value must contain the substrings *a,b* and *e,f* for this document to return. For example, if the MISC field, for example, has the value *a,b,c,d,e,f* or *0=e,fx 1=da,ba*, the document returns.

SUBSTRING

The SUBSTRING field specifier (case sensitive) allows you to return documents whose field value is a substring of a specified string (or equal to a specified string).

```
FieldText=SUBSTRING{yourStrings}:yourFields
```

where:

<i>yourStrings</i>	<p>is one or more strings. A document returns only if one of <i>yourFields</i> contains a substring of one of the specified strings.</p> <p>If you want to specify multiple strings, separate them with commas (there must be no space before or after a comma). FieldText queries that include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284.</p>
<i>yourFields</i>	<p>is one or more fields. A document returns only if it contains one of these fields, and if the value in this field is a substring of <i>yourStrings</i>.</p> <p>If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>

Examples:

```
FieldText=SUBSTRING{Telecommunications,Technology}:SECTOR
```

The SECTOR field must contain a string that is a substring of *Telecommunications* or *Technology*. For example, if the SECTOR field has the value *Telecom* or *Technology*, the document returns. If the SECTOR field has the value *Latest Technology*, the document does not return.

Fields whose Values Match Specific Terms or Phrases

You can use the following field specifiers (case sensitive) to return documents in which specified fields contain specified terms or phrases.

TERM

The TERM field specifier (case sensitive) allows you to find documents with a specified field whose value contains a conceptual match for one or more terms that you specify. A conceptual match exists if a term you specify matches a term in a specified field after it has been stemmed.

Note: If the language that you use does not match the `DefaultLanguageType` specified in the IDOL Server configuration file, add the `LanguageType` parameter to your query action (see ["Specify the Language Type of a Query" on page 110](#)).

```
FieldText=TERM{yourTerms}:yourFields
```

where:

<i>yourTerms</i>	<p>is one or more terms. A document returns only if one of <i>yourFields</i> contains a value that includes a term which conceptually matches one of the specified terms.</p> <p>To specify multiple terms, separate them with commas (there must be no space before or after a comma). FieldText queries that include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284.</p>
<i>yourFields</i>	<p>is one or more fields. A document returns only if it contains one of these fields, and if a term in this field conceptually matches one of <i>yourTerms</i>.</p> <p>To specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>

Examples:

```
FieldText=TERM{shopping,centers}:DRETITLE
```

The DRETITLE field must contain a term that conceptually matches *shopping* or *centers* for the document to return. For example, if the DRETITLE field has the value *shop* the document returns, but if it has the value *bookshopping*, it does not return.

```
FieldText=TERM{training,football}:ITEM:PRODUCT
```

The ITEM or PRODUCT field must contain a term that conceptually matches *trainers* or *football* for the document to return. For example, if the ITEM or PRODUCT field has the value *train* or *footballers*, the document returns, while if it has the value *trainer* or *soccer*, it does not return.

TERMALL

The TERMALL field specifier (case sensitive) allows you to find documents with a specified field whose value contains conceptual matches of several terms that you specify. A conceptual match exists if the terms that you specify match terms in a specified field after they have been stemmed.

Note: If the language that you are using does not match the `DefaultLanguageType` specified in the IDOL Server configuration file, add the `LanguageType` parameter to your query action (see ["Specify the Language Type of a Query" on page 110](#)).

`FieldText=TERMALL{yourTerms}:yourFields`

where:

<code>yourTerms</code>	<p>is multiple terms. A document returns only if one of <code>yourFields</code> contains a value that includes terms which conceptually match the specified terms.</p> <p>Separate the terms with commas (there must be no space before or after a comma). <code>FieldText</code> queries that include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284.</p>
<code>yourFields</code>	<p>is one or more fields. A document returns only if it contains one of these fields, and if a term in this field conceptually matches one of <code>yourTerms</code>.</p> <p>To specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>

Examples:

`FieldText=TERMALL{shopping,centers}:DRETITLE`

The DRETITLE field value must contain a term that conceptually matches *shopping* or *centers* for the document to return. For example, if the DRETITLE field has the value *town center shop*, the document returns.

`FieldText=TERMALL{walk,climb}:DRETITLE:TITLE`

The DRETITLE or TITLE field value must contain a term that conceptually matches *walking* or *climbing* for the document to return. For example, if the DRETITLE or TITLE field has the value *hill walking and rock climbing*, the document returns.

TERMEXACT

The TERMEXACT field specifier (case sensitive) allows you to find documents with a specified field that contains an exact match of any of the terms that you specify.

Note: If the language that you are using does not match the `DefaultLanguageType` specified in the IDOL Server configuration file, add the `LanguageType` parameter to your query action (see ["Specify the Language Type of a Query" on page 110](#)).

`FieldText=TERMEXACT{yourTerms}:yourFields`

where:

<i>yourTerms</i>	<p>is one or more terms. A document returns only if one of <i>yourFields</i> contains a value that exactly matches one of the specified terms.</p> <p>To specify multiple terms, separate them with commas (there must be no space before or after a comma). FieldText queries that include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284.</p>
<i>yourFields</i>	<p>is one or more fields. A document returns only if it contains one of these fields, and if this field contains an exact match of one of <i>yourTerms</i>.</p> <p>To specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>

Examples:

```
FieldText=TERMEXACT{help,helped}:DRETITLE
```

The DRETITLE field value must contain the term *help* or *helped* for the document to return. For example, if the DRETITLE field has the value *helps* or *helping*, the document does not return.

```
FieldText=TERMEXACT{Word,Excel}:FILE:DATEI
```

The FILE or DATEI field value must contain the term *Word* or *Excel* for the document to return. For example, if the FILE or DATEI field has the value *WordPerfect*, the document does not return.

TERMEXACTALL

The TERMEXACTALL field specifier (case sensitive) allows you to find documents with a specified field that contains an exact match of all terms that you specify.

Note: If the language that you are using does not match the `DefaultLanguageType` specified in the IDOL Server configuration file, add the `LanguageType` parameter to your query action (see ["Specify the Language Type of a Query" on page 110](#)).

```
FieldText=TERMEXACTALL{yourTerms}:yourFields
```

where:

<i>yourTerms</i>	<p>is multiple terms. A document returns only if one of <i>yourFields</i> contains exact matches of the specified terms.</p> <p>Separate the terms with commas (there must be no space before or after a comma). FieldText queries that include commas and braces in the query have specific percent-encoding requirements. For information about percent-encoding, see "FieldText" on page 284.</p>
<i>yourFields</i>	<p>is one or more fields. A document returns only if it contains one of these fields, and if this field contains an exact match of all <i>yourTerms</i>.</p> <p>If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).</p>

Examples:

```
FieldText=TERMEXACTALL{rabbits,eating,carrots}:DRETITLE
```

This query returns only documents whose DRETITLE field contains all the specified terms (in their specified form). For example, a document whose DRETITLE field has the value *Rabbits like eating carrots*, or *The carrots were there but the rabbits ate all the cabbage*, returns as a result, but a document with a field that contains *Rabbits like to eat a carrot each day* does not return.

```
FieldText=TERMEXACTALL{flour,milk,eggs}:DRETITLE:TITLE
```

This query returns only documents whose DRETITLE or TITLE field contains all the specified terms (in their specified form). For example, a document whose DRETITLE or TITLE field has the value *Most cake recipes include milk, eggs and flour* return as a result, but a document with a field that contains *Use a cup of milk, two cups of flour and one egg* does not return.

TERMEXACTPHRASE

The TERMEXACTPHRASE field specifier (case sensitive) allows you to return documents in which a specified field contains an exact match of a phrase specified by you. IDOL Server matches your phrase before it applies stemming (it does not remove stop words). It ignores any punctuation in the specifier or field.

Note: If the language that you are using does not match the `DefaultLanguageType` specified in the IDOL Server configuration file, add the `LanguageType` parameter to your query action (see ["Specify the Language Type of a Query" on page 110](#)).

```
FieldText=TERMEXACTPHRASE{yourPhrase}:yourFields
```

where:

<i>yourPhrase</i>	is a phrase. A document returns only if one of <i>yourFields</i> contains an exact match of the specified phrase.
<i>yourFields</i>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains an exact match of <i>yourPhrase</i> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

Examples:

```
FieldText=TERMEXACTPHRASE{Batman! and Robins}:FILM
```

A document whose FILM field contains *Showing now, Batman and Robin's film*, returns as a result, but a document whose FILM field contains *Showing now, 'Batman and Robin' the movie* does not return.

```
FieldText=TERMEXACTPHRASE{gift horse }:DRETITLE:TITLE
```

A document whose DRETITLE or TITLE field contains *looking a gift horse in the mouth* returns as a result, but a document whose DRETITLE or TITLE field contains *the gift horse's mouth had rotting teeth* does not return.

TERMPHRASE

The TERMPHRASE field specifier (case sensitive) allows you to return documents in which a specified field contains a conceptual match of a phrase that you specify. IDOL Server matches your phrase after it applies stemming (it does not remove stop words). It ignores any punctuation in the specifier or field.

Note: If the language that you are using does not match the `DefaultLanguageType` specified in the IDOL Server configuration file, add the `LanguageType` parameter to your query action (see ["Specify the Language Type of a Query" on page 110](#)).

`FieldText=TERMPHRASE{yourPhrase}:yourFields`

where:

<code>yourPhrase</code>	is a phrase. A document returns only if one of <code>yourFields</code> contains a conceptual match of the specified phrase.
<code>yourFields</code>	is one or more fields. A document returns only if it contains one of these fields, and if this field contains a conceptual match of <code>yourPhrase</code> . If you want to specify multiple fields, separate them with colons (there must be no space before or after a colon).

Examples:

`FieldText=TERMPHRASE{Batman! and Robins}:FILM`

A document whose `FILM` field contains *Showing now: 'Batman and Robin'*, returns as a result.

`FieldText=TERMPHRASE{gift horse }:DRETITLE:TITLE`

A document whose `DRETITLE` or `TITLE` field contains *the gift horse's mouth had rotting teeth* returns.

Field Specifiers to Bias Result Scores

- **BIAS.** The `BIAS` field specifier (case sensitive) allows you to bias the score of results according to the numerical proximity of the specified field to a particular value.
You can also boost the percentage relevance that is given to query results by setting up specific field process or by using multipliers. See ["Manipulate Result Relevance" on page 317](#) for details on `BIAS` and other methods that allow you to manipulate result scores.
- **BIASDATE.** The `BIASDATE` field specifier (case sensitive) allows you to boost the score of result documents by a specified percentage, based on how close the date in a specified field is to a specified date.
- **BIASDISTCARTESIAN.** The `BIASDISTCARTESIAN` field specifier allows you to boost the score of any document according to its distance from a specified point using Cartesian coordinates (X/Y).
- **BIASDISTSPHERICAL.** The `BIASDISTSPHERICAL` field specifier allows you to boost the score of any document according to its distance from a specified point using spherical coordinates (latitude and longitude).
- **BIASVAL.** The `BIASVAL` field specifier (case sensitive) allows you to bias the score of result

documents by a specified percentage, based on whether they include a specific value in the specified field.

Field Specifier for Linked Queries

Linking queries allows you to query for documents based on criteria in the document, and in a connected document. For example, you might want to connect documents written by a particular user with values in the user profile.

The LINK field specifier allows you to form linked queries. For more information, see ["Link Queries" on page 274](#).

Fuzzy Search

If you are not quite sure how to spell some of the words that you want to query for, you can use the Query action to submit a fuzzy query to IDOL Server. A fuzzy query returns results that contain words which are similar to the query string.

Fuzzy Query Syntax

If you want to submit a fuzzy query, you must specify the Query action Text parameter in one of the following formats:

- `Text=myQueryTextDREFUZZY(fuzzyQueryText)`

For example:

```
http://IDOLhost:port/action=Query&Text=best selling author DREFUZZY(Rowlling)
```

- `Text=DREFUZZY(fuzzyQuerytext)`

For example:

```
http://IDOLhost:port/action=Query&Text=DREFUZZY(Caroll Jabberwalky)
```

- You can also use Boolean and proximity operators within fuzzy queries.

```
Text=DREFUZZY(fuzzyQueryTextOPERATOROtherFuzzyQueryText)
```

For example:

```
http://IDOLhost:port/action=Query&Text=DREFUZZY(Caroll AND Jabberwalky)
```

Adjust the Tolerance Level of a Fuzzy Search

By default, DREFUZZY internally determines an appropriate tolerance level when it judges whether words in a result document are similar enough to the words that you specify in the query. However, in exceptional circumstances you can postfix DREFUZZY with a numerical value to adjust this tolerance level:

```
DREFUZZYN(fuzzyQuerytext)
```

This value operates like a slider that increases or decreases the tolerance level. The higher the value is, the more words in result documents count as eligible matches to the words specified in the query. The lower the value is, the fewer words in result documents count as eligible matches to the words specified in the query.

HPE does not recommend you specify a value higher than 6, because this can result in fuzzy matching that is so flexible that results are not related to the query.

Examples:

```
action=Query&Text=DREFUZZY1(Caroll Jabberwalky)
```

```
action=Query&Text=DREFUZZY3(Caroll Jabberwalky)
```

The second query in this example is more tolerant in accepting matches for the specified words than the first query, which means that it might return more results.

Parametric Search

The `GetTagValues` and `GetQueryTagValues` actions allow you to perform parametric searches.

A parametric search allows you to search for items by their characteristics (values in certain fields). When you provide fixed values in parametric fields, the parametric search returns consistent values in the nonfixed parametric fields. For example, you can search an IDOL Server wine database for specific wine varieties from a specific region by specifying which fields must match these characteristics. Only wines that match the specified variety and region return.

Configure IDOL Server for Parametric Fields

Before you perform parametric searches, configure IDOL Server to recognize parametric fields.

Note: You must configure parametric field recognition before you index the data that you want to search.

To configure IDOL Server to recognize parametric fields

1. Open the IDOL Server configuration file in a text editor.
2. In the `[Server]` section, set the `ParametricRefinement` parameter to `True`. (If the section does not contain this parameter, add it.)

Note: If you want to define parametric fields or add extra parametric fields, but have already indexed content into IDOL Server, also set `RegenerateParametricIndex` to `True`. This parameter allows IDOL Server to generate the files that it requires to internally identify parametric fields on startup, so that you need only to restart IDOL Server to use parametric fields, rather than having to reindex all your data.

You can also use the `DREREGENERATE` index action to regenerate the parametric index while the server is running.

3. List a parametric field process in the `[FieldProcessing]` section.

For example:

```
[FieldProcessing]
0=MyFirstProcess
1=ParametricFields
```

4. Create a section for each field process that you listed, in which you create a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process. For example:

```
[MyFirstProcess]
Property=MyProperty
PropertyFieldCSVs=*/MyField,*/MyOtherField
```

```
[ParametricFields]
Property=Parametric
PropertyFieldCSVs=*/Grape,*/Color,*/Region,*/Price
```

Note: The properties that you create must not have the same name as the processes.

5. Create a section for the parametric property in which you set the `ParametricType` parameter to `True`. This property enables IDOL Server to recognize the associated `PropertyFieldCSVs` fields as parametric fields. For example:

```
[Parametric]
ParametricType=True
```

6. Save and close the configuration file. Restart IDOL Server for your changes to take effect. You can now index your data into IDOL Server.

Perform a Parametric Search

After you configure IDOL Server to index and recognize parametric fields, you can use the following actions to perform a parametric search.

GetTagValues

This action allows you to specify one or more parametric fields and return all values that these fields contain in IDOL Server. It includes values in documents that you do not have access to, and values in documents that were deleted (unless you compacted the IDOL Server data index after the documents were deleted).

For example:

```
action=GetTagValues&FieldName=Grape
```

This action requests the different values of the IDOL Server `Grape` fields. It returns a list of all grape varieties stored in an IDOL Server wine database, for example.

You can also restrict the action, so that it returns `Grape` field values only if they are in a document that also contains other specific fields that have specific values. For example:

```
action=GetTagValues&FieldName=Grape&Restriction=MATCH{Barossa Valley}:Region+MATCH{Red}:Color
```

This action returns Grape field values only if they are in a document that also contains a Region field that has the value *Barossa Valley* and a Color field that has the value *Red*.

GetQueryTagValues

This action combines query text with one or more parametric fields. When IDOL Server performs the query, it finds documents that match the specified query text, and returns the values of the specified parametric fields for these documents. Unlike the `GetTagValues` action, the `GetQueryTagValues` action does not return field values in documents that you do not have access to, or in documents that were deleted.

For example:

```
http://localhost:5552/action=GetQueryTagValues&FieldName=GRAPE,COUNTRY&Text= A smooth red wine that complements game
```

This action requests the different values of the GRAPE and COUNTRY fields of documents that are conceptually similar to the specified Text.

You can also restrict the action by combining it with various action parameters. For example:

```
http://localhost:5552/action=GetQueryTagValues&FieldName=GRAPE,COUNTRY&Text= A smooth red wine that complements game&MaxValues=10&Sort=Alphabetical
```

This action requests the 10 top values of the GRAPE and COUNTRY fields of documents that are conceptually similar to the specified Text. IDOL Server displays the values in alphabetical order when it returns them.

```
http://localhost:5552/action=GetQueryTagValues&FieldName=GRAPE,COUNTRY&Text= A smooth red wine that complements game&DocumentCount=True
```

This action requests the different values of the GRAPE and COUNTRY fields of documents that are conceptually similar to the specified Text. The `DocumentCount` parameter instructs IDOL Server to return the number of documents that contain each value.

```
http://localhost:5552/action=GetQueryTagValues&FieldName=GRAPE,COUNTRY&Text= A smooth red wine that complements game&FieldDependence=True
```

This action requests the different values of the GRAPE and COUNTRY fields of documents that are conceptually similar to the specified Text. The `FieldDependence` parameter instructs IDOL Server to find sets of values that occur together. If IDOL Server finds documents that contain the first parametric field listed, it checks if they also contain the subsequently listed parametric fields and returns them. You can also use the `FieldDependenceMultiLevel` parameter to display these results in a hierarchical structure.

For further details on available parameters for the `GetTagValues` and `GetQueryTagValues` actions, refer to the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)

Proper Names Search

If you want IDOL Server to recognize names and treat them as a unit, you must enable proper names searches.

Note: To search for exact matches of phrases as well as names, enable `AdvancedSearch` before you index your content into IDOL Server, and set the `ProperNames` parameter to 7 (see below).

Related Topics

- ["Phrase Search" on page 208](#)

Enable Proper Names Searches

Note: You must enable proper names searches before you index the data that you want to query against.

To enable proper names searches

1. Open the IDOL Server configuration file in a text editor.
2. Before you store content in IDOL Server, terms are always stemmed and stop words are always discarded. If you want to store proper name terms (adjacent terms that begin with a capital letter) in addition to the normal content, you can set the `ProperNames` parameter in the `[LanguageTypes]` section to one of the following values.

Value	Meaning
0	Proper name terms are not stored.
1	Adjacent capitalized terms are compounded, then stemmed and indexed as a unit. For example, <i>Sam James</i> is indexed as <i>SAMJAM</i> .
2	Adjacent terms are compounded (regardless of capitalization), then stemmed and indexed as a unit. For example, <i>bottlenose dolphins</i> is indexed as <i>BOTTLENOSEDOLPHIN</i> . Note: This setting considerably increases the number of terms in the IDOL Server index, which can slow down its performance.

Use the following `ProperNames` options only if you need to query for proper names that contain stop words (for example, *The Who* or *The Queen*).

Value	Meaning
3	Adjacent capitalized stop words are compounded, then stemmed and indexed as a unit. For example, <i>And His</i> is indexed as <i>ANDHI</i> . Adjacent capitalized terms are compounded, then stemmed and indexed as a unit. For example, <i>Sam James</i> is indexed as <i>SAMJAM</i> .

Value	Meaning
	Capitalized stop words adjacent to capitalized terms are treated as individual terms. For example, <i>The Queen</i> is treated as <i>THE</i> and <i>QUEEN</i> , according to your stop word rules.
4	Capitalized stop words are compounded with adjacent capitalized terms, then stemmed and indexed as a unit. For example, <i>The Bells</i> is indexed as <i>THEBEL</i> , and <i>Calling Will</i> is indexed as <i>CALLINGWIL</i> . Adjacent capitalized stop words are compounded, then stemmed and indexed as a unit. Adjacent capitalized terms are compounded, then stemmed and indexed as a unit.
5	Adjacent capitalized stop words are compounded and indexed unstemmed as a unit. For example, <i>And His</i> is indexed as <i>ANDHIS</i> . Adjacent capitalized terms are compounded and indexed unstemmed as a unit. For example, <i>Sam James</i> is indexed as <i>SAMJAMES</i> Capitalized stop words adjacent to capitalized terms are treated as individual terms.
6	Capitalized stop words are compounded with adjacent capitalized terms, and indexed unstemmed as a unit. For example, <i>The Bells</i> is indexed as <i>THEBELLS</i> , and <i>Calling Will</i> is indexed as <i>CALLINGWILL</i> . Adjacent capitalized stop words are compounded and indexed unstemmed as a unit. Adjacent capitalized terms are compounded and indexed unstemmed as a unit.
7	Capitalized stop words are compounded with adjacent capitalized terms, and indexed unstemmed as a unit. Adjacent capitalized stop words are compounded and indexed unstemmed as a unit. Adjacent capitalized terms are treated as individual terms. For example, <i>Sam James</i> is indexed as <i>SAM</i> and <i>JAME</i> . <div style="background-color: #e0e0e0; padding: 5px;">Note: HPE recommends that you use this setting if you set <code>AdvancedSearch</code> to <code>True</code> in the [Server] section of the IDOL Server configuration file.</div>

You must set this parameter for each of the languages that you want to enable name recognition for (if the language settings do not include the `ProperNames` parameter, you must add it). For example:

```
[LanguageTypes]
DefaultLanguageType=English
LanguageDirectory=C:\HewlettPackardEnterprise\IDOLServer\common\langfiles
0=English
1=Deutsch
2=Francais

[English]
Encodings=UTF8:englishUTF8
ProperNames=1
```

```
[Deutsch]
Encodings=UTF8:germanUTF8
ProperNames=1
```

```
[Francais]
Encodings=UTF8:frenchUTF8
ProperNames=1
```

3. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
4. Index documents into IDOL Server. After you finish indexing, IDOL Server treats any Query action as a proper name query.

Example Proper Name Searches

The following table describes how the ProperNames setting affects the terms that IDOL Server stores for the sentence *Tom Jones And His greatest hits*.

Original	Tom		Jones		And His	greatest		hits
0	TOM		JONE			GREAT		HIT
1	TOM	TOMJON	JONE			GREAT		HIT
2	TOM	TOMJON	JONE			GREAT	GREATESTHIT	HIT
3	TOM	TOMJON	JONE		ANDHI	GREAT		HIT
4	TOM	TOMJON	JONE	JONESAND	ANDHI	GREAT		HIT
5	TOM	TOMJONES	JONE		ANDHIS	GREAT		HIT
6	TOM	TOMJONES	JONE	JONESAND	ANDHIS	GREAT		HIT
7	TOM		JONE	JONESAND	ANDHIS	GREAT		HIT

If IDOL Server contains these documents, the following queries produce different results according to your ProperNames settings.

Doc 1:	Doc 2:
Tom Waits and The The in concert with Norah Jones	Tom Jones and the the in concert with Katie Melua

- action=Query&Text=Tom Jones

If you set ProperNames to 0 or 7, both documents return with the same relevance (in both cases, the query to IDOL Server has the terms *TOM* and *JONE*, which match both documents).

If you set ProperNames to 1, 2, 3, 4, 5, or 6, Doc 2 returns with a higher relevance than Doc 1 (because it matches not just the terms *TOM* and *JONE*, but also *TOMJON* or *TOMJONES*).

- action=Query&Text=tom jones

If you set `ProperNames` to 0, 1, 3, 4, 5, 6, or 7, both documents return with the same relevance (in both cases, the query to IDOL Server has the terms *TOM* and *JONE*, which match both documents).

If you set `ProperNames` to 2, Doc 2 returns with a higher relevance than Doc 1 (because it matches not just the terms *TOM* and *JONE*, but also *TOMJON*).

- `action=Query&Text=The The`

If you set `ProperNames` to 0, 1, or 2, the query returns no results (because IDOL Server discards both instances of the word *The* as stop words).

If you set `ProperNames` to 3, 4, 5, 6, or 7, only Doc 1 returns (because in all these cases the query to IDOL Server has the term *THETH* or *THETHE*, which matches only Doc 1).

- `action=Query&Text=the the`

If you set `ProperNames` to 0, 1, 2, 3, 4, 5, 6, or 7, no results return (because IDOL Server discards both instances of the word *the* as stop words).

Soundex Keyword Search

If the spelling of a keyword is not quite accurate but phonetically correct, a Soundex keyword query returns results that contain the keyword and phonetically similar keywords (using a Soundex algorithm).

Enable Soundex Keyword Searches

Note: You must enable Soundex keyword searches before you index the data that you want to search.

To enable Soundex keyword searches

1. Open the IDOL Server configuration file in a text editor.
2. In the [Server] section, set the `Soundex` parameter to one of the following values. If the [Server] section does not contain the `Soundex` parameter, add it.

1	The HPE Soundex extension.
2	Standard Soundex.
4	Double Metaphone.
8	The HPE Double Metaphone extension.

3. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
4. Index documents into IDOL Server. After you finish indexing, you can perform Soundex keyword searches by using the `Query` action. The phonetic indexing and query uses the algorithm that you selected.

Perform Soundex Keyword Searches

You can use Soundex with single or multiple keywords, or as part of a query text string. For example:

```
http://localhost:5552/action=Query&Text=SOUNDEX(einstine)
```

```
http://localhost:5552/action=Query&Text=albert SOUNDEX(einstine) examined the  
phenomenon discovered by max SOUNDEX(plank) in 1905
```

Synonym Search

A synonym query returns results which are conceptually similar to the terms in a Query action Text parameter or conceptually similar to the synonyms that are available for the Text terms.

You can set up synonym searching by using one of these methods:

- Enable synonym searches in IDOL Server.
HPE recommends this method if you require synonym matching for approximately a hundred terms. The synonym lists that this method uses can contain terms and phrases.
- Set up an additional Synonym IDOL Server in which each document is a synonym list.
HPE recommends this method if you require synonym matching for a much larger number of terms (several hundred or more). The synonym lists that this method uses can contain only terms, not phrases.

Related Topics

- ["Enable Synonym Searches" below](#)
- ["Set up an Additional Synonym IDOL Server" on page 267](#)

Enable Synonym Searches

Use the following procedure to enable synonym searches in IDOL Server.

To enable synonym searches

1. Set up a synonym file.
2. Configure IDOL Server to use a synonym file.
3. Perform a synonym query.

For details on the settings that the [Synonym] sections can contain and on how you can configure them, see the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)
- ["Set up a Synonym File" on the next page](#)
- ["Configure IDOL Server to Use a Synonym File" on page 266](#)
- ["Perform Synonym Searches" on page 267](#)

Set up a Synonym File

Note: You must set up a synonym file before you index the data that you want to search.

To set up a synonym file

1. Create a text file and save it in the IDOL Server `IDOL/content` directory, using the file name that you specified in the `[SynonymType]` section of the IDOL Server configuration file.
2. Create a section for each language. The section name must be the name of a valid language type configured in IDOL Server (that is, a language and encoding).

Note: You do not need to create a section for each encoding in the same language. IDOL Server uses the same synonyms for this language, in all encodings. For example, if you add an `[EnglishUTF8]` section, you do not also need to add an `[EnglishASCII]` section.

For example:

```
[EnglishUTF8]
[GermanUTF8]
```

3. In each section, create a line for each word for which you want to list synonyms. Use the same encoding that you specified in the section name. For example:

```
[EnglishUTF8]
cat
dog
```

```
[GermanUTF8]
Katze
Hund
```

4. List synonym strings next to each word and save the file. You must separate the word and each string with commas. There must be no space before or after a comma. The individual terms can contain spaces but must not contain any punctuation. For example:

```
[EnglishUTF8]
cat,feline,grimalkin,moggy,mouser,puss,tabby
dog,cur,hound,mans best friend,mongrel,mutt,pooch,puppy
```

```
[GermanUTF8]
Katze,Mietze,Mietzekatze,Mietzekater,Kater,Mulle,Kätzchen
Hund,Wau Wau,Hündin,Töle,Kläffer,Hündchen,Welpe
```

Note: For Asian languages, you can also use Asian byte-order marks.

Related Topics

- ["Configure IDOL Server to Use a Synonym File" on the next page](#)

Configure IDOL Server to Use a Synonym File

Note: You must configure IDOL Server to use the synonym file before you index the data that you want to search.

To configure IDOL Server to use a synonym file

1. Open the IDOL Server configuration file in a text editor.
2. In the [FieldProcessing] section of the IDOL Server configuration file, set up a synonym process. This process allows IDOL Server to determine when it must apply synonym settings. For example:

```
[FieldProcessing]
θ=SynonymMatch
```

3. Create a section for the synonym process you listed, in which you create a property for the process (synonym properties always point to a defined synonym job). Identify the fields that you want to associate with the process. For example:

```
[SynonymMatch]
Property=ApplySynonymMatch
PropertyFieldCSVs=*/DRETITLE,*/DRECONTENT
```

In this example, IDOL Server returns documents for synonym queries only if their DRETITLE or DRECONTENT field values match the query.

Note: The property that you create must not have the same name as the process.

To identify the fields, use the format /FieldName to match root-level fields, */FieldName to match all fields except root-level, or /Path/FieldName to match fields that the specified path points to.

4. Create a section for the property in which you set the SynonymType parameter to the name of the synonym job that specifies which settings IDOL Server must apply to synonym queries.

```
[ApplySynonymMatch]
SynonymType=Synonym_job
```

5. In the [Synonym] section of the IDOL Server configuration file, list the synonym job whose settings you want to apply when you send a synonym query to IDOL Server. You can set up multiple jobs. However, you normally require only one. For example:

```
[Synonym]
θ=Synonym_job
```

6. Define a section for your synonym job in which you specify the settings that you want to apply to synonym queries. The section must have the same name as the synonym job. For example:

```
[Synonym_job]
File=animals.txt
MaxExpandLevel=1
```

7. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Perform Synonym Searches

After you create a synonym file and configure IDOL Server to use it, you can turn any Query action that you send to IDOL Server into a synonym query by adding `&Synonym=True` to it. For example:

```
http://localhost:5552/action=Query&Text=Felix is a great mouser&Synonym=True
```

This query returns documents that conceptually match the term *mouser*, as well as documents that conceptually match any of the terms listed as synonyms for the term *mouser* in the synonym file.

Update Synonym Files

IDOL Server processes synonyms at index time. When you want to update or add a synonym in your synonym file, you must reindex your content for the modified synonyms to be available over all content. If you do not reindex, the synonym changes apply only to new documents.

Set up an Additional Synonym IDOL Server

Use the following procedure to set up an additional Synonym IDOL Server.

To set up an additional IDOL Server

1. Install the Synonym IDOL Server.
2. Create a synonym file and index it.
3. Perform a synonym query.

Related Topics

- ["Install the Synonym IDOL Server" below](#)
- ["Create and Index a Synonym File" below](#)
- ["Perform Synonym Searches" on the next page](#)

Install the Synonym IDOL Server

Follow the IDOL Server installation instructions to install the IDOL Server component. If you are installing the Synonym IDOL Server on the same machine as your existing IDOL Server, ensure that the servers use different ports. Refer to the *IDOL Getting Started Guide* for details of how to install IDOL Server.

Create and Index a Synonym File

You can obtain the synonym file to store in your Synonym IDOL Server by spidering a Thesaurus site (using HTTP Connector), or by creating the file manually. A synonym file must be a text file that contains the fields described in the following table.

Field	Description
#DRREFERENCE	Enter a unique reference string for the synonym document. Usually this reference is a file name, URL, or unique code number.

Field	Description
#DRECONTENT	Enter a list of associated single words, separated by carriage returns or spaces (you cannot list phrases).
#DREENDDOC	A delimiter that indicates the end of the document.

For example:

```
#DREREFERENCE Syn1.txt
#DRECONTENT cat feline grimalkin moggy mouser tabby siamese kitten
#DREENDDOC
#DREREFERENCE Syn2.txt
#DRECONTENT dog cur hound mongrel mutt pooch puppy
#DREENDDOC
...
```

If you use HTTP Connector to create the synonym file, you can use the connector to index the file. If you create the file manually, you can index it by using a DREADD index action.

Related Topics

- ["DREADD: Index IDX and XML Files Directly" on page 44](#)

Perform Synonym Searches

Use the following procedure to perform a synonym search.

To perform synonym searches

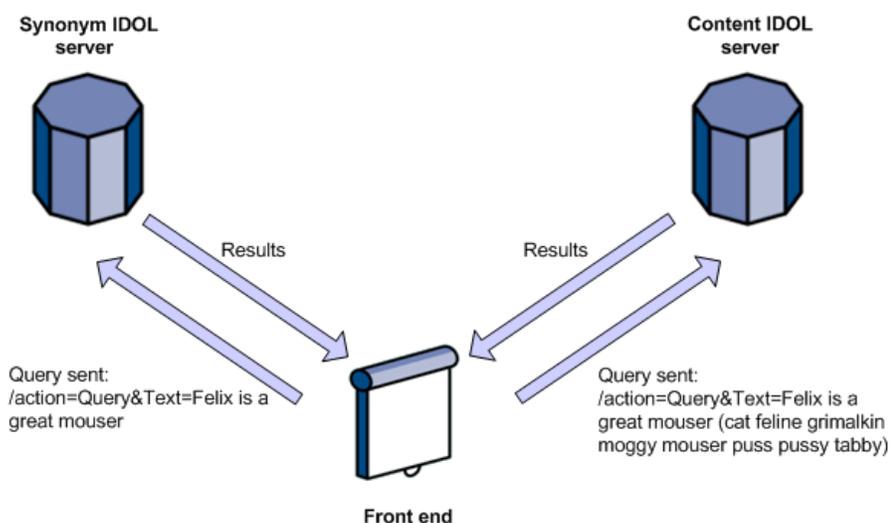
1. Send a query to the Synonym IDOL Server. For example:

```
http://synonymServerHost:synonymServerPort/action=Query&Text=mouser
```

2. When the Synonym IDOL Server returns the synonym results, add the results to the query string and send the query to your normal Content IDOL Server (normally you set up a front end to do this). For example:

```
http://IDOLhost:port/action=Query&Text=mouser+(cat feline grimalkin moggy
mouser tabby siamese kitten)
```

This query returns documents that conceptually match the term *mouser*, as well as documents that conceptually match any of the terms that the Synonym IDOL Server lists as synonyms for the term *mouser*.



Analytics Functions

Analytics functions allow you to perform an operation on document fields at query time, and create a temporary field that contains the result, which you can use in other query operations such as `FieldText`. The `UserMetaFields` parameter for the `Query` and `GetQueryTagValues` actions specifies an operation that you want to perform, and the name of the field that you want to create.

For example, you can use `UserMetaFields` to create a new field that contains the mean of the values of specified fields in a document. You can then use the `average` in the `FieldText` parameter to find documents with a particular mean value, or in the `Sort` parameter to sort the results by the mean value.

For operations that you want to use regularly, you might want to use Connector Framework Server (CFS) to process documents before you index, and add a field to the document that you can use in queries. In general, storing the results as permanent fields gives better query performance. However, you can use `UserMetaFields` to perform operations that you do not need regularly, or to create and use fields that you did not consider when you originally indexed the data. You can also use `UserMetaFields` if you want to update fields without recalculating the value of the stored field.

Create Metafields

The `UserMetaFields` parameter has the following syntax:

```
UserMetaFields=OPERATION{UserFieldName}:DocumentFieldName
```

where:

<i>OPERATION</i>	is the operation that you want to perform on the contents of the specified fields in your documents.
<i>UserFieldName</i>	is the name of the user metafield that you want to create.
<i>DocumentFieldName</i>	is the name of the fields in your documents that you want to run the specified

	operation on. To specify multiple fields, separate the field names with colons (:) (there must be no space before or after a colon).
--	--

Note: UserMetaFields operations are more efficient when you use them on fields that are configured as NumericType, except for the COUNTA operation, which is more efficient on fields that are configured as CountType, MatchType, or ParametricType.

To specify multiple operations, separate the operations with a plus sign (+) or a space.

You can optionally use a constant in the UserMetaFields definition, in the following form:

UserMetaFields=OPERATION{UserFieldName, Constant}:DocumentFieldName

The constants can be positive or negative, and they can include decimals (for example 1.23) and scientific notation (for example 1E2). The meaning of the constants depends on the operation that you use.

Note: IDOL Server uses the constant only if the DocumentFieldName contains values. For example, for the operation MIN{MINSCORE, 10}:SCORE, IDOL Server does not create a MINSCORE user metafield if the SCORE field is missing for a document.

The following table describes the operations that you can perform on document fields.

Operation	Description	Constant
ADD	The sum of all the values in the specified fields.	The initial value to add the field values to.
SUB	The value of the first field name (or the sum of the values in all instances of the first field name), minus the values of the remaining fields.	An additional value to subtract.
MUL	The product of all the values in the specified fields.	The initial value to multiply the field values by.
DIV	The value of the first field name (or the product of the nonzero values in all instances of the first field name), divided by the nonzero values from the remaining fields.	An additional value to divide by. If the constant value is 0, the constant is ignored.
MIN	The smallest value in the specified fields.	An initial value to include in the minimum calculation.
MAX	The highest value in the specified fields.	An initial value to include in the maximum calculation.
MEAN	The mean of the values in the specified fields.	No effect.
COUNT	The number of numeric values in the specified fields (it does not count any non-numeric values).	No effect.
COUNTA	The number of non-empty occurrences of the specified field, regardless of their value.	No effect.
ABS	The absolute value of the value in the specified field (that	No effect.

Operation	Description	Constant
	is, removing any negative sign).	
ROUND	The nearest integer to the value in the specified field, or the nearest multiple of the specified constant.	The accuracy to round to. For example, 0.01 rounds to two decimal places.
FLOOR	The nearest integer that is lower than the value in the specified field, or the nearest multiple of the specified constant lower than the value.	The accuracy to round down to. For example, 0.01 rounds to two decimal places.
LOG	The log (to the base of the specified constant) of the value of the first instance in a document of the specified field. The constant is required for this function.	The base of the log.
POW	The value of the first instance in a document of the specified field, raised to the power of the specified constant. The constant is required for this function.	The power to use.

Note: The ABS, ROUND, FLOOR, LOG, and POW operations act on a single value for each document. HPE recommends that you use these operations on fields that have only a single value, or on the output of other user metafield operations. If you apply these operations to a field with multiple values, it returns the result of the operation on the first field value, which might depend on your configuration.

You can also use the following distance operations. In this case, you must specify two constants and two field names that contain the location information.

Operation	Description	Constant
DISTSPHERICAL	The distance in kilometers between the location specified by the constants, and the location specified by the document fields, in latitude and longitude. You must specify two fields for this operator, in the form :LAT_FIELD:LONG_FIELD.	The latitude and longitude of the location you want to calculate distances from. Specify these values as <i>Lat, Long</i> .
DISTCARTESIAN	The distance between the point specified by the constants, and the point specified by the document fields (in Cartesian coordinates). You must specify two fields for this operator, in the form :X_FIELD:Y_FIELD.	The X and Y coordinate of the point that you want to calculate distances from. Specify these values as <i>X, Y</i> .

Note: The DISTSPHERICAL and DISTCARTESIAN operations act on a single pair of values for each document (one value for each coordinate field). HPE recommends that you use these operations on fields that have only a single value, or the output of other user metafield operations. If you apply these operations to a field with multiple values, it returns the result of the operation on the first field value, which might depend on your configuration.

If both of your coordinate fields are configured as `NumericType`, HPE recommends that you set `XMLFullStructure` to `True` in your configuration. With this additional configuration, distance-based metafields produce results consistent with distance-based sorting.

You can apply `UserMetaFields` operations to other user metafields created in the same query. For example, you might use an operation to create one field, and then use that field in a second operation. You must create a metafield before you reference it in another operation (that is, the operation to create the field must appear to the left of the operation that uses it in the `UserMetaFields` parameter).

When you use your metafield in another operation, the field name has the format `autn_user_UserFieldName`, where `UserFieldName` is the value that you specified in the `UserMetaField` parameter.

For example, the following query creates the `TaxAmount` user metafield from the document fields, and then uses this new metafield to calculate the total price.

```
UserMetaFields=MUL{taxAmount}:basePrice:taxRate,ADD{totalPrice}:basePrice:autn_user_taxAmount
```

Use Metafields

IDOL Server does not permanently store the values for user metafields, but you can use the results in the `Query` or `GetQueryTagValues` action.

When you use your metafield in another operation, the field name has the format `autn_user_UserFieldName`, where `UserFieldName` is the value you specified in the `UserMetaField` parameter.

FieldText

You can use metafields in `FieldText` operators. For example:

```
action=Query&Text=Television&UserMetaFields=ADD{TOTAL}:PRICE:TAX&FieldText=EQUAL{500}:autn_user_TOTAL
```

In this example, the `UserMetaFields` operation adds the value of the `PRICE` and `TAX` fields in documents that match the query for `Television`. It creates the `TOTAL` field for each document to store this result, and returns only documents where the new `TOTAL` field has a value of `500`.

Related Topics

- ["FieldText Search" on page 222](#)

Sort

You can use metafields in the `Sort` parameter to sort your result documents. For example:

```
action=Query&Text=Television&UserMetaFields=MIN{LOWSCORE}:REVIEWSCORE&Sort=autn_user_LOWScore:decreasing
```

In this example, the `UserMetaFields` operation takes the lowest value of the `REVIEWSCORE` fields in a document that matches the query for `Television`. It creates a `LOWSCORE` field for each document to store this result, and returns documents in decreasing order of the `LOWSCORE` field.

Related Topics

- ["Change Result Sorting \(Display Order\)" on page 289](#)

PrintFields

You can specify metafields in the `PrintFields` parameter to display the metafields in the results.

Note: By default, when you set `Print` to `all`, metafields do not return in results. In this case, you can set the `PrintFields` parameter to a list of metafields to display them.

```
action=Query&Text=Television&UserMetaFields=MIN  
{LOWSCORE}:REVIEWSCORE&Print=Fields&PrintFields=autn_user_LOWScore
```

In this example, the `UserMetaFields` operation takes the lowest value of the `REVIEWSCORE` fields in a document that matches the query for `Television`. It creates a `LOWSCORE` field for each document to store this result, and displays the `LOWSCORE` field in the results.

GetQueryTagValues FieldName

In the `GetQueryTagValues` action, you can use metafields as the `FieldName` that you want to return results for. For example:

```
action=GetQueryTagValues&Text=Television&UserMetaFields=MEAN  
{AVERAGE}:PRICE&FieldName=autn_user_AVERAGE
```

In this example, the `UserMetaFields` operation calculates the mean value of the `PRICE` fields in documents that match the query for `Television`. It creates an `AVERAGE` field for each document to store this result, and then returns the values that occur in the `AVERAGE` field as tags for the parametric search.

```
action=GetQueryTagValues&Text=Restaurants&UserMetaFields=DISTSPHERICAL  
{HOME,51.3,0.73}:LAT:LONG&FieldName=autn_user_HOME&Ranges=FIXED{0,5,10,20,.}:autn_  
user_HOME&DocumentCount=true
```

In this example, the `UserMetaFields` operation finds documents that match the query for `Restaurants`, and uses the document `LAT` and `LONG` fields to calculate the distance from a location in London. It creates a `HOME` field for each document to store this distance, and then returns the number of documents that occur within the specified distances from the original point, grouped by the specified ranges (0-5, 5-10, 10-20, 20+).

Related Topics

- ["Parametric Search" on page 257](#)

AgentBoolean

You can use metafields as part of an agent, and use the metafield in `AgentBoolean` queries.

For example, you might create the following agent `IDX` file, and index it into the `Agentstore` component:

```
#DREREFERENCE UserMetaFields  
#DREFIELD AGENTBOOL="usermeta"  
#DREFIELD FTF="GREATER{5}:autn_user_X"
```

```
#DREFIELD AGENTPARAMS="UserMetaFields=MEAN{X}:NUMBER"  
#DRECONTENT  
usermeta  
#DREENDDOC
```

This IDX file includes the `UserMetaFields` parameter in the `AGENTPARAMS` field, and a `FieldText` expression that uses the resulting metafield in the `FTF` field.

For this example, when you send a `Query` action to the `Agentstore` to return agents that match a document, you also set the following parameters in the action:

- `TextParse=True`
- `AgentBooleanField=AGENTBOOL`
- `FieldTextField=FTF`
- `AgentParamsField=AGENTPARAMS`

For a document to match this example agent, it must include the term `usermeta`. When the document matches the `usermeta` term, the agent calculates the mean of the values in the `NUMBER` fields, and creates an `X` field to contain the mean. For the agent to return as a match for the document, the document `X` metafield must have a value greater than 5.

Related Topics

- ["AgentBoolean Agents and Categories" on page 169](#)
- ["Perform AgentBoolean Queries" on page 172](#)

Link Queries

You can use link queries to query for documents based on criteria in the document, and in a connected document. For example, you might want to connect documents written by a particular user with values in the user profile.

To use linked queries, you must configure a `ReferenceMemoryMappedType` field that occurs in the documents that you want to query, which contains values that occur in the document references for the connected documents. `ReferenceMemoryMappedType` fields provide a fast lookup between child documents and a parent document. For example, for a user profile, this might be a `UserName` field in the documents that specifies the user that wrote a document, and the value of that field is the document reference for the user profile.

Configure IDOL Server for Linked Queries

To use link queries, you must configure a `ReferenceMemoryMappedType` field. You then index the documents. You must either ensure that you index the parent documents first (in which the `ReferenceMemoryMapped` values occur in the document reference), or use the `DREREGENERATE` index action to fix the parent-child ordering.

To configure IDOL Server for linked queries

1. Open the IDX file of the documents that you want to index.
2. Find the field that you want to use to link documents. This field must contain a value that occurs in a normal field in the documents that you want to search, and in a reference field in the parent documents. For example, it might be a `UserName` field in the documents, and the value of this field is the document reference for the user profile.
3. Open the IDOL Server configuration file in a text editor.
4. In the `[FieldProcessing]` section, go to the end of the list of field processing sections and add the next number. Set this parameter to the name of the field process that you want to create. For example:

```
[FieldProcessing]
...
10=SetNumericDateFields
11=SetReferenceMemoryMappedFields
```

5. Create a section for the field process. Create a property for the process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[SetReferenceMemoryMappedFields]
Property=ReferenceMemoryMappedFields
PropertyFieldCSVs=*/UserName,*/Author
```

6. Create a section for the new property, in which you set the `ReferenceMemoryMappedType` property to `True`. For example:

```
[ReferenceMemoryMappedFields]
ReferenceMemoryMappedType=True
```

7. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
8. Index the parent documents, and then index the rest of the documents. Alternatively, you can index all the documents and then send a `DREREGENERATE` index action to fix the parent-child relationships. For example:

```
http://localhost:9001/DREREGENERATE?Type=ReferenceMemoryMapped
```

Related Topics

- ["Configure a Field Process" on page 76](#)

Send Linked Queries

Linked queries rely on the `LINK` field specifier, and the `LinkFieldText` action parameter. You can use these options in the `Query`, `Suggest`, `SuggestOnText`, and `GetQueryTagValues` actions.

To create a linked query, set the following parameters:

Text	The query text that you want to find in the documents.
FieldText	<p>A LINK field expression, setting the ReferenceMemoryMappedType field that connects the results documents to the parent documents that you want to link to. For example:</p> <pre>LINK{}:UserName</pre> <p>You can also add any other FieldText expressions that you want to use to restrict the results documents.</p>
LinkFieldText	<p>A FieldText expression that you want to apply in the parent documents (the documents in which the ReferenceMemoryMapped values occur in a ReferenceType field).</p> <p>The LinkFieldText expression must be a simple FieldText expression, that is, it must use either all AND or all OR Boolean expressions. The fields that you use in the expressions must be MatchType, NumericDateType, NumericType, or ParametricType (when ParametricNumericMapping is enabled).</p> <p>You cannot use metadata fields in the LinkFieldText expression, including the IDOL default metadata fields, and fields created by using the UserMetaFields parameter.</p>

Example

A Users database has a set of user profile documents that use the user name as the document reference, and list interests and other user information. For example:

```
<DOCUMENT>
  <DREREFERENCE>User01</DREREFERENCE>
  <INTEREST>Soccer</INTEREST>
  <COUNTRY>USA</COUNTRY>
  <JOINED>01/01/2012</JOINED>
</DOCUMENT>
<DOCUMENT>
  <DREREFERENCE>User02</DREREFERENCE>
  <INTEREST>Food</INTEREST>
  <COUNTRY>UK</COUNTRY>
  <JOINED>02/06/2011</JOINED>
</DOCUMENT>
```

The same index has a SocialMedia database, which contains posts by these users. For example:

```
<DOCUMENT>
  <DREREFERENCE>SM0001</DREREFERENCE>
  <USERNAME>User01</USERNAME>
  <DRECONTENT>Just got my tickets for the final in Brazil!</DRECONTENT>
</DOCUMENT>
<DOCUMENT>
  <DREREFERENCE>SM0002</DREREFERENCE>
  <USERNAME>User02</USERNAME>
```

```
<DRECONTENT>Brazil nuts are so moreish...</DRECONTENT>  
</DOCUMENT>
```

The following standard query returns both of these documents:

```
action=Query&Text=Brazil
```

You can use a linked query to specify additional restrictions. The following query finds posts about Brazil by users who are interested in soccer:

```
action=Query&Text=Brazil&FieldText=LINK{}:USERNAME&LinkFieldText=MATCH  
{Soccer}:INTEREST
```

This returns the document *SM0001*, but not the document *SM0002*.

This query requires the following field configuration:

- USERNAME is configured as ReferenceMemoryMappedType.
- INTEREST is configured as MatchType.

Combine Different Search Types

This section describes how you can combine search types.

Synonym and Boolean Searches

If you combine a synonym search with a Boolean search, IDOL Server obeys the Boolean rules while it processes the synonym query. For example:

```
action=Query&Text=cat+AND+dog&Synonym=True
```

If the synonym file contains synonyms for the terms *cat* and *dog*, IDOL Server returns only documents that contain both the term *cat* or one of the synonyms for *cat*, and the term *dog* or one of the synonyms for *dog*. For example, documents that contain *mouser* and *cur*, *cat* and *man's best friend*, and so on.

Synonym Search and Field Restrictions

You can use field restrictions within a synonym search to return only documents that contain the synonym within a specific field. For example:

```
action=Query&Text=cat:Title&Synonym=True
```

If the synonym file lists synonyms for the term *cat*, IDOL Server returns only documents that contain the term *cat* or one of the synonyms for *cat* in their *Title* fields.

Soundex and Proper Names Searches

You can send a proper name search to IDOL Server, which includes a SOUNDEX keyword search. However, because the SOUNDEX keywords are spelled phonetically, IDOL Server cannot match proper names for them.

Soundex and Boolean Searches

If you combine a Soundex keyword search with a Boolean search, IDOL Server obeys the Boolean rules while it processes the Soundex keyword search. For example:

```
action=Query&Text=Munich+AND+SOUNDEX(einstine)
```

This query returns only documents that contain both the term *Munich* and a term that phonetically matches *einstine* (for example, *Einstein*).

Soundex and Proximity Searches

If you combine a Soundex keyword search with a proximity search, IDOL Server obeys the proximity rules while it processes the Soundex keyword search. For example:

```
action=Query&Text=Munich+NEAR2+SOUNDEX(einstine)
```

This query returns only documents in which the term *Munich* is no more than two terms away from a term that phonetically matches *einstine* (for example, *Einstein*).

Soundex Search and Field Restrictions

You can use field restrictions to restrict a Soundex keyword search to specific fields. For example:

```
action=Query&Text=SOUNDEX(einstine:Name)
```

This query returns only documents that contain a term that phonetically matches *einstine* (for example, *Einstein*) in their Name fields.

Exact Phrase and Boolean Searches

If you combine an exact phrase search for multiple phrases (in which the individual phrases are framed by quotation marks) with a Boolean search, IDOL Server obeys the Boolean rules while it processes the exact phrase search. To ensure that Boolean operators tie to an entire phrase, enclose the phrase in brackets. For example:

```
action=Query&Text=("Egyptian cats")+AND+("Phoenician sailors")
```

This query returns only documents that match the phrase *Egyptian cats*, as well as the phrase *Phoenician sailors* (after stemming and stopping).

```
action=Query&Text=("Egyptian cats")+NOT+("Phoenician sailors")
```

This query returns only documents that match the phrase *Egyptian cats*, but not the phrase *Phoenician sailors* (after stemming and stopping).

Exact Phrase and Proximity Searches

If you combine an exact phrase search for multiple phrases (in which you frame the individual phrases in quotation marks) with a proximity search, IDOL Server obeys the proximity rules while it processes

the exact phrase search. To ensure that proximity operators tie to an entire phrase, enclose the phrases in brackets. For example:

```
action=Query&Text=("Egyptian cats")+ NEAR2+("Phoenician sailors")
```

This query returns only documents in which a match of the phrase *Egyptian cats* is no further than two words away from a match of the phrase *Phoenician sailors* (after stemming and stopping).

```
action=Query&Text=("Egyptian cats")+BEFORE+("Phoenician sailors")
```

This query returns only documents that match the phrase *Egyptian cats* and the phrase *Phoenician sailors* (after stemming and stopping). The phrase *Egyptian cats* must occur before the phrase *Phoenician sailors*.

Exact Phrase Search and Field Restrictions

You can use field restrictions to restrict an exact phrase search to specific fields. For example:

```
action=Query&Text="birds of prey" "bird watching":DRETITLE
```

This query returns documents that contain a match for *birds of prey* in any field or a match for *bird watching* in the DRETITLE field (after stemming and stopping).

Boolean and Proximity Searches

Boolean and proximity operators have the following precedence:

```
Highest:      NOT
              NEAR; DNEAR; XNEAR; YNEAR
              AND; BEFORE; AFTER; WHEN; SENTENCE; DSENTENCE; PARAGRAPH
Lowest:      OR; WNEAR; EOR
```

Operators that have the same level of precedence have neither left or right associativity. You can use brackets to bind terms together as appropriate. Proximity operators must have terms on either side, and cannot be adjacent to brackets.

Boolean Search and Field Restrictions

If you combine a field restriction keyword search with a Boolean search, IDOL Server obeys the Boolean rules while returning only documents that contain the specified values in the specified fields. For example:

```
action=Query&Text=cat:Animal AND dog:Animal:Fauna
```

This query returns only documents that contain the term *cat* in their *Animal* fields and the term *dog* in their *Animal* or *Fauna* fields.

Proximity Searches and Field Restrictions

If you combine a field restriction keyword search with a proximity search, IDOL Server obeys the proximity rules while returning only documents that contain the specified values in the specified fields.

For example:

```
action=Query&Text=cat NEAR2 dog:Animal
```

This query returns only documents that contain the term *cat* and term *dog* in their *Animal* fields. The terms must be no more than two terms away from each other.

Wildcards in Queries

Use Wildcard matching sparingly, because it slows IDOL Server performance.

You can use the following Wildcard characters in query `Text` and `FieldText` queries:

?	to match one character.
*	to match zero, one, or more characters.

You can use the `UnstemmedMinDocOccs` parameter in the `[Server]` section of the configuration file to specify the number of documents in which a term must occur to consider it in a Wildcard search.

Note: To use Wildcards to search for numbers, set `SplitNumbers` (described in the `[Server]` configuration section of the *IDOL Server Reference*) to `False`.

Wildcards in Query Text

You can use Wildcards in the `Text` parameter of the `Query` action.

You can use the `Text` action parameter to specify field restrictions for result documents, as long as the fields that you restrict results to are `Index` fields in IDOL Server. If you match fields, you can use Wildcards and match multiple fields simultaneously by separating them with colons.

Related Topics

- ["Set up the Field Index Process" on page 38](#)

Example Wildcard Queries

The following examples demonstrate how to use Wildcards in query text.

Example 1

```
action=Query&Text=Mi?rotech
```

This query returns documents that contain the term *Mikrotech* or *Microtech*.

Example 2

```
action=Query&Text="Co*ins":Name:Author+Arm?dale:Title
```

This query returns documents that contain a `Name` or `Author` field whose value matches the Wildcard string `Co*ins` (for example, *Collins*) and documents that contain a `Title` field whose value matches the Wildcard string `Arm?dale` (for example *Armada*).

Example 3

Consider the following scenarios:

A. The IDOL Server has two indexed documents. One document contains the word *rollerskating*, and the other contains the word *rollerskaters* (both of which stem to *ROLLERSK*).

B. The IDOL Server has only one indexed document. It contains the word *rollerskater* (which also stems to *ROLLERSK*).

- Send the following query:

```
action=Query&Text=rollersk*
```

This query returns all documents that contain any terms that stem to *ROLLERSK*. It returns both documents in situation A and the one document in situation B.

- Send the following query:

```
action=Query&Text=rollerskatin*
```

A Wildcard query returns documents only if the IDOL Server index contains one or more matches to the Wildcard term. If the term matches, it is then stemmed, and all occurrences of the stem (in any document) are also considered matches.

Therefore, this query returns both documents in situation A (because *rollerskatin** matches *rollerskating*, and because it stems to *ROLLERSK*, which has matches in both documents). It does not return the document in situation B (because *rollerskatin** does not match *rollerskater*).

- Send the following query when the `AdvancedSearch` configuration parameter is set to `True`:

```
action=Query&Text="rollerskatin*"
```

This query returns documents that contain the Wildcard term. However, because `AdvancedSearch` is enabled *and* the Wildcard term is in quotation marks, the Wildcard term is not stemmed and additional matches to its stem do not return.

Therefore, with this query, only the *rollerskating* document returns in situation A, and no document returns in situation B.

Wildcards in FieldText Queries

If you use a `Query`, `Suggest`, or `SuggestOnText` action to send a `FieldText` query to IDOL Server (by using the `FieldText` action parameter), you can use Wildcards to match a single string or to match multiple strings.

Note: When identifying fields, use the format `/FieldName` to match root-level fields, `FieldName` to match all fields except root-level, or `/Path/FieldName` to match fields that the specified path points to.

All string matching is case insensitive.

Matches for One or More Strings

The field specifier `WILD` allows you to Wildcard match:

- one or more strings
- one or more strings that consist of several words
- one or more strings that contain punctuation
- one or more strings that consist of several words and punctuation

Strings can contain punctuation (except braces), which means that if you want to match a string that contains HTML with IDOL Server content, percent-encode the HTML to avoid confusion with “&” and so on.

To match a string that contains a comma, percent-encode the comma as `%2C`, otherwise IDOL Server reads it as a separator.

You can match multiple fields simultaneously by separating them with colons.

Examples:

```
action=Query&FieldText=WILD{wom?n }:Clothes
```

The `Clothes` field must contain a word that matches the specified Wildcard string (for example, *woman* or *women*) for the document to return as a result.

```
action=Query&FieldText=WILD{of mice and m?n}:Title:Book
```

The `Title` or `Book` field must contain a string that matches the specified Wildcard string (for example, *of mice and man* or *of mice and men*) for the document to return as a result.

```
action=Query&FieldText=WILD{Glory is fleeting\, but * is forever}:QuotesNapoleon
```

The `QuotesNapoleon` field must contain a string that matches the specified Wildcard string (for example, *Glory is fleeting, but obscurity is forever*) for the document to return as a result.

```
action=Query&FieldText=WILD{*.html,* .htm}:URL
```

The `URL` field value must end with *.html* or *.htm* for the document to return as a result.

```
action=Query&FieldText=WILD{passi*incarnata}:Climbers
```

The `Climbers` field must contain a phrase that begins with *passi* and ends with *incarnata* (for example, *passionflower incarnata* or *passiflora incarnata*) for the document to return as a result.

```
action=Query&FieldText=WILD{ passi*incarnata,passi*alata*}:Climbers
```

The `Climbers` field must contain a string that matches one of the specified Wildcard strings (for example, *passionflower incarnata*, *passiflora incarnata*, *passionflower alata*, *passiflora alata*, *passionflower alata shannon*, or *passiflora alata shannon*) for the document to return as a result.

```
action=Query&FieldText=WILD{*www.example.com*.txt,*www.example.com*.pdf}:PATH:URL
```

The `PATH` or `URL` field must contain a path that contains *www.example.com* and ends with *.txt* or *.pdf* (for example, *http://www.example.com/files/doc.txt* or *http://www.example.com/fields/technicalbrief.pdf*) for this document to return as a result.

Wildcard Searches in Japanese, Chinese, Korean, and Thai

Asian languages do not include spaces or word boundaries. For this reason IDOL Server applies *sentence breaking* to Asian text when it processes it, to split the text into individual words or *terms*.

You can carry out Wildcard searches in Japanese, Chinese, Korean, and Thai, provided that you query IDOL Server with one or more terms rather than a single string in which words are not delimited by spaces.

The question mark (?) Wildcard might not behave as expected, because it represents a single character, and each Asian *letter* actually consists of multiple characters (usually two). For example, if you want to use a ? single-character Wildcard in a multibyte language query, you must use one ? character for each byte (for example, ??? for a single Japanese character).

Query for Nonalphanumeric Characters

Although IDOL Server does not store nonalphanumeric characters, you can query for them using FieldText queries. To speed up the processing time of the query, combine the FieldText queries with an ordinary text query, by using the Text parameter and the FieldText parameter as follows.

Text

Specify your entire query string including the nonalphanumeric characters. If you include any special characters, you must treat them appropriately, as shown in the following table.

Treat...	Like this:
&	Percent-encode ampersands that are part of the query string.
~ [] * ? : () "	<p>If you have not changed the IDOL Server default behavior, replace the character with a space.</p> <p>If you configured IDOL Server not to treat any of these characters as a separator by specifying them in the <code>DiminishSeparator</code> configuration parameter, remove the character from the search string.</p> <p>Alternatively, you can instruct IDOL Server to interpret the following characters as normal characters in query syntax by setting the <code>IgnoreSpecials</code> action parameter to <code>True</code> for the <code>Query</code> or <code>GetQueryTagValues</code> action:</p> <p>*?": () and Boolean or proximity operators, for example AND, NOT, OR, NEAR, and so on.</p> <p>This disables Wildcard searching, phrase queries, field restrictions, and Boolean operations.</p>

FieldText

You must percent-encode all strings in `FieldText`.

You must distinguish punctuation in IDOL query syntax (such as commas and braces) from the same punctuation that might occur in strings. Use the following percent-encoding guidelines:

- percent-encode an ampersand (&) with %26
- percent-encode a backslash (\) with %5C
- percent-encode a comma (,) with %2C
- percent-encode a percentage sign (%) with %25
- To distinguish between braces in strings from query syntax braces, percent-encode left ({) and right (}) braces twice, with %257B and %257D.
- To distinguish between commas in strings from separator commas, percent-encode commas within strings twice (%252C).
- Leave commas that separate multiple items, and braces that start and end lists, as regular punctuation. There must be no spaces before or after these commas.

Use the `STRING` field specifier to search for your entire query string, including the nonalphanumeric characters, in the appropriate field in IDOL Server.

Examples

To search for *Auto**:

```
action=Query&Text=Auto&FieldText=STRING{Auto*}:DRECONTENT
```

To search for *yahoo!*:

```
action=Query&Text=yahoo!&FieldText=STRING{yahoo!}:DRECONTENT
```

To search for *AT&T*:

```
action=Query&Text=AT%26T&FieldText=STRING{AT%26T}:DRECONTENT
```

To search for *"r-t"*

```
action=Query&Text=r-t&FieldText=STRING{r-t}:DRECONTENT
```

To search for *"eat, drink and be merry"*

```
action=Query&Text=eat, drink and be merry&FieldText=STRING{eat%2c drink and be merry}:DRECONTENT
```

To search for *"politics [and] their effects"*

```
action=Query&Text=politics and their effects&FieldText=STRING{politics [and] their effects}:DRECONTENT
```

To search for *"*NSYNC"*

```
action=Query&Text=*NSYNC&IgnoreSpecials=True
```

Optimize Retrieval of Tagged Documents

To include one or more attributes that are specific to a document, you can create fields in the document when you store it in IDOL Server. When you send a query to IDOL Server, you can then use any of the fields that you created to restrict which documents IDOL Server returns.

For example, you can create fields that contain authors, categories, folders, product types, or any other attribute. You can then restrict a query to return only documents that were, for example, written by a specific author or belong to one or more specific categories.

When you restrict a query in this way, its processing time might increase. To keep the processing time to a minimum, use the appropriate query syntax for the fields that you create.

Query Syntaxes

Query processing time depends on the syntax that the query uses. Different syntaxes are available, and some require you to create fields in a specific way.

Fastest

Syntax:	<code>action=Query&Text=text&FieldText=EQUAL {numericalAttribute}:fieldName</code>
Requirements:	<ul style="list-style-type: none"> • create a numeric field for each attribute • store these fields as numeric fields in IDOL Server
Example:	<p>Four attributes are available to indicate which categories a document belongs to: England, France, Germany, USA</p> <p>It uses numbers to indicate which attributes apply to a document: 1—England, 2—France, 3—Germany, 4—USA</p> <p>In documents, you create a numeric field for each category that they belong to. For example, if a document belongs to the categories <i>France</i> and <i>USA</i>:</p> <pre>#DREFIELD Cat1="2" #DREFIELD Cat2="4"</pre> <p>The following query returns documents that match the specified <i>Text</i> and contain the value <i>4</i> in one of their <i>Cat</i> fields (for example, <i>Cat1</i>). The value <i>4</i> indicates that the documents belong to the <i>USA</i> category.</p> <pre>action=Query&Text=presidential election&FieldText=EQUAL{4}:Cat*</pre>
Syntax:	<code>action=Query&Text=text&FieldText=MATCH{attribute}:fieldName</code>
Requirements:	<ul style="list-style-type: none"> • create a field for each attribute

Example:	<p>Four attributes are available to indicate which categories a document belongs to: England, France, Germany, USA</p> <p>In documents, you create a field for each category that they belong to. For example, if a document belongs to the categories <i>France</i> and <i>USA</i>:</p> <pre>#DREFIELD Cat1="France" #DREFIELD Cat2="USA"</pre> <p>The following query returns documents that match the specified Text and contain the value <i>France</i> in one of their Cat fields (for example, Cat1).</p> <pre>action=Query&Text=presidential_election&FieldText=MATCH {France}:Cat*</pre>
Syntax:	<pre>action=Query&Text=text&FieldText=BITAND {numericalAttribute}:fieldName</pre>
Requirements:	<ul style="list-style-type: none"> • assign each attribute a <i>bit</i> • create a numeric field for all attributes (if you have more than 32 bits, you need more fields) • store this field as a numeric field in IDOL Server
Example:	<p>Four attributes are available to indicate which categories a document belongs to: England, France, Germany, USA</p> <p>Assign a bit value to each attribute: 1—France, 2—England, 4—Germany, 8—USA</p> <p>In documents, you create a field for the categories that they belong to. For example, if a document belongs to the categories <i>France</i> and <i>USA</i>:</p> <pre>#DREFIELD Cat="9"</pre> <p>The following query returns documents that match the specified Text and contain the value <i>10</i> in its Cat field. The value <i>10</i> indicates that documents must belong to the <i>England</i> or the <i>USA</i> category.</p> <pre>action=Query&Text=presidential_election&FieldText=BITAND{10}:Cat</pre>

Slowest

Syntax:	<pre>action=Query&Text=text&FieldText=STRING{attribute}:fieldName</pre>
Requirements:	<ul style="list-style-type: none"> • create a field that contains a comma-separated list of the document attributes
Example:	<p>Four attributes are available to indicate which categories a document belongs to: England, France, Germany, USA</p>

In documents, you create a field that contains a comma-separated list of the categories that the documents belong to. For example, if a document belongs to the categories *France* and *USA*:

```
#DREFIELD Cat="France,USA"
```

The following query returns documents that match the specified `Text` and contain the value *France* in their `Cat` field.

```
action=Query&Text=presidential election&FieldText=STRING  
{France}:Cat
```


Chapter 15: Customize Results

You can change the IDOL Server results display from its default state. You can set the number of results to display, batch results, change the order in which to display results, set additional fields to display, apply templates, or cluster results.

You can also choose to generate document summaries, display hyperlinks, display query summaries, and suggest alternate spellings for query terms.

- [Change the Results Display](#)289
- [Change the Field Display](#) 293
- [Use XSLT Templates to Change Output Format](#)297
- [Generate Summaries](#) 299
- [Cluster Results](#) 300
- [Generate Hyperlinks](#) 302
- [Provide Spell Correction](#) 303
- [Automatic Query Guidance](#) 304
- [Generate Query Summaries \(Dynamic Thesaurus\)](#) 307
- [Generate Dynamic Clusters](#) 310

Change the Results Display

You can control the number of results to display in a results list, and the sort order of the returned documents.

Set the Number of Results to Display

When you perform a query, IDOL Server by default returns a maximum of six results. You can modify the number of results that IDOL Server returns by adding the `MaxResults` parameter to your query and setting it to the number of results to return. For example:

```
http://IDOLhost:port/action=Query&Text=Harry Potter and the Goblet of  
Fire&MaxResults=10
```

In this example, IDOL Server returns 10 results for the query, if at least 10 results are available.

Change Result Sorting (Display Order)

By default, IDOL Server lists query results in order of relevance. To weight and rank the documents it returns by statistical relevance, IDOL Server uses complex algorithms that use a combination of information theory and Bayesian methods. It makes use of information theoretic values calculated dynamically for all concepts on indexing, which allows it to evaluate relevance both as a percentage, and in the case of agents, as absolute values.

In practice, the relevance acts as a measure of the conceptual overlap between the query text and the text within a document. You can affect the relevance in several ways. You can apply extra weight to certain fields by associating a weighting factor with them at indexing time. For example, you can give extra weight when query terms appear in the document title as opposed to the body of the text.

You can change the order in which IDOL Server returns query results by adding the `Sort` parameter to your `Query`, `Suggest`, `SuggestOnText`, `GetTagValues`, or `GetQueryTagValues` action.

Sort Options for Query, Suggest, and SuggestOnText

The following table lists the `Sort` options that are available for the `Query`, `Suggest`, and `SuggestOnText` actions.

Off	Displays results unsorted.
AutnRank	Displays results in order of the value in their <code>AutnRankType</code> field. This method lists the document with the highest <code>AutnRankType</code> field value first.
Cluster	Displays results in order of cluster (in decreasing cluster ID order), if you also add <code>Cluster=True</code> to the action. If you specify <code>Cluster</code> as one of multiple sorting options, it automatically takes precedence over the other sorting methods, even if you did not put it in first place.
Database	Displays results in order of database number (in increasing order). Define the database numbers in the IDOL Server configuration file.
Date	Displays results in order of their date (the date contained in the <code>DateType</code> fields). This method lists the most recent document first. If several documents have the same date, their display order is determined by their <code>autn:docid</code> (document ID) number (the highest <code>autn:docid</code> is listed first).
Distcartesian	Displays results according to their distance from a specified point using Cartesian coordinates (X/Y). The option has this format: <code>sort=Distcartesian{coordX,coordY}:X:Y</code> where: <ul style="list-style-type: none"> • <code>coordX</code> is the specified x coordinate. • <code>coordY</code> is the specified y coordinate. • <code>X</code> is the document field that contains the X coordinate. • <code>Y</code> is the document field that contains the Y coordinate. You must specify two fields in the order X:Y.
Distspherical	Displays results according to their distance from a specified point using spherical coordinates (latitude and longitude). The option has this format: <code>sort=Distspherical{lat,long}:Latfield:Longfield</code> where:

	<ul style="list-style-type: none"> • <i>Lat</i> is the specified latitude. Specify latitude positions south of the equator as negative. • <i>Long</i> is the specified longitude. Specify longitude positions west of the Greenwich Meridian as negative. • <i>Latfield</i> is the document field that contains the latitude. • <i>Longfield</i> is the document field that contains the longitude. <p>You must specify two fields in the order <code>latitude:longitude</code>.</p>
DocIDDecreasing	Displays results in order of their <code>autn:docid</code> (document ID) number. This method lists the document with the highest <code>autn:docid</code> first.
DocIDIncreasing	Displays results in order of their <code>autn:docid</code> (document ID) number. This method lists the document with the lowest <code>autn:docid</code> first.
<i>fieldName</i> : <i>sortMethod</i>	<p>Displays results in the order specified by <i>sortMethod</i>, based on the value of the IDOL Server field <i>fieldName</i>. This option is optimized if <i>fieldName</i> is a <code>MatchType</code> field. The following sort methods are available:</p> <ul style="list-style-type: none"> • alphabetical. Determines the display order of results by the string contained in <i>fieldName</i>. This method lists results in alphabetical order. (Sorting is faster if <i>fieldName</i> is <code>SortType</code>.) • decreasing. Determines the display order of results by the number or string contained in <i>fieldName</i>. This method lists the result with the highest number or the result that is last in alphabetical order first. For <code>NumericType</code> fields, this method is equivalent to <code>numberdecreasing</code>. For <code>SortType</code> fields, this method is equivalent to <code>reversealphabetical</code>. • increasing. Determines the display order of results by the number or string contained in <i>fieldName</i>. This method lists the result with the lowest number or the result that is first in alphabetical order first. For <code>NumericType</code> fields, this method is equivalent to <code>numberincreasing</code>. For <code>SortType</code> fields, this method is equivalent to <code>alphabetical</code>. • numberdecreasing. Determines the display order of results by the number contained in <i>fieldName</i>. This method lists the result with the highest number first. (Sorting is faster if <i>fieldName</i> is <code>NumericType</code>.) • numberincreasing. Determines the display order of results by the number contained in <i>fieldName</i>. This method lists the result with the lowest number first. (Sorting is faster if <i>fieldName</i> is <code>NumericType</code>.) • reversealphabetical. Determines the display order of results by the string contained in <i>fieldName</i>. This method lists results in reverse alphabetical order. (Sorting is faster if <i>fieldName</i> is <code>SortType</code>.)
Random	Displays results in random order.
Relevance	<p>Displays results in order of their relevance. This method lists the document with the highest relevance first.</p> <p>If documents have the same relevance, this method determines their display order by their <code>autn:docid</code> (document ID) number, and lists the highest <code>autn:docid</code> first.</p>

ReverseDate	<p>Displays results in order of their date (the date contained in the DateType fields). It lists the oldest document first.</p> <p>If several documents have the same date, this method determines their display order by their autn:docid (document ID) number, and lists the highest autn:docid first.</p>
-------------	--

If you want to sort results by several criteria, you can specify them as follows:

sortOption1+sortOption2+...

Example 1:

`http://MyHost:20000/action=Query&Text=presidential elections&Sort=Date`

In this example, IDOL Server displays results in order of the document date.

Example 2:

`http://MyHost:20000/action=Query&Text=presidential elections&Sort=DRETITLE:reversealphabetical`

In this example, IDOL Server displays results in reverse alphabetical order by their DRETITLE.

Example 3:

`http://MyHost:20000/action=Query&Text=presidential elections&Sort=Relevance+DRETITLE:alphabetical+Date`

In this example, results order by Relevance, then by their DRETITLE, and then by their Date.

Sort for GetTagValues and GetQueryTagValues

The following table lists the Sort options that are available for the GetTagValues and GetQueryTagValues actions.

Off	Displays results unsorted.
Alphabetical	Determines the display order of results by the string contained in the IDOL Server field. Displays results in alphabetical order.
Date	Determines the display order of results by the date contained in the IDOL Server field. This method lists the most recent result first.
NumberDecreasing	Determines the display order of results by the number contained in the IDOL Server field. This method lists the result with the highest number first.
NumberIncreasing	Determines the display order of results by the number contained in the IDOL Server field. This method lists the result with the lowest number first.
ReverseAlphabetical	Determines the display order of results by the string contained in the IDOL Server field. This method displays results in reverse alphabetical order.
ReverseDate	Determines the display order of results by the date contained in the IDOL Server field. This method lists the oldest result first.

For the `GetQueryTagValues` action, these `Sort` options are also available:

<code>DocumentCount</code>	If you set the <code>DocumentCount</code> action parameter to <code>True</code> , you can use this option to display results in order of their document count (in decreasing order).
<code>ReverseDocumentCount</code>	If you set the <code>DocumentCount</code> action parameter to <code>True</code> , you can use this option to display results in the reverse order of their document count (in increasing order).

For example:

```
http://MyHost:20000/action=GetQueryTagValues&FieldName=GRAPE,COUNTRY &Text=A smooth red wine that complements game&Sort=Alphabetical
```

In this example, IDOL Server displays results in alphabetical order.

Batch (Page) Results

If you are running a `Query`, `Suggest`, or `SuggestOnText` action, you can use the `MaxResults` parameter with the `Start` parameter to display only `MaxResults` number of results, from the `Start` position onwards.

The `Start` value must be smaller than the `MaxResults` value because if the position it specifies does not fall within the generated range of results, no results return for the query, even though results might be available.

Example 1:

```
http://IDOLhost:port/action=Query&Text=Harry Potter and the Goblet of Fire&MaxResults=6&Start=6
```

In this example, if a total of 10 results is available for the query, only the sixth result returns for the query.

Example 2:

```
http://IDOLhost:port/action=Query&Text=Harry Potter and the Goblet of Fire&MaxResults=100&Start=6
```

In this example, if a total of 10 results is available for the query, results 6 through 10 (inclusive) return for the query.

Example 3:

```
http://IDOLhost:port/action=Query&Text=Harry Potter and the Goblet of Fire&MaxResults=5&Start=6
```

In this example, no results return for the query (even if results are available).

Change the Field Display

To minimize query processing time, by default IDOL Server returns only a subset of fields for each query result. You can display additional fields (both metafields and document fields).

Returned Fields

By default, IDOL Server returns the following fields for each result:

<autn:reference>	The reference of the result document.
<autn:id>	The ID of the result document.
<autn:section>	The section of the document.
<autn:weight>	The relevance weight of the result document.
<autn:links>	The terms in the query text that match in the result document.
<autn:database>	The database that contains the result document.
<autn:title>	The title of the result document.

You can modify the set of fields that IDOL Server returns, as described in the following section.

Display Additional Metafields

You can display all metafields that are available for results by adding `XMLMeta=True` to your query. For example:

```
http://MyHost:20000/action=Query&Text=Choreographic techniques&XMLMeta=True
```

This action instructs IDOL Server to return all available metafields for documents that match the text specified by the `Text` parameter.

Related Topics

- ["Metadata Fields" on page 96](#)

Display Document Fields

You can configure IDOL Server to display specific document fields whenever it returns query results. Alternatively, you can specify the document fields that IDOL Server returns when you perform an individual query.

Configure IDOL Server to Always Display Specific Fields

To display specific document fields every time IDOL Server returns query results, create a field process in the IDOL Server configuration file that identifies the document fields that you want to display. After you create this field process, the fields that you identified always display when IDOL Server returns query results (provided that you set the query action `Print` parameter to `Fields`, which is its default value).

To configure IDOL Server to always display specific fields

1. Open the IDOL Server configuration file in a text editor.

2. In the `[FieldProcessing]` section, list a new field process whose name indicates that the process identifies which fields are displayed.

For example:

```
[FieldProcessing]
0=MyFirstProcess
1=PrintFields
```

3. Create a section for the new field process that you listed, in which you create a property for the process (you define the property later by setting one or more applicable configuration parameters). Use the `PropertyFieldCSVs` parameter to identify the fields that you want to display.

Note: The property that you create must not have the same name as the process.

For example:

```
[PrintFields]
Property=Print
PropertyFieldCSVs=*/SUMMARY,*/DRECONTENT
```

4. Create a section for the property in which you set the `PrintType` parameter to `True`. This displays the associated `PropertyFieldCSVs` fields for query results. For example:

```
[Print]
PrintType=True
```

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

Every time you perform a `Query`, `Suggest`, `SuggestOnText`, or `GetContent` query, IDOL Server now returns `SUMMARY` and `DRECONTENT` field of the result documents, in addition to the metafields that it displays by default.

Display Specific Fields for Individual Queries

To return specific document fields for individual queries, use the `Print` or `PrintFields` action parameters when you perform a `Query`, `Suggest`, `SuggestOnText`, or `GetContent` query action.

- **Print.** Use one of the following options to determine the types of fields to display:

All	All XML fields. Note: This option does not display temporary fields that you create in a Query action by using the <code>UserMetaFields</code> parameter. To display those fields, you must list them in the <code>PrintFields</code> parameter.
AllSections	All fields, and all sections in section breaking fields.
Date	Fields with the <code>DateType</code> property.
Fields	Fields with the <code>PrintType</code> property.
Index	Fields with the <code>Index</code> property.
IndexText	The contents of fields with the <code>Index</code> property.
NegativeFields	All fields, except those listed in the <code>PrintFields</code> action parameter.
None	The first reference field only.
NoResults	No results or fields. The query returns only the number of hits.
Parametric	Fields with the <code>ParametricType</code> property.
Reference	Fields with the <code>ReferenceType</code> property.
Source	Fields with the <code>SourceType</code> property.

For example:

```
http://12.3.4.56:4000/action=Query&Text=Hogwarts school of witchcraft and wizardry&Print=Index
```

This query returns results that are conceptually similar to the specified query text. Each result returns with fields that were set up as `Index` fields in the IDOL Server configuration file.

- **PrintFields.** A list of fields that you want to display in the results.

If you also set `Print`, `Print` generally takes precedence. However:

- when you set `Print` to `Fields`, only the specified `PrintFields` are displayed.
- when you set `Print` to `NegativeFields`, the specified `PrintFields` are not displayed.

For example:

```
http://12.3.4.56:4000/action=Query&Text=Hogwarts school of witchcraft and wizardry&PrintFields=Author,Title
```

This query returns results that are conceptually similar to the specified query text. Each result returns with its `Author` and `Title` fields.

Related Topics

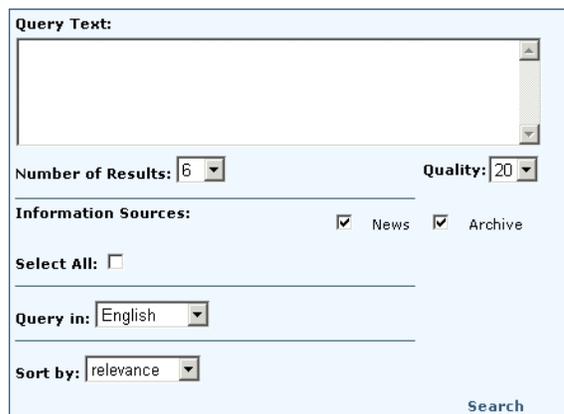
- ["Display Online Help" on page 30](#)

Use XSLT Templates to Change Output Format

You can use XSLT templates to format the results that IDOL Server returns. You can write your own XSLT templates or use the templates that the IDOL Server installation includes in the `acitemplates` directory:

- `QueryForm.tmp1`. You can apply this template to the `Query` action to produce a simple HTML query interface from which you can query IDOL Server.

The default version of this template has the following appearance.



The screenshot shows a web form titled "Query Text:" with a large text input area. Below the input area are two dropdown menus: "Number of Results:" set to "6" and "Quality:" set to "20". Underneath is a section for "Information Sources:" with checkboxes for "News" and "Archive", both of which are checked. There is a "Select All:" checkbox which is unchecked. Below that is a "Query in:" dropdown menu set to "English". At the bottom left is a "Sort by:" dropdown menu set to "relevance". A "Search" button is located at the bottom right of the form.

- `GetStatusForm.tmp1`. You can apply this template to the `GetStatus` action to generate HTML formatted status information.

The default version of this template has the following appearance.

Autonomy IDOL Server Version 5.0.4 Build 65249

ACI Port: 9000
Index Port: 9001
Service Port: 9002
Directory: D:\Autonomy\IDOLserver\IDOL
ACI Threads: 4
Licensed Languages: English

Documents: 7185
Document Sections: 7185
Committed Sections: 7196
Terms: 77134
Total Terms: 94421
Total Hashes: 1048576
Record Size: 53
Max Occurrences: 273
Terms Per Doc: 50
Suggest Terms: 40

Caches:

Term Cache		Index Cache	
Terms:	10292	Terms:	0
Used:	6814 Kb	Used:	0 Kb
Max:	102400 Kb	Max:	102400 Kb
Requests:	17713	Blocks:	1
Hit Rate:	38		

Databases: 2

No.	Name	Documents	Document Sections	Readonly	Expiry Hours	Expiry Action
1	News	7185	7185	false	-	-
2	Archive	0	0	false	-	-

Enable the XSLT Templates

Use the following procedure to enable the XSLT templates.

To enable the XSL templates

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Server] section.
3. Set the XSLTempLates setting to True. (If the [Server] section does not contain this setting, add it.)
4. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Apply XSLT Templates to Actions

To apply a template to action output, add the following parameters to the action:

Template	Specify the name of the template that you want to use to format action output.
OutputEncoding	Set this parameter to UTF8.

Example 1:

```
action=GetStatus&Template=QueryForm&OutputEncoding=UTF8
```

In this example, the `QueryForm.tmp1` XSLT template is applied to a `GetStatus` action.

Example 2:

```
action=GetStatus&Template=GetStatusForm&OutputEncoding=UTF8
```

In this example, the `GetStatusForm.tmp1` XSLT template is applied to a `GetStatus` action.

Note: If the action returns an error response, IDOL Server does not apply the XSLT template to the response.

Generate Summaries

You can specify several types of document summaries to return with query results, or you can directly generate summaries for arbitrary documents or blocks of text.

Types of Summaries

IDOL Server can automatically generate one of these summary types for the results it produces. It generates all summaries in real time.

- **Concept.** A conceptual summary of each result document. A concept summary contains sentences that are typical of the result content (these sentences can be from different parts of the result document).
- **Context.** A conceptual summary of each result document, biased by any terms in the query `Text` and `FieldText` parameters. A context summary contains sentences that are particularly relevant to the terms in the query. These sentences can be from different parts of the result document.

Note: To ensure that the summary contains the query terms, use the `Characters` action parameter, rather than `Sentences`. You can also use the `MaxSourceCharacters` configuration parameter to use more than the first 10,000 characters of a document to generate summaries.

- **Quick.** A brief summary of each result document. A quick summary contains the first few sentences of the result document.
- **ParagraphConcept.** A conceptual summary of each result document. The summary contains the paragraphs that are most typical of the content of the result document. These paragraphs can be from different parts of the result document.
- **ParagraphContext.** A conceptual paragraph summary of each result document, biased by any terms in the query `Text` and `FieldText` parameters. This summary contains paragraphs that are particularly relevant to the terms in the query.

Return Summaries with Query Results

Use the following procedure to return summaries with query results.

To generate summaries for query results

1. Send a `Query`, `Suggest`, or `SuggestOnText` action to IDOL Server that includes the `Summary` parameter. Set the `Summary` parameter to the type of summary that you want to return for results (`Concept`, `Context`, `Quick`, `ParagraphConcept`, or `ParagraphContext`). For example:

```
action=Query&Text=Undulant fever&Summary=Concept
```

Each result of this query returns with a conceptual summary.

2. Optionally, make the following settings in the IDOL Server configuration file, depending on the type of summary that you want to generate:
 - **For Concept, Context, or Quick summaries.** Use the `SourceType` property to specify the fields to generate the summary from.
 - **For Concept or Context summaries.** In the `[Summary]` section, set the `MinWordsPerSentence` parameter to the minimum number of words that a sentence must contain to consider it as a sentence for the summary.
 - **For Context summaries.** In the `[Server]` section, set the `ContextSummaryQueryTermWeight` parameter to the weight that it must use for the terms in user queries. The context summary gives this weight to sentences that contain terms in common with the query text. The other terms are given their APCM weight.
3. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Related Topics

- ["About Fields" on page 73](#)

Note: If you remove fields from the Source fields list, you must first perform a `DREINITAL` index action and then reindex the content into the IDOL Server. You cannot remove source fields after you index the data.

Summarize Text or Documents

To generate summaries for text or documents, send a `Summarize` action to IDOL Server that includes the `Summary` parameter. Set the `Summary` parameter to the type of summary to generate (`Concept`, `Quick`, `ParagraphConcept`, or `Concepts`).

To generate a summary for text, identify the text in the `Text` parameter. To generate a summary for one or more documents, identify them in the `ID` or `Reference` parameters.

For example:

```
action=Summarize&ID=30&Summary=Concept
```

This action generates a concept summary from the content of the document with the ID 30.

Cluster Results

If you perform a `Query`, `Suggest`, or `SuggestOnText` query, you can set `Cluster` to `True` in the action to cluster the results that the query produces.

IDOL Server returns an `<autn:cluster>` field for each result that contains the ID of the cluster into which the result has been grouped. ID 1 is given to the cluster that forms around the most relevant result document. After this cluster forms, the cluster with the ID 2 forms around the most relevant of the remaining results, and so on.

The `ClusterThreshold` parameter in the IDOL Server configuration file sets the percentage relevance that results must have to each other to group them in one cluster.

For example:

```
action=Query&Text=cats&Cluster=True
```

IDOL Server clusters the results of the query above, and returns the ID of the cluster that each result belongs to in an `<autn:cluster>` field:

```
<autn:hit>
  <autn:reference>http://example.wikipedia.org/wiki/Big_Cat
Rescue</autn:reference>
  <autn:id>595644</autn:id>
  <autn:section>0</autn:section>
  <autn:weight>88.04</autn:weight>
  <autn:cluster>1</autn:cluster>
  <autn:clustertitle>Big Cat Rescue, Easy Street</autn:clustertitle>
  <autn:links>CAT</autn:links>
  <autn:database>Wikipedia</autn:database>
  <autn:title>Big Cat Rescue</autn:title>
</autn:hit>
<autn:hit>
  <autn:reference>http://example.wikipedia.org/wiki/The_Cat_in_the
Hat</autn:reference>
  <autn:id>229422</autn:id>
  <autn:section>1</autn:section>
  <autn:weight>86.93</autn:weight>
  <autn:cluster>2</autn:cluster>
  <autn:clustertitle>Allan, ballad, Hat, Krinklebine</autn:clustertitle>
  <autn:links>CAT</autn:links>
  <autn:database>Wikipedia</autn:database>
  <autn:title>The Cat in the Hat</autn:title>
</autn:hit>
<autn:hit>
  <autn:reference>http://example.wikipedia.org/wiki/The_Cat_in_the
Hat</autn:reference>
  <autn:id>229421</autn:id>
  <autn:section>0</autn:section>
  <autn:weight>86.93</autn:weight>
  <autn:cluster>3</autn:cluster>
  <autn:clustertitle>sight vocabulary, Cat, Hat, Seuss</autn:clustertitle>
  <autn:links>CAT</autn:links>
  <autn:database>Wikipedia</autn:database>
  <autn:title>The Cat in the Hat</autn:title>
</autn:hit>
<autn:hit>
```

```
<autn:reference>http://example.wikipedia.org/wiki/Cat</autn:reference>  
<autn:id>56711</autn:id>  
<autn:section>18</autn:section>  
<autn:weight>86.93</autn:weight>  
<autn:cluster>4</autn:cluster>  
<autn:clustertitle>breeds Polydactyl cat, Halloween, List,  
pets</autn:clustertitle>  
<autn:links>CAT</autn:links>  
<autn:database>Wikipedia</autn:database>  
<autn:title>Cat</autn:title>  
</autn:hit>
```

Generate Hyperlinks

When IDOL Server returns results, it automatically generates hyperlinks in real time that connect to contextually similar content.

About Hyperlinks

You can use hyperlinks in results to recommend related articles, documents, affinity products or services, or media content that relates to textual content.

Because IDOL Server inserts links automatically at the time that it retrieves the document, they can include references to documents and articles written long before. Similarly, hyperlinks from archived material can link to the latest news or material on that subject.

Here are some example applications of hyperlinks:

- **New Media.** When a user is viewing an article on a news media Internet site, IDOL Server can dynamically link to contextually similar content and recommend related articles in real time.
- **Corporate.** Within a corporate environment, while an employee is reading or writing a document, IDOL Server can suggest contextually similar documents from various sources through dynamically created hyperlinks to documents, multimedia content, and related emails.
- **Legal.** Hyperlinking facilitates the ability to suggest contextually relevant legal content pertinent to the legal issues being researched, reducing the time taken to navigate to the right information and identify previous precedents.
- **CRM.** When a customer service representative attends a customer enquiry, IDOL Server presents answers to the frequently asked questions and related emails in the form of dynamic hyperlinks. This process raises the level of customer service and shortens cycle times.

Implement Hyperlinks

If you connect IDOL Server to an HPE interface application, the interface generates hyperlinks automatically, for example, when it returns query results or when a user refines a query.

If you connect IDOL Server to a third-party interface application, you can generate hyperlinks automatically by running a Suggest action when you return query results. Refer to the ACI API documentation for details on the functions that you require for this.

Provide Spell Correction

When you perform a query, you can set `Spellcheck` to `True` in your query string to instruct IDOL Server to check the spelling of the query text. It suggests corrections for the terms that it detects to be misspelled.

How Spell Correction Works

To enable spell checking, set the parameters `SpellCheckMaxCheckTerms`, `SpellCheckIncorrectMaxDocOccs`, and `UnstemmedMinDocOccs` in the `[Server]` section of the configuration file before you index content. When you perform a query that includes `Spellcheck=True`, IDOL Server uses these settings in the spell checking process, as shown below:

1. IDOL Server determines if the query is eligible for spell checking.
IDOL Server checks how many terms the query text contains (it ignores stop words, proper-name terms and hyphenated terms). If the number does not exceed the specified `SpellCheckMaxCheckTerms`, the query is eligible for spell checking.
2. IDOL Server determines which terms are misspelled.
IDOL Server checks how many times each query term occurs in its data index. If a term occurs fewer times than the specified `SpellCheckIncorrectMaxDocOccs`, IDOL Server assumes that the term is misspelled.
3. IDOL Server finds correct spellings and suggests them.
IDOL Server uses a proprietary term-distancing algorithm to find terms in its data index that are closest to the misspelled terms. It then checks how many times these terms occur. If a term occurs at least the specified number of `UnstemmedMinDocOccs` times, it uses it as a spell check suggestion.
IDOL Server returns the corrected terms as a comma-separated list in an `<autn:spelling>` field. It also returns a corrected version of the query text in an `<autn:spellingquery>` field.
4. When you shut down IDOL Server, it creates a spelling correction file.
The spelling correction file stores the corrections that you make. You can add further corrections to the file or amend existing corrections.

Related Topics

- ["Spell Correction File" below](#)

Spell Correction File

If IDOL Server has performed spell checking, it generates a `prx.db` spelling correction file in its `IDOL > content > main` directory when it is shut down. The spelling correction file stores the corrections that you make.

For example:

```
http://IDOLhost:port/action=Query&Text=san+fransisco&Spellcheck=True
```

If you perform this action, IDOL Server returns the obvious correction and a corrected version of the query text.

After IDOL Server stops, it creates the `prx.db` spelling correction file that contains the corrections it has made in a human-readable form. For example:

```
<?xml version="1.0" encoding="UTF-8" ?><PROXIMALS>  
<PROXIMAL ORIG="FRANSISCO" CORRECT="FRANCISCO" />  
</PROXIMALS>
```

You can add further corrections to the file or amend any existing ones. You must specify the original and corrected spellings in upper case.

Your changes take effect when you start IDOL Server again.

You can exempt words from being spell checked by entering them and omitting the corrected attribute, for example:

```
<PROXIMAL ORIG="TELEFONE" />
```

Note: Any changes that you make to the `prx.db` file must be valid XML, otherwise they are ignored.

Note: You can use the `SpellCheckCacheMaxSize` setting in the `[Server]` section of the configuration file to increase the number of spelling corrections that the cache and the `prx.db` file can store.

Related Topics

- ["Edit the Spelling Correction Cache" on page 399](#)

Automatic Query Guidance

When it performs a query, IDOL Server can find the most salient terms and phrases in the query results. It can automatically cluster these terms and phrases and use the clustered phrases to provide a hierarchical set of queries that guide users to the result area that they are looking for.

About Automatic Query Guidance

Here is an example of the use of Automatic Query Guidance:



In this example, IDOL Server generates three clusters from documents that match the query **apollo**, and lists the top terms or phrases in each cluster as alternative queries. You can click any of these queries to perform them. In this example, you can display results that do not fall into any of the clusters by clicking **Other**.

To set up Automatic Query Guidance

1. Configure IDOL Server to enable Automatic Query Guidance.
2. Send an action with the `QuerySummary` parameter.

Enable Automatic Query Guidance

Use the following procedure to enable automatic query guidance.

To configure IDOL Server to generate query summaries

1. Open the IDOL Server configuration file in a text editor.
2. In the `[Server]` section, configure the following parameters:

<code>QuerySummaryAdvanced</code>	Set this parameter to <code>True</code> to enable the advanced algorithm that IDOL Server requires to provide query guidance. This algorithm clusters the best terms and phrases in the query results, and uses the phrases of the top clusters as query summaries. The default value is <code>False</code> .
<code>QuerySummaryLength</code>	Set this parameter to the number of cluster phrases to generate. To provide good query guidance phrases, enter a number that is large enough to allow IDOL Server to generate high-quality clusters. However, setting <code>QuerySummaryLength</code> too high might slow the

	query performance (a sensible maximum value might be 25–100). The default value is 10.
QuerySummaryTerms	Set this parameter to the number of terms that the algorithm for QuerySummaryLength can use to generate the query summaries (a sensible value might be 50–500). The default value is 0. In this case, IDOL Server uses the specified TermsPerDoc number, which in turn defaults to 50.

Tip: You can also set these parameters in the Query action so that you can test different values without reconfiguring IDOL Server, or tune QuerySummary functionality for different uses.

About the QuerySummary Parameter

To perform an action with the QuerySummary parameter, add the following parameters to your Query, Suggest, or SuggestOnText action:

QuerySummary=True	IDOL Server identifies the three most relevant terms and phrases of the clustered phrases, and returns them in an <autn:querysummary> field. It also returns a list of the specified QuerySummaryLength number of the best terms and phrases.
Combine=Simple	If multiple sections in a result document match the query, IDOL Server displays only the section with the highest conceptual similarity (rather than returning different sections of the same result document). This increases the relevance of the final phrases.
Print=NoResults	The purpose of the query is to provide query guidance, but not return results at the same time. Printing the results slows IDOL Server performance.
MaxResults=N	Set N to the number of results from which to generate the clustered terms and phrases. IDOL Server can produce high-quality clusters only if it has enough result documents available. However, setting MaxResults too high can slow the query performance (a sensible maximum value might be 50–500).

For example:

```
action=Query&Text=Apollo&QuerySummary=True&Combine=Simple&Print=NoResults&MaxResults=100
```

In this example, if you have set QuerySummaryLength to 10 in the configuration file, IDOL Server clusters the results that the query produces and lists the top 10 relevant terms or phrases that the results contain:

```
<autn:element pdocs="52" pocc="108" cluster="0" docs="56">Apollo
program</autn:element>
<autn:element pdocs="29" pocc="83" cluster="0" docs="34">Neil
Armstrong</autn:element>
<autn:element pdocs="22" pocc="43" cluster="0" docs="27">command
module</autn:element>
```

```
<autn:element pdocs="18" poccs="61" cluster="0" docs="54">Lunar  
Module</autn:element>  
<autn:element pdocs="13" poccs="13" cluster="1" docs="13">Greek  
mythology</autn:element>  
<autn:element pdocs="15" poccs="22" cluster="2" docs="35">Royal Navy</autn:element>  
<autn:element pdocs="22" poccs="28" cluster="2" docs="26">HMS Apollo</autn:element>  
<autn:element pdocs="14" poccs="21" cluster="0" docs="14">moon  
landings</autn:element>  
<autn:element pdocs="8" poccs="8" cluster="2" docs="9">Leander-class  
frigate</autn:element>  
<autn:element pdocs="12" poccs="14" cluster="1" docs="15">Greek god  
Apollo</autn:element>
```

The information for the top phrases and terms in the clusters that IDOL Server has generated consists of the following elements:

pdocs	The number of documents in which the entire phrase appears.
poccs	The total number of times that the phrase appears in all documents.
cluster	The number of the cluster that the phrase falls into. A negative number indicates that the associated documents do not fall into any of the generated clusters.
docs	The number of documents in which all the terms appear.

For example:

```
<autn:element pdocs="52" poccs="108" cluster="0" docs="56">Apollo  
program</autn:element>
```

In this example, the phrase “Apollo program” occurs in 52 documents out of the 100 results. The phrase appears 108 times in total in these documents. The phrase falls into cluster 0.

The terms “Apollo” and “program” appear in 56 documents.

IDOL Server also returns the best phrases or terms of the top three clusters in a comma-separated list in the <autn:querysummary> field:

```
<autn:querysummary>Apollo program, Greek mythology, Royal Navy</autn:querysummary>
```

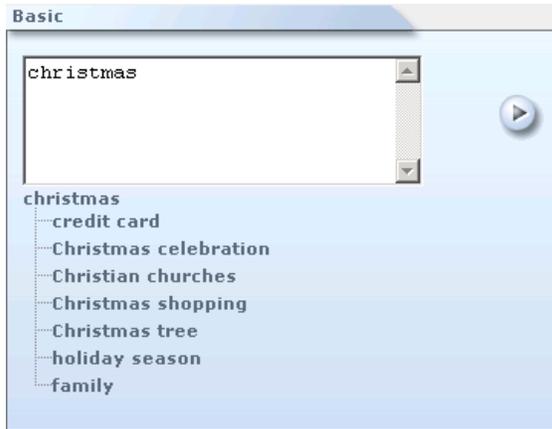
You can use each of these top cluster phrases as an alternative query, as well as the less relevant terms and phrases in each cluster.

Generate Query Summaries (Dynamic Thesaurus)

When it performs a query, IDOL Server can automatically generate query summaries that contain the most salient terms and phrases of the query result documents. It can use these terms and phrases to suggest alternative queries to users, allowing them to further refine queries and quickly produce a variety of relevant result sets.

About Query Summaries

Here is an example of the use of Query Summaries:



In this example, the IDOL Server has identified the seven phrases that are most relevant to the query **christmas** that result documents contain, and listed them as query summaries. Users can click any of these phrases to perform a new query.

To automatically generate query summaries

1. Configure IDOL Server to generate query summaries.
2. Perform an action with the `QuerySummary` parameter.

Configure IDOL Server to Generate Query Summaries

Use the following procedure to enable query summary generation.

To configure IDOL Server to generate query summaries

1. Open the IDOL Server configuration file in a text editor.
2. In the `[Server]` section, set the following parameters:

<code>QuerySummaryLength</code>	<p>Set this parameter to the number of query summaries to return when you perform an action that includes the <code>QuerySummary</code> parameter.</p> <p>For example, if you set <code>QuerySummaryLength</code> to 5, IDOL Server identifies the five top phrases or terms that the result documents contain, and returns them in a comma-separated list. If IDOL Server cannot find your requested number of relevant phrases or terms, it returns fewer.</p> <p>The default value is 10</p>
<code>QuerySummaryAdvanced</code>	<p>Set this parameter to <code>True</code> to use an advanced algorithm to generate the specified <code>QuerySummaryLength</code> number of query summaries. This algorithm clusters the results and uses the phrases of the top clusters as query summaries. This process produces better query</p>

	<p>summaries, but is more system intensive.</p> <p>Set this parameter to <code>False</code> if you want query summaries without first clustering the results.</p> <p>The default value is <code>False</code>.</p>
QuerySummaryTerms	<p>If you set <code>QuerySummaryAdvanced</code> to <code>True</code>, <code>QuerySummaryTerms</code> allows you to specify the number of terms that the algorithm uses to generate the query summaries (a sensible value might be 50–500). If you set <code>QuerySummaryTerms</code> to <code>0</code>, IDOL Server uses the specified value of <code>TermsPerDoc</code>, which in turn defaults to 50.</p> <p>The default value is <code>0</code>.</p>

Tip: You can also set these parameters in the `Query` action so that you can test different values without reconfiguring IDOL Server, or tune `QuerySummary` functionality for different uses.

Perform an Action with the QuerySummary Parameter

To perform an action with the `QuerySummary` parameter, you can add the following parameters to your `Query`, `Suggest`, or `SuggestOnText` action:

QuerySummary=True	IDOL Server identifies the most relevant terms and phrases that the query result documents contain, and returns them in an <code><autn:querysummary></code> field. You can use each term or phrase in this field as an alternative query suggestion.
Combine=Simple	If multiple sections in a result document match the query, IDOL Server displays only the section with the highest conceptual similarity (rather than returning different sections of the same result document). This option increases the relevance of the final phrases.
Print=NoResults	The purpose of the query is to generate query summaries but not return results at the same time. Printing the results slows IDOL Server performance.
MaxResults=N	Enter the number of results from which you want to generate the query summaries. IDOL Server can produce good query summaries only if it has enough result documents available from which it can identify sufficiently different phrases. However, setting <code>MaxResults</code> too high might slow the query performance (a sensible value might be 50–500).

For example:

```
action=Query&Text=Date&QuerySummary=True&Combine=Simple&Print=NoResults&MaxResults=100
```

In this example, if you have set `QuerySummaryLength` to 5 in the configuration file, IDOL Server finds the top five phrases or terms in results that match the specified `Text` value, and returns them in a comma-separated list in the `<autn:querysummary>` field:

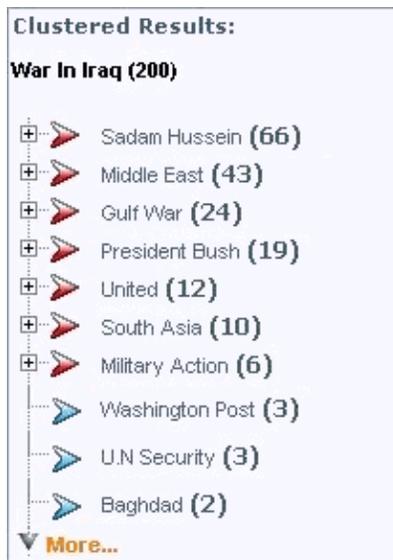
```
<autn:querysummary>Date conversion, rendezvous, stuffed dates, appointment, go steady</autn:querysummary>
```

It can use each of these terms and phrases as an alternative query.

Generate Dynamic Clusters

Every time it runs a query, IDOL Server can automatically cluster the results that are available for this query, and then in turn cluster the first set of clusters further to produce subclusters. This process allows you to generate a hierarchy of dynamic clusters that allows users to navigate quickly to their area of interest.

For example:



In this example, IDOL Server has generated 10 dynamic clusters from documents that match the query **War In Iraq**. A plus sign (+) and a red arrow next to a cluster in this interface indicates that you can expand the cluster to display subclusters. A blue arrow indicates clusters that cannot cluster further.

IDOL Server lists each cluster with the best phrase it contains. The number in brackets indicates how many documents the cluster contains. The user can click the best phrase for a cluster to run a query with this phrase and automatically cluster the results that this query produces.

Dynamic clustering can be faster than clustering in the background, through `ClusterCluster` and `ClusterServe2DMap`. However, using `ClusterCluster` and `ClusterServe2DMap` is more powerful and accurate. Also, dynamic clustering is more suited to finding trends in smaller data sets. `ClusterServe2DMap` is better for clustering an entire database.

Related Topics

- ["Cluster Information" on page 179](#)

To generate dynamic clusters

1. Configure IDOL Server to enable dynamic clustering.
2. Perform an action with the `QuerySummary` parameter.

Configure IDOL Server to Enable Dynamic Clusters

Use the following procedure to enable dynamic clustering in IDOL Server.

To configure IDOL Server to enable dynamic clustering

1. Open the IDOL Server configuration file in a text editor.
2. In the [Server] section, set the following parameters:

QuerySummaryAdvanced	Set this parameter to <code>True</code> to enable the advanced algorithm that provides dynamic clustering. This algorithm clusters the best terms and phrases from the query results, and lists them. The default value is <code>False</code> .
QuerySummaryLength	Set this parameter to the number of cluster phrases to generate. Setting <code>QuerySummaryLength</code> too high can slow the query performance (a sensible value might be 25–250). The default value is <code>10</code> .
QuerySummaryIDs	Set this parameter to <code>True</code> to return the document IDs of the parent documents that phrases come from. The IDs return in the list that IDOL Server generates when you perform an action with the <code>QuerySummary</code> parameter. You can use the IDs to display the number of documents that each dynamic cluster contains. The default value is <code>False</code> .
QuerySummaryTerms	Specify the number of terms that the algorithm for <code>QuerySummaryLength</code> can use to generate the query summaries (a sensible value might be 50–500). If you set this parameter to <code>0</code> , IDOL Server uses the specified <code>TermsPerDoc</code> number, which in turn defaults to <code>50</code> . The default value is <code>0</code> .

Tip: You can also set these parameters in the `Query` action so that you can test different values without reconfiguring IDOL Server, or tune `QuerySummary` functionality for different uses.

Perform an Action with the QuerySummary Parameter

To perform an action with the `QuerySummary` parameter, add the following parameters to your `Query`, `Suggest`, or `SuggestOnText` action:

QuerySummary=True	IDOL Server identifies the three most relevant terms and phrases of the clustered phrases, and returns them in an <code><autn:querysummary></code> field. It also returns a list of the specified <code>QuerySummaryLength</code> number of the best terms and phrases.
Combine=Simple	If multiple sections in a result document match the query, IDOL Server

	displays only the section with the highest conceptual similarity (rather than returning different sections of the same result document). This process increases the relevance of the final phrases.
Print=NoResults	The purpose of the query is to provide dynamic clustering but not return results at the same time. Printing the results slows IDOL Server performance.
MaxResults=N	Set this parameter to the number of results from which you want to generate the terms and phrases. Setting MaxResults too high can slow the query performance (a sensible value might be 50–500).

For example:

```
action=Query&Text=War In
Iraq&QuerySummary=True&Combine=Simple&Print=NoResults&MaxResults=100
```

Display Cluster Information

In the previous example, if you have set QuerySummaryLength to 10 in the configuration file, IDOL Server lists the top 10 relevant terms or phrases that the results contain:

```
<autn:element pdocs="63" poccs="132" cluster="0" docs="66"
ids="398867,388941,...">Saddam Hussein</autn:element>
<autn:element pdocs="43" poccs="78" cluster="0" docs="43"
ids="398867,343399,...">Middle East</autn:element>
<autn:element pdocs="22" poccs="41" cluster="0" docs="24"
ids="388941,338798,...">Gulf War</autn:element>
<autn:element pdocs="17" poccs="32" cluster="0" docs="19"
ids="409508,398867,...">President Bush</autn:element>
<autn:element pdocs="12" poccs="17" cluster="1" docs="12"
ids="326892,326496,...">United</autn:element>
<autn:element pdocs="10" poccs="15" cluster="1" docs="10"
ids="388941,326892,...">South Asia</autn:element>
<autn:element pdocs="4" poccs="7" cluster="1" docs="6"
ids="428429,326497,...">Military Action</autn:element>
<autn:element pdocs="3" poccs="4" cluster="1" docs="3"
ids="326496,227758,...">Washington Post</autn:element>
<autn:element pdocs="2" poccs="4" cluster="1" docs="3" ids="324404,227756,...">U.N.
Security</autn:element>
<autn:element pdocs="2" poccs="3" cluster="1" docs="2"
ids="398567,343199,...">Baghdad</autn:element>
```

The information for the top phrases and terms that IDOL Server has generated consists of the following elements:

pdocs	The number of documents in which the entire phrase appears.
poccs	The total number of times that the phrase appears in documents.
cluster	The number of the Automatic Query Guidance cluster that the phrase falls into. A negative

	number indicates that the associated documents do not fall into any of the generated clusters.
docs	The number of documents in which all the terms appear.
ids	The IDs of the documents in which the phrases appear. You can use the IDs in a front end to display how many documents a dynamic cluster contains.

Related Topics

- ["Automatic Query Guidance " on page 304](#)
- ["Display the Number of Documents a Dynamic Cluster Contains" below](#)

For example:

```
<autn:element pdocs="63" poccs="132" cluster="0" docs="66"
ids="398867,388941,...">Saddam Hussein</autn:element>
```

In this example, the phrase "Saddam Hussein" occurs in 63 documents out of the 100 results. The phrase appears 132 times in total. The phrase falls into Automatic Query Guidance cluster 0. A total of 66 documents contain the terms "Saddam" and "Hussein".

IDOL Server also returns the best phrases or terms of the top three Automatic Query Guidance clusters in a comma-separated list in the <autn:querysummary> field. However, dynamic clustering does not use these terms.

```
<autn:querysummary>Saddam Hussein, Middle East, Gulf War</autn:querysummary>
```

Display the Number of Documents a Dynamic Cluster Contains

You can use the document IDs that dynamic clustering returns to display how many documents a dynamic cluster contains in a front-end application.

For example, assume that a query for Apollo returns 10 result documents with the document IDs listed in the following table.

Dynamic cluster name	IDs of documents in this dynamic cluster									
Neil Armstrong		2			5		7		9	10
command module					5		7		9	
moon landing		2			5			8	9	
Greek mythology	1		3	4						
Zeus	1		3							

When IDOL Server displays the total number of documents that a dynamic cluster contains, it counts documents for a dynamic cluster only if they were not already counted for a previous dynamic cluster. The clusters have the following totals:

- Neil Armstrong (5)
- Greek mythology (3)

- moon landing (1)
- other (1)

IDOL Server determines the totals in the following way:

- The Neil Armstrong cluster contains five documents.
- The command module cluster contains only documents that were counted for the previous cluster, so it is not listed.
- The moon landing cluster contains one document because the documents with the IDs 2, 5, and 9 were counted for previous clusters.
- The Greek myth cluster contains three documents.
- The Zeus cluster contains only documents that were counted for the previous cluster, so it is not listed.
- One document that the query returned did not fit into any of the clusters. IDOL Server lists this document in the other cluster.

Create a Cluster Map

You can use the `ClusterMapFromResults` action to create a map of the clusters that you generate from your query.

You provide XML-formatted cluster information to `ClusterMapFromResults` as the value of the `ImportXML` parameter, and the action returns a 2-D cluster map as binary image data. The file is by default in JPEG format, but (as in the case of the `ClusterServe2DMap` action) you can specify another format by setting the `PictureFormat` configuration parameter in the `[Cluster]` section of the IDOL Server configuration file.

`ClusterMapFromResults` also creates and stores a new set of cluster results that includes the map coordinates of the clusters. If your application uses these cluster results to support mouse-over effects when displaying the cluster map, you can retrieve the results by using the `ClusterResults` action. In this case, use the `TargetJobName` from the `ClusterMapFromResults` action as the `SourceJobName` in the `ClusterResults` action.

When you use the `ImportXML` parameter to send data to `ClusterMapFromResults`, you must use the following format:

```
<?xml version='1.0' encoding='UTF-8' ?>
<autn:clusters xmlns:autn='http://schemas.autonomy.com/aci/'>
  <autn:numclusters>5</autn:numclusters>
  <autn:cluster>
    <autn:title>MyClusterTitle</autn:title>
    <autn:score>5.63</autn:score>
    <autn:term>CAT~[85]</autn:term>
    <autn:term>DOG~[76]</autn:term>
    <autn:term>RABBIT~[65]</autn:term>
    <autn:term>HAMST~[64]</autn:term>
    <!-- more terms -->
    <autn:docs>
      <autn:numdocs>3</autn:numdocs>
      <autn:doc>
```

```
        <autn:title>MyDocTitle</autn:title>  
        <autn:ref>http://foo.com</autn:ref>  
        <autn:score>5</autn:score>  
    <autn:doc>  
    <!-- more docs-->  
    </autn:docs>  
</autn:cluster>  
<!-- more clusters-->  
</autn:clusters>
```

This XML format is similar to the format of the XML results of a Query action. If you pass this data to the ClusterMapFromResults action in the ImportXML parameter, you must first percent-encode it.

Note: As an alternative to providing XML data to ClusterMapFromResults, you can provide it with a job name from a previous ClusterCluster action.

Related Topics

- ["Generate a Cluster Map after You Cluster" on page 183](#)

Chapter 16: Manipulate Result Relevance

This section describes the methods that you can use to boost the relevance of a result in a results list.

- [Boost Relevance](#) 317
- [Use a Field Process to Boost Relevance](#) 317
- [Use the BIAS Field Specifier to Boost Relevance](#) 319
- [Use Multipliers to Boost Relevance](#) 323
- [Use the AutnRankType Field to Boost Relevance](#) 324

Boost Relevance

You can boost the percentage relevance that IDOL Server gives to query results by using the following methods:

- **Use a field process.** You can set up a field process in the IDOL Server configuration file to boost the percentage relevance of query results according to the number of times terms in specified fields match the query terms.
- **Use BIAS.** You can use the BIAS field specifier at query time to boost the percentage relevance of query results according to the numerical proximity of a specified field to a particular value.
- **Use multipliers.** You can multiply the weight of individual query terms to boost the relevance of results that match these terms accordingly.
- **Use the AutnRankType field.** You can create a field in documents that indicates their importance and instruct IDOL Server to take the value of this field into account when it calculates the relevance that the document is given if it returns as a result.

Use a Field Process to Boost Relevance

You can set up a field process that identifies specific fields in documents and manipulates the weight of terms in these fields if they match the query terms.

For example, use the following procedure to boost the weight of results that contain query terms in their DRETITLE and SUMMARIES fields.

1. For each field whose content you want to use to determine whether to boost result weights, list a process in the [FieldProcessing] section of the configuration file that indexes the field and manipulates its term weights. You can use one field process to boost terms in several fields by the same factor.

For example:

```
[FieldProcessing]
0=IndexAndWeightHigher1
1=IndexAndWeightHigher2
```

2. Create a section for each of the processes that you listed, in which you create a property for the

process (you define the property later by setting one or more applicable configuration parameters). Identify the fields that you want to associate with the processes.

Note: The properties that you create must not have the same name as the processes.

For example:

```
[IndexAndWeightHigher1]
Property=IndexHigherWeight1
PropertyFieldCSVs=*/DRETITLE
```

```
[IndexAndWeightHigher2]
Property=IndexHigherWeight2
PropertyFieldCSVs=*/SUMMARY
```

3. Create a section for each of the properties and specify configuration settings for each. The `Index` parameter ensures that IDOL Server indexes the fields that you associate with the field process. The `Weight` parameter determines the factor by which to boost terms in the associated `PropertyFieldCSVs` fields if they match query terms.

For example:

```
[IndexHigherWeight1]
Index=True
Weight=4
```

```
[IndexHigherWeight2]
Index=True
Weight=2
```

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

In future queries, IDOL Server boosts the relevance of the result to the query according to how many times the terms in the result `SUMMARY` and `DRETITLE` fields match the query terms.

For example, the following query boosts results whose `SUMMARY` and `DRETITLE` field matches the query terms *cat* and *dog*:

```
http://IDOLhost:port/action=Query&Text=cat and dog
```

This means that these results return in the following order:

- **Result 1:**

Title = Cats & Dogs

Summary = **Cats** and **dogs** duke it out in this live action feature about a professor on the brink of discovering a cure for **dog** allergies. The **dogs** assign an agent to protect the professor and his family from a feline invasion.

Content = Unbeknownst to humans, **dogs** have fought for thousands of years to keep mankind from falling under the rule of **cats**. Using combinations of live animals, animatronic puppets, and digital wizardry, this film has just enough imagination to match its effects, climaxing with a feline global-domination scheme involving mice sprayed with chemicals that will make all humans allergic to their canine friends.

- **Result 2:**

Title = Garfield

Summary = Garfield comes to life in an all new live action major motion picture.

Content = Garfield is a fat **cat**. A **cat** that eats lots of Lasagne. A **cat** that is lazy and sleeps as much as possible. Nevertheless, Garfield is a clever **cat**, always able to outwit his owner, Jon and the neighbor's **dog**, Odie. Garfield is a cool and sarcastic **cat** but he is also a **cat** with a heart as is shown when he comes to the rescue of Odie the **dog**, in the movie that is coming out this year. The hapless pup disappears and is kidnapped by a nasty **dog** trainer, and Garfield feels responsible. Pulling himself away from the TV, Garfield springs into action. Maybe it's friendship for **cat** and **dog** after all.

- **Result 3:**

Title = Tom and Jerry: The movie

Summary = The celebrated **cat** and mouse team meets a young runaway who desperately needs their help to find her missing father. Along the way they run into her evil Aunt who tosses them into a pet prison. Bonding together, Tom & Jerry outwit the Aunt and mastermind a great escape to set off on the wildest adventures of their **cat** and mouse careers.

Content = The popular animated duo team up again to appear this time on the big screen. Homeless, the 'toons end up helping out a young girl who stays with a nasty auntie while she is separated from her father. Will the young Robyn be reunited with her loving father? Will the odd pair make it on the streets? Will they find a home? Those are some of the burning questions that may plague the minds of young viewers of this fun adventure.

Without the boosting to the weight of the SUMMARY and DRETITLE field, Result 2 is the top result, with Result 1 following in second place.

Result 3 does not rank higher than Result 2 in either case. Although its weight is slightly boosted because its SUMMARY field contains one of the query terms, this boost is not sufficient to outrank Result 2.

Use the BIAS Field Specifier to Boost Relevance

The BIAS field specifier allows you to bias the score of results at query time according to the numerical proximity of the specified field to a particular value. It ignores initial dollar (\$), pound sign (£), or hyphen (-) characters in the field names.

Specify BIAS in the format:

BIAS{*optimum*,*range*,*percentage*}

where:

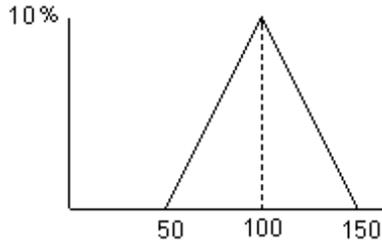
<i>optimum</i>	is the value that the field must contain to increase or decrease the result weight by the maximum <i>percentage</i> .
<i>range</i>	is a positive number that determines the range of the <i>optimum</i> . If the specified field contains a value that is in the range of (<i>optimum</i> - <i>range</i>) to (<i>optimum</i> + <i>range</i>), the result weight increases or decreases according to the specified <i>percentage</i> .
<i>percentage</i>	is a percentage in the range -100 to 100. If the value of the specified field is in the

specified range, the score of the result increases or decreases according to how close the value is to the specified <i>optimum</i> .

For example:

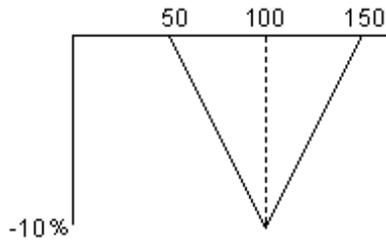
- `http://IDOLhost:port/action=Query&FieldText=BIAS{100,50,10}:/PRICE`

A document whose PRICE field value is within the range 50 either side of 100 has its weight increased on a linear scale from 10 percent if the price is 100, to 0 percent if the price is 50 or 150:



- `http://IDOLhost:port/action=Query&FieldText=BIAS{100,50,-10}:/PRICE`

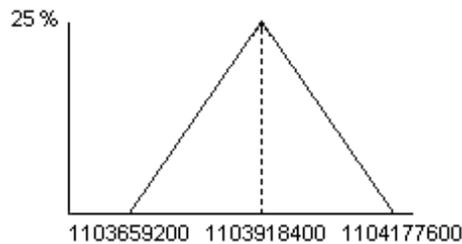
A document whose PRICE field value is within the range 50 either side of 100 has its weight decreased on a linear scale from -10 percent if the price is 100, to 0 percent if the price is 50 or 150:



- You can also use the BIAS field specifier to bias the score of results according to the numerical proximity in their `autn_date` metafield to a particular value. For example:

`FieldText=BIAS{1103918400,259200,25}:autn_date`

A document whose `autn_date` field value is within the range 259200 either side of 1103918400 has its weight increased on a linear scale from 25 percent if the price is 1103918400, to 0 percent, if the date is 1103659200 or 1104177600.



Related Topics

- ["Metadata Fields" on page 96](#)

BIASDATE

The BIASDATE field operator works in a similar way to the BIAS field operator. It allows you to bias the score of results at query time according to the proximity of the specified field to a particular date.

Specify BIASDATE in the format:

`BIASDATE{optimumDate, range, percentage}`

where:

<i>optimumDate</i>	is the date that the field must contain to increase or decrease the result weight by the maximum <i>percentage</i> . See "Date formats " on page 230 for a list of the available date formats.
<i>range</i>	is a positive value that determines the range in seconds of the specified optimum. If the specified field contains a value that is in the range (<i>optimum</i> – <i>range</i>) to (<i>optimum</i> + <i>range</i>), the result weight is increased or decreased according to the specified <i>percentage</i> .
<i>percentage</i>	is a percentage in the range –100 to 100. If the value of the specified field is within the specified range, the score of the result increases or decreases according to how close the value is to the specified optimum. When you set <code>Absweight</code> to <code>True</code> , this is the absolute value by which to boost the weight, and it is then not limited by +/- 100.

For example:

- `FieldText=BIASDATE{1/12/2008,864000,10}:DATE`

A document whose DATE field value is within a range of 10 days (864000 seconds) either side of 1/12/2008 has its weight increased on a linear scale from 10 percent if the date is 1/12/2008, to 0 percent if the date is 20/11/2008 or 11/12/2008.

- `FieldText=BIASDATE{1/12/2008,86400,-10}:DATE`

A document whose DATE field value is within a range of 10 days (864000 seconds) either side of 1/12/2008 has its weight decreased on a linear scale from –10 percent if the date is 1/12/2008, to 0 percent if the date is 20/11/2008 or 11/12/2008.

BIASDISTCARTESIAN

The BIASDISTCARTESIAN field operator works in a similar way to the BIAS field operator. It allows you to bias the score of results at query time according to the proximity of the specified field to a particular location, specified using Cartesian coordinates.

Specify BIASDISTCARTESIAN in the format:

`FieldText=BIASDISTCARTESIAN{coordX, coordY, range, percentage}:X:Y`

where:

<i>coordX</i>	is the X coordinate.
---------------	----------------------

<i>coordY</i>	is the Y coordinate.
<i>range</i>	is the distance in kilometers from the coordinates. If the fields contain the coordinates of a location that is within this distance of the coordinates, the results weight increases or decreases according to the specified percentage.
<i>percentage</i>	is a percentage in the range –100 to 100. If the value of the field is within the specified range, the score of the result increases or decreases according to how close the value is to the optimum.
<i>X</i>	is the document field that contains the X coordinate.
<i>Y</i>	is the document field that contains the Y coordinate.

You must specify two fields in the order X:Y. The fields must be of `NumericType`.

For example:

```
FieldText=BIASDISTCARTESIAN{10,11,5,7}:X:Y
```

In this example, all documents whose (X/Y) position is within a distance of 5 units of the point (10,11) are given a relevance boost. The maximum boost of 7 percent is given to documents at the specified point. The boost decreases linearly down to zero boost at 5 units.

BIASDISTSPHERICAL

The `BIASDISTSPHERICAL` field operator works in a similar way to the `BIAS` field operator. It allows you to bias the score of results at query time according to the proximity of the specified field to a particular location, specified using a latitude and longitude value.

Specify `BIASDISTSPHERICAL` in the format:

```
FieldText=BIASDISTSPHERICAL{Lat,Long,range,percentage}:LATFIELD:LONGFIELD
```

where:

<i>Lat</i>	is the latitude. Specify latitude positions south of the equator as negative.
<i>Long</i>	is the longitude. Specify longitude positions west of the Greenwich Meridian as negative.
<i>range</i>	is the distance in kilometers from the specified coordinates. If the fields contain the coordinates of a location that is within this distance of the specified coordinates, the results weight increases or decreases according to the specified percentage.
<i>percentage</i>	is a percentage in the range –100 to 100. If the value of the field is within the specified range, the score of the result increases or decreases according to how close the value is to the specified optimum.
<i>LATFIELD</i>	is the document field that contains the latitude.
<i>LONGFIELD</i>	is the document field that contains the longitude.

You must specify two fields in the order latitude:longitude. The fields must be of `NumericType`.

For example:

```
FieldText=BIASDISTSPHERICAL{52.2,0.1,100,7}:LAT:LONG
```

In this example, all documents within 100 kilometers of the point (lat,long)=(52.2,0.1) are given a relevance boost. IDOL Server gives the maximum boost of 7 percent to documents at the specified point. The boost decreases linearly down to zero boost at 100 kilometers.

Use Multipliers to Boost Relevance

You can add multipliers to individual query terms to boost the relevance of results that match these terms accordingly.

To apply a multiplier to a query term, use this format:

```
queryTerm[*N]
```

where:

<i>queryTerm</i>	is the query term whose weight to multiply.
<i>N</i>	is the factor to multiply the specified query term weight by. This weight can be any positive number.

For example:

```
http://IDOLhost:port/action=Query&Text=bread[*2.5]+brown+loaf
```

This action multiplies the weight of the query term *bread* by 2.5. The weight of the query terms *brown* and *loaf* does not change.

When results return for the query, the relevance of documents that contain the term *bread* is boosted relative to those that do not.

```
http://IDOLhost:port/action=Query&Text=SOUNDEX(bred)+bred[*4]
```

In this example, a supermarket wants to ensure that an online search for *bread* returns appropriate results. The supermarket has found that customers tend to misspell *bread* as *bred*. If a customer queries for *bread*, appropriate results return as usual. If a customer queries for *bred*, the term is submitted twice; once as a Soundex keyword search, and once with a multiplier. This ensures that if results exist that match *bred* (for example, a new CD by a band called *bred*), they return with a higher relevance than results that match *bred* phonetically.

Similarly, you can use multipliers to reduce the influence of individual query terms. For example:

```
http://IDOLhost:port/action=Query&Text=cat[*0.5]+dog
```

In this example, the weight of the query term *cat* is halved by multiplying it by 0.5. The weight of the query term *dog* does not change.

When you return results for the query, the relevance of documents that contain the term *cat* is reduced relative to those that do not.

Related Topics

- ["Soundex Keyword Search" on page 263](#)

Use the AutnRankType Field to Boost Relevance

Each document that is stored in IDOL Server is given an `AutnRank` value which indicates the importance of the document. By default IDOL Server ignores this value and gives all documents an `AutnRank` value of 0.

You can boost the relevance of documents according to how important they are to you by creating a field in the documents that indicates their importance, and instructing IDOL Server to read the `AutnRank` value from this field. If two documents match a query equally well, the one that has the higher value in this field returns with a higher relevance rating.

To boost results according to their `AutnRank` value

1. In the `[Server]` section of the configuration file, set `AutnRank` to `True`.
This instructs IDOL Server to take the `AutnRank` value of a document into account when it calculates the relevance that the document is given if it returns as a result.
2. In the `[FieldProcessing]` section, set up a ranking process. This process allows IDOL Server to identify which field in a document contains its `AutnRank` value.

For example:

```
[FieldProcessing]
0=SetAutnRankField
```

3. Create a section for the ranking process that you listed, in which you create a property for the process. Identify the field that you want to associate with the process (when identifying the field, use the format `/FieldName` to match root-level fields, `*/FieldName` to match all fields except root-level, or `/Path/FieldName` to match fields that the specified path points to).

Note: The property that you create must not have the same name as the process.

For example:

```
[SetAutnRankField]
Property=ReadAutnRank
PropertyFieldCSVs=*/AUTNRANK
```

In this example IDOL Server reads the `AutnRank` value of the document from its `AUTNRANK` field.

4. Create a section for the property in which you set the `AutnRankType` parameter to `True`.

```
[ReadAutnRank]
AutnRankType=True
```

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

Chapter 17: Manipulate the Results Set

If you are running a Query, Suggest, or SuggestOnText action, you can use several parameters to manipulate the generation of results. You can also store the state of your results from any of those actions, for later retrieval and processing.

- [Combine Parameter](#) 325
- [FieldCheck Parameter](#) 328
- [Predict Parameter](#) 328
- [Store and Retrieve the Result State](#) 328

Combine Parameter

The `Combine` action parameter combines results that:

- derive from the same document.
- contain the same content.
- have the same value in a specific `ReferenceType` field.

By default, IDOL Server displays the result with the highest relevance. However, you can use the `Sort` action parameter to set alternative sorting methods.

You can set `Combine` to one of the following options:

- `Simple`
- `FieldCheck`
- `ReferenceTypeFields`

Simple

This is the HPE recommended `Combine` option.

When IDOL Server indexes very long texts, by default it breaks them into sections, and indexes them as individual documents. Each document has its own ID, but they all have the same document reference. This process stabilizes indexing and ensures that the most relevant section of a text returns when you query IDOL Server (for example, rather than an entire book). However, if several sections match the query, each section returns as a result. A query can contain multiple results with the same document reference, which belong to the same text, for example, different pages of the same book. If you display each result with the `Print` action parameter set to `AllSections`, you see the same text every time.

You can prevent IDOL Server from returning different sections of the same source text by adding `Combine=Simple` to the query. IDOL Server displays only the section that has the highest conceptual similarity to the query (unless you add `Print=AllSections` to the query, in which case it displays the entire source text). If multiple sections have the same conceptual relevance, IDOL Server returns the one with the lowest section number.

For example:

```
http://IDOLhost:port/action=Query&Text=The Moonstone&Combine=Simple
```

In this example, if several results derive from the same source text, IDOL Server displays only the result that has the highest relevance to the query text.

FieldCheck

The `FieldCheck` option combines results based on the hash value of their `FieldCheckType` field. The `FieldCheckType` field holds a value that is frequently used to restrict results (for example, a field that stores category names). When IDOL Server indexes a `FieldCheckType` field, it stores it in a fast lookup table in memory, so that it can return quickly.

Related Topics

- ["FieldCheckType Fields" on page 88](#)

For example:

```
http://IDOLhost:port/action=Query&Text=The best thing to do in your spare  
time&Combine=FieldCheck
```

In this example, IDOL Server is configured to store the `Category` field as a `FieldCheckType` field.

If IDOL Server contains 50 documents that match the query text, of which eight contain a `Category` field with the value `Sport`, five contain a `Category` field with the value `Gardening`, and one contains a `Category` field with the value `Cooking`, the above query returns only three results:

- the most relevant of the documents whose `Category` contains the value `Sport`.
- the most relevant of the documents whose `Category` contains the value `Gardening`.
- the document whose `Category` contains the value `Cooking`.

Note: If you set `URLAnalysis` to `True` in the `[Server]` section of the IDOL Server configuration file, you cannot identify a field as a `FieldCheckType` field, because IDOL Server automatically uses the domain of the URL it finds in the document `ReferenceType` fields as the `FieldCheck` value.

ReferenceTypeFields

For this option, `ReferenceTypeFields` is a plus-, space-, or comma-separated list of `ReferenceType` fields. If a query produces several results that contain the same value in one or more of the specified `ReferenceType` fields, IDOL Server returns only the most relevant result. If several results have the same relevance, the result with the highest `DocID` returns (unless you enable a `Sort` option that overrides this).

For example:

```
http://IDOLhost:port/action=Query&Text=The Moonstone&Combine=DRETITLE
```

In this example, if several results contain the same value in the `DRETITLE` field, it displays only the result that has the highest relevance to the query text.

When you combine using a `ReferenceType` field, IDOL Server automatically uses all the fields that you list in the configuration file in the same `PropertyFieldCSVs` parameter as this field. To ensure that IDOL Server combines using only the specified field, you must set up an individual field process to identify this field as a `ReferenceType` field.

If you want to use multiple `ReferenceType` fields to combine, you might want to create a process that identifies all these fields as `ReferenceType`.

Related Topics

- ["Configure a Field Process" on page 76](#)

For example:

```
[SetupReferenceFields]
Property=ReferenceFields
PropertyFieldCSVs=*/DREREFERENCE,*/url
```

```
[CombineField1]
Property=ReferenceFields
PropertyFieldCSVs=*/DRETITLE
```

```
[CombineField2]
Property=ReferenceFields
PropertyFieldCSVs=*/CombineField
```

In this example, if you set `Combine` to `DREREFERENCE`, IDOL Server combines using the `DREREFERENCE` and `url` fields. If you set `Combine` to `DRETITLE`, it uses only the `DRETITLE` field to combine.

Multiple Options

You can combine the `Simple` and `FieldCheck` options, in which case you must specify `Simple` first. For example:

```
Combine=Simple+FieldCheck
```

If you set `Combine` to `ReferenceTypeFields`, you cannot combine the fields with another `Combine` option.

Exceptions

In some cases, you might not want to combine documents. If you set `Combine` to `FieldCheck` or `ReferenceTypeFields`, by default IDOL Server combines all documents that do not contain the field, and returns only one set.

To return these results without combining them, set the `CombineIgnoreMissingValue` configuration parameter to `True`. For example:

```
[Server]
CombineIgnoreMissingValue=True
```

FieldCheck Parameter

You can use the `FieldCheck` action parameter to return only documents whose `FieldCheckType` field matches the value you specify, for example a category name. You must identify a `FieldCheckType` field before you store content in IDOL Server.

If you set `UrlAnalysis` to `True` in the IDOL Server configuration file, enter a domain name for `FieldCheck` to restrict results to documents that were aggregated from this domain. For example:

```
http://IDOLhost:port/action=Query&Text=A fast sports car&FieldCheck=Red
```

In this example, IDOL Server is configured to store the `Color` field as a `FieldCheckType` field. The query returns only those results whose content matches the specified `Text` and whose `FieldCheckType` field has the value `Red`.

Predict Parameter

You can use the `Predict` parameter to instruct IDOL Server to use statistical sampling to estimate the total number of results that are available. You must also set `TotalResults` to `True` for your query action. Prediction can increase the query speed.

If you set `Predict` to `False`, IDOL Server counts all results and in addition prints the number of results for each database. For example:

```
http://IDOLhost:IDOLport/action=Query&Text=A fast sports  
car&TotalResults=True&Predict=True
```

In this example, if you have set `TotalResultsPredictionThreshold` to `100`, IDOL Server uses statistical sampling to estimate the total number of results that are available for the query. If fewer than 100 results are available for the query, IDOL Server returns the exact number of total results.

If you do not want IDOL Server to display the number of results for each database, you can set the `TotalResultsPrintDatabase` configuration parameter to `False`.

Store and Retrieve the Result State

Whenever you use the `Query`, `Suggest`, or `SuggestOnText` actions, you can save the result state (the `DocID` values of all returned results) in a *state token*. In subsequent actions, you can use this token to represent the documents from that result set.

A state token is an array of `DocID` values. Its name is of the form `ABCDEFGH-N`, where `ABCDEFGH` is a random alphanumeric string (generated at query time), and `N` is the number of document IDs that the token holds. You can also use document references, rather than document IDs, to create state tokens. In this case, use the `StoredStateField` parameter to specify the name of the field that contains the references.

When you have a state token, supply the token name to specify the entire set of documents (for example, `CWQ4FJ9LZSE5-6`). You can also use a (zero-based) array notation to specify a subset of the documents (for example, `CWQ4FJ9LZSE5-6[3]` to select the fourth document in the array).

Tokens in the array are in the same order as in the returned results.

IDOL Server stores tokens persistently, in the directory *installDir/agentstore/storedstate*.

Store the Result State

To create a state token from your query results, use the `StoreState` parameter when you issue the query:

```
http://localhost:20000/action=Query&Text=apple&StoreState=True
```

To use document references, rather than DocIDs, add the `StoredStateField` parameter:

```
http://localhost:20000/action=Query&Text=apple&StoreState=True&StoredStateField=Reference
```

The returned results include a tag set that holds the token name, for example `<autn:state>CWQ4FJ9LZSE5-6</autn:state>`.

If the query is likely to return a very large data set, you can optimize performance by not printing the query results (assuming that you want only the state token):

```
http://localhost:20000/action=Query&Text=apple&StoreState=True&MaxResults=5000&Print=noresults
```

Query with the State Token

When you query IDOL Server, you can use the `StateID`, `StateMatchID`, and `StateDontMatchID` parameters to pass a state token. These parameters are similar to the `ID`, `MatchID`, and `DontMatchID` parameters that specify individual document IDs.

Action	State parameters supported
GetQueryTagValues	StateMatchID, StateDontMatchID
GetContent	StateID
List	StateMatchID, StateDontMatchID
Query	StateMatchID, StateDontMatchID
Suggest	StateID, StateMatchID, StateDontMatchID
SuggestOnText	StateMatchID, StateDontMatchID
Summarize	StateID
TermGetBest	StateID

If you use `StateMatchID` and `StateDontMatchID` in the same action, IDOL removes any documents in the `StateDontMatchID` list from the `StateMatchID` list.

Examples:

- **Retrieve all documents.** Use the `StateID` parameter to retrieve all documents in the token:

```
action=GetContent&StateID=CWQ4FJ9LZSE5-6
```

- **Query only the specified documents.** Use the StateMatchID parameter to, for example, query only the first four documents in the stored result set:

```
action=Query&Text=pear&StateMatchID=CWQ4FJ9LZSE5-6[0-3]
```

- **Query all but the specified documents.** Use the StateDontMatchID parameter to, for example, query all IDOL Server content except for the documents specified in the token:

```
action=Query&Text=pear&StateDontMatchID=CWQ4FJ9LZSE5-6
```

Note: A stored-state-aware DAH forwards any query that includes a state token to the IDOL Server that originally created the token.

Use a State Token with Index Actions

You can use the StateID and StateMatchID parameters to pass a state token in the DRELETEDOC, DREEXPORTIDX, DREEXPORTXML, DREEXPORTREMOTE, and DRECHANGEMETA actions.

Index action	State parameters supported
DRELETEDOC	StateID
DREEXPORTIDX	StateMatchID
DREEXPORTXML	StateMatchID
DREEXPORTREMOTE	StateMatchID
DRECHANGEMETA	StateID

Examples:

- **Change document database.** Use the StateID parameter to assign all 100 documents identified by state token CTBJ1V4Q04I5-100 (as well as the indexed documents with document IDs 1 and 2) to the archive database:

```
DRECHANGEMETA?Type=database&NewValue=archive&Docs=1+2&StateID=CTBJ1V4Q04I5-100
```

- **Delete documents.** Use the StateID parameter to delete the first five documents identified by state token CTBJ1V4Q04I5-100:

```
DRELETEDOC?StateID=CTBJ1V4Q04I5-100[0-4]
```

- **Export documents.** Use the StateMatchID parameter to export the second, fourth, and sixth documents identified by state token CTBJ1V4Q04I5-100:

```
DREEXPORTXML?StateMatchID=CTBJ1V4Q04I5-100[1+3+5]
```

Expire State Tokens

You can manually expire stored state tokens by using the TokenManagement action. For example:

```
action=TokenManagement&TokenAction=Expire&StateID=B8UGI95FKJG-23
```

This action expires the state token B8UGI95FKJG-23 from IDOL Server.

For more information about the TokenManagement action, refer to the *IDOL Server Reference*.

Chapter 18: View Documents

IDOL Server can display documents in a Web browser. This section describes the View service, which converts documents to HTML for viewing. The View service is primarily used to display result documents.

- [About the View Service](#) 333
- [Configure the View Service](#) 333
- [View Documents](#) 339
- [View Templates](#) 343

About the View Service

The View component uses IDOL KeyView filters to convert documents into HTML format for display in a Web browser. It can convert locally stored documents, as well as documents from an intranet or Internet source. It can also retrieve the document in its original format.

View can convert documents to HTML from many different formats, including:

- word processing documents
- spreadsheets
- presentations
- graphics files
- CAD Drawings
- email messages

When you convert documents, you can select terms for highlighting. During the conversion process, View sends a `Highlight` action to the Content component. When the document is displayed, it includes highlighting for the specified terms.

Configure the View Service

You can configure the View service as part of IDOL Server by using the IDOL Server configuration file. Most View settings are found in the `[Viewing]` section. You might also need to configure settings in the `[Paths]` section.

The `[Viewing]` section contains general settings that enable View to access documents to convert them to HTML. There are also settings that determine how View manages its cache.

For full details of the available settings for the View service, refer to the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)
- ["Set up SSL for the View Component" on page 360](#)

Enable View to Access Documents

By default, View cannot access documents in the local file system. Additionally, by default, it cannot access UNC paths to Windows shared folders that contain dollar symbols (\$). You must enable View to access documents in these directories by using the IDOL Server configuration file.

To enable View to access documents

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Viewing] section, or create one if it does not exist.
3. Set the ViewLocalDirectoriesCSVs parameter to a comma-separated list of directories that contain documents that you want to allow the View service to access. You must list the full paths. For example:

```
[Viewing]
ViewLocalDirectoriesCSVs=C:\Shared,C:\PDFs
```

View can access these directories and any subdirectories, and it can convert documents within these directories to HTML for display.

4. To allow the View service to access Windows shared folders that contain dollar symbols (\$), you must set ViewAllowedSpecialUNCPathsCSVs to a comma-separated list of these paths. For example:

```
ViewAllowedSpecialUNCPathsCSVs=C:\Shared$Files\
```

5. To allow the View service to access only the listed directories, set RestrictToAllowedDirectories to True. This parameter prevents the View service from accessing any directory or URL that is not listed. By default, only local disk access is restricted and the View service can access any (non-special) UNC path or Web URL.
6. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Configure View to Block Particular URLs or Hosts

By default, the View service attempts to access and process any URL or Web address that you send in the Reference action parameter for the View action. You can configure View to block certain URLs, host names or IP addresses. In this case, View does not process viewing requests for those URLs.

To configure View to block a URL or host

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Viewing] section.
3. Set ViewBlockedURLsCSVs to a comma-separated list of URLs that you want to block. You can use Wildcard values in the URLs. The View service matches the URL string, and does not retrieve files for any matching URLs. For example:

```
ViewBlockedURLsCSVs=*restart*,*example.com,15.*.*.*
```

This example blocks any URL that contains the string restart, and also blocks any Web address that ends with example.com, or an IP address that starts with 15. It does not block the IP address that example.com maps to, or the name of a host that uses an IP address that starts with 15.

4. Set `ViewBlockedHostCSVs` to a comma-separated list of IP addresses or host names that you want to block. The View service performs a host name lookup and blocks both the IP address and the host name. For example:

```
ViewBlockedHostCSVs=12.34.56.78,www.example.com
```

In this example, View blocks the IP address 12.34.56.78 and any host name that points to it. It also blocks `www.example.com` and the corresponding IP address.

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

Configure View to Use a Proxy Server

To allow the View service to access HTTP documents using a proxy server, you must configure the details for the proxy server in the IDOL Server configuration file.

To configure a proxy server for the View service

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[Viewing]` section.
3. Set `ProxyHost` to the IP address or host name of your proxy server. For example:

```
ProxyHost=proxy.example.com
```

4. Set `ProxyPort` to the port that View must use to communicate with the proxy server. For example:

```
ProxyPort=8080
```

5. If the proxy server requires NTLM authentication, set `NTLMProxy` to `True`. For example:

```
NTLMProxy=True
```

6. If the proxy server requires a user name and password, set `ProxyLogin` and `ProxyPassword`. For example:

```
ProxyLogin=idolview  
ProxyPassword=9szJwMkD28A
```

7. If you set `NTLMProxy` to `True`, you must set `ProxyLogin` to a fully qualified NTLM login. For example:

```
ProxyLogin=mydomain\\myusername
```

8. Set `NoProxyHostsCSVs` to a comma-separated list of host names for which the View service does not use the Proxy server. You can use Wildcard values. For example:

```
NoProxyHostsCSVs=localhost,*.companyname.com
```

9. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Configure View to Highlight Terms

You might need to add additional parameters to your configuration file to use View to highlight link terms in returned documents.

To configure View to highlight link terms

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Viewing] section.
3. Set the following parameters in the [Viewing] section:

IDOLHost	The host name where the Content component runs.
IDOLPort	The value of the ACI port for this Content component.
IDOLTimeout	The amount of time (in seconds) that View waits for a response from IDOL Server.

These parameters determine where View sends Highlight actions. By default, View uses the IDOL Server. You must configure these parameters if you want to send Highlight actions to a different Content component, or if you use View in a stand-alone configuration. For information on using stand-alone components, refer to the *IDOL Getting Started Guide*.

4. Set HighlightChunkSize to the maximum size (in bytes) of the chunks that View splits text into before highlighting. Enter -1 for an unlimited chunk size.
5. Set DefaultLanguageType to the language type that View uses for the link terms when highlighting. This parameter ensures that IDOL Server stems the terms correctly when highlighting. You can override this parameter by using the LanguageType parameter in the View action. See "[Highlight Expressions in Different Languages](#)" on page 342.
6. Set DefaultStartTag and DefaultEndTag to the default opening and closing HTML tags to use to highlight link terms.
7. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Configure View to Embed Images

By default, the View service converts any images to links when it converts a document to HTML. When your browser displays the HTML document, it retrieves the image by accessing the link in a subsequent request. Alternatively, you can configure the View service to embed each image as base64-encoded data in the HTML output.

Embedding the images in this way means that View must process only a single action for the entire document, rather than processing the subsequent requests for images. This process means that you can use a load balancer to process View requests, without having to ensure that the image requests go to the correct View component.

To configure View to embed images

1. Open the IDOL Server configuration file in a text editor.
2. In the [Viewing] section, set EmbedImages to True. For example:

```
EmbedImages=True
```

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

You can override the EmbedImages configuration parameter for an individual View action by using the EmbedImages action parameter. This action parameter uses the configuration setting as its default value.

Configure the View Cache

When the View service converts a document to HTML, it caches the results. If it receives a request for the same document again, it retrieves the version from the cache, rather than converting it again.

You can use an internal View cache, or you can cache documents on a memcached server instance. For details about memcached servers, refer to <http://memcached.org>.

You can also share a cache between multiple View services.

Configure the Internal View Cache

By default, the View service uses an internal cache, stored on disk. You can configure how this cache behaves.

To configure the internal View cache

1. Open the IDOL Server configuration file in a text editor.
2. In the [Viewing] section, set CacheType to Internal. For example:

```
CacheType=Internal
```

3. In the [Viewing] section, set CacheExpirySeconds to the length of time (in seconds) that you want to keep the cache. Use a large value to ensure that the View service does not have to regularly convert the same documents. For example:

```
CacheExpirySeconds=604800
```

4. In the [Paths] section, set ViewCacheDirectory to the directory in which you want the View service to store the cache. The View service must have write access to this directory. For example:

```
ViewCacheDirectory=C:\IDOL\IDOLServer\IDOL\view\Cache
```

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

Use a Memcached Server View Cache

The View service can store cache documents in a memcached server instance.

Note: By default, the memcached program has a maximum entry size limit of 1 megabyte, so it cannot process files bigger than 1 megabyte.

To configure View to use a memcached cache

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Viewing] section.
3. Set CacheType to Memcached. For example:

```
CacheType=Memcached
```

4. Set `CacheServers` to a comma-separated list of `host:port` pairs, where:

<i>host</i>	is the host name or IP address of the memcached server.
<i>port</i>	is the port that View must use to contact the memcached server.

For example:

```
CacheServers=localhost:11211,cacheserver:21532
```

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

Use a Shared View Cache

You can share a View cache between multiple View services. To use a shared cache, you must have a memcached server instance that each View service can access. This memcached server stores information about where to find the converted cache documents, so that each View service can retrieve them.

When you use a shared cache, the `ResetCache` action and automatic cache expiration are unavailable. The `CacheExpirySeconds` parameter specifies only how often to delete old, duplicate job files, but it does not automatically delete the newest instance of a job.

To use a shared View cache

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[Viewing]` section.
3. Set `CacheType` to `Internal`. For example:

```
CacheType=Internal
```

4. Set `SharedCache` to `True`. For example:

```
SharedCache=True
```

5. Set `CacheServers` to a comma-separated list of `host:port` pairs, where:

<i>host</i>	is the host name or IP address of the memcached server.
<i>port</i>	is the port that View must use to contact the memcached server.

6. In the `[Paths]` section, set `ViewCacheDirectory` to the directory to use to store the cache. For example:

```
ViewCacheDirectory=C:\View\Cache
```

7. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
8. Repeat [Step 1](#) to [Step 7](#) for each View service that must share the cache.

Distribute View Servers

When you have multiple View servers set up to process View requests, you can use the Distributed Action Handler (DAH) to distribute requests between the servers.

To distribute View Servers with a DAH, you must configure the DAH to distribute the View actions by reference. The DAH distribute by reference mode ensures that it always sends repeated requests for the same document to the same View child server. This process ensures that each document is cached in only one View server, and the cache is used in subsequent requests for the same document.

DAH distribution by reference is available only for the View, GetLink, and ViewGetDocInfo actions. When you upload documents to view by using the View action with the ViewUpload parameter, you can add the Reference parameter to ensure consistent routing.

Note: If you do not configure DAH in distribute by reference, images in converted documents might fail to load.

For more information about this method, refer to the *Distributed Action Handler Administration Guide*.

View Documents

To convert a document into HTML, use the View action. For example:

```
http://IDOLhost:IDOLport/action=View&Reference=DocumentReference
```

where:

<i>IDOLhost</i>	is the name or IP address of the host on which IDOL Server runs.
<i>IDOLport</i>	is the IDOL Server ACI port.
<i>DocumentReference</i>	<p>is the full reference of the document that you want to view. The reference can be:</p> <ul style="list-style-type: none">• A document in the local file system, for example: Reference=C:\Documents\report.doc• A document on an intranet site, for example: Reference=//intranet/report.doc• A Web page or document accessible on the Internet, for example: Reference=http://news.bbc.co.uk <p>Note: You can also upload a file to view. See "View an Uploaded Document" on page 341.</p>

View the Document Directly in the Web Browser

By default, the document returns in the standard ACI XML format. The document is written as a string of base64-encoded data. If you pass the results to a client application for further processing, you might want the document in this format.

To view the document directly in your Web browser, set the NoACI parameter to True in the View action. This returns the document in a format that the Web browser can read, so that you can view the HTML directly.

For example:

```
http://localhost:9000/action=View&Reference=C:\PDFs\report.pdf&NoACI=True
```

Use IDOL Admin to View Documents

As an alternative to using the `View` action, you can use the IDOL Admin interface to convert documents into HTML to display in a Web browser.

To use IDOL Admin to view a document in your Web browser

1. In the **Control** menu, click **View**.
2. Type the document reference into the text box. The reference can be:
 - a document in the local file system, for example:
C:\Documents\report.doc
 - a document on an intranet site, for example:
//intranet/report.doc
 - a Web page or document accessible on the Internet, for example:
http://news.bbc.co.uk
3. If your document references have a common format, you can configure a standard prefix in the IDOL configuration file to add to each document reference. If you select the **Use reference prefix** check box, IDOL Server adds this prefix to the `View` action automatically.
For more information on using the `ReferencePrefix` and `UseReferencePrefix` parameters, refer to the *IDOL Server Reference*.
4. Click **View**.
The document opens in your Web browser.

View the Latest Version of a Document

`View` stores converted documents in the cache. It stores them for a length of time determined by one of the following configuration parameters:

- `CacheExpirySeconds` for non-secured documents
- `SecureCacheExpirySeconds` for secured documents

If a user requests a document that exists in the cache, `View` retrieves it from the cache to improve performance.

You can set the `IgnoreCache` parameter in a `View` action to convert the document again, rather than retrieving the cached version. For documents that change frequently, this parameter ensures that `View` returns the most recent form of the document. For example:

```
http://localhost:9000/action=View&Reference=C:\PDFs\report.pdf&IgnoreCache=True
```

Tip: You can delete all the files from the `View` cache directory by using the **Reset View Cache** feature in the **Controls** section of the **Overview** tab in the **Status** page of the IDOL Admin interface. For more information, refer to the *IDOL Admin User Guide*.

View an Uploaded Document

You can view a document that the View service cannot access directly on a Web site or file system, by uploading the file to View in the View action. To upload a document, you must send the View action as a POST request, and upload the document as `multipart/form-data` in the `ViewUpload` parameter.

For example:

```
<form name="example" method="post" enctype="multipart/form-data"
action="http://localhost:14000/a=view&output=html&noaci=true" target="view_output">
View This File:(input type="file" name="viewupload" />
    <br><br>
    Highlight: <input type="text" name="links" />
    <br><br>
    <input type="submit" value="Submit" />
</form>
```

This example shows a POST request for the View action, including the usage of `ViewUpload`.

Highlight Terms

When you convert documents to HTML you can highlight link terms in the returned documents.

To highlight terms in a View action

1. Set the `Links` parameter in a View action to the name of the terms that you want to highlight.
2. Set the `StartTag` and `EndTag` parameters to the HTML tags to use to highlight the term.

For example:

```
action=View&Reference=C:\Documents\report.doc&NoACI=True&Links=price&StartTag=<font
color="red">&EndTag=</font>
```

When the View service receives this action, it sends the document text to the Content component as part of a `Highlight` action. For example:

```
action=Highlight&Text=<document text>&Links=price&StartTag=<font
color="red">&EndTag=</font>
```

The response from Content contains the specified HTML tags around the link terms, which View incorporates into the HTML conversion of the document.

In this example, View converts `report.doc` to HTML for viewing directly by the Web browser. In the returned document, all instances of the term `price` are highlighted using the HTML tags `` and ``. For example:

In some cases the `price` has risen considerably.

When you view this document in a Web browser, these terms display in red text.

Highlight Boolean Expressions

To use Boolean expressions in your link terms, set the `Boolean` parameter to `True` in the `View` action. This process is similar to setting the `Boolean` parameter to `True` for a `Highlight` action. IDOL Server then treats terms in the `Links` parameter as query text, rather than a list of terms. For example:

```
action=View&Reference=C:\Documents\report.doc&NoACI=True&Boolean=True&Links=price  
AND risen&StartTag=<font color="red">&EndTag=</font>
```

This action highlights the terms `price` and `risen`, if both terms appear in the document.

Highlight Expressions in Different Languages

You can specify the language type of the text in the `Links` parameter by adding the `LanguageType` parameter to the `View` action. This setting ensures that IDOL Server stems words correctly before highlighting.

Use this parameter if the language type of the text to highlight is not the same as the language type you set in the `DefaultLanguageType` configuration parameter. For example:

```
action=View&Reference=C:\Documents\report.doc&NoACI=True&Links=price&StartTag=<font  
color="red">&EndTag=</font>&LanguageType=EnglishUTF8
```

Highlight Multiple Link Terms

You can use the `MultiHighlight` action parameter to highlight different link terms with different HTML tags.

To highlight multiple terms with different HTML tags

1. In the `View` action, set `MultiHighlight` to `True`.
2. Set `Links` to a semicolon-separated list of terms or groups of terms.
3. Set `StartTag` and `EndTag` to a semicolon-separated list of opening and closing HTML tags respectively. The first pair of tags apply to the first link term, the second pair of tags apply to the second link term and so on.

For example:

```
action=View&MultiHighlight=True&Links=dog+OR+cat;rabbit;"apples and  
pears"&StartTag=<b>;<i>;<font color="red">&EndTag=</b>;</i>;</font>&Boolean=True
```

This action highlights the terms `dog` or `cat` in bold, the term `rabbit` in italic, and the phrase `apples and pears` in red.

Specify Document Processing

When you send a `View` action, use the `OutputType` parameter to specify the way that the `View` service processes the document. Set `OutputType` to one of the following values:

HTML	Converts the document to HTML. It also modifies relative or inline URL links with a prefix specified by the <code>URLPrefix</code> parameter.
------	---

ReplaceURLs	Modifies all relative or inline URL links with a prefix specified by the URLPrefix action parameter. If you do not specify a URLPrefix, the View service uses the value of the DefaultURLPrefix configuration parameter. This output type does not convert the document to HTML.
Raw	Returns the original document without any conversion or modification. In this case, highlighting is not available, unless the original document is in HTML format.
Redirect	Directs the client to the original Web site to display the original document. In this case, highlighting is not available.

By default, OutputType is set to:

- ReplaceURLs when the document reference is a URL with the prefix http:, https:, msx:, or notes.
- HTML for all other documents.

View Document Information

You can use the ViewGetDocInfo action to return information about the View action for a document. For example, this action returns the document type and the number of pages that result from the KeyView HTML conversion.

The ViewGetDocInfo action accepts all View action parameters. It returns information for the particular View action that you send. For example, if you specify the Links parameter, ViewGetDocInfo returns information about which pages contain highlighted terms in the converted document.

View Templates

The View service can apply *templates* when it converts documents to HTML, which alter the appearance of the returned document. These templates are text files with .ini file extensions, which contain section names in square brackets, and key=*value* pairs. For example:

```
[KVHTMLOptionsEx]
OutputCharSet=KVCS_UTF8
bUseDocumentColors=True
```

There are several basic templates installed with IDOL Server. By default, these are found in:

```
installDir\common\keyview\templates
```

If you move the templates folder, you must edit the value of the ViewingTemplatesPath configuration parameter in the [Paths] section of your IDOL Server configuration file.

The following table lists the provided templates.

View service templates

Template	Description
k2lowband.ini	This template is useful when you need to provide information to a mobile workforce that might not always have access to fast connections.

View service templates, continued

Template	Description
	<ul style="list-style-type: none">• Creates a single HTML file.• Creates a table of contents at the top of the HTML document.• Suppresses embedded graphics from the source document.
k2lowbandunix.ini	Similar to k2lowband.ini except that it saves vector graphics in a vector format to view using a Java applet.
k2frames.ini	<ul style="list-style-type: none">• Divides word processing documents, spreadsheets, and presentations into multiple files according to the heading levels in the document.• Creates two frames. The table of contents (based on source document heading levels and page breaks) appears in the left frame, and the HTML files appear in the right frame.• Inserts Previous and Next links at the end of each block.
k2framesunix.ini	Similar to k2frames.ini except that it saves vector graphics in a vector format to view using a Java applet.
k2onefile.ini	<ul style="list-style-type: none">• Creates a single HTML file.• Creates a table of contents at the top of the HTML document.• Lists all metadata (Title, Subject, Author, Comments, and so on).• Converts graphics to JPEG with 640 x 640 resolution.
k2pdfonefile.ini	Similar to k2onefile.ini except that it converts graphics to JPEG and preserves the original resolution.
k2onefileunix.ini	Similar to k2onefile.ini except that it saves vector graphics in a vector format to view using a Java applet.

Apply a Template to a Document

To view a document with one of these templates, add the `ViewTemplate` parameter to the `View` action. For example:

```
action=View&Reference=C:\PDFs\report.pdf&NoACI=True&ViewTemplate=k2frames.ini
```

Apply a Default Template to All Documents

To define a default template to use for all documents in all `View` actions, set the `DefaultTemplate` parameter in the `[Viewing]` section of your IDOL Server configuration file.

```
[Viewing]  
DefaultTemplate=k2onefile.ini
```

Modify the HTML Output for Documents

You can edit the supplied templates to change the appearance of the HTML output. Some basic configuration parameters are discussed in this section. All available sections and parameters are fully described in the *KeyView HTML Export SDK C and COM Programming Guide*. These are advanced parameters that can improve the fidelity and accuracy of the HTML output. They are for users who are already familiar with using template files.

To modify the HTML output

1. Open the template file that you want to update, or create a new template file with any other options that you want to use.
2. Add a [KVHTMLConfig] configuration section.
3. Add the parameters that you require (the parameters are described in the following sections). For example:

```
[KVHTMLConfig]
KVCFG_INCLREVISIONMARK=True
KVCFG_LOGICALPDF=LPDF_RTL
KVCFG_SETTEXTROTATE=True
KVCFG_BLANKPICTURE=True
```

4. Save and close the template file.
5. To use the template, send a View action with ViewTemplate set to the name of the template file with these modifications. For example:

```
action=View&Reference=C:\PDFs\report.pdf&NoACI=True&ViewTemplate=MyTemplate.ini
```

Modify the HTML Output for PDF Files

You can modify the HTML output from the conversion of PDF files by setting the following parameters in the [KVHTMLConfig] section of the template file.

Template configuration options for PDF files

Parameter	Description
KVCFG_SETHIFIPDF	Converts each page of a PDF document to a JPEG file, which provides a high-fidelity conversion of the document. By default, PDF files are converted to HTML.
KVCFG_SUPPRESSTOCPRINTIMAGE	Prevents the use of bookmarks in a PDF file to generate a table of contents in the HTML output (the default behavior).
KVCFG_DELSOFTHYPHEN	Removes soft hyphens in the source document and joins the hyphenated words in the HTML output. By default, soft hyphens are maintained.
KVCFG_LOGICALPDF	Determines the order in which to extract paragraphs from a PDF. By default, the View service extracts PDF paragraphs in the order in which they are stored in the file (unstructured reading order), not the order in

Template configuration options for PDF files, continued

Parameter	Description
	<p>which they appear on the visual page (logical reading order). You can specify the following paragraph directions:</p> <ul style="list-style-type: none"> • LPDF_LTR. Logical reading order and left-to-right paragraph direction. Use this option when most of your documents are in a language that uses left-to-right reading order, such as English or German. • LPDF_RTL. Logical reading order and right-to-left paragraph direction. Use this option when most of your documents are in a language that uses right-to-left reading order, such as Hebrew or Arabic. • LPDF_AUTO. Logical reading order. The PDF reader determines the paragraph direction for each PDF page, and then sets the direction accordingly. This option is used when no paragraph direction is specified. • LPDF_RAW. Unstructured paragraph flow. This value is the default behavior. If you enable logical reading order, and you want to return to an unstructured paragraph flow, set this flag.
KVCFG_SETTEXTROTATE	<p>Displays rotated text in a file at zero degrees at the bottom of the page that it appears on. The View Service enlarges the page to accommodate the text.</p> <p>By default, rotated text in a file is displayed in its original position, at the original font size, and at zero degrees rotation. Because the text is the original size, but might be displayed in a smaller space, the text might overlap adjacent text in the HTML output. Use the KVCFG_SETTEXTROTATE option to avoid this problem.</p> <p>HTML markup does not support text rotation.</p>
KVCFG_SETPDFINVISIBLETEXTTOGGLE	<p>Displays a JavaScript button in the upper right corner of the exported page, which you can click to switch between invisible and normal text.</p> <p>When you turn on invisible text, the invisible text is displayed and the normal content is hidden; when you turn off invisible text, the invisible text is hidden.</p>
KVCFG_SETPDFINVISIBLETEXTOPACITY	<p>Displays invisible text with an image in the PDF.</p> <p>Invisible text often occurs in PDF documents when the PDF software processes rasterized images through optical character recognition and then inserts the text in the PDF. You might want to display both the invisible text and the rasterized image.</p> <p>Set the invisible text <i>opacity</i> to an integer between 0 and 100, where 0 hides the invisible text and 100 displays it fully.</p>

Related Topics

- ["Modify the HTML Output for Documents" on page 345](#)

Hide Graphics

To hide graphics when you display a document, but still maintain the text flow of the original document, set the `KVCFG_BLANKPICTURE` parameter to `True` in the `[KVHTMLConfig]` section of the template file. KeyView does not convert graphics in a document, but it generates an image tag with an empty `src` attribute, creating an empty placeholder for the graphic. For example:

```
<img src="" height="136" width="101">
```

This parameter applies only to word processing formats, and is disabled by default.

To hide graphics completely, and exclude the image tags from the converted HTML document, set the `bNoPictures` parameter to `True` in the `[KVHTMLConfig]` section of the template file. This parameter is disabled by default.

Related Topics

- ["Modify the HTML Output for Documents" on page 345](#)

Show Revised Content and Revision Information

You can show content that was deleted from a document with revision tracking enabled by setting the `KVCFG_INCLREVISIONMARK` parameter to `True` in the `[KVHTMLConfig]` section of the template file. When you enable this parameter, revision information (revision title, reviewer name, and revision date and time) for deletions and insertions of content display in a tooltip.

To reset the parameter and exclude deleted content and revision information from the HTML output, set the parameter to `False`. The default is `False`.

Format Revised Content

You can display revised content with a specified title and revision information. By default, the title is either the text string *inserted:* or *deleted:*, and it includes the reviewer name, and date and time.

You can also define a unique HTML style (such as, `color: red; background: orange`) to apply to the modifications for each reviewer. This allows you to easily differentiate between edits from multiple reviewers. For example, changes made by JSmith are highlighted in red, changes made by RBrown are highlighted in blue, and so on.

For example, you can configure View server to generate the following markup for inserted text:

```
<ins style="color: red" title="Inserted: JohnD, 2006-04-24T14:47:00"  
cite="mailto:JohnD" datetime="2006-04-24T14:47:00">This text was added</ins> in a  
previous version.
```

This text is displayed in the browser as:

This text was added in a previous version.

When you hover the cursor over the underlined text in the browser, the text **"Inserted: JohnD, 2006-04-24T14:47:00"** is displayed as a tooltip.

Use the following parameters to define the display of revised content.

Parameter	Description
KVCFG_ INSTITLEPREFIX	Specifies a string to include at the beginning of the tooltip for insertions. By default, the string <i>inserted:</i> is included.
KVCFG_ INSTITLEFLAG	Specifies whether to display the reviewer name and the revision date and time in the tooltip for insertions. The following values are available: <ul style="list-style-type: none"> • RMT_OFF. Does not display the information with the insertion. • RMT_AUTHOR. Displays the name of the reviewer who made the insertion. • RMT_DATETIME. Displays the date and time of the insertion. • RMT_AUTHORDATETIME. Displays the reviewer, date, and time of the insertion. This is the default.
KVCFG_ DELTITLEPREFIX	Specifies a string to include at the beginning of the tooltip for deletions. By default, the string <i>deleted:</i> is included.
KVCFG_ DELTITLEFLAG	Specifies whether to display the reviewer name and the revision date and time in the tooltip for deletions. The following values are available: <ul style="list-style-type: none"> • RMT_OFF. Does not display the information with the deletion. • RMT_AUTHOR. Displays the name of the reviewer who made the deletion. • RMT_DATETIME. Displays the date and time of the deletion. • RMT_AUTHORDATETIME. Displays the reviewer, date, and time of the deletion. This is the default.
KVCFG_ AUTHORSTYLE <i>N</i>	Specifies a list of HTML style attributes to apply to the changes for each reviewer.

For example:

```
[KVHTMLConfig]
KVCFG_INCLREVISIONMARK=True
KVCFG_INSTITLEFLAG=RMT_AUTHOR
KVCFG_INSTITLEPREFIX=New:
KVCFG_DELTITLEFLAG=RMT_AUTHOR
KVCFG_DELTITLEPREFIX=deleted:
KVCFG_AUTHORSTYLE0=color: red; background: white
KVCFG_AUTHORSTYLE1=color: yellow; background: blue
KVCFG_AUTHORSTYLE2=color: green; background: black
```

This configuration:

- enables revision marks.
- displays only the name of the reviewer in the tooltip.
- adds the string *New:* to the beginning of the tooltip.
- defines styles to use for each reviewer.

Related Topics

- ["Modify the HTML Output for Documents" on page 345](#)

Show Hidden Content

Microsoft Word, Excel, or PowerPoint documents contain hidden information, some of which is shown by default when exported and some of which is hidden by default. There are several options that allow you to determine exactly which types of hidden data to export.

Hidden Content in Microsoft Documents

You can display four types of hidden data from Microsoft Word, Excel, and PowerPoint documents, each of which has a corresponding parameter, which you can change to determine whether to show the hidden data. The table below lists each data type, its default behavior, and its corresponding configuration parameter.

Hidden data settings

Hidden data type	Default behavior	Configuration API flag
Microsoft Word		
Comments	Shown ¹	KVCFG_WP_NOCOMMENTS
Hidden text	Hidden	KVCFG_WP_SHOWHIDDENTEXT
Date field codes	Hidden	KVCFG_WP_SHOWDATEFIELDPCODE
File name field codes	Hidden	KVCFG_WP_SHOWFILENAMEFIELDPCODE
Microsoft Excel		
Hidden information	Hidden	KVCFG_SS_SHOWHIDDENINFOR
Comments	Hidden	KVCFG_SS_SHOWCOMMENTS
Formulas	Hidden	KVCFG_SS_SHOWFORMULA
Microsoft PowerPoint		
Hidden slides	Shown	KVCFG_PG_HIDEHIDDENSLIDE
Comments	Shown ²	KVCFG_PG_HIDECOMMENT
Comments slide	Hidden	KVCFG_PG_SHOWCOMMENTSSSLIDE ³
Slide notes	Hidden	KVCFG_PG_SHOWSLIDENOTES

¹ Shown by default in Microsoft Word 97 to 2003 documents.

² Shown by default in Microsoft PowerPoint 97 to 2000 documents.

³ This setting affects PowerPoint 2003 and 2007 only.

Related Topics

- ["Modify the HTML Output for Documents" on page 345](#)

Part V: Administration and Maintenance

This section explains how to administer your IDOL Server installation and how to perform routine maintenance.

- ["Set up Security"](#)
- ["Add Users to IDOL Server"](#)
- ["Mail"](#)
- ["Administer IDOL Server "](#)
- ["Back up the IDOL Server"](#)
- ["Troubleshoot IDOL Server"](#)

Chapter 19: Set up Security

This section describes the process of applying security settings to documents, and the process of setting up an SSL connection between IDOL components.

- [Set up Security on Documents](#) 353
- [Set up an SSL Connection](#) 356

Set up Security on Documents

You can apply custom security settings to the documents that you index into IDOL Server. To do this, you identify fields in these documents that determine the security settings that are appropriate for each document.

Alternatively, you can specify the security property of a document every time that you index it by sending an additional parameter with the index action.

For more details on the settings that the [Security] section can contain and on how you can configure them, refer to the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)
- ["Edit the Configuration File" on page 35](#)
- ["Encrypt Passwords" on page 457](#)

To set up automatic security application for documents

1. Open the IDOL Server configuration file in a text editor.
2. In the [Security] section, set the `SecurityInfoKeys` parameter to specify the security encryption keys to use to encrypt and decrypt the security information used by IDOL Server. You can set the `SecurityInfoKeys` parameter either to a comma-separated list of four signed 32-bit integers, or to the name of an AES key file. For information about how to generate a key file with the `outpassword` command-line tool, see ["Encrypt Passwords" on page 457](#).

For example:

```
[Security]
SecurityInfoKeys=123456789,144135468,56443234,2000111222
```

3. In the [Security] section, list the security types that you want to use.

```
0=NT
1=Netware
2=Notes
3=Exchange
```

4. Create a section for each of the security types you defined (the section must have the same name as the security type). For each section, provide settings that determine how IDOL Server handles that security type. For example:

```
[NT]
SecurityCode=1
Library=nt_security.dll
Type=AUTONOMY_SECURITY_V4_NT_MAPPED
ReferenceField=*/AUTONOMYMETADATA
```

```
[Netware]
SecurityCode=2
Library=netware_security.dll
Type=AUTONOMY_SECURITY_NETWARE_MAPPED
ReferenceField=*/AUTONOMYMETADATA
```

```
[Notes]
SecurityCode=3
Library=notes_security.dll
Type=AUTONOMY_SECURITY_V4_NOTES_MAPPED
ReferenceField=*/AUTONOMYMETADATA
```

```
[Exchange]
SecurityCode=4
Library=exchange_security.dll
Type=AUTONOMY_SECURITY_EXCHANGE_MAPPED
ReferenceField=*/AUTONOMYMETADATA
```

5. In the [FieldProcessing] section, set up processes that allow IDOL Server to recognize the security type of documents (unless you send an additional parameter to specify the security property of a document every time you index a document).

If you use a version 4 security type (for example, AUTONOMY_SECURITY_V4_NOTES_MAPPED), you must include a process that defines how to handle metadata. For example:

```
[FieldProcessing]
0=DetectNT
1=DetectNetware
2=DetectNotes
3=DetectExchange
4=DefineMetaData
```

6. Create a section for each of the processes that you listed, in which you create a property for the process (security properties always point to a defined security type). Identify the field that you want to associate with the processes.

Note: The property that you create must not have the same name as the process.

When identifying the fields, use the format /FieldName to match root-level fields, */FieldName to match all fields except root-level, or /Path/FieldName to match fields that the specified path points to.

You can use the `PropertyMatch` parameter to identify a specific value that fields must have to be processed.

For example:

```
[DetectNT]
Property=SetNTProperty
PropertyFieldCSVs=*/DRESECURITYTYPE
PropertyMatch=*nt
```

```
[DetectNetware]
Property=SetNetwareProperty
PropertyFieldCSVs=*/DRESECURITYTYPE
PropertyMatch=*netware
```

```
[DetectNotes]
Property=SetNotesProperty
PropertyFieldCSVs=*/DRESECURITYTYPE
PropertyMatch=*notes
```

```
[DetectExchange]
Property=SetExchangeProperty
PropertyFieldCSVs=*/DRESECURITYTYPE
PropertyMatch=*exchange
```

```
[DefineMetaData]
Property=HideMetaData
PropertyFieldCSVs=*/AUTONOMYMETADATA
```

7. Create a section for each of the properties and specify appropriate configuration settings for each property. These configuration parameters define the processes to apply to all the fields (or all documents that contain the fields) that you previously associated with the processes.

If you use a version 4 security type (for example, `AUTONOMY_SECURITY_V4_NOTES_MAPPED`), you must set `ACLType` to `True` in the section that sets up how IDOL Server handles metadata, to implement optimized security.

```
[SetNTProperty]
SecurityType=NT
```

```
[SetNetwareProperty]
SecurityType=Netware
```

```
[SetNotesProperty]
SecurityType=Notes
```

```
[SetExchangeProperty]
SecurityType=Exchange
```

```
[HideMetaData]
HiddenType=True
ACLType=True
```

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

Note: For details of how to ensure security in an IDOL infrastructure, refer to the *Intellectual Asset Protection (IAS) Administration Guide*.

Tip: You can view information about the configured security types in IDOL Server on the **Security Types** tab on the Status page in the IDOL Admin interface. For more information, refer to the *IDOL Admin User Guide*.

Set up an SSL Connection

There are several ways to set up Secure Socket Layer (SSL) connections for IDOL Server. For example, you can:

- Configure an SSL *gateway*. You configure incoming communications to IDOL Server to use SSL connections, but communications between components in IDOL Server are plain.
- Configure SSL between all IDOL components in a unified IDOL Server. All communications into IDOL, and between components, are configured with SSL connections.
- Configure SSL between stand-alone IDOL components.

In all cases the basic principle of configuring SSL is the same, but the exact configuration varies.

To configure SSL connections

1. Set the `SSLConfig` parameter to the name of the section in which you define SSL options. The configuration sections where you set `SSLConfig` vary depending on your setup. In general:
 - For incoming ACI calls, set the `SSLConfig` parameter in the `[Server]` section.
 - For incoming Index actions, set the `SSLConfig` parameter in the `[IndexServer]` section.
 - For incoming Service actions, set the `SSLConfig` parameter in the `[Service]` section.
 - For outgoing ACI calls to IDOL components, set the `SSLConfig` parameter in each component section. For example, `[AgentDRE]`.

For example:

```
[Server]
SSLConfig=SSLOption1
```

2. For each `SSLOption` you define, create a new configuration section to contain the SSL options.

For example:

```
[SSLOption1]
```

3. Within each SSL options section, you can specify the following SSL parameters:

SSLMethod	Determines which SSL protocol to use: SSLV2, SSLV3, SSLV23, or TLSV1. In most cases, SSLV23 is appropriate.
SSLCertificate	The SSL Certificate file to use to identify this component to a peer.

	The certificate can be in either ASN1 or PEM format. HPE recommends that you use the PEM format. This parameter requires a matching SSLPrivateKey value.
SSLPrivateKey	The private security key for the SSL certificate. The security key can be in either ASN1 or PEM format. This parameter requires a matching SSLCertificate value. The private key can be password protected. See SSLPrivateKeyPassword.
SSLCACertificate	The Certificate Authority certificate indicating that this component trusts only communication with a peer that offers a certificate signed by the specified CAs.
SSLCheckCertificate	Requests a certificate signed by a trusted authority from peers. Setting SSLCACertificate implicitly sets this parameter to True. If you set SSLCACertificate to False, IDOL Server encrypts communications, but does not request certificates from peers.
SSLCheckCommonName	Determines whether the host name listed in the peer certificate (that is, the CommonName or "CN" attribute) resolves to the same IP address as the peer itself, as determined by the network connection. This parameter helps verify the identity of the peer. For example, if the host name in a certificate is eip.example.com and resolves to an IP address of 12.3.4.56, the peer must share the same IP address.
SSLPrivateKeyPassword	If the file defined in SSLPrivateKey is password protected, use this parameter to specify the password. The password can be in plain text or in basic or AES encryption format.

Related Topics

- ["Password Encryption" on page 457](#)

Set up SSL between IDOL components

If you are using a unified IDOL Server configuration, you can enable SSL communication between IDOL components. Set the SSLIDOLComponents parameter to True in the [Server] section.

You can configure Secure Socket Layer (SSL) connections for communication between the following components and other IDOL components:

- Agentstore
- Category
- Community
- Content
- IDOL Proxy
- View

You can set `SSLConfig` in the following configuration sections for SSL communications between IDOL components:

- `[Server]` to configure SSL communications for incoming ACI calls for all components.
- `[IndexServer]` to configure incoming SSL communications to the IDOL Server index port. This option implicitly includes any indexing components (such as Content).
- `[Service]` to configure incoming SSL communications to the IDOL Server service port.
- `[Agent]` to configure outgoing SSL communications from the Category component to the Content component where the IDOL Server agent index is stored (Agentstore).
- `[AgentDRE]` to configure outgoing SSL communications from IDOL components to the Content component where the IDOL Server agent index is stored (Agentstore).
- `[CatDRE]` to configure outgoing SSL communications from IDOL components to the Content component where the IDOL Server category index is stored (Agentstore).
- `[DataDRE]` to configure outgoing SSL communications from IDOL components to the Content component where the IDOL Server data index is stored.

Note: For SSL communication with the Agentstore component, you must also configure SSL settings in the Agentstore configuration file.

For example:

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

```
[AgentDRE]
SSLConfig=SSLOptions2
...
```

```
[DataDRE]
SSLConfig=SSLOptions2
...
```

For Omni Group Servers:

```
[Note]
GroupServerHost=...
GroupServerPort=...
SSLConfig=SSLOptions2
```

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

```
[SSLOptions2]           //SSL options for outgoing connections
SSLMethod=SSLV23
SSLCertificate=host2.crt
```

```
SSLPrivateKey=9s7BxMjD2d3M3t7awt/J8A  
SSLCACertificate=trusted.crt
```

Set up SSL for Shared Communications

To define one set of SSL options to share between multiple communications, define only one `SSLOption` section. Set all the `SSLConfig` parameters to the single `SSLOption` section in each configuration section. For example:

```
[Server]  
SSLConfig=SSLOptions  
...
```

```
[AgentDRE]  
SSLConfig=SSLOptions  
...
```

```
[DataDRE]  
SSLConfig=SSLOptions  
...
```

For Omni Group Servers:

```
...  
[Note]  
GroupServerHost=hostname  
GroupServerPort=1.2.3.4  
SSLConfig=SSLOptions
```

```
[SSLOptions]  
...
```

Set up SSL for Mailer

You can set up Secure Socket Layer (SSL) connections between the Mailer component and Community and Category.

To configure SSL for Mailer, set the following parameters in the `[Email]` section:

<code>SSLConfig</code>	The SSL options for a secure connection with Community.
<code>ClassificationSSLConfig</code>	The SSL options for a secure connection with Category.

For example:

```
[Email]  
SSLConfig=SSLOption  
ClassificationSSLConfig=CatSSLConfig
```

Set up SSL for the View Component

To configure SSL for the View component, you can set the following parameters in the [Viewing] section:

SSLConfig	SSL options for a secure connection to the Content component to use for highlighting.
OutgoingSSLConfig	SSL options for a secure connection to other applications. The View component uses this option if someone requests to view a document that exists on a server that is secured by SSL.

For example,

```
[Viewing]
IdolHost=localhost
IdolPort=9000
SSLConfig=SSLOption1
OutgoingSSLConfig=SSLOption2
```

Set up SSL for Communications to Remote Servers

You can use SSL connections to the remote server when you use the DREEXPORTREMOTE index action.

You must configure an SSLOptions configuration section in the IDOL Server configuration file. For example:

```
[SSLOptions1]
SSLMethod=SSLV23
SSLCertificate=host1.crt
SSLPrivateKey=host1.key
SSLCACertificate=trusted.crt
```

When you send the DREEXPORTREMOTE index action, set the SSLConfig action parameter to the name of the configuration section where you configure the SSL options. For example:

```
http://localhost:20001/DREEXPORTREMOTE?&TargetEngineHost=banff&TargetEnginePort=20200&Delete=True&Blocking=True&BatchSize=10000&SSLConfig=SSLOptions1
```

In this example, IDOL Server uses the SSL configuration options from the [SSLOptions1] configuration section to contact the remote server.

Log SSL Settings

You can define a log file for logging SSL setting details:

```
[LOGGING]
0=...
1=SSL_LOG_STREAM
```

```
[SSL_LOG_STREAM]
```

```
LogFile=ssl.log  
LogTypeCSVs=SSL  
LogLevel=Normal  
LogTime=True  
LogEcho=False  
LogMaxSizeKbs=1024
```

Check SSL Status

You can check whether your service port and index port SSL settings are correct by using ACI actions.

- The `GetChildren` action returns the `<autn:serviceport_ssl_enabled>` tag. The value is `True` when SSL is enabled for the service port.
- The `GetStatus` action returns the `<indexport_ssl_enabled>` tag. The value is `True` when SSL is enabled for the index port.

Chapter 20: Add Users to IDOL Server

This section describes how to create and manage IDOL user accounts.

- [Create IDOL Users](#)363
- [Integrate with a Third-Party User Structure](#)365
- [Implement User Account Security](#) 365

Create IDOL Users

You must store users in IDOL Server to use any of the following IDOL Server features:

- **Agents.** Users can store queries in the form of agents to ensure that they always have the latest available information. Users can edit and retrain their agents.
- **Profiling.** A profile is a set of agents that are trained using the documents that the user is looking at, and that return data that matches their interests. You can set up your application so that every time a user looks at a document, the profile determines whether the document is relevant to its agent training. The profile then either updates the training with the document content, or creates a new profile agent for the user.
- **Collaboration.** You can match users that have common agents or similar profiles.
- **Alerting.** When the IDOL Server receives new content that matches user agents, the server immediately notifies the user by email or a third-party system (for example, by SMS or a pager).
- **Mailing.** IDOL Server matches the agents and profiles against its document content at regular intervals, and automatically notifies users of documents that match their agents and profiles by sending them an email message.
- **Expertise.** IDOL Server accepts a natural language or Boolean search string and returns users who own matching agents or profiles. This process allows instant identification of experts in any subjects at hand.

You can create a user structure that is either flat (all users have equal roles and responsibilities in IDOL) or hierarchical (you assign roles to users, which can hierarchically relate to other roles).

Flat Structure

To create a flat user structure, use the `UserAdd` action to create users. For example:

`http://IDOLhost:port/action=UserAdd&UserName=JaneBrown&Password=Sesame`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine where the IDOL Server is installed.
<i>port</i>	is the IDOL Server action port (specified as the <code>Port</code> value in the <code>[Server]</code> section of the IDOL Server configuration file).

Hierarchical Structure

To use a hierarchical user structure, you must create a hierarchical set of roles. You can then assign users to the roles.

To create a hierarchical user structure

1. Decide which groups you want to use to structure your users. You can, for example, group them according to roles and responsibilities.

2. Use the `RoleAdd` action to create a role for each user group. For example:

```
action=RoleAdd&RoleName=Sales
```

3. Use the `RoleAddRoleToRole` action to create a hierarchical structure of roles. For example:

```
action=RoleAddRoleToRole&RoleName=Sales&ParentRoleName=TeleSales
```

4. Use the `UserAdd` action to create individual IDOL users. For example:

```
action=UserAdd&UserName=JaneBrown&Password=Sesame
```

5. Use the `RoleAddUserToRole` action to associate each user with a role. For example:

```
action=RoleAddUserToRole&UserName=JaneBrown&RoleName=TeleSales
```

Create Users in IDOL Admin

As an alternative to using actions, you can use the IDOL Admin interface to create users.

To create users in IDOL Admin

1. In the **Control** menu, click **Users**.
2. Click **+Add**.
The Add User dialog box opens.
3. Type a user name and password. Type the password again in the **Confirm Password** field.
4. Click **Add User**.
The Add User dialog box closes and the user appears in the user list.

Manage Roles in IDOL Admin

As an alternative to submitting actions, you can create roles and assign them to users by using the IDOL Admin interface.

To manage roles in IDOL Admin

1. In the Control menu, click **Roles**.
2. Click **Add Role**.
The Add Role dialog box opens.
3. Type the name of the role that you want to create, then click **Add Role**.
IDOL Admin creates the role and adds it to the list of roles.

4. Select the role that you want to associate with the user, then click **Add User**.
The Add User(s) To Role dialog box opens, displaying a list of IDOL users.
5. Select the check boxes beside the users to add to the role. To select all users, select the check box alongside **User**. The check boxes next to all users are automatically selected.
6. Click **Ok**.
The Add User(s) to Role dialog box closes and the users are added to the user list.

To remove a user from the selected role, click **X** beside the user. The Delete Role User Confirmation dialog box opens; click **Delete** to remove the user from the role.

Integrate with a Third-Party User Structure

The `DeferLogin` option allows you to integrate IDOL Server with a third-party system (such as SiteMinder, Windows NT, LDAP, or Lotus Notes) to manage authentication. The entitlements of the users are set to the ones given to the IDOL Server default (root) role.

To use the `DeferLogin` option

1. Set `DeferLogin` to `True` in the `[Server]` section of the IDOL Server configuration file, and restart IDOL Server.
2. Add `DeferLogin=True` to any user action that you send.

When a user uses IDOL Server for the first time, IDOL Server creates a user with that name and allocates the default role permissions and settings to this user.

Implement User Account Security

IDOL Server provides several security options for user accounts.

- In addition to passwords, you can assign PIN codes to users for authentication with the IDOL Server.
- You can set a minimum length for user names and passwords, and a minimum strength for user passwords.
- You can also specify how often users must change passwords and PIN codes, and ensure that they do not use the same passwords again.
- You can set up the IDOL Server to protect against *brute force attacks*. IDOL Server can lock user accounts if there are too many incorrect logon attempts from a user in a certain time period.

Create User PIN Codes

You can create user PIN codes to use for authentication in addition to passwords. IDOL Server can lock users out if they fail to authenticate using the PIN code.

To configure IDOL Server to use PIN codes

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[User]` section, or create one if it does not exist.

3. Set the `PincodeLength` parameter to the length of PIN code that you want to use. Created PIN codes must have the specified length, and must consist of alphanumeric characters. For example:

```
[User]  
PincodeLength=8
```

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

Add a PIN Code for a User

When you enable PIN codes in the configuration file, you can assign them to users by sending a `UserPin` action.

To add or change a PIN code

- Send the following action to IDOL Server:

```
http://IDOLhost:IDOLport/action=UserPin&UserName=Username&Pincode=PinValue
```

where:

<i>IDOLhost</i>	is the name or IP address of the host on which IDOL Server runs.
<i>IDOLport</i>	is the IDOL Server ACI port.
<i>Username</i>	is the name of the user for which you want to add a PIN code.
<i>PinValue</i>	is the value of the new pin.

Authenticate Users with PIN Codes

You authenticate users with PIN codes by using the `UserPin` action. This action can check certain characters in the PIN code, rather than the whole PIN code.

To authenticate a PIN code

- Send the following action to IDOL Server:

```
http://  
IDOLhost  
:IDOLport/action=UserPin&UserName=Username&Positions=Positions&Values=Values
```

where:

<i>IDOLhost</i>	is the name or IP address of the host on which IDOL Server runs.
<i>IDOLport</i>	is the IDOL Server ACI port.
<i>Username</i>	is the user name of the user whose account you want to authenticate.
<i>Positions</i>	is a comma-separated list of the positions in the PIN code that you want to check.
<i>Values</i>	is a comma-separated list of the values that IDOL Server must check against the specified positions.

For example, to authenticate by using the third, fifth, and sixth characters from a PIN code, you must set `Positions` to 3,5,6. The `Values` parameter then contains the values that the user provides, for example y,9,a. IDOL Server checks that the specified values occur at the specified positions in the PIN code.

Set User Name and Password Restrictions

You can specify certain restrictions on user account names and passwords to ensure that it is difficult to gain unauthorized access.

To configure user account name and password restrictions

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[User]` section, or create one if it does not exist.
3. Set the `MinUsernameLength` parameter to the minimum number of characters allowed for user names. For example:

```
MinUsernameLength=6
```

4. Set the `MinPasswordLength` parameter to the minimum number of characters allowed for passwords. For example:

```
MinPasswordLength=6
```

5. Set the `PasswordStrength` parameter to the minimum allowed strength of passwords. This parameter uses a scale from 1 (weakest) to 10 (strongest). For example:

```
PasswordStrength=6
```

To increase password strength, users must create passwords that contain a mixture of upper- and lowercase letters, numbers, and punctuation.

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

Enable Password and PIN Code Time Restrictions

To help prevent unauthorized access, you can ensure that users change their passwords and PIN codes regularly.

You can ensure that users do not re-use passwords. IDOL Server can store a list of used passwords, and prevent users from re-using a stored password.

IDOL Server can also ensure that users keep passwords for a minimum duration. This option prevents users from immediately changing their password several times to return to a previous password.

Note: You can use the `UserReadUserListDetails` action to check the time remaining (in seconds) until password expiration for a user. If the password has expired, -1 is displayed.

To configure password and PIN code time restrictions

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[User]` section, or create one if it does not exist.
3. Set the `PasswordChangeDuration` parameter to the time interval after which users must change

their password. For example:

```
PasswordChangeDuration=60days
```

4. Set the `PincodeChangeDuration` parameter to the time interval after which users must change their PIN code. For example:

```
PincodeChangeDuration=60days
```

5. Set the `MaxNumPasswordPerUser` parameter to the number of passwords that you want to store in IDOL Server. Users cannot change their passwords to any of the stored passwords. For example:

```
MaxNumPasswordPerUser=5
```

6. Set the `KeepPasswordDuration` parameter to the length of time that users must keep a password before they are allowed to change it again. For example:

```
KeepPasswordDuration=5days
```

7. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
8. Notify your users of your password and PIN policies.

Set Maximum Login Attempts

To protect against brute force attacks on user accounts, you can configure IDOL Server to lock user accounts when there are too many incorrect login attempts within a specified time period.

To set a maximum number of login attempts

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[User]` section, or create one if it does not exist.
3. Set the `LoginMaxAttempts` parameter to the maximum number of incorrect login attempts to allow in the time period.
4. Set the `LoginExpiryTime` parameter to the time (in seconds) before the current number of login attempts resets. IDOL Server locks the user account if there are too many incorrect login attempts within this time period. For example:

```
LoginMaxAttempts=3  
LoginExpiryTime=60
```

In this example, the user account locks if there are three incorrect login attempts within 60 seconds of each other.

5. To automatically unlock users, set the `LockRemovalDuration` parameter to the length of time that the user remains locked. For example:

```
LockRemovalDuration=24hours
```

Set `LockRemovalDuration` to `-1` to disable it.

6. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
7. Notify your users of your password and PIN policies.

Users must contact a system administrator to unlock their accounts, unless you configure the `LockRemovalDuration` parameter.

Lock and Unlock User Accounts

If genuine users are locked out, you can reinstate them by using a `UserLock` action. You can also use this action to lock user accounts manually.

Tip: You can also click **Lock** or **Unlock** in the User Details page in the IDOL Admin interface to lock and unlock users.

To unlock a user account

- Send the following action to IDOL Server:

```
http://IDOLhost:IDOLport/action=UserLock&UserName=Username&Unlock=True
```

where:

<i>IDOLhost</i>	is the name or IP address of the host on which IDOL Server runs.
<i>IDOLport</i>	is the IDOL Server ACI port.
<i>Username</i>	is the user name of the user whose account you want to unlock.

To lock a user account

- Send the following action to IDOL Server:

```
http://IDOLhost:IDOLport/action=UserLock&UserName=Username&Unlock=False
```

where:

<i>IDOLhost</i>	is the name or IP address of the host on which IDOL Server runs.
<i>IDOLport</i>	is the IDOL Server ACI port.
<i>Username</i>	is the user name of the user whose account you want to lock.

Chapter 21: Mail

IDOL Server matches user agents and subscription channels against its document content at regular intervals. It automatically sends users email to notify them of documents that match their agents and the channels that they are subscribed to. You determine the format of the emails by using customizable templates.

- [Automatically Email Agent and Channel Results](#) 371
- [Send Custom Emails](#) 373
- [Send Emails in Batches](#) 374
- [Mailer Templates](#) 374

Automatically Email Agent and Channel Results

You can configure IDOL Server to automatically email users the results that their agents and channels produce. You can schedule the emailing of agent and channel results and optionally store lists of sent results to prevent the duplication of email.

Note: IDOL Server is configured to email agent and channel results to users by default.

To set up automatic email of agent and channel results

1. Open the IDOL Server configuration file in a text editor.
2. Find the [UserCustom] section. This section lists all the custom processes that IDOL Server runs.
3. Check whether the [UserCustom] section lists a section for emailing. If it does not, add one. For example:

```
[UserCustom]
0=Email
```

4. Create a configuration file section for the emailing process that you listed. For example:

```
[Email]
```

5. In your new section, set `Library` to `InstallDir/community/modules/user_email`. Set `RunMailer` to `True` and `DefaultSendEmail` to `True` to enable the IDOL Server mailing operation.
6. Specify a `TestUser`. While you are configuring mailing, IDOL Server sends all mail to the `TestUser` email address.
7. If you are using a proxy server, specify the `ProxyHost`, `ProxyPort`, your `ProxyUsername`, and your `ProxyPassword`.
8. Use `SMTPHost` and `SMTPPort` to specify the details of your mail server.
9. Use `Cycles` and `Interval` to determine how many times the mailing operation must run, and the time span that you want to elapse between the sending of email. Set `StartTime` to `now`, so that you can test the mailing operation immediately when you start IDOL Server.

10. Set `Retries` to the number of times that IDOL Server attempts to connect to its agent index before it times out. Set `TimeoutMS` to how long each of these attempts can take.
11. Use `From`, `FromHost`, and `FromName` to set the details to display as the sender of email that the mailing operation sends. You can also use `FromField` and `FromNameField` to specify a user field that contains the sender details.
12. Specify the `DefaultSubject` to display as the mail subject line.
13. Use `XSLTemplate` to specify the template to use for the email. The `DefaultEmailFormat` and `DefaultEmailResultsType` settings allow you to specify the email format and whether to send results individually or in sets.
14. Set `DefaultAddSetToReadDocuments` to `True` to automatically add the documents in the email to the list of documents that the user has viewed. Set `DefaultExcludeReadDocuments` to `True` to exclude documents that the user has recently viewed from the email (so that they receive each result only once).

You must set `DreTemplateReferenceStart` and `DreTemplateReferenceEnd` to ensure that IDOL Server can extract document references and determine if they were viewed.

15. To include channel results in the email that the mailing operation sends, configure the following parameters:
 - `ClassificationServerXSLTemplate`. The template to use to display channel results.
 - `ClassificationServerNumResults`. The maximum number of channel results to include in the email.
 - `ClassificationServerThreshold`. The quality of channel results to include in the email.
 - `ClassificationServerParams`. Parameters that must be included in the channels query that the mailing operation sends to the IDOL Server category index.
 - `ClassificationServerValues`. The values of the specified `ClassificationServerParams` parameters.
 - `ClassificationServerRetries`. The number of times that the mailing operation attempts to connect to the IDOL Server category index.
 - `ClassificationServerTimeout`. Specifies how long each of the `ClassificationServerRetries` can take, before the mailing operation times out.

Note: Users receive channel results only for categories that they subscribe to. You can subscribe a user to one or more categories by sending a `UserEdit` action to IDOL Server. Use the `CategorySubscribe` action parameter to specify the categories whose results you want to mail to the user. (You can unsubscribe a user by using a `UserEdit` action with the `CategoryUnsubscribe` action parameter set to the categories whose results must no longer be included in the email to the user.)

Note: To include channel results from another IDOL Server installation in the emails that the mailing operation sends, use `ClassificationServerHost` and `ClassificationServerPort` to specify the location of that IDOL Server.

16. To minimize the impact that the mailing operation has on your system resources, you can set `SleepBetweenRequests` and `MaxEmailsPerUser` to values that are appropriate for your environment.
17. Save the IDOL Server configuration file and restart IDOL Server. The mailing operation starts immediately because you set `StartTime` to `now`. It sends mail to the `TestUser` address that you specified. Ensure that the mail process works smoothly.
18. Make any adjustments to your settings that you need, then save the configuration file again and restart IDOL Server. You can enable `VerboseLogging` if you experience problems with the mailing operation.

When you are satisfied with the mailing operation:

1. Open the IDOL Server configuration file in a text editor. Find the `[UserCustom]` section.
2. Delete the email address that you specified for `TestUser`, and set `StartTime` to the time when you want the mailing operation to start.
3. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Related Topics

- ["Edit the Configuration File" on page 35](#)

Send Custom Emails

You can configure IDOL Server to send an email that contains details of a specific single document to a user.

To configure IDOL Server to send custom emails

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[UserCustom]` section.

If you already added a custom section to automatically email results to users, the same settings enable the sending of custom emails. If you are using this existing section, ensure that you specify the template to use for custom emails by using the `EmailActionXSLTemplate` parameter. Continue with [Step 7](#).

If you want IDOL Server to send custom emails without enabling automatic emailing of the agent and channel results, specify a new custom section. Continue with [Step 2](#).

Note: For details of configuration parameters, refer to the *IDOL Server Reference*.

3. Add a section to the configuration file with the name that you specified in the `[UserCustom]` section.
4. In your new section, set `Library` to `InstalLDir/community/modules/user_email` to enable the mailing operation.
5. If you are using a proxy server, specify the `ProxyHost`, `ProxyPort`, your `ProxyUsername`, and your `ProxyPassword`.
6. Use `SMTPHost` and `SMTPPort` to specify the details of your mail server.
7. Specify the template to use to create custom emails as the value of the `EmailActionXSLTemplate` parameter.

8. Save and close the configuration file. Restart IDOL Server for your changes to take effect.
9. Send a Custom action to IDOL Server:
 - Set Function to `email`.
 - Set Library to the name of the custom section in the IDOL Server configuration file that sets up the mailing operation.

Refer to the *IDOL Server Reference* for details on the Custom action.

10. The mailing operation uses the template that you specified as the value of the `EmailActionXSLTemplate` parameter to create the email that it sends to the specified user.

Related Topics

- ["Display Online Help" on page 30](#)
- ["Automatically Email Agent and Channel Results" on page 371](#)

Send Emails in Batches

By default, IDOL Server sends emails individually. To speed up the email process, you can configure IDOL Server to send emails in batches. The mailer can continue generating emails while sending batches of emails.

To set up batch emails

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[Email]` section.
3. Add the following configuration parameter:

<code>MaxEmailsToSendBatchProcess</code>	Specify the maximum number of emails to store in the queue before sending as a batch. The default value is <code>20</code> .
--	---

4. You can slow the email process by using the following configuration parameter:

<code>SleepBetweenSentEmails</code>	Specify a sleep time (in milliseconds) after the mailer sends each email. The default value is <code>0</code> .
-------------------------------------	--

For example:

```
[Email]
MaxEmailsToSendBatchProcess=100
SleepBetweenSentEmails=50
```

Mailer Templates

The following table describes the XSL templates that the IDOL Server installation includes for the mailing operation.

For automatically emailing agent and channel results:	
email.xss	The main template that the user_email library uses for results emails. email.xss specifies the overall structure of emails, and includes specific instructions for displaying individual agent results.
channels.xss	The template that the user_email library uses for formatting the channel results that the email.xss template includes.
For sending custom emails:	
ondemand.xss	The template that specifies how to display the e-mails that IDOL Server sends in response to a Custom action.

You can modify these templates to customize email layout.

Edit Templates

The XSL templates use XPath and XSLT to identify and display fields from the XML returned in response to actions sent to IDOL Server.

You identify the XML fields that the template uses to create emails by using the select attribute in the template XSL tags. To identify the XML fields that a template can use, use a Web browser to send the HTTP action for which IDOL Server uses the template to display results. You can then determine available field names from the autn tags in the XML that returns. The action to send depends on the template that you are editing.

Template	Action
email.xss	AgentGetResults
channels.xss	CategoryQuery
ondemand.xss	Custom

For details of how to send these actions, refer to the *IDOL Server Reference*.

Related Topics

- ["Send Custom Emails" on page 373](#)
- ["Display Online Help" on page 30](#)

For example, if you send an AgentGetResults action to IDOL Server, the following XML might return:

```
<?xml version='1.0' encoding='UTF-8' ?>
<autnresponse xmlns:autn='http://schemas.autonomy.com/aci/'>
  <action>AGENTGETRESULTS</action>
  <response>SUCCESS</response>
  <responsedata>
    <autn:agent>
      <autn:aid>2-A2</autn:aid>
      <autn:training />
      <autn:parent>2</autn:parent>
      <autn:agentname>agent21</autn:agentname>
    </autn:agent>
  </responsedata>
</autnresponse>
```

```
<autn:fields>
<retrained>>true</retrained>
<private>>false</private>
<fromdocument>>true</fromdocument>
</autn:fields>
<autn:results>
<autn:numhits>1</autn:numhits>
<autn:hit>

<autn:reference>http://193.115.251.40/ArchiveData/encarta/38000/msdata3
9439.htm</autn:reference>
<autn:id>1254</autn:id>
<autn:section>0</autn:section>
<autn:weight>70.77</autn:weight>

<autn:links>TAPESTRI,REVIV,WEAV,REACH,EUROPEAN,OCCUR,PRACTIC,TRADIT,REM
AIN,EUROP,EXAMPL,WESTERN,ALTHOUGH,EAR</autn:links>
<autn:database>News</autn:database>
<autn:title>Tapestry Tapestry weaving may have been practiced in Europe
as...</autn:title>
<autn:summary>Tapestry Tapestry weaving may have been practiced in
Europe as ... . Tapestry Tapestry weaving may have been practiced in
Europe as early as the 8th century, although no examples remain. Western European tapestry
reached its greatest development between the 14th and 18th centuries. During the
19th and 20th centuries, however, revivals of the tapestry tradition occurred. .
</autn:summary>
<autn:content>
<DOCUMENT>

<DREREFERENCE>http://193.115.251.40/ArchiveData/encarta/38000/msdata
39439.htm</DREREFERENCE>
<DRETITLE>Tapestry Tapestry weaving may have been practiced in
Europe as ... </DRETITLE>
<BLANK />
<IMAGE>archiv</IMAGE>
<PAPER />
<SUMMARY>Tapestry Tapestry weaving may have been practiced in
Europe as early as the 8th century, although no examples remain. Western European
tapestry reached its greatest development between the 14th and 18th
centuries</SUMMARY>
<DOCTYPE>ARCHIVE</DOCTYPE>
<DREDATE>907347778</DREDATE>
<DREDBNAME>ARCHIVE</DREDBNAME>
<DRECONTENT>Tapestry Tapestry weaving may have been practiced in
Europe as early as the 8th century, although no examples remain. Western European
tapestry reached its greatest development between the 14th and 18th centuries.
During the 19th and 20th centuries, however, revivals of the tapestry tradition
occurred. </DRECONTENT>
<autn:content>
```

```
        </autn:hit>  
      </autn:results>  
    </autn:agent>  
  </respondedata>  
</autnresponse>
```

In this example, you can see from the XML that IDOL Server returns that the following fields are available as values for the `select` attribute:

agent	private	section
aid	fromdocument	weight
training	results	links
parent	numhits	database
agentname	hit	title
fields	reference	summary
retrained	id	content

You can include these fields as values in the XSL tags. For example, to display the value of the `<autn:title>` tag for each result document, include these lines in your template:

```
<xsl:for-each select=respondedata/hit">  
<xsl:value-of select="title">  
</xsl:for-each>
```

You must remove the `autn:` part of the tag from the XSL tag that you specify. For example, if the XML that IDOL Server returns contains a tag called `autn:title`, specify the tag as `title` (as in `select="title"`, in the example here).

Chapter 22: Administer IDOL Server

You can administer IDOL Server by performing the tasks described in this section.

- [Enable Configuration Changes](#) 379
- [Delete and Restore Documents](#) 380
- [Locate Duplicate Documents](#) 382
- [Create and Delete Databases](#) 383
- [Expire Documents](#) 388
- [Change Document Metadata](#) 390
- [Change Document Field Values](#) 391
- [Edit the Spelling Correction Cache](#) 399
- [Compact the Data Index](#) 399
- [Initialize the Data Index](#) 401
- [Validate the Data Index](#) 402

Enable Configuration Changes

When you change the IDOL Server configuration file, you must stop and restart IDOL Server so that it reads the new configuration.

To stop and restart IDOL Server

- **For Windows.** Display the Windows Services dialog box, stop the IDOL Server service, then start it again.
- **For UNIX.** Use the `Stop.sh` stop script to stop IDOL Server, then use the `Start.sh` script to start it again. These scripts are supplied with the IDOL Server installation.
- **For any platform.** In the Console page in IDOL Admin, click the **Service Control** tab, then click **Stop Service**. Then manually restart the service by using the start script (UNIX), or the Windows Services dialog box (Windows).

Some configuration changes require you to reindex your data. In these cases, HPE recommends that you use the `DREEXPORTIDX` or `DREEXPORTXML` index actions to export your content to an IDX or XML document. You can then initialize your index, make your configuration changes, and index the IDX or XML file back into IDOL Server.

Related Topics

- ["Export IDX Documents from IDOL Server"](#)
- ["Export XML Documents from IDOL Server"](#)

Delete and Restore Documents

This section describes how to delete documents from the IDOL Server data index, and how to restore documents that were deleted by using the DREDELETEDOC index action.

Delete Documents by Reference

If you identify one or more documents by their reference field, you can delete them from the IDOL Server data index by sending a DREDELETEREF action (case sensitive) from your Web browser:

```
http://
IDOLhost
:indexPort/DREDELETEREF?Docs=docReferences&Field=fields&DREDbName=databaseName
```

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).
<i>docReferences</i>	is a list of the (percent-encoded) references of the documents that you want to delete. If the references contain plus symbols (+) or spaces, percent-encode each reference, then separate multiple references with plus symbols (there must be no space before or after a plus symbol).
<i>fields</i>	<p>is the field that contains the reference specified in the <code>Docs</code> parameter when a document has more than one reference. Separate multiple references with commas or spaces (there must be no space before or after a comma). For example, the following action:</p> <pre>DREDELETEREF?Docs=myref1.txt&Field=REFFIELD1</pre> <p>deletes a document that contains the following references:</p> <pre>#DREFIELD REFFIELD1="myref1.txt" #DREFIELD REFFIELD2="myref2.txt"</pre> <p>but does not delete a document that contains the following references:</p> <pre>#DREFIELD REFFIELD1="myref2.txt" #DREFIELD REFFIELD2="myref1.txt"</pre>
<i>databaseName</i>	<p>is the name of the database that the documents that you want to delete belong to.</p> <p>This parameter is optional. If you do not specify a database, IDOL Server deletes the document from all databases that contain it.</p>

For example:

```
http://12.3.4.56:20001/DREDELETEREF?Docs=http%3A%2F%2Fnews%2Enewssite%2Ecom%2Findex%2Ehtml+http%3A%2F%2Fnews%2Enewssite%2Ecom%2Fcoverstory%2Ehtml
```

This action uses port 20001 to delete the documents with the specified URLs from IDOL Server, which is located on a machine with the IP address 12.3.4.56.

Delete Documents and Ranges of Documents

If you identify individual documents or ranges of documents by their ID, you can delete them from the IDOL Server data index by sending a DREDELETEDOC action (case sensitive) from your Web browser:

`http://IDOLhost:indexPort/DREDELETEDOC?Docs=docIDs`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).
<i>docIDs</i>	<p>is one or more individual document IDs or ranges of document IDs that specify the documents that you want to delete. Use one or a combination of the following two formats:</p> <ul style="list-style-type: none">• <i>docID</i>. Specify the IDs of one or more documents. To specify multiple document IDs, separate them with plus symbols (there must be no space before or after a plus symbol).• <i>range=[min_docID,max_docID]</i>. Enter the document ID of the first and last document in a range of documents that you want to delete. <p>If you want to combine the formats, separate them with plus symbols (there must be no space before or after a plus symbol).</p>

For example:

`http://MyHost:20001/DREDELETEDOC?Docs=3+5+range=[7,10]`

This action uses port 20001 to delete the documents with the DocIDs 3, 5, 7, 8, 9, and 10 from IDOL Server, which is located on a machine with the host name MyHost.

Delete Documents in IDOL Admin

After you use the Test Action tab in the Console page in the Control section of IDOL Admin, you can delete individual documents or sets of documents from the results list in the IDOL Admin interface.

To delete a set of documents in IDOL Admin

1. In the results list, click **Delete these documents**.

The Delete Documents dialog box opens.

2. Click **Delete Documents**.

To delete an individual document in IDOL Admin

1. In the results list, click **Options** next to the document that you want to delete.

The Edit query result dialog box opens.

2. Click **Delete**.

The Delete dialog box opens.

3. Click **Delete**.

Restore Deleted Documents

If you used a DREDELETEDOC action to delete documents from the IDOL Server data index, you can use a DREUNDELETEDOC action (case sensitive) to restore some or all the individual documents that you deleted. You can restore documents only if you have not run a DRECOMPACT index action since they were deleted (DRECOMPACT permanently removes unused documents and space from IDOL Server).

`http://IDOLhost:indexPort/DREUNDELETEDOC?Docs=docIDs`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).
<i>docIDs</i>	<p>is one or more individual document IDs or ranges of document IDs that specify deleted documents that you want to restore. Use one or a combination of the following two formats:</p> <ul style="list-style-type: none">• <i>docID</i>. Specify the IDs of one or more deleted documents. If you want to specify multiple document IDs, separate them with plus symbols (there must be no space before or after a plus symbol).• <code>range=[min_docID,max_docID]</code>. Enter the document ID of the first and last document in a range of deleted documents that you want to restore. <p>If you want to combine the formats, separate them with plus symbols (there must be no space before or after a plus symbol).</p>

For example:

`http://MyHost:20001/DREUNDELETEDOC?Docs=3+5+range=[7,10]`

This action uses port 20001 to restore the documents with the DocIDs 3, 5, 7, 8, 9, and 10 to IDOL Server, which is located on a machine with the host name MyHost.

Locate Duplicate Documents

You can locate duplicate documents in the data index after indexing has taken place by using the DREDUPLICATE index action. This action locates duplicates in a specified subset of the content, and then removes them, tags a field, or moves the duplicate documents to another database.

`http://IDOLhost:indexPort/DREDUPLICATE?ReferenceField=Field&DuplicateAction=Action`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
-----------------	--

<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).
<i>Field</i>	is a <code>ReferenceType</code> field used as the initial determination of whether two documents are a match.
<i>Action</i>	is the action to perform on a duplicate. The following options are available: <ul style="list-style-type: none">• <code>Delete</code>. Deletes all duplicate documents.• <code>Database</code>. Moves all duplicate documents to a database. If you select the <code>Database</code> action, you must specify the database in the <code>Database</code> parameter.• <code>Tag</code>. Tags a specified field in the duplicate document. You must specify the field in the <code>TagField</code> index parameter. You can also specify a value to tag the field with by using the <code>TagValue</code> parameter. If you do not specify a <code>TagValue</code>, the field is tagged with the value <code>1</code>.

For example:

```
http://MyHost:20001/DREDUPLICATE?ReferenceField=DOCUMENT/DRREFERENCE&DuplicateAction=Database&Database=Duplicates
```

This action uses port `20001` to remove duplicates from the IDOL Server that is located on the machine with the host name `MyHost`. IDOL Server uses the `DRREFERENCE` field to identify duplicate documents, and moves them to the `Duplicates` database.

```
http://MyHost:20001/DREDUPLICATE?ReferenceField=DOCUMENT/DRREFERENCE&DuplicateAction=Tag&TagField=DOCUMENT/DRETITLE&TagValue=Duplicate
```

In this example, IDOL Server initially uses the `DRREFERENCE` field to identify the duplicate documents, and then it changes the `DRETITLE` field to the value `Duplicate`.

To prevent IDOL Server from indexing duplicate documents, use the `KillDuplicates` parameter with the `DREADD` and `DREADDATA` index actions.

For details on the other parameters that are available for the `DREDUPLICATE` index action, refer to the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)
- ["Prevent Duplicate Documents" on page 58](#)

Create and Delete Databases

This section describes how to create and delete databases in IDOL Server.

Create a New Database

You can create a new database in IDOL Server by using one of the following methods:

- Send a DRECREATEDBASE index action to IDOL Server.

Note: This action does not complete if the configuration file is read-only.

- Add a database to the IDOL Server configuration file.
- Create a new database in IDOL Admin.

Send a DRECREATEDBASE Index Action

To create a new database by this method, send a DRECREATEDBASE action (case sensitive) from your Web browser:

`http://IDOLhost:indexPort/DRECREATEDBASE?DREDbName=databaseName`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <i>IndexPort</i> in the [Server] section of the IDOL Server configuration file).
<i>databaseName</i>	is the name of the database that you want to create.

For example:

`http://12.3.4.56:20001/DRECREATEDBASE?DREDbName=Archive`

This action uses port 20001 to create a new database named *Archive* on the IDOL Server machine with the IP address 12.3.4.56.

For information about other available parameters, refer to the *IDOL Server Reference*.

Add a Database to the IDOL Server Configuration File

Use the following procedure to add a database to the IDOL Server configuration file.

To add a new database to the IDOL Server configuration file

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Databases] section. This section contains the NumDBs setting, which indicates how many databases IDOL Server currently contains. It also contains a section for each of these databases, with settings that apply to them. The names of the individual database sections use the format *DatabaseN*, where *N* numbers the databases in consecutive order, starting from 0 (zero).

For example:

```
[Databases]  
NumDBs=2
```

```
[Database0]  
Name=News
```

```
[Database1]  
Name=Archive
```

3. Increase the NumDBs setting by one.

For example:

```
[Databases]  
NumDBs=3
```

4. Create a section for the new database. The name of the section must use the format Database*N*, where *N* numbers the databases in consecutive order, starting from 0 (zero).

Use the Name parameter to specify a name for your new database.

For example:

```
[Databases]  
NumDBs=3
```

```
[Database0]  
Name=News
```

```
[Database1]  
Name=Archive
```

```
[Database2]  
Name=myNewDatabase
```

5. Add any other optional parameters for the database. For more information, refer to the *IDOL Server Reference*.
6. Save and close the configuration file. Restart IDOL Server for your changes to take effect.

Create a new database in IDOL Admin

You can add a new database for the Content or Agentstore components, or IDOL Proxy, from the **Databases** page in the Control section of the IDOL Admin interface.

To add a new database to the IDOL Server configuration file

1. Click **+ Add New Database**. The Add New Database dialog box opens.
2. In the **Database Name** box, type a name for the database.
3. Select the **Internal** check box if you want the database to be internal, and therefore hidden.
4. Select the **Read Only** check box if you want to prevent documents being added to the database.
5. Select a priority for the index action. This determines how IDOL Server queues the action.
6. Click **Create**.

Related Topics

- ["Display Online Help" on page 30](#)

Modify a Database Configuration

You can modify the configuration for an IDOL Server database either by updating the parameters for the database in the configuration file, or by using the DREDBSETOPTIONS index action. For more information, refer to the *IDOL Server Reference*.

Alternatively, you can modify the configuration for a database by using the IDOL Admin interface.

To edit a database in IDOL Admin

1. In the **Control** menu, click **Databases**.
2. In the **Database Control** column, click  next to the database that you want to edit.
The Edit Database Properties dialog box opens.
3. Update the settings as required, then click **Edit**.

Delete a Database and All its Documents

You can delete an IDOL Server database and all the documents that it contains either by sending a DREREMOVEDBASE action, or by using the IDOL Admin interface.

Delete a Database and All its Documents by Sending an Action

You can delete an IDOL Server database and all the documents that it contains by sending a DREREMOVEDBASE action (case sensitive) from your Web browser:

`http://IDOLhost:indexPort/DREREMOVEDBASE?DREDbName=databaseName`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as IndexPort in the [Server] section of the IDOL Server configuration file).
<i>databaseName</i>	is the name of the database that you want to delete. The documents in this database are deleted as well.

For example:

`http://MyHost:20001/DREREMOVEDBASE?DREDbName=Archive`

This action uses port 20001 to delete the Archive database and all documents that it contains from the IDOL Server machine with host name MyHost.

Note: This index action does not complete if the configuration file is read-only.

Delete a Database and All its Documents by using IDOL Admin

Use the following procedure to delete a database and all its documents by using the IDOL Admin interface.

To delete a database and all its documents by using IDOL Admin

1. In the **Control** menu, click **Databases**.
2. In the **Database Control** column, click  next to the database that you want to delete.
The Delete Database dialog box opens.
3. Click **Delete**.

Delete All Documents from a Database

You can delete all documents from an IDOL Server database either by sending a DREDELBASE action, or by using the IDOL Admin interface.

Delete All Documents from a Database by Sending an Action

You can delete all documents from an IDOL Server database by sending a DREDELBASE action (case sensitive) from your Web browser:

`http://IDOLhost:indexPort/DREDELBASE?DREDbName=databaseName`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).
<i>databaseName</i>	is the name of the database whose documents you want to delete.

For example:

`http://12.3.4.56:20001/DREDELBASE?DREDbName=Archive`

This action uses port 20001 to delete all documents from the `Archive` database on a machine with the IP address 12.3.4.56.

Note: This index action does not complete if the configuration file is read-only.

Delete All Documents from a Database by using IDOL Admin

Use the following procedure to delete all documents from a database by using the IDOL Admin interface.

To delete all documents from a database by using IDOL Admin

1. In the **Control** menu, click **Databases**.

2. In the **Database Control** column, click  next to the database from which you want to delete documents.

The Delete Documents dialog box opens.

3. Click **Delete**.

Expire Documents

To ensure that the documents in your IDOL Server are current, you can run an expiration operation, which deletes or archives documents that reach a specific age. You can expire documents:

- immediately, by sending a DREEXPIRE action
- at regular intervals, by setting up a scheduled expiration operation

By default, IDOL Server deletes documents when they expire. To archive them instead, enter the name of the database to use for archiving as the value of the `ExpireIntoDatabase` parameter in each of the IDOL Server configuration file database sections.

IDOL Server can read the date that determines when a document expires from:

- a field in the document.
- the expiration time that you set for the database that contains the document.

IDOL Server does not expire the document if it is unable to determine whether a document must expire (for example, because it does not contain the expiration field, and the database has no expiration time).

Set up a Field Process for Expiration Dates

Use the following procedure to set up fields that determine when documents expire.

To set up fields that determine when documents expire

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[FieldProcessing]` section.
3. Add a new field process to the list of field processes that the `[FieldProcessing]` section contains.

For example:

```
[FieldProcessing]
0=IndexFields
1=IndexAndWeightHigher
```

4. To add a new field process, add a new line to the `[FieldProcessing]` section:

```
[FieldProcessing]
0=IndexFields
1=IndexAndWeightHigher
2=ExpireDateFields
```

The listed field processes are numbered in consecutive order, starting from 0 (zero).

5. Create a section for your new field process in the configuration file. Create a property for the new process, and use the `PropertyFieldCSVs` parameter to identify the document fields that determine whether documents expire (a document expires when the time in this field elapses).

For example:

```
[ExpireDateFields]
Property=SetExpireDate
PropertyFieldCSVs=*/DREEXPIRE,*/valid_time
```

6. Create a section for your new property in the configuration file, and set the `ExpireDateType` to `true` to indicate that the associated `PropertyFieldCSVs` fields hold the document expiration date.

To include an extra offset from the expiration date in the field, set `ExpireAfterDelay` to the number of hours (positive or negative) by which to offset the field expiration date.

For example:

```
[SetExpireDate]
ExpireDateType=True
ExpireAfterDelay=12
```

Expire Immediately

To expire documents immediately, send a DREEXPIRE index action (case sensitive) from your Web browser:

```
http://IDOLhost:indexPort/DREEXPIRE
```

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the <code>[Server]</code> section of the IDOL Server configuration file).

For example:

```
http://MyHost:20001/DREEXPIRE
```

This action uses port 20001 to expire the documents from the IDOL Server located on a machine with host name MyHost.

Expire at Regular Intervals

Use the following procedure to expire documents at regular intervals.

To expire documents at regular intervals

1. Open the IDOL Server configuration file in a text editor.
2. Find the `[Schedule]` section. If the configuration file does not contain a `[Schedule]` section, add one.

3. Specify the following parameters in the [Schedule] section:
 - `Expire`. Set to `True` to enable an expiration schedule.
 - `ExpireTime`. Specify the time (hh:mm) at which you want the expiration operation to start.
 - `ExpireInterval`. Specify the number of hours that elapse between individual expiration operations. Set to 0 (zero) to perform the operation daily.

For example:

```
[Schedule]
Expire=True
ExpireTime=00:00
ExpireInterval=24
```

4. Set the following parameter in the individual database sections to specify where to archive the database documents when they expire:
 - `ExpireIntoDatabase`. Specify the name of the database in which to archive expired documents. To delete documents when they expire, do not add this setting.

For example:

```
[News]
ExpireIntoDatabase=Archive
```

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

Change Document Metadata

Send a `DRECHANGEMETA` index action (case sensitive) from your Web browser to change the metadata (index date, expire date, or database) of documents. You can identify individual documents or document ranges by their references or IDs. (You can specify ranges only with IDs.)

```
http://
IDOLhost
:indexPort/DRECHANGEMETA?Type=type&Refs=docReferences&Docs=docIDs&NewValue=value
```

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).
<i>type</i>	is the type of metadata whose value you are changing: <ul style="list-style-type: none">• <code>date</code>. The index date to assign to the documents. The date you specify must be in a format that you set for <code>DateFormatCSVs</code> in the IDOL Server configuration file.• <code>expiredate</code>. The expire date to assign to the documents. The date you specify must be in a format that you set for <code>DateFormatCSVs</code> in the IDOL Server

	<p>configuration file.</p> <ul style="list-style-type: none"> • database. The database to move the documents to.
<i>docReferences</i>	<p>is a list of (percent-encoded) references to the documents whose metadata is to change. If the references contain plus symbols (+) or spaces, percent-encode each reference, then separate multiple references with plus symbols (there must be no space before or after a plus symbol) and percent-encode the whole string again (using Internet Explorer or the ACI client).</p>
<i>docIDs</i>	<p>is one or more individual document IDs or a ranges of document IDs for the documents whose metadata is to change. Use one or a combination of the following formats:</p> <ul style="list-style-type: none"> • <i>docID</i>. Specify the IDs of one or more documents. To specify multiple document IDs, separate them with plus symbols (there must be no space before or after a plus symbol). • <i>range=[min_docID,max_docID]</i>. Enter the document ID of the first and last document in a range of documents. <p>To combine the formats, separate them with plus symbols (there must be no space before or after a plus symbol).</p>
<i>value</i>	<p>is the new metadata value to write into all the specified documents.</p>

For example:

```
http://12.3.4.56:20001/DRECHANGEMETA?Type=database&Docs=3+5+range=[7,10]
&NewValue=Archive
```

This action uses port 20001 to reassign the documents with the IDs 3, 5, 7, 8, 9, and 10 to the Archive database. IDOL Server is located on a machine with the IP address 12.3.4.56.

Note: By default, when you set the *Refs* parameter, IDOL Server searches all document references for the specified value. If your documents have multiple reference fields, and you want to search only one field name, set the *Field* parameter to the name of the reference field that you want to search.

Change Document Field Values

To change field values in documents, use the DREREPPLACE index action, or the IDOL Admin interface.

Related Topics

- ["About Fields" on page 73](#)

Change Document Field Values by Running an Index Action

You can change field values in documents by using the following index action:

```
DREREPPLACE?DatabaseMatch=Databases\n\nData#DREENDDATA\n\n
```

Note: This action requires a POST request method.

where:

<i>Databases</i>	is a list of one or more databases in which to replace the specified data. If you specify multiple databases, separate them with plus symbols (there must be no space before or after a plus symbol).
<i>Data</i>	<p>is a list of field values to replace or delete in IDOL Server. Specify the fields and field values as follows:</p> <pre>#DREALL #DREDBNAME <i>databaseName</i> #DREDOCID <i>id</i> or #DREDOCREF <i>url</i> or #DRESTATEID <i>stateID</i> #DREFIELDNAME <i>fieldName</i> #DREFIELDVALUE <i>fieldValue</i> #DREFIELDVALUEIFNOTFOUND <i>fieldValue</i> #DREDELETFIELDVALUE <i>fieldValue</i> #DREDELETESINGLEFIELDVALUE <i>fieldValue</i> #DREDELETFIELD <i>fieldName</i> #DREFIELDBITOR <i>value</i> (#DREFIELDBITAND <i>value</i> or #DREFIELDBITXOR <i>value</i>)</pre> <p>The data fields are described in the table below.</p>
#DREENDDATA	must exist at the end of an action parameter block to signify the end of the action.

DREREPPLACE Data Fields

Field	Description
#DREALL	Match all documents. This option is restricted only by the DatabaseMatch parameter supplied as part of the DREREPPLACE request, or by database read-only status.
#DREDBNAME <i>databaseName</i>	The databases in which to match all documents. To specify more than one database, separate them with plus signs (+). IDOL Server ignores the DatabaseMatch parameter for the documents identified by #DREDBNAME .
#DREDOCID <i>id</i>	The ID of the document that contains the field to change or delete.
#DREDOCREF <i>url</i>	The reference of the document that contains the field to change or delete.
#DRESTATEID <i>stateID</i>	The stored state ID of an array of documents that contains the field to change or delete. The state ID returns from a query where StoreState=True.
#DREFIELDNAME <i>fieldName</i>	The name of the field whose value you want to change or delete. Note that in XML documents,

DREREPLACE Data Fields, continued

Field	Description
	<i>fieldName</i> must be fully qualified (for example, XML/DOCUMENT/MYFIELD).
#DREFIELDVALUE <i>fieldValue</i>	The new value for the field specified by <i>fieldName</i> .
#DREFIELDVALUEIFNOTFOUND <i>fieldValue</i>	The new value for the field specified by <i>fieldName</i> if the <i>fieldName/fieldValue</i> pair does not already exist.
#DREFIELDVALUEIFNOFIELD <i>fieldValue</i>	The new value for the field specified by <i>fieldName</i> if the <i>fieldName</i> does not already exist in the document.
#DREDELETEFIELDVALUE <i>fieldValue</i> #DREDELETENOCASEFIELDVALUE <i>fieldValue</i>	The value that the field specified by <i>fieldName</i> must contain for the field to be deleted. By default, the value matching is not case sensitive (the NOCASE specifier is optional), but you can use #DREDELETECASEFIELDVALUE to match case sensitively.
#DREDELETECASEFIELDVALUE <i>fieldValue</i>	The value (case sensitive) that the field specified by <i>fieldName</i> must contain for the field to be deleted.
#DREDELETESINGLEFIELDVALUE <i>fieldValue</i> #DREDELETESINGLENOCASEFIELDVALUE <i>fieldValue</i>	The value that the field specified by <i>fieldName</i> must contain for IDOL Server to delete it. This option deletes only the first instance of the <i>fieldName/fieldValue</i> pair. By default, the value matching is not case sensitive (the NOCASE specifier is optional), but you can use #DREDELETESINGLECASEFIELDVALUE to match case sensitively.
#DREDELETESINGLECASEFIELDVALUE <i>fieldValue</i>	The value (case sensitive) that the field specified by <i>fieldName</i> must contain for IDOL Server to delete it. This option deletes only the first instance of the <i>fieldName/fieldValue</i> pair.
#DREDELETEWILDFIELDVALUE <i>fieldValue</i> #DREDELETEWILDCASEFIELDVALUE <i>fieldValue</i>	The Wildcard value that the field specified by <i>fieldName</i> must contain for the field to be deleted. This operation matches Wildcards for UTF-8 (that is, ? matches a single UTF-8 character). By default, the value matching is case sensitive (the CASE specifier is optional), but you can use #DREDELETEWILDNOCASEFIELDVALUE to match case insensitively.
#DREDELETEWILDNOCASEFIELDVALUE <i>fieldValue</i>	The Wildcard value (case insensitive) that the field specified by <i>fieldName</i> must contain for the field to

DREREPLACE Data Fields, continued

Field	Description
	be deleted. This operation matches Wildcards for UTF-8 (that is, ? matches a single UTF-8 character).
#DREDELETESINGLEWILDFIELDVALUE <i>fieldValue</i> #DREDELETESINGLEWILDCASEFIELDVALUE <i>fieldValue</i>	The Wildcard value that the field specified by <i>fieldName</i> must contain for the first instance of the field to be deleted. This operation deletes only a single instance of the <i>fieldName/fieldValue</i> pair. It matches Wildcards for UTF-8 (that is, ? matches a single UTF-8 character). By default, the value matching is case sensitive (the CASE specifier is optional), but you can use #DREDELETESINGLEWILDNOCASEFIELDVALUE to match case insensitively.
#DREDELETESINGLEWILDNOCASEFIELDVALUE <i>fieldValue</i>	The Wildcard value (case insensitive) that the field specified by <i>fieldName</i> must contain for the first instance of the field to be deleted. This operation deletes only a single instance of the <i>fieldName/fieldValue</i> pair. It matches Wildcards for UTF-8 (that is, ? matches a single UTF-8 character).
#DREDELETFIELD <i>fieldName</i>	The field to delete. You do not require a #DRE*VALUE parameter.
#DREFIELDBITOR <i>value</i>	Generally a 32-bit decimal integer. <ul style="list-style-type: none"> • If this field is a NumericType field with the NumericIntegerOnly property, the value is a signed 64-bit integer. • If the field is a BitFieldType field, the value must be a hexadecimal value. IDOL Server performs the bitwise OR operation on the field identified by the previous #DREFIELDNAME. For NumericType fields, if the field is not present in the document, it adds a field with value 0000000000 first, and then applies the bit operation.
#DREXMLFIELDVALUE	A block of XML data to insert into an XML document under the previous #DREFIELDNAME.

To specify multiple instances of #DREDOCID, #DREDOCREF, or #DRESTATEID, separate the entries with a percent-encoded plus sign or comma.

For example: #DREDOCID 1,3,7-10 or #DREDOCID REF1+REF2+REF3. You must also set ReplaceAllRefs to True to perform actions on multiple documents.

Note: HPE generally recommends that you do not replace values in Index or ACLType fields,

because IDOL Server must reindex the documents being changed, which can slow the update process. Changes to numerical fields, numerical date fields, or fields with other properties, occur without reindexing, and so occur quickly.

Related Topics

- ["Send Data with a POST Method" on page 51](#)

Change Document Field Values by Using IDOL Admin

In the **Replace** tab on the Console page in the Control section of IDOL Admin, you can use the wizard to change field values or delete fields from indexed documents. For more information, refer to the *IDOL Admin User Guide*.

Caution: You cannot recover your original data after you have replaced or deleted values from indexed documents.

To replace or delete field values in indexed documents

1. On the **Data Source** page, click an option to specify the fields to replace or delete:
 - a. **Build the data to replace or delete manually.** Click this option to add a new data source.
 - b. **UTF-8 encoded local file containing DREREPPLACE parameters.** Click **Choose Files** to browse to a file that already specifies the preceding information.
 - c. **Introduce some data that defines the fields and values to replace or delete.** Copy and paste in text that specifies:
 - i. the documents in which to delete or replace the field values
 - ii. the values to substitute, or whether to delete the field
 - iii. #DREENDDATANOOP to mark the end of the index action parameters
2. Click **Next**.
 - a. If you chose to build the data to replace or delete manually, the Choose Data page opens. You can add a new data source and specify the documents and field values that you want to edit.
 - b. If you chose to replace or delete data from local files, or to copy and paste text that specifies the fields and values to replace or delete, the Choose Database page opens. Select one or more databases in which to change the data. Alternatively, you can read the target database from the data.
3. Click **Next**. The Summary page opens.
4. Select a priority for the DREREPPLACE action, which determines how IDOL Server queues the action. The Summary section displays the action that will be sent to the server. The action is automatically constructed as you complete the wizard. It is shown for reference only; you cannot edit the action directly.
5. Click **Replace**.

IDOL Server performs the replace action on the specified documents. The changes are not visible in index searches until the index cache is flushed to disk. Click **Sync** to flush the index cache.

If you need to edit any of your settings, you can click **Previous** to move back through the pages of the wizard, or click **Reset** to clear all settings and start again from the beginning.

Examples

```
DREREPPLACE?DATABASEMATCH=News+Archive HTTP/1.0\nContent-Length:203\n\n
```

```
#DREDOCID 1\n#DREFIELDNAME Price\n#DREFIELDVALUE 10\n#DREFIELDNAME Color\n#DREFIELDVALUE Red\n
```

```
#DREDOCREF http://www.example.com/example/dynamic/autopage442.shtml\n#DREFIELDNAME Country\n#DREFIELDVALUE UK\n#DREFIELDNAME Region\n#DREFIELDVALUE South East\n#DREFIELDNAME OnSale\n#DREDELETEFIELDVALUE Yes\n
```

```
#DRESTATEID abcdefg-6\n#DREFIELDNAME Fruit\n#DREFIELDVALUEIFNOTFOUND mango\n#DREDELETESINGLEFIELDVALUE apple\n#DREDELETEFIELD XML/DOC/DELETEME\n
```

```
#DREENDDATANOOP\n\n
```

In this example, the DREREPPLACE makes these changes in the News and Archive databases.

- In the document with the ID 1:
 - The value of the Price field changes to 10.
 - The value of the Color field changes to Red.
- In the document with the reference <http://www.example.com/example/dynamic/autopage442.shtml>:
 - The value of the Country field changes to UK.
 - The value of the Region field changes to South East.
 - The field OnSale is removed from the document if it has the value Yes.
- In the documents referenced in the state ID abcdefg-6:
 - The value of the Fruit field changes to mango if the Fruit/mango pair does not already exist.
 - If the Fruit field contains the value apple, a single instance of the Fruit/apple pair is deleted.

- All instances of the XML/DOC/DELETED field are deleted.
- #DREENDDATANOOP signifies the end of the action parameters.

Edit BitField Values

The following example demonstrates how to use a DREREPPLACE action to edit the set information held in BitFieldType fields.

```
DREREPPLACE?ReplaceAllRefs=True HTTP/1.0\nContent-Length:203\n\n
```

```
#DREALL \n#DREFIELDNAME BitField\n#DREFIELDVALUEIFNOFIELD A008\n#DREFIELDBITOR A008\n
```

```
#DREDOCID 1274\n#DREFIELDNAME Workbook\n#DREFIELDBITXOR 98\n
```

```
#DREDBNAME News\n#DREFIELDNAME NewsSets\n#DREFIELDBITAND D9\n
```

```
#DREENDDATANOOP\n\n
```

Note: The BitField, Workbook, and NewsSets fields used in this example must be BitFieldType fields.

In this example, the DREREPPLACE makes these changes.

- In all documents in the IDOL Server:
 - If there is no BitField field, IDOL Server adds one with the value A008.
 - IDOL Server changes the value in the BitField field by performing a bitwise OR operation between the current value of the field and the value A008.

For example, if the current value of the field is 8080:

Hexadecimal	Binary	
8080	1000 0000 1000 0000	
A008	1010 0000 0000 1000	
	1010 0000 1000 1000	The new field value is A088

The result of the bitwise OR is that any binary digits that were set to 1 (indicating that the document was part of the set) are left as they are. In addition, all the documents have the bits set

to 1 for the third, thirteenth, and fifteenth sets, if it was not already set. In this way, all the documents are marked as part of sets 3, 13, and 15.

In this example, the document is marked as part of set 3, while remaining a part of sets 7, 11, 13, and 15.

- In the document with ID 1274:
 - IDOL Server changes the value in the `Workbook` field by performing a bitwise XOR operation between the current value of the field and the value 98. For example, if the current value of the field is 6C:

Hexadecimal	Binary	
6C	0110 1100	
98	1001 1000	
	1111 0100	The new field value is F4

The result of the bitwise XOR is that any binary digits that are set to 1 in either number (but not in both) are set to 1 in the result. If the document was not already part of sets 3, 4, or 7, IDOL Server adds it to those sets. However, if the bit is set to 1 or 0 in both numbers, it is set to 0 in the result. If a document is already part of sets 3, 4, or 7, it is removed from them.

In this example, IDOL Server adds the document to sets 4 and 7. It remains part of sets 2, 5, and 6, but IDOL Server removes it from set 3.

- In all documents in the `News` database:
 - IDOL Server changes the value in the `NewsSets` field by performing a bitwise AND operation between the current value of the field and the value D9.

For example, if the current value of the field is 6C:

Hexadecimal	Binary	
6C	0110 1100	
D9	1101 1001	
	0100 1000	The new field value is 48

The result of the bitwise AND is that any binary digits that are set to 1 in both numbers are set to 1 in the result. If a document is already part of set 0, 3, 4, 6, or 7, it remains a part of those sets, but it is not be added if it was not originally part of those sets. It is also removed from any other sets it was initially part of.

In this example, the document remains in set 3 and set 6, but IDOL Server removes it from sets 2 and 5.

- `#DREENDDATANOOP` signifies the end of the action parameters.

Related Topics

- ["BitFieldType Fields" on page 94](#)

Edit the Spelling Correction Cache

You can use the the DRESPELLCHECKCACHE index action edit entries in the IDOL Server spelling correction cache without stopping the service. This action can edit or remove entries in the cache, or write the cache to disk.

`http://IDOLhost:indexPort/DRESPELLCHECKCACHE?Action=Action`

where:

<i>IDOLhost</i>	is the IP address or host name of the IDOL Server.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).
<i>Action</i>	is the action to perform in the spelling correction cache. The following options are available: <ul style="list-style-type: none">• Update. Add a correction for the spelling, overwriting any current entry. Set <code>Original</code> to the incorrect spelling, and <code>Corrected</code> to the correct spelling for this word. If you do not specify the <code>Corrected</code> parameter, IDOL Server marks the word in the cache as not to be corrected.• Remove. Remove any entry for the specified incorrect spelling. Set the <code>Original</code> parameter to the incorrect spelling to remove.• Clear. Remove all entries from the cache.• Flush. Write the spelling correction cache (<code>prx.db</code>) to disk.

For example:

`http://localhost:9001/DRESPELLCHECKCACHE?Action=Update&Original=Barnana&Corrected=Banana`

This action uses port 9001 to edit the spelling correction cache in the IDOL Server which is located on the local machine. It adds Banana as the correction for Barnana, overwriting any current entry.

`http://localhost:9001/DRESPELLCHECKCACHE?Action=Remove&Original=Barnana`

This action uses port 9001 to edit the spelling correction cache in the IDOL Server which is located on the local machine. It removes any entry for barnana from the cache.

For details on other parameters that are available for the DRESPELLCHECKCACHE index action, refer to the *IDOL Server Reference*.

Related Topics

- ["Display Online Help" on page 30](#)

Compact the Data Index

You can reduce the space that the documents in the IDOL Server data index take up by running a compact operation. In this operation, new documents fill up the space that has been created through

the deletion of documents (similar to defragmentation).

You can compact the IDOL Server data index

- immediately, by sending a DRECOMPACT action.
- at regular intervals, by setting up a scheduled Compact operation.

Note: You can automatically back up the IDOL Server data index whenever you send a DRECOMPACT action. HPE recommends that you back up IDOL Server before you compact it.

Related Topics

- ["Back up the Data Index Automatically" on page 409](#)
- ["Back up the Entire IDOL Server Data Index" on page 407](#)

Compact the Data Index Immediately

You can compact the data index either by sending an action from your Web browser, or by using the IDOL Admin interface.

To compact the data index by sending an action from your Web browser

Send a DRECOMPACT action (case sensitive) from your Web browser:

`http://IDOLhost:indexPort/DRECOMPACT`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).

For example:

`http://MyHost:20001/DRECOMPACT`

This action uses port 20001 to compact the data content of an IDOL Server that is located on a machine with host name MyHost.

You can restrict compaction to the `NodeTable` (document storage) stage by using the `index` parameter `Type`.

For example:

`http://MyHost:20001/DRECOMPACT?Type=NodeTable`

After you start a DRECOMPACT operation, you can pause the index action by sending an `IndexerGetStatus` action with the `IndexAction` parameter set to `pause`. You can also cancel the index action by sending an `IndexerGetStatus` action with the `IndexAction` parameter set to `Cancel`. In this case, the next DRECOMPACT operation that you send resumes where the previous compaction stopped.

To compact the data index by using IDOL Admin

1. In IDOL Admin, go to **Control > Console > Service Control**.
2. Click **Compact**.

A confirmation message appears.

3. Click **Compact**.

You can monitor the progress of the DRECOMPACT action in the Recent Tasks panel.

Compact the Data Index at Regular Intervals

Use the following procedure to compact the data index at regular intervals.

To set up a schedule for compaction

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Schedule] section. If the configuration file does not contain a [Schedule] section, add one.
3. Specify the following parameters in the [Schedule] section:
 - Compact. Set to True to enable a compacting schedule.
 - CompactTime. Specify the time (hh:mm) when you want the Compact operation to start.
 - CompactInterval. Specify the number of hours that elapse between individual compaction operations. Set to 0 (zero) if you want the operation to take place daily.

For example:

```
[Schedule]
Compact=True
CompactTime=00:00
CompactInterval=24
```

Initialize the Data Index

You can discard the data that your IDOL Server data index contains and reset it to the state it was in when you first installed IDOL Server. This operation also clears the contents of the index queue, including any queued index actions. Your configuration file is not reset to its initial state.

You can initialize the data index either by sending an action from your Web browser, or by using the IDOL Admin interface.

To initialize the data index by sending an action from your Web browser

Send a DREINITIAL action (case sensitive) from your Web browser:

```
http://IDOLhost:indexPort/DREINITIAL?
```

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as IndexPort in the [Server] section of the IDOL Server configuration file).

For example:

`http://12.3.4.56:20001/DREINITIAL?`

This action uses port 20001 to reset the data index of an IDOL Server that is located on a machine with the IP address 12.3.4.56 to its original state.

To initialize the data index by using IDOL Admin

1. In the **Service Control** tab in the Console page, click **Initialize IDOL Content**.
A confirmation message appears.
2. Click **Delete it**.

You can monitor the progress of the DREINITIAL action in the Recent Tasks panel.

Validate the Data Index

You can check that your data indexes are correctly populated by validating the index. Validation checks the consistency of each of the subindexes in IDOL (node table, numeric, and so on) and logs information about any issues in the application log.

You can perform validation:

- at any time, using an index action or the IDOL Admin interface.
- on startup, by configuring validation parameters.
- automatically, following a server interruption.

Note: The validation checks can take some time to run, depending on the size of your index.

Validate the Data Index Immediately

You can validate the data index either by sending an action from your Web browser, or by using the IDOL Admin interface.

To validate the data index by sending an action from your Web browser

To validate part of the data index immediately, send a DREVALIDATE index action (case sensitive) from your Web browser:

`http://IDOLhost:indexPort/DREVALIDATE?&Type=type`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the [Server] section of the IDOL Server configuration file).
<i>type</i>	is the type of subindex that you want to validate. You can specify only one subindex for each action. Use one of the following values: <ul style="list-style-type: none">• <code>DiskIndex</code>• <code>NodeTable</code>• <code>Numeric</code>

	<ul style="list-style-type: none">• RefIndex• SecIndex• Unstemmed
--	---

For example:

`http://12.3.4.56:20001/DREVALIDATE?&Type=SecIndex`

This index action uses port 20001 to validate the Content security index files of an IDOL Server that is located on a machine with the IP address 12.3.4.56.

To validate the data index by using IDOL Admin

1. On the **Service Control** tab of the Console page, click **Validate**.
The Validate dialog box opens.
2. Select the check boxes next to the subindexes that you want to validate.
3. If you want to fail validation as soon as an error is found, select the **Fail validation immediately** check box.
4. Select the **Index Action Priority**, which determines how the action is queued.
5. Click **Validate**.

You can use the Recent Tasks panel to monitor the progress of the validation.

Validate Subindexes On Startup

You can configure IDOL Server to validate specified subindexes on startup. You can set the validation configuration parameters in the [Server] section, to specify which subindexes to validate every time you restart IDOL Server.

For more information, refer to the *IDOL Server Reference*.

Validate the Data Index Automatically

When IDOL Server restarts after an indexing interruption, it automatically runs all the internal validation checks on its subindexes.

These checks can take some time to run. If this is not desirable, you can turn off this validation check by setting the `ValidateOnInterruption` configuration parameter to `False`.

```
[Server]  
ValidateOnInterruption=False
```

You can also use the `ValidateOptions` configuration parameter to configure other methods for speeding up the validation process. For more information, refer to the *IDOL Server Reference*.

Repair an Index After Validation Fails

You can regenerate a data index if the validation step fails. There are three ways to regenerate an index:

1. Use the `RegenerateIndex` configuration parameters and restart the server.
2. Use the `DREREGENERATE` index action while the server is running.
3. Use the IDOL Admin interface.

All these methods recreate all the files for a particular data index without reindexing all your content.

Note: You can also use these methods to regenerate an index after you make a field configuration change. However, you cannot use the `RegenerateRefIndex` or `DREREGENERATE RefIndex` option to change the configuration for `ReferenceType` fields.

Regenerate with a Server Restart

The `RegenerateIndex` configuration parameters allow you to recreate the data indexes.

To regenerate an index by using the Regenerate configuration parameters

1. Open the IDOL Server configuration file in a text editor.
2. In the `[Server]` section, add the `RegenerateIndex` configuration parameter for the index that you want to regenerate, and set it to `True`. The following parameters regenerate the indexes that correspond to some of the validation steps.
 - `RegenerateNumericIndex`
 - `RegenerateRefIndex`
 - `RegenerateSecIndex`
 - `RegenerateUnstemmedIndex`

For more information about these parameters, refer to the *IDOL Server Reference*.

Note: If validation fails for the `nodetable` or `diskindex` stages, you must reindex your content to repair the index.

3. Save and close the configuration file. Restart IDOL Server for your changes to take effect. On startup, IDOL Server regenerates the specified index.
4. In the configuration file, delete the `RegenerateIndex` parameter, or set it to `False`. This step ensures that IDOL Server does not waste time regenerating the index again when you next restart IDOL Server.

Regenerate with an Index Action

The `DREREGENERATE` action allows you to recreate the data indexes while the server is running. Use this method if you need IDOL Server to be accessible while you regenerate.

To regenerate the data index immediately, send a `DREREGENERATE` index action (case sensitive) from your Web browser:

```
http://IDOLHost:indexPort/DREREGENERATE?&Type=type
```

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified as <code>IndexPort</code> in the <code>[Server]</code> section of the IDOL Server configuration file).
<i>type</i>	<p>is the type of subindex that you want to regenerate. You can specify multiple indexes in a comma-separated list. The following values are available:</p> <ul style="list-style-type: none">• BitField• Count• Match• Numeric• NumericDate• Parametric• RefIndex• SecIndex• Sort <p>Note: You can also use the <code>DREREGENERATE</code> index action to regenerate the <code>ReferenceMemoryMappedType</code> field index to restore parent-child relationships for documents that were indexed out of order. For more information, refer to the <i>IDOL Server Reference</i>.</p>

For example:

`http://12.3.4.56:20001/DREREGENERATE?&Type=Numeric,NumericDate`

This index action uses port 20001 to regenerate the `NumericType` and `NumericDateType` field indexes of an IDOL Server that is located on a machine with the IP address 12.3.4.56.

Regenerate with IDOL Admin

You can also regenerate the data index by using the **Regenerate** function in the **Service Control** tab in the Console page of the IDOL Admin interface.

To regenerate the data index by using IDOL Admin

1. In the **Service Control** tab in the Console page, click **Regenerate**.
The Regenerate dialog box opens.
2. In the **Type** list, select the type of subindex that you want to regenerate. The following values are available:
 - Bit Field
 - Count Index
 - Match Index
 - Numeric Date

- Numeric Index
 - Parametric
 - Reference Memory Mapped
 - Security Index
 - Sort Field
3. In the **Priority** list, select the priority for the action, which determines how the action is queued.
 4. Click **Regenerate**.

Chapter 23: Back up the IDOL Server

HPE recommends that you back up IDOL Server at regular intervals to ensure that you always have current copies of the data that IDOL Server stores.

- [Back up Content](#)407
- [Archive Index Actions](#) 415
- [Restore Content](#) 416
- [Export Users, Roles, Agents, and Profiles](#) 418
- [Import Users, Roles, Agents, and Profiles](#) 418
- [Back up Categories, Taxonomies, and Cluster Jobs](#) 419
- [Restore Categories, Taxonomies, and Cluster Jobs](#) 420

Back up Content

IDOL Server stores content data in its data index. You can back up the entire data index, or export IDX and XML documents from specific IDOL databases.

Note: To restore backed up content, see "[Restore Content](#)" on page 416.

Back up the Entire IDOL Server Data Index

You can back up the entire IDOL Server data index in several ways:

- Immediately, using a DREBACKUP index action.
- At regular intervals, by setting up a scheduled backup.
- Automatically, whenever you send a DRECOMPACT index action.
- Dynamically, by taking a snapshot of the data.

Related Topics

- ["Back up the Data Index Immediately" below](#)
- ["Back up the Data Index at Regular Intervals" on the next page](#)
- ["Back up the Data Index Automatically" on page 409](#)
- ["Back up the Data Index Dynamically" on page 410](#)

Back up the Data Index Immediately

Issue a DREBACKUP action (case sensitive) from your Web browser to copy all the IDOL Server data index *.DB files to a new location:

`http://IDOLhost:indexPort/DREBACKUP?path`

where:

<i>IDOLhost</i>	is the IP address or host name of the IDOL Server machine.
<i>indexPort</i>	is the IDOL Server index port (specified by the <i>IndexPort</i> parameter in the [Server] section of the IDOL Server configuration file).
<i>path</i>	is the path to the location where you want to create the IDOL Server backup. In Windows, you can use a Unicode path.

For example:

`http://MyHost:20001/DREBACKUP?E:\Backup`

This action uses port 20001 to create a backup of the IDOL Server data index on E:\Backup. The IDOL Server whose data index is backed up is located on a machine with the host name MyHost.

You can set the optional parameter *CheckIndexUpdates* to *True* if you want to check for updates to the index since the last backup. In this case, IDOL Server skips the backup if the index has not been modified.

DREBACKUP also accepts the optional parameter *HostDetails*. Set *HostDetails* to *True* to write the backup to a subdirectory in the specified directory path. IDOL Server names this directory using its host and port. In this way, a series of servers under a Distributed Index Handler (DIH) can all accept the same directory as input, and they write to a subdirectory named with their own host and port.

Back up the Data Index at Regular Intervals

Use the following procedure to back up the data index at regular intervals.

To back up the data index at regular intervals

1. Open the IDOL Server configuration file in a text editor.
2. Find the [Schedule] section. If the configuration file does not contain a [Schedule] section, add one.
3. Specify the following parameters in the [Schedule] section:

<i>Backup</i>	Set to <i>True</i> to enable a schedule backup.
<i>BackupCheckIndexUpdates</i>	Set to <i>True</i> to check for updates to the index before IDOL Server runs a backup. When the index has not changed, IDOL Server skips the scheduled backup.
<i>BackupCompression</i>	Set to <i>True</i> to compress the IDOL Server data index before the backup.
<i>BackupTime</i>	Specify the time (hh:mm) when you want the backup to start.
<i>BackupInterval</i>	Specify the number of hours that elapse between individual backups. Set to 0 (zero) to back up daily.
<i>BackupDirN</i>	Set to the path to the location where you want to create the data index backup. You must specify one directory for each of the

	NumberOfBackups.
BackupMaintainDirStructure	Set to True to maintain the directory structure when the data index is backed up.
NumberOfBackups	Specify the number of times that you want to back up the IDOL Server data index. You can cycle the backing up procedure by specifying multiple backups (the number of backups you specify must correspond to the number of BackupDirN directories that you specify). IDOL Server runs multiple backups in the following way: <ul style="list-style-type: none"> • It creates the first backup at the specified BackupTime. • It creates the next one after the specified BackupInterval, and so on. • After the data index has been backed up as many times as you specified for NumberOfBackups, it overwrites the first backup the next time that the specified BackupInterval elapses. This process means that you always have a current set of data index backups.
BackupRetryAttempts	Specify the number of times that you want IDOL Server to reattempt a backup if it fails. After this number of attempts, IDOL Server cancels the operation.
BackupRetryPause	The number of seconds that you want IDOL Server to wait between backup attempts.

For example:

```
[Schedule]
Backup=True
BackupCompression=True
BackupTime=00:00
BackupInterval=24
BackupMaintainStructure=True
BackupRetryAttempts=3
BackupRetryPause=5
NumberOfBackups=3
BackupDir0=E:\DataIndex_Backup0
BackupDir1=E:\DataIndex_Backup1
BackupDir2=E:\DataIndex_Backup2
```

Back up the Data Index Automatically

Use the following procedure to back up the data index automatically whenever you send a DRECOMPACT action.

To back up the data index automatically

1. Open the IDOL Server configuration file in a text editor.

2. Find the [Schedule] section. If the configuration file does not contain a [Schedule] section, add one.
3. Specify the following parameters in the [Schedule] section:

PreCompactionBackup	<p>Set this parameter to True to perform a backup of key files automatically whenever a DRECOMPACT action is sent (either from a Web browser, or by an IDOL Server schedule). IDOL Server backs up the files before it compacts the data index, so that you can restore them. True is the default value.</p> <p>The backup maintains the IDOL Server directory structure. You can compress the backup by setting BackupCompression to True and setting BackupCompressionLevel.</p>
PreCompactionBackupPath	<p>If you set PreCompactionBackup to True, you can use PreCompactionBackupPath to specify the path to the directory where you want to back up files. By default, it uses the internalbackup directory in the IDOL Server data directory.</p> <p>If IDOL Server shuts down without completing a compaction, it uses the contents of this directory to restore itself.</p>

Back up the Data Index Dynamically

If your IDOL storage is a SAN with disk-snapshot capabilities, you can perform a hot backup (snapshot) of the data index.

To back up the data index dynamically

1. Send the DREFLUSHANDPAUSE action to prepare IDOL Server for the hot backup by flushing all files to disk and pausing indexing:

```
http://IDOLhost:indexPort/DREFLUSHANDPAUSE?
```

2. Note the index ID returned from this action (for example, INDEXID=41); you need this ID later to restart the paused indexing queue.
3. Poll IDOL Server with the IndexerGetStatus action to view the status of the flush and pause process:

```
<autnresponse>
<action>INDEXERGETSTATUS</action>
<response>SUCCESS</response>
<respondedata>
<item>
  <id>41</id>
  <origin_ip>127.0.0.1</origin_ip>
  <received_time>2006/10/06 13:47:15</received_time>
  <start_time>2006/10/06 13:47:16</start_time>
  <end_time>Finished</end_time>
  <duration_secs>0</duration_secs>
  <percentage_processed>0</percentage_processed>
  <status>-16</status>
```

```
<description>Indexing Paused</description>*
<index_command>/DREFLUSHANDPAUSE? HTTP/1.1</index_command>
</item>
</responsedata>
</autnresponse>
```

When the returned status is -16 (Indexing Paused), you can perform the hot backup.

4. Take the snapshot according to the procedures defined by your SAN.
5. After the snapshot completes, restart the indexing queue. Issue another IndexerGetStatus action, this time specifying a Restart action:

```
action=IndexerGetStatus&Index=indexNum&IndexAction=restart
```

where *indexNum* is the index ID (for example, 41) returned from DREFLUSHANDPAUSE.

Export IDX Documents from IDOL Server

You can send a DREEXPORTIDX action (case sensitive) from your Web browser to export IDX documents from one or more IDOL Server databases. (Use DREEXPORTXML to export XML documents.) Use the following syntax:

```
http://
IDOLhost
:
indexPort
/DREEXPORTIDX?FileName=
fileName
&Compress=True|False&DatabaseMatch=
databaseCSV&BatchSize=size&MinDate=minDate&MaxDate=maxDate
```

where:

<i>IDOLhost</i>	The IP address or name of the IDOL Server machine.
<i>indexPort</i>	The IDOL Server index port (the value of IndexPort in the [Server] section of the IDOL Server configuration file).
<i>fileName</i>	The path to the location where you store the IDX files to export. Double-byte file names are acceptable. The path must include a basic file name, which IDOL Server postfixes with incremental numbers and an appropriate extension. If you do not specify a file name, IDOL Server exports the files to the current working directory (IDOLserver\IDOL\content), and IDOL Server creates a file name in the format AUTN-IDX-EXPORT-host-port-date>-time-incrementalNumber.extension.
Compress	Whether to compress the exported files. Set Compress to False if you do not want to compress the files.
<i>databaseCSV</i>	If you do not want to export documents from all IDOL Server databases, enter one or more databases to restrict the export to. If you want to specify multiple databases, separate them with plus symbols, commas, or spaces (there must be no space before or after plus symbols or commas).

<i>size</i>	The maximum number of document sections that you want to export to one IDX file. The default value is 100,000 sections.
<i>minDate</i>	The earliest creation date or time that a document can have to export.
<i>maxDate</i>	The latest creation date or time that a document can have to export.

Note: For additional action parameters, refer to the *IDOL Server Reference*.

Example 1:

```
http://12.3.4.56:20001/DREEXPORTIDX?FileName=/export/data/backup/output&Compress=True&DatabaseMatch=News,Archive&BatchSize=1000&MinDate=01/01/2003&MaxDate=01/01/2004
```

In this example, IDOL Server exports all IDX documents that have dates between the first of January 2003 and the first of January 2004. It exports from the *News* and *Archive* databases to a series of compressed files in the */export/data/backup* directory. It names the files created in this directory *output-0.idx.gz*, *output-1.idx.gz* and so on.

Example 2:

```
http://12.3.4.56:20001/DREEXPORTIDX?
```

In this example, IDOL Server exports all IDX documents to a series of compressed files in the IDOL Server current working directory (*IDOLserver\IDOL\content*). It names the files created in this directory *AUTN-IDX-EXPORT-12.04.2005-02.15.41-0.idx.gz*, *AUTN-IDX-EXPORT-12.04.2005-02.15.41-1.idx.gz* and so on.

Note: IDOL Server does not split the sections of a multiple section document across batches. If the exact *BatchSize* splits a multiple-section document, IDOL Server does not use the exact value.

Note: You do not need to uncompress compressed IDX files before you index them. For example, the action *DREADD?output-0.idx.gz* indexes the *output-0.idx.gz* file correctly without you having to uncompress the file first.

Export XML Documents from IDOL Server

You can send a *DREEXPORTXML* action (case sensitive) from your Web browser to export XML documents from one or more IDOL Server databases (use *DREEXPORTIDX* to export IDX documents):

```
http://  
IDOLhost  
:  
indexPort  
/DREEXPORTXML?FileName=  
fileName  
&Compress=True|False&DatabaseMatch=  
databaseCSV&BatchSize=size&MinDate=minDate&MaxDate=maxDate
```

where:

<i>IDOLhost</i>	The IP address or name of the IDOL Server machine.
<i>indexPort</i>	The IDOL Server index port (the value of <i>IndexPort</i> in the [Server] section of the IDOL Server configuration file).
<i>fileName</i>	The path to the location where you store the XML files to export. Double-byte file names are acceptable. The path must include a basic file name which IDOL Server postfixes with incremental numbers and an appropriate extension. If you do not specify a file name, IDOL Server exports the files to the current working directory (IDOLserver\IDOL\content), and IDOL Server creates a file name in the format <i>AUTN-XML-EXPORT-host-port-date-time-incrementalNumber.extension</i> .
Compress	Whether to compress the exported files. Set <i>Compress</i> to <i>False</i> if you do not want to compress the files.
<i>databaseCSV</i>	One or more databases to export documents from. By default, IDOL Server exports from all databases. To specify multiple databases, separate them with plus symbols, commas, or spaces (there must be no space before or after plus symbols or commas).
<i>size</i>	The maximum number of document sections to export to one IDX file. The default value is 100,000 sections.
<i>minDate</i>	The earliest creation date or time that a document can have to export.
<i>maxDate</i>	The latest creation date or time that a document can have to export.

Note: For additional action parameters, refer to the *IDOL Server Reference*.

Example 1:

```
http://12.3.4.56:20001/DREEXPORTXML?FileName=/export/data/backup/output&Compress=True&DatabaseMatch=News,Archive&BatchSize=1000&MinDate=01/01/2003&MaxDate=01/01/2004
```

In this example, IDOL Server exports all XML documents that have dates between 1 January 2003 and 1 January 2004. It exports from the *News* and *Archive* databases to a series of compressed files in the */export/data/backup* directory. IDOL Server names the files created in this directory *output-0.xml.gz*, *output-1.xml.gz* and so on.

Example 2:

```
http://12.3.4.56:20001/DREEXPORTXML?
```

In this example, IDOL Server exports all XML documents to a series of compressed files in the IDOL Server current working directory (IDOLserver\IDOL\content). It names the files created in this directory *AUTN-XML-EXPORT-12.04.2005-02.15.41-0.xml.gz*, *AUTN-XML-EXPORT-12.04.2005-02.15.41-1.xml.gz* and so on.

Note: IDOL Server does not split multiple-section documents across chunks, so the specified

BatchSize is not used exactly if it splits a multiple-section document.

Note: You do not need to uncompress compressed XML files before you index them. For example, the action `DREADD?output-0.xml.gz` indexes the `output-0.xml.gz` file correctly without you having to uncompress the file first.

Use IDOL Admin to Export Indexed Documents

You can export IDX or XML documents from one or more IDOL Server databases. You can also export the entire data index.

To export indexed documents

1. In the **Control** menu, click **Console**.
2. Click the **Export** tab.
3. In the **Export Location** box, type the path to the directory in which to store the exported documents.
4. In the **Export Index Action** box, select the type of files you want to export—either **IDX** or **XML**. This selects either the `DREEXPORTIDX` or `DREEXPORTXML` action, which is displayed and automatically updated as you fill in the form.
5. Click **Select Databases**, then select the check boxes next to all the databases you want to export documents from. If you do not select any databases, IDOL Server exports documents from all databases.
6. Complete the following fields to specify the export task in detail, including which documents to export.

Index Action Priority	Change or set the priority of the exporting job.
Compress	Select whether to compress the exported files.
Delete Documents	Select whether to delete the documents from IDOL Server after exporting them. (Documents are deleted only if the export is successful.)
Add Host Details	Select whether to add host details to the exported file name.
Batch Size	Type the number of document sections to export to one file. Click X to delete the value.
Max Date	Select the latest creation date and time that a document can have to be exported. Click Now in the calendar to automatically select the current date and time. Click X to delete the value.
Min Date	Select the earliest creation date and time that a document can have to be exported. Click Now in the calendar to automatically select the current date and time. Click X to delete the value.

Max ID	Type the maximum document ID number (inclusive) to export. Click X to delete the value.
Min ID	Type the minimum document ID number (inclusive) to export. Click X to delete the value.
Match ID	If you do not want to export all documents, type the IDs of documents to export. Separate multiple IDs with spaces or plus symbols. There must be no space before or after a plus symbol. You can also specify a range of IDs, using a hyphen. Click X to delete all values.
Match Reference	If you do not want to export all IDX documents, type the references of the documents to export. If the references contain plus symbols (+) or spaces, percent-encode each reference, then separate multiple references with plus symbols (there must be no space before or after a plus symbol) and percent-encode the whole string again (using Internet Explorer or the ACI client). Click X to delete all values.
State Match ID	Type the state token of a list of documents to export. If you specify the token name only, all documents listed in the token are exported. If you add a (zero-based) index range, or individual numbers separated by plus symbols (+), in square brackets after the token name, only that range or set of documents are exported. Click X to delete all values.

7. Click **Export** to export the documents to the specified location.

Archive Index Actions

You can configure IDOL Server to store a record of all index actions that it runs. You can use the archive of index actions to replay index actions, for example after you restore to a backup.

To configure IDOL Server to archive index actions

1. Open the IDOL Server configuration file in a text editor.
2. In the [Paths] section, set the `ArchivePath` parameter to the path where you want to store the archive. For example:

```
[Paths]  
ArchivePath=./archive
```

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

When you have configured `ArchivePath`, IDOL Server saves all index actions to the specified path. It also archives any state tokens that have been referenced in the `DREREPLACE`, `DRECHANGEMETA`, `DREDELETEDOC`, `DREEXPORTIDX`, `DREEXPORTXML`, and `DREEXPORTREMOTE` index actions.

During a `DREINITIAL` index action, IDOL Server stores the contents of the archive directory in a subdirectory, so that you can replay actions after you restore from a backup. If you restore to a time, IDOL Server automatically replays archived index actions between the last backup file and the restore time.

Tip: When you configure an `ArchivePath`, you can optionally stop IDOL Server from indexing a particular index action by using the `NoArchive` index action parameter for any index action.

However, if you use the `NoArchive` parameter for an index action that changes the state of the index, and then use `DREINITIAL` and `RestoreToTime` to restore the index, the restore process does not include these changes, and the final index state might be different.

Related Topics

- ["Restore to a Time"](#)

Restore Content

This section describes how to restore your IDOL Server data index from a backup file or an exported file.

Return a List of Backup Files

To find out what backup files are available for your IDOL Server, you can send the `GetBackupData` action. This action returns a list of the backup files that you can use to restore your index.

`http://IDOLhost:ACIPort/action=GetBackupData`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>ACIPort</i>	is the IDOL Server ACI port (specified by the <code>Port</code> parameter in the <code>[Server]</code> section of the IDOL Server configuration file).

To find out which backup file IDOL Server would use to restore to a particular backup time, you can set the `GetBackupForRestoreTime` parameter to the backup time that you want. In this case, the action returns the single backup file that is used to restore to this time. For a list of available date formats, see ["Date formats " on page 230](#).

Restore from a Backup File

To restore a data index to an IDOL Server, you can send a `DREINITIAL` action (case sensitive) from your Web browser, including the path to a backup file that you created by using the `DREBACKUP` index action:

`http://IDOLhost:indexPort/DREINITIAL?path`

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified by the <code>IndexPort</code> parameter in the <code>[Server]</code> section of the IDOL Server configuration file).
<i>path</i>	is the path to the location of the IDOL Server backup, created by using the <code>DREBACKUP</code>

	index action. In Windows, you can use a Unicode path.
--	---

For example:

```
http://12.3.4.56:20001/DREINITIAL?E:\DataIndex_Backup
```

This action uses port 20001 to restore the files backed up on E:\DataIndex_Backup to an IDOL Server located on a machine with the IP address 12.3.4.56.

If you create a backup by using the `HostDetails` parameter, you can restore these backups by using the `HostDetails` parameter with your `DREINITIAL`:

```
http://IDOLhost:port/DREINITIAL?path&HostDetails=True
```

Note: When you restore from a backup by using `DREINITIAL`, IDOL Server does not reindex the data.

When you want to update a configuration parameter that requires you to reindex your data, you can use `DREEXPORTIDX` or `DREEXPORTXML` to export an IDX or XML file. You can then initialize your index, make your configuration changes, and index the IDX or XML file back into IDOL Server.

Related Topics

- ["Export IDX Documents from IDOL Server"](#)
- ["Export XML Documents from IDOL Server"](#)

Restore to a Time

To restore your index to a particular time, you can send a `DREINITIAL` index action (case sensitive), and set the `RestoreToTime` parameter.

```
http://IDOLhost:indexPort/DREINITIAL?RestoreToTime=Time
```

where:

<i>IDOLhost</i>	is the IP address or host name of the machine on which IDOL Server is installed.
<i>indexPort</i>	is the IDOL Server index port (specified by the <code>IndexPort</code> parameter in the [Server] section of the IDOL Server configuration file).
<i>Time</i>	is the time that you want to restore to. For a list of available time and date formats, see "Date formats " on page 230.

When you restore to a time, IDOL Server:

- restores to the latest backup from before the specified time.
- runs any archived index actions from between the backup time and the specified restore time. If you have not configured an `ArchivePath`, the index remains in the backup state.

To find out which backup file IDOL Server would use to back up to a particular time, use the `GetBackupData` action, with the `GetBackupForRestoreTime` parameter. See ["Return a List of Backup Files" on the previous page.](#)

Restore Exported Content

To restore index data from an exported IDX or XML file, created with the DREEXPORTIDX or DREEXPORTXML index actions, you can index the file by using the DREADD index action.

You can use this method to reindex content after you make a configuration change that requires a reindex.

Related Topics

- ["DREADD: Index IDX and XML Files Directly" on page 44](#)

Export Users, Roles, Agents, and Profiles

You can export IDOL Server users, roles, agents, and profiles to a specified XML file from where they can be imported into an IDOL Server again. For example, you can use this process to transfer IDOL Server to a different platform.

`http://IDOLhost:ACIPort/action=UserExport&ExportFileName=fileName`

where:

<i>IDOLhost</i>	is the IP address or name of the IDOL Server machine.
<i>ACIPort</i>	is the IDOL Server ACI port (the value of <code>Port</code> in the <code>[Server]</code> section of the IDOL Server configuration file).
<i>fileName</i>	is the name of the XML file to export IDOL Server users, roles, agents and profiles to. You must specify the path to the file if the XML file is not stored in the same directory as IDOL Server.

For example:

`http://12.3.4.56:20000/action=UserExport&ExportFileName=MyFile.xml`

This action uses port 20000 to export IDOL Server users, roles, agents, and profiles to the `MyFile.xml` file.

Import Users, Roles, Agents, and Profiles

You can import IDOL Server users, roles, agents, and profiles from a specified XML file (into which you previously exported them from an IDOL Server by using the `UserExport` action). You can use this process, for example, to transfer IDOL Server to a different platform.

`http://
IDOLhost:ACIPort/action=UserImport&ImportFileName=fileName&UserFields=fieldCSV`

where:

<i>IDOLhost</i>	is the IP address or name of the IDOL Server machine.
-----------------	---

<i>ACIPort</i>	is the IDOL Server ACI port (the value of <i>Port</i> in the [Server] section of the IDOL Server configuration file).
<i>fileName</i>	is the name of the XML file that contains the users, roles, agents, and profiles that you want to import. If the XML file is not stored in the same directory as IDOL Server, specify the path to the file as well.
<i>fieldCSV</i>	(optional) allows you to restrict the import of user fields by specifying a Wildcard list of fields to import. If you specify multiple fields, separate them with commas (there must be no space before or after a comma). IDOL Server imports only user fields that match a listed field.

For example:

```
http://12.3.4.56:20000/action=UserImport&ImportFileName=MyFile.xml
```

This action uses port 20000 to import IDOL Server users, roles, agents, and profiles from the MyFile.xml file.

Back up Categories, Taxonomies, and Cluster Jobs

You can back up categories, taxonomies, and cluster jobs (snapshots, cluster results, and spectrographs) either by submitting an action, or by using the IDOL Admin interface.

To back up IDOL Server categories, taxonomies, and cluster jobs by submitting an action

- Run a BackupServer action, with the Path parameter set to the path to the directory where you want to store the backup for your categories, taxonomies, and cluster jobs:

```
http://IDOLhost:Categoryport/action=BackupServer&Path=C:\Backups\Category\
```

where:

<i>IDOLhost</i>	is the IP address or name of the machine on which IDOL Server is installed.
<i>Categoryport</i>	is the IDOL Server ACI port of the Category component. You must send the BackupServer action directly to the Category Component (not to the default IDOL Proxy port).

Note: During the backup process, IDOL Server stores temporary files in the directory that you can configure with the BackupDir configuration parameter. It stores the final backup file in the path that you specify.

Note: While IDOL Server is processing a BackupServer action, you cannot add, change, or remove categories, or run cluster jobs. You can, however, run actions that do not impact the data that is backed up. For example, you can use the ClusterResults, ClusterSGDataServe, CategoryQuery actions.

To back up IDOL Server categories, taxonomies, and cluster jobs by using IDOL Admin

- On the **Category Backup/Restore** tab, click **Backup**.

The Backup maintenance action backs up the category data to the backup directory that you specified by using the BackupDir parameter in the [Server] section of the category.cfg configuration file. The **Backup Action String** field shows the action that is sent to IDOL Server.

Related Topics

- ["Restore Categories, Taxonomies, and Cluster Jobs" below](#)

Restore Categories, Taxonomies, and Cluster Jobs

You can restore backups of categories, taxonomies, and cluster jobs either by running a RestoreServer action, or by using the IDOL Admin interface.

To restore data from a backup by running an action

1. Send the RestoreServer action directly to the Category component (do not send it to the main IDOL Proxy port). Set the FileName parameter to the name of the backup file that you want to restore from. For example:

```
http://localhost:9020/action=RestoreServer&FileName=C:\Backups\Category\mybackup.zip
```

2. Send the CategorySyncCatDRE action to synchronize the restored Category component with the Agentstore component. For example:

```
http://localhost:9020/action=CategorySyncCatDRE
```

Note: While IDOL Server is processing a RestoreServer action, you cannot run any category, cluster, or taxonomy actions.

To restore data from a backup by using IDOL Admin

- On the **Category Backup/Restore** tab in the Console page, click **Restore**.
IDOL Server restores the category data from your most recent backup file, and displays a confirmation message. The **Restore Action String** field shows the action that is sent to IDOL Server.

Related Topics

- ["Back up Categories, Taxonomies, and Cluster Jobs" on the previous page](#)

Chapter 24: Troubleshoot IDOL Server

To troubleshoot IDOL Server, consult the log files. The entries in these files provide details of possible problems or incorrect configurations. They are useful in tracking and fixing, for example, configuration errors.

- [IDOL Server Log Files](#) 421
- [Customize Logging](#) 423
- [Create a Diagnostics Package](#) 424
- [IDOL Statistics Server](#) 425

IDOL Server Log Files

By default IDOL Server creates the following log files:

agentstore_application.log	Logs general application errors, warnings, and information relating to the IDOL Server agent index.
agentstore_index.log	Logs messages relating to the indexing, deletion, and updating of agents.
agentstore_query.log	Logs messages relating to the querying of agents.
application.log	Logs general application errors, warnings, and information relating to IDOL Server indexes.
community_application.log	Logs general application errors, warnings, and information relating to user settings (for example, preferences and authentication).
community_term.log	Logs terms as they are added to profiles and agents.
community_user.log	Logs when users and agents are added or deleted (or when this fails).
category_application.log	Logs general application errors, warnings, and information relating to the IDOL Server category index.
category_category.log	Logs messages relating to category actions that read or manipulate the categories, including errors, warnings, and progress information.
category_cluster.log	Logs messages relating to cluster actions, including errors, warnings, and progress information.
category_schedule.log	Logs messages relating to the running of the analysis schedules that are specified in the configuration file.

category_ taxonomy.log	Logs messages relating to the TaxonomyGenerate action, including errors, warnings, and progress information.
content_ application.log	Logs general application errors, warnings, and information relating to the IDOL Server data index.
content_ index.log	Logs messages relating to the indexing, deletion, and updating of documents.
content_ query.log	Logs messages relating to query processes.
content_ queryterms.log	Logs the query terms.
index.log	Logs the index actions that IDOL Server receives.
mailer.log	Logs messages relating to the mailer and scheduled mail tasks.
query.log	Logs all the requests that IDOL Server receives.
stats_index.log	Logs IDOL Server statistics.

You can use the IDOL Admin interface to view the Request log and any other log streams that are enabled.

To select a log stream to display

1. In the **Monitor** menu, click **Logs**.
2. Click the relevant tab.

If there is more than one log file available for that log stream, a list displays the options.

When you select a log stream, it is generated in the user interface.

Note: IDOL Admin shows only the logs for the component that you are currently connected to.

To update the log stream at any point

- Click **Refresh**. You can also enable Autorefresh by selecting a refresh interval in the **Refresh every [number] seconds** list. If Autorefresh is enabled, only the latest log entries are visible.

Note: You can refresh only the current log file for each log stream. If you are viewing a previous log file (the log name is appended with a date), the refresh options are unavailable.

To search within the displayed log stream

- Type a search term into the box in the top-right corner of the page.
If the term appears in the log stream, the search box turns green and the instances in the text are highlighted. The log stream automatically scrolls to the first instance. If there are no results for the term, the search box turns red.

Warning or error entries in the log streams are written in orange text. The number of warning and error messages in a log stream is displayed beside its tab name. Hover over the number to open either a Warnings in [logstream] window or an Errors in [logstream] window, containing links to the relevant entries in the log stream.

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 IDOL 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 *IDOL 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.

Create a Diagnostics Package

You can automatically generate a .ZIP package that includes files that are useful for troubleshooting. When you need to report a problem to HPE Big Data customer support, create a diagnostics package and send it to help them identify and solve your problem.

You can create a diagnostics package either by sending an action, or by using IDOL Admin.

To create a diagnostics package by sending an action

- Send the `Diagnostic` action to the service port of the component that you want to create the diagnostics package for.
 - Set the `File` parameter to the file name and path for the .ZIP file to create.
 - (Optional) Set the `Logs` parameter to one of the following options to determine the type of logs to include in the diagnostics package:

Current	Include only current logs.
Full	Include all logs, including archives.
None	Do not include logs.

HPE recommends that you include logs in the diagnostics package if you send it to HPE Big Data customer support.

For example:

```
http://localhost:9012/action=Diagnostic&File=C:\Diagnostics\Content_10.2.zip&Logs=Current
```

This action creates a diagnostic .ZIP file for the `Content` component on the local machine with service port 9012.

To create a diagnostics package by using IDOL Admin

1. In the IDOL Admin interface, go to the **Diagnostics** tab on the Console page.
2. In the **File path on IDOL server** box, type a file path on your server to save the diagnostics bundle to. Clicking the icon on the right generates a unique file name using the format `server-install-directory/diagnostics-timestamp.zip`.
3. Select the level of log information to include in the diagnostics bundle:

Current	Include only current logs.
Full	Include all logs, including archives.
None	Do not include logs.

4. Click **Create Diagnostics Bundle** to export the diagnostics ZIP file to the specified location. A confirmation message appears.

IDOL Statistics Server

The IDOL Statistics server monitors interactions between one or more IDOL databases and an end user. Interactions can occur in two ways: a user can send actions directly to an IDOL Server through a Web browser; alternatively, a user can interact with IDOL through a front-end application such as IDOL Search Optimizer. The Statistics server monitors IDOL log files for queries or actions that users send to the database, then uses that data to report statistics.

For more information about Statistics Server, refer to the *Query Manipulation Server Administration Guide*.

Part VI: Appendixes

This section includes the following appendixes:

- "IDOL Server Configuration File"
- "Password Encryption"
- "Languages and Language Files"
- "Manually Create IDX Files "
- "Category XML Format"
- "GetStatus Action Response"
- "Error Codes "

Appendix A: IDOL Server Configuration File

This section describes the IDOL Server configuration file.

- [IDOL Server Configuration File](#)429
- [Configuration File Sections](#) 430

IDOL Server Configuration File

The IDOL Server configuration file contains the parameters that determine how IDOL Server operates. You can modify the configuration parameters to customize IDOL Server. You can find this file in the following directory:

```
InstallDir\idolserver.cfg
```

where *InstallDir* is the installation directory for your IDOL Server.

You can validate the configuration file, or search for text within the configuration file, by using the Config page in the **Monitor** section of the IDOL Admin user interface. However, you cannot edit the configuration file by using IDOL Admin. For more information, refer to the *IDOL Admin User Guide*.

Related Topics

- ["Edit the Configuration File" on page 35](#)

Modify Configuration Parameter Values

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

Note: You must stop and restart IDOL 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="\",<p>

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

Configuration File Sections

The IDOL Server configuration file contains several sections, each of which represent a specific area that you can configure by setting appropriate configuration parameters. For details on all available configuration parameters, refer to the *IDOL Server Reference*.

The configuration file sections for each configuration parameter are listed in the *IDOL Server Reference* under **Configuration Sections**.

Related Topics

- ["Display Online Help" on page 30](#)

The configuration file can contain the following sections:

[ACIEncryption]	[IndexCache]	[Security]
[Agent]	[IndexQueue]	[Server]
[AgentDRE]	[IndexServer]	[Service]
[AnalysisSchedules]	[LanguageTypes]	[SSLOptionN]
[Category]	[License]	[Summary]
[CatDRE]	[Logging]	[Synonym]
[Cluster]	[MemoryCache]	[Taxonomy]
[Community]	[MyProperty]	[TermCache]
[DRE]	[Paths]	[User]
[DataDRE]	[Profile]	[UserCustom]
[Databases]	[ProfileNamedAreas]	[UserSecurity]
[DistributionSettings]	[Role]	[UserSecurityFields]
[DocumentTracking]	[Schedule]	[Viewing]
[FieldProcessing]	[SectionBreaking]	

Note: Some of these sections might not be present in the IDOL Server configuration file immediately after installation. The sections that you require depend on the operations that you need your IDOL Server to carry out.

[ACIEncryption] Section

You can encrypt communications between ACI servers and applications that use the HPE ACI API by configuring an [ACIEncryption] section in each of the application configuration files. For example:

```
[ACIEncryption]
CommsAllowUnencrypted=False
CommsEncryptionType=GSS
ServiceName=kerberos
```

[Agent] Section

The [Agent] section determines how agents operate. For example:

```
[Agent]
DynamicAgentFields=True
DreCombine=Simple
DreSentences=3
DreCharacters=300
DreSummary=Context
DontCopyAgentFields=EmailAddress
AgentIndexFieldCSVs=DRELanguageType
```

Related Topics

- ["Agents" on page 139](#)

[AgentDRE] Section

The [AgentDRE] section contains details of the machine that hosts the IDOL Server agent index. When you distribute IDOL Server across multiple machines, you must configure an [AgentDRE] section in the Community configuration file. IDOL Server stores agents and profiles in its agent index. For example:

```
[AgentDRE]
ACIPort=9002
Host=1.23.45.6
Timeout=5000
```

[AnalysisSchedules] Section

The [AnalysisSchedules] section summarizes the number of classification schedules to run. You must create a subsection to specify details for each of these schedules. You can schedule the following actions:

- ClusterSnapshot
- ClusterCluster
- ClusterSGDataGen
- TaxonomyGenerate

For example:

```
[AnalysisSchedules]  
Number=3
```

```
[AnalysisSchedule0]  
ScheduleStartTime=12:00  
ScheduleInterval=1day  
ScheduleCycles=-1  
ScheduleAction=ClusterSnapshot  
TargetJobname=myjob
```

```
[AnalysisSchedule1]  
ScheduleStartTime=12:15  
ScheduleInterval=1day  
ScheduleCycles=-1  
ScheduleAction=ClusterCluster  
SourceJobName=myjob  
TargetJobName=myjob_clusters  
DoMapping=True
```

```
[AnalysisSchedule2]  
ScheduleStartTime=12:15  
ScheduleInterval=1day  
ScheduleCycles=-1  
ScheduleAction=ClusterCluster  
SourceJobName=myjob  
TargetJobName=myjob_clusters_new  
WhatsNew=True  
Interval=86400
```

Related Topics

- ["Set up Schedules" on page 187](#)

[CatDRE] Section

The [CatDRE] section contains details of the machine that hosts the IDOL Server category index. When you distribute IDOL Server across multiple machines, you must configure a [CatDRE] section in the Category configuration file. IDOL Server stores categories in its category index. For example:

```
[CatDRE]  
ACIPort=9002  
Host=1.23.45.6
```

[Category] Section

The [Category] section contains details for the categories that IDOL Server creates in its category index. For example:

```
[Category]
AuthorizeWithoutUser=True
```

Related Topics

- ["Categorization" on page 143](#)

[Cluster] Section

The [Cluster] section contains the details for clustering. For example:

```
[Cluster]
ResultExpiryDays=30
SnapshotExpiryDays=30
SGExpiryDays=30
DownloadDocAction=DREContents
TitleFromSummary=True
SummaryField=autn:summary
```

Related Topics

- ["Cluster Information" on page 179](#)

[Community] Section

The [Community] section determines how community queries operate. For example:

```
[Community]
DreMinScore=20
DreWeighFieldText=False
ExpandQuery=False
ExpandQueryLog=False
ExpandQueryMinScore=60
ExpandQueryMaxResults=30
ExpandQueryMaxScore=80
```

[DAHEngines] Section

When you run the DAH in a unified IDOL Server configuration, you can configure child servers for DAH and DIH separately. This process allows you increased flexibility when querying and indexing content in remote servers. For example, you can configure different child servers to perform actions or index actions exclusively. For more information, refer to the *Distributed Action Handler Administration Guide*.

Use the [DAHEngines] section to set the total number of IDOL Servers to which the DAH sends actions.

You must create a [DAHEngineN] subsection for each of the child servers that the DAH sends actions to, and specify the configuration details for that server.

For example:

```
[DAEngines]  
Number=2
```

[DAEngineN] Section

In the [DAEngines] section you must create a [DAEngineN] subsection for each IDOL Server that the DAH sends actions to. Each [DAEngineN] subsection specifies the host and ACI port of a child IDOL Server. You must number child IDOL Server sections in consecutive order, starting from 0 (zero).

For example:

```
[DAEngines]  
Number=2
```

```
[DAEngine0]  
Host=localhost  
Port=9000
```

```
[DAEngine1]  
Host=12.3.4.56  
Port=9000
```

[Databases] Section

The [Databases] section lists the databases in which IDOL Server stores its data, and contains a subsection for each of the databases, where you specify settings that apply only to this database. If you index documents in multiple languages, you do not need to create a database for each language. For example:

```
[Databases]  
NumDBs=2
```

```
[Database0]  
Name=News
```

```
[Database1]  
Name=Archive
```

Related Topics

- ["Create and Delete Databases" on page 383](#)

[DataDRE] Section

The [DataDRE] section contains details of the machine that hosts the IDOL Server data index. When you distribute IDOL Server across multiple machines, you must configure a [DataDRE] section in the Category and Community configuration files. For example:

```
[DataDRE]  
Host=7.89.01.2
```

```
ACIPort=6002  
Timeout=5000
```

[DIHEngines] Section

When you run the DIH in a unified IDOL Server configuration, you can configure child servers for DAH and DIH separately. This process allows you increased flexibility when querying and indexing content in remote servers. For example, you can configure different child servers to perform indexing or actions exclusively. For more information, refer to the *Distributed Index Handler Administration Guide*.

Use the [DIHEngines] section to set the total number of IDOL Servers that the DIH sends index actions to.

You must create a [DIHEngineN] subsection for each of the child servers that the DIH sends index actions to. In these sections you specify the server configuration details.

For example:

```
[DIHEngines]  
Number=2
```

[DIHEngineN] Section

In the [DIHEngines] section you must create a [DIHEngineN] subsection for each IDOL Server to which the DIH sends index actions. Each [DIHEngineN] subsection specifies the host and ACI port of a child IDOL Server. You must number child IDOL Server sections in consecutive order, starting from 0 (zero).

For example:

```
[DIHEngines]  
Number=2
```

```
[DIHEngine0]  
Host=dih1.company.com  
Port=16100
```

```
[DIHEngine1]  
Host=12.3.4.56  
Port=16200
```

[DistributionIDOLServers] Section

When you run the DAH and DIH in a unified IDOL Server configuration, use the [DistributionIDOLServers] section to set the total number of IDOL Servers with which the DAH and DIH communicate. Child servers defined in this section handle both ACI actions and index actions. For more information refer to the *Distributed Action Handler Administration Guide* and the *Distributed Index Handler Administration Guide*.

You must create an [IDOLServerN] subsection for each of the IDOL Servers or other child servers that the DAH and DIH communicate with, in which you can specify the server configuration details.

```
[DistributionIDOLServers]  
Number=2
```

[DistributionSettings] Section

Use the [DistributionSettings] section when you are using a unified configuration with an IDOL component. There are no specific configuration options that belong to this section. Enter the options that normally appear in the [Server] section of the *IDOLComponent.cfg* file in a stand-alone configuration of the component. For more information, refer to the *Distributed Action Handler Administration Guide* and the *Distributed Index Handler Administration Guide*.

For example, a unified configuration with the DIH:

```
[DistributionSettings]  
Port=16000  
DIHPort=16001  
IndexClients=*. *.*.*.*  
MirrorMode=True
```

[DocumentTracking] Section

The [DocumentTracking] section contains parameters that enable the tracking of documents through the import and indexing process. For example:

```
[DocumentTracking]  
Backend=IDOL  
DatabaseName=DocTracking  
TargetHost=IDOL1  
TargetPort=9001
```

[DRE] Section

The [DRE] section allows you to list Query, TermGetBest, and TermGetInfo action parameters to enable for agent and profile queries. After you enable the parameters in the configuration file, you can set them for the *DREQueryParameter* in the [Agent] and [Profile] section of the configuration file, or for the *DREQueryParameter* parameter of AgentGetResults and Community actions. For example:

```
[DRE]  
AdditionalDREQueryParameters=Characters,MaxDate  
AdditionalDRETermGetBestParameters=Weights  
AdditionalDRETermGetInfoParameters=OnlyExisting
```

Related Topics

- ["Agents" on page 139](#)
- ["Profiles" on page 191](#)

[FieldProcessing] Section

The [FieldProcessing] section lists the processes to apply to fields, and contains a subsection for each of the processes, in which you define the process. For example:

```
[FieldProcessing]
0=SetIndexFields
1=SetIndexAndWeightHigher
2=SetSectionBreakFields
3=SetDateFields
4=SetDatabaseFields
5=SetReferenceFields
6=SetTitleFields
7=SetHighlightFields
8=SetSourceFields
9=DetectNT_V4Security
10=DetectNotes_V4Security
11=DetectNetware_V4Security
12=DetectExchange_V4Security
13=DetectDocumentum_V4Security
14=HideAutonomyMetaDataField
15=SetNumericFields
16=SetFieldCheckFields

[SetIndexFields]
Property=IndexFields
PropertyFieldCSVs=*/DRECONTENT,*/DRETITLE

[SetIndexAndWeightHigher]
Property=IndexWeightFields
PropertyFieldCSVs=*/SUMMARIES

[SetSectionBreakFields]
Property=SectionFields
PropertyFieldCSVs=*/DRESECTION

[SetDateFields]
Property=DateFields
PropertyFieldCSVs=*/DREDATE,*/DATE

[SetDatabaseFields]
Property=DatabaseFields
PropertyFieldCSVs=*/DREDBNAME,*/DATABASE

[SetReferenceFields]
Property=ReferenceFields
PropertyFieldCSVs=*/DREREFERENCE,*/REFERENCE
```

```
[SetTitleFields]  
Property=TitleFields  
PropertyFieldCSVs=*/DRETITLE,*/TITLE
```

```
[SetHighlightFields]  
Property=HighlightFields  
PropertyFieldCSVs=*/DRETITLE,*/DRECONTENT
```

```
[SetSourceFields]  
Property=SourceFields  
PropertyFieldCSVs=*/DRETITLE,*/DRECONTENT
```

```
[DetectNT_V4Security]  
Property=SecurityNT_V4  
PropertyFieldCSVs=*/SECURITYTYPE  
PropertyMatch=nt
```

```
[DetectNotes_V4Security]  
Property=SecurityNotes_V4  
PropertyFieldCSVs=*/SECURITYTYPE  
PropertyMatch=*notes_v4
```

```
[DetectNetware_V4Security]  
Property=SecurityNetware_V4  
PropertyFieldCSVs=*/SECURITYTYPE  
PropertyMatch=*netware_v4
```

```
[DetectExchange_V4Security]  
Property=SecurityExchange_V4  
PropertyFieldCSVs=*/SECURITYTYPE  
PropertyMatch=*exchange_v4
```

```
[DetectDocumentum_V4Security]  
Property=SecurityDocumentum_V4  
PropertyFieldCSVs=*/SECURITYTYPE  
PropertyMatch=*documentum
```

```
[HideAutonomyMetadataField]  
Property=HideMetaFields  
PropertyFieldCSVs=*/AUTONOMYMETADATA
```

```
[SetNumericFields]  
Property=NumericFields  
PropertyFieldCSVs=*/NONEXISTENT
```

```
[SetFieldCheckFields]  
Property=FieldCheckFields  
PropertyFieldCSVs=*/NONEXISTENT
```

Related Topics

- ["Fields" on page 73](#)

[FlushLock] Section

The [FlushLock] configuration section contains settings that control disk flushing and locking.

Flushing is the process of writing cached data to disk during indexing. It occurs periodically, and it is resource-intensive. It is important to prevent two flushes from occurring simultaneously.

If a lock file exists, a server writes information to it to notify other servers that it is flushing, and other servers wait until it is finished before writing to the lock file and performing their flushes.

You can configure a lock file by using the FlushLockFile configuration parameter in the [Server] section. However, HPE does not recommend that you use a FlushLockFile if you are using networked storage.

For networked storage, you can configure the [FlushLock] and [MyLockServer] sections with the details of Redis servers. In this case, the Content component attempts to contact all the configured Redis servers to obtain a lock, and flushes only if it can obtain a lock from more than half of the Redis servers.

For failover purposes, HPE recommends that you configure at least three Redis servers for locking. In this case, if one of the servers is unavailable, the Content component can still obtain more than half of the locks.

Note: Your lock servers must use Redis version 2.8 or later.

```
[FlushLock]
0=LockServer1
1=LockServer2
2=LockServer3
```

```
[LockServer1]
Host=12.34.56.78
Port=6379
```

```
[LockServer2]
Host=12.34.56.89
Port=6389
```

```
[LockServer3]
Host=12.34.56.12
Port=6389
```

Related Topics

- ["\[MyLockServer\] Sections" on page 444](#)

[IDOLServerM] Section

In the [DistributionIDOLServers] section, you must create an [IDOLServerM] subsection for each IDOL Server that the DAH communicates with. Each [IDOLServerM] subsection specifies the host and ACI port of a child IDOL Server. You must number child IDOL Server sections in consecutive order, starting from 0 (zero).

For example:

```
[DistributionIDOLServers]
Number=2
```

```
[IDOLServer0]
Host=localhost
Port=9000
```

```
[IDOLServer1]
Host=1.23.45.6
Port=9000
```

[IndexCache] Section

The [IndexCache] section contains parameters that determine how much memory IDOL Server uses to cache data for indexing. For example:

```
[IndexCache]
IndexCacheMaxSize=102400
```

[IndexNotify] Section

The [IndexNotify] section contains parameters that control and enable the automatic generation of index job information for a specified host. For example:

```
[IndexNotify]
Host=10.1.1.10
ACIPort=9992
BatchSize=1
BatchTimeout=10000
ConnectRetries=1
ConnectTimeout=5000
```

[IndexQueue] Section

The [IndexQueue] section contains parameters that control the index queue. For example:

```
[IndexQueue]
IndexQueueInitialSize=30000
```

```
IndexQueueMaxHistory=4000  
IndexQueueMaxPendingItems=100
```

[IndexServer] Section

The [IndexServer] section contains settings for the IDOL Server index port. Use this section to set Secure Socket Layer (SSL) communications for the index port. For example:

```
[IndexServer]  
SSLConfig=SSLOption1
```

Related Topics

- ["Set up an SSL Connection" on page 356](#)

[LanguageTypes] Section

The [LanguageTypes] section lists the language types that you want to use. It contains a section for each language listed, in which you configure the parameters that determine how to handle each language. For example:

```
[LanguageTypes]  
DefaultLanguageType=englishUTF8  
DefaultEncoding=UTF8  
LanguageDirectory=C:\IDOLServer\IDOL\langfiles  
0=afrikaans  
1=albanian  
2=arabic  
3=armenian  
4=azeri  
5=basque  
6=belorussian  
7=bengali  
8=bosnian
```

```
[afrikaans]  
Encodings=UTF8:afrikaansUTF8  
IndexNumbers=1
```

```
[albanian]  
EncodingsUTF8:albanianUTF8  
IndexNumbers=1
```

```
[arabic]  
Encodings=ARABIC_ISO:arabicARABIC_ISO,ARABIC:arabicARABIC,UTF8:arabicUTF8  
IndexNumbers=1
```

```
[armenian]  
Encodings=UTF8:armenianUTF8  
IndexNumbers=1
```

```
[azeri]
Encodings=UTF8:azeriUTF8
IndexNumbers=1
```

```
[basque]
Encodings=UTF8:basqueUTF8
Stoplist=basque.dat
IndexNumbers=1
```

```
[belorussian]
Encodings=CYRILLIC:belorussianCYRILLIC,UTF8:belorussianUTF8
IndexNumbers=1
```

```
[bengali]
Encodings=UTF8:bengaliUTF8
IndexNumbers=1
```

```
[bosnian]
Encodings=UTF8:bosnianUTF8
IndexNumbers=1
```

Related Topics

- ["Language Support" on page 99](#)
- ["Languages and Language Files" on page 461](#)

[License] Section

The [License] section contains licensing details, which you must not change. For example:

```
[License]
LicenseServerHost=127.0.0.1
LicenseServerACIPort=20000
LicenseServerTimeout=600000
LicenseServerRetries=1
```

[Logging] Section

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

```
[Logging]
LogArchiveDirectory=C:\idol\IDOLserver\IDOL\logs\archive
LogDirectory=C:\idol\IDOLserver\IDOL\logs
LogTime=True
LogEcho=TRUE
```

```
LogLevel=TRUE  
LogExpireAction=compress  
LogOldAction=move  
LogMaxSizeKbs=10240  
0=ApplicationLogStream  
1=QueryLogStream  
2=IndexLogStream  
3=QueryTermsLogStream  
4=UserLogStream  
5=CategoryLogStream  
6=ClusterLogStream  
7=TaxonomyLogStream  
8=ScheduleLogStream  
9=CommunityTermLogStream
```

```
[ApplicationLogStream]  
LogFile=application.log  
LogTypeCSVs=application
```

```
[QueryLogStream]  
LogFile=query.log  
LogTypeCSVs=query
```

```
[IndexLogStream]  
LogFile=index.log  
LogTypeCSVs=index
```

```
[QueryTermsLogStream]  
LogFile=queryterms.log  
LogTypeCSVs=queryterms
```

```
[UserLogStream]  
LogFile=user.log  
LogTypeCSVs=user
```

```
[CategoryLogStream]  
LogFile=category.log  
LogTypeCSVs=category
```

```
[ClusterLogStream]  
LogFile=cluster.log  
LogTypeCSVs=cluster
```

```
[TaxonomyLogStream]  
LogFile=taxonomy.log  
LogTypeCSVs=taxonomy
```

```
[ScheduleLogStream]  
LogFile=schedule.log
```

```
LogTypeCSVs=schedule
```

```
[CommunityTermLogStream]  
LogFile=term.log  
LogTypeCSVs=term
```

Note: All queries are truncated to 4,000 characters in query logs.

Related Topics

- ["Troubleshoot IDOL Server" on page 421](#)

[MemoryCache] Section

The [MemoryCache] section contains parameters that control caching of specified fields to memory. For example:

```
[MemoryCache]  
EvictWhenFull=True  
MaxSize=1073741
```

[MyLockServer] Sections

The [MyLockServer] configuration sections contain details of your Redis servers to use for flushing and locking.

Flushing is the process of writing cached data to disk during indexing. It occurs periodically, and it is resource-intensive. It is important to prevent two flushes from occurring simultaneously.

If a lock file exists, a server writes information to it to notify other servers that it is flushing, and other servers wait until it is finished before writing to the lock file and performing their flushes.

You can configure a lock file by using the FlushLockFile configuration parameter in the [Server] section. However, HPE does not recommend that you use a FlushLockFile if you are using networked storage.

For networked storage, you can configure the [FlushLock] and [MyLockServer] sections with the details of Redis servers. In this case, the Content component attempts to contact all the configured Redis servers to obtain a lock, and flushes only if it can obtain a lock from more than half of the Redis servers.

For failover purposes, HPE recommends that you configure at least three Redis servers for locking. In this case, if one of the servers is unavailable, the Content component can still obtain more than half of the locks.

Note: Your lock servers must use Redis version 2.8 or later.

```
[FlushLock]  
0=LockServer1  
1=LockServer2  
2=LockServer3
```

```
[LockServer1]
```

```
Host=12.34.56.78  
Port=6379
```

```
[LockServer2]  
Host=12.34.56.89  
Port=6389
```

```
[LockServer3]  
Host=12.34.56.12  
Port=6389
```

Related Topics

- ["\[FlushLock\] Section" on page 439](#)

[*MyProperty*] Sections

The [*MyProperty*] sections list the properties that you created for the processes that you listed in the [FieldProcessing] section. You must create a subsection for each property. Within this section, you set configuration parameters that you apply to associated fields. For example:

```
[IndexFields]  
Index=True
```

```
[IndexWeightFields]  
Index=True  
Weight=2
```

```
[SectionFields]  
SectionBreakType=True
```

```
[DateFields]  
DateType=True
```

```
[DatabaseFields]  
DatabaseType=True
```

```
[ReferenceFields]  
ReferenceType=True  
TrimSpaces=True
```

```
[TitleFields]  
TitleType=True
```

```
[HighlightFields]  
HighlightType=True
```

```
[SourceFields]  
SourceType=True
```

```
[SecurityNT_V4]  
SecurityType=NT_V4
```

```
[SecurityNotes_V4]  
SecurityType=Notes_V4
```

```
[SecurityNetware_V4]  
SecurityType=Netware_V4
```

```
[SecurityExchange_V4]  
SecurityType=Exchange_V4
```

```
[SecurityDocumentum_V4]  
SecurityType=Documentum_V4
```

```
[HideMetaDataFields]  
HiddenType=True  
ACLType=True
```

```
[NumericFields]  
NumericType=True
```

```
[FieldCheckFields]  
FieldCheckType=True
```

Related Topics

- ["Fields" on page 73](#)

[Paths] Section

The [Paths] section contains parameters that allow you to split the database into multiple partitions, and parameters that indicate the location of files that IDOL Server uses. For example:

```
[Paths]  
DyntermPath=./dynterm  
NodetablePath=./nodetable  
RefIndexPath=./refindex  
MainPath=./main  
StatusPath=./status  
UserPath=./users  
Modules=C:\idol\IDOLserver\IDOL\modules  
ClusterDirectory=./cluster  
TaxonomyDirectory=./taxonomy  
CategoryDirectory=./category  
ImExDirectory=./imex  
TemplateDirectory=./templates
```

Related Topics

- ["Store IDOL Server Data Files on Multiple Disks" on page 36](#)

[Profile] Section

The [Profile] section contains parameters that determine how to match profiles. For example:

```
[Profile]
DreCombine=Simple
DreSentences=3
DreCharacters=300
DrePrint=All
DreSummary=Context
DreMaxQueryTerms=20
```

Related Topics

- ["Profiles" on page 191](#)

[ProfileNamedAreas] Section

The [ProfileNamedAreas] section determines the names of the areas that contain the profiles that IDOL Server creates when users read or write documents. For example:

```
[ProfileNamedAreas]
0=default
1=authored
```

[Role] Section

The [Role] section contains parameters that determine the default role and which database a role can access. For example:

```
[Role]
DefaultRolename=everyone
AutoSetDatabases=True
DatabasePrivilege=databases
```

Related Topics

- ["Add Users to IDOL Server" on page 363](#)

[Schedule] Section

The [Schedule] section contains parameters that allow you to schedule when to compact IDOL Server and when to expire documents from databases. For example:

```
[Schedule]
Compact=True
Expire=True
CompactTime=00:00
CompactInterval=672
```

```
ExpireTime=00:00  
ExpireInterval=24
```

Related Topics

- ["Compact the Data Index at Regular Intervals" on page 401](#)

[SectionBreaking] Section

The [SectionBreaking] section contains parameters that determine the size of the sections that IDOL Server divides documents into before it indexes them. For example:

```
[SectionBreaking]  
MinFieldLength=80  
MaxSectionLength=2000
```

[Security] Section

The [Security] section lists the security modules that you are using, and contains a subsection for each of the security modules. Within each subsection, you can specify the settings to apply to each module. For example:

```
[Security]  
SecurityInfoKeys=123456789,144135468,56443234,2000111222  
0=NT_V4  
1=Netware_V4  
2=Notes_V4  
3=Exchange_V4  
4=Documentum_V4
```

```
[NT_V4]  
SecurityCode=1  
Library=C:\idol\IDOLserver\IDOL\modules\mapped_security  
Type=AUTONOMY_SECURITY_V4_NT_MAPPED  
ReferenceField=*/AUTONOMYMETADATA
```

```
[Netware_V4]  
SecurityCode=2  
Library=C:\idol\IDOLserver\IDOL\modules\mapped_security  
Type=AUTONOMY_SECURITY_V4_NETWARE_MAPPED  
ReferenceField=*/AUTONOMYMETADATA
```

```
[Notes_V4]  
SecurityCode=3  
Library=C:\idol\IDOLserver\IDOL\modules\mapped_security  
Type=AUTONOMY_SECURITY_V4_NOTES_MAPPED  
ReferenceField=*/AUTONOMYMETADATA
```

```
[Exchange_V4]  
SecurityCode=4
```

```
Library=C:\idol\IDOLserver\IDOL\modules\mapped_security  
Type=AUTONOMY_SECURITY_V4_EXCHANGE_GRPS_MAPPED  
ReferenceField=*/AUTONOMYMETADATA
```

```
[Documentum_V4]  
SecurityCode=5  
Library=C:\idol\IDOLserver\IDOL\modules\mapped_security  
Type=AUTONOMY_SECURITY_V4_DOCUMENTUM_MAPPED  
ReferenceField=*/AUTONOMYMETADATA
```

Note: Use the `[FieldProcessing]` and `[MyProperty]` sections to identify fields that determine the security type of documents and the processes to apply to these fields or documents.

Note: If you run IDOL Server on a UNIX platform, specify the `LD_LIBRARY_PATH` to ensure that IDOL Server can find the shared objects that it requires to implement security.

Related Topics

- ["Set up Security on Documents" on page 353](#)

[Server] Section

The `[Server]` section contains general parameters. For example:

```
[Server]  
IndexClients=*. *.*.*  
AdminClients=*. *.*.*  
IndexPort=20001  
Port=20000  
Threads=4  
MaxInputString=16000  
DelayedSync=True  
AutoDetectLanguagesAtIndex=True  
XSLTemplates=False  
DateFormatCSVs=SHORTMONTH#SD+#SYYYY,DD/MM/YYYY,YYYY/MM/DD,YYYY-MM-DD  
KillDuplicates=*/DRREFERENCE  
DocumentDelimiterCSVs=*/DOCUMENT  
  
CantHaveFieldCSVs=*/CHECKSUM,*/DREWORDCOUNT,*/DRETYPE,*/IMPORTBODYLEN,*/IMPORTMETAL  
EN,*/IMPORTLINKLEN,*/IMPORTTITLELEN,*/IMPORTQUALITY,*/DREPAGE,*/DREFILENAME,*/dredo  
ctype  
InactiveSchedules=all
```

[Service] Section

The `[Service]` section contains parameters that determine which machines are permitted to use and control the IDOL Server service. For example:

```
[Service]
ServicePort=40010
ServiceControlClients=127.0.0.1
ServiceStatusClients=127.0.0.1
```

[SSLOptionN] Section

The [SSLOptionN] section contains settings that determine incoming or outgoing SSL connections for IDOL Server. For example:

```
[SSLOption0]
SSLMethod=SSLV23
SSLCertificate=host1.crt
SSLPrivateKey=host1.key
```

```
[SSLOption1]
SSLMethod=SSLV23
SSLCertificate=host2.crt
SSLPrivateKey=host2.key
SSLPrivateKeyPassword=sample1XQ
SSLCheckCommonName=True
```

Note: You must create an SSLOption section for each unique value set by the SSLConfig parameter in the [Server] section, or other section.

Related Topics

- ["Set up an SSL Connection" on page 356](#)

[Summary] Section

The [Summary] section contains parameters that determine how to generate summaries. For example:

```
[Summary]
MinWordsPerSentence=10
```

Related Topics

- ["Return Summaries with Query Results" on page 299](#)

[Synonym] Section

The [Synonym] section lists the parameters that determine how IDOL Server handles synonym queries. A synonym query returns results which are conceptually similar to the query terms, or conceptually similar to the synonyms that are available for the query terms. For example:

```
[Synonym]
0=PC_Syn
```

```
[PC_Syn]
```

```
File=myfile.txt  
MaxExpandLevel=1
```

Note: To send synonym queries to IDOL Server, you must also set up a synonym file and add the Synonym action parameter to your query.

Related Topics

- ["Synonym Search" on page 264](#)

[Taxonomy] Section

The [Taxonomy] section contains parameters that determine how to generate taxonomies when you run a TaxonomyGenerate action. For example:

```
[Taxonomy]  
MaxConcepts=100  
RelevanceThreshold=20  
DistributionThreshold=10  
ConceptThreshold=400  
MinConceptOccs=15  
CompoundRelevance=40  
SiblingStrength=20  
MinChildren=1  
OnlyMatchSubset=0  
MaxQNum=5000
```

Related Topics

- ["Create Taxonomies" on page 153](#)

[TermCache] Section

The [TermCache] section contains parameters that determine which query terms to store in memory, and how much memory to allocate for cached query terms.

For example:

```
[TermCache]  
TermCachePersistentKB=100000
```

[User] Section

The [User] section contains parameters that determine how many agents each user can have, and which fields belong to these agents. For example:

```
[User]  
MaxAgents=10  
IndexFieldCSVs=DRELanguageType
```

Related Topics

- ["Add Users to IDOL Server" on page 363](#)

[UserCustom] Section

The [UserCustom] section allows you to add custom functionality to IDOL Server. It lists the functionality that you add, and contains a subsection for each item listed. Within each subsection, you can specify the settings that apply to this functionality (for example, the shared library it uses). For example:

```
[UserCustom]
0=Email

[Email]
Library=C:\IDOLserver\IDOL\modules\user_email
FromHost=127.0.0.1
SMTPHost=smtp.company.com
SMTPPort=25
DrePrint=all
XSLTemplate=C:\IDOLserver\IDOL\templates\email.xss
EmailActionXSLTemplate=C:\IDOLserver\IDOL\templates\ondemand.xss
ClassificationServerXSLTemplate=C:\IDOLserver\IDOL\templates\channels.xss
RunMailer=False
Retries=2
TimeoutMS=15000
StartTime=9:00
Interval=1 day
Cycles=-1
FromName=IdolMailer
DefaultSendEmail=True
DefaultEmailFormat=text/html
DefaultExcludeReadDocuments=True
DefaultAddSetToReadDocuments=True
DefaultSubject=USERNAME 's Results
DefaultMimeVersion=1.0
MaxEmailsPerUser=20
From=user@company.com
```

Related Topics

- ["Mail" on page 371](#)

[UserSecurity] Section

The [UserSecurity] section lists your security repositories, specifies generic parameters for them, and contains a subsection for each listed security repository. Within each subsection, you specify the parameters that apply to that repository.

Note: You can list up to eight security types. Each security type must have a matching

configuration section.

Note: If you rename a security type, IDOL Server treats it as a new security type.

Note: If you remove a security type, IDOL Server leaves the corresponding user security fields intact.

For example:

```
[UserSecurity]
DefaultSecurityType=0
DocumentSecurity=True
SyncRolesFromGroups=False
SecurityUsernameDefaultToLoginUsername=False
0=Autonomy
1=NT
2=Notes
3=LDAP
4=Documentum
5=Exchange
6=Netware

[Autonomy]
Library=C:\idol\IDOLserver\IDOL\modules\user_autnsecurity
EnableLogging=False
DocumentSecurity=False
SecurityFieldCSVs=none

[NT]
CaseSensitiveUserNames=False
CaseSensitiveGroupNames=False
Library=C:\idol\IDOLserver\IDOL\modules\user_ntsecurity
EnableLogging=False
DocumentSecurity=True
V4=True
SecurityFieldCSVs=username, domain
Domain=DOMAIN
DocumentSecurityType=NT_V4
```

[UserSecurityFields] Section

The [UserSecurityFields] section lists the security fields. For example:

```
[UserSecurityFields]
0=username
1=password
2=group
3=domain
```

```
[Notes]
Library=C:\idol\IDOLserver\IDOL\modules\user_notessecurity
EnableLogging=False
NotesAuthURL=http://notesserver/names.nsf
DocumentSecurity=True
CaseSensitiveUserNames=False
CaseSensitiveGroupNames=False
SecurityFieldCSVs=username
DocumentSecurityType=Notes_V4
```

```
[LDAP]
Library=C:\idol\IDOLserver\IDOL\modules\user_ldapsecurity
EnableLogging=False
RDNAttribute=CN
Group=OU=Users,0=Company
LDAPServer=127.0.0.1
LDAPPort=389
FieldCSVs=email,emailaddress,telephone
LDAPAllAttributeValues=True
LDAPAttributeValueSeparatorChar=,
SecurityFieldCSVs=none
DocumentSecurity=False
CaseSensitiveUserNames=False
CaseSensitiveGroupNames=False
```

```
[NT]
CaseSensitiveUserNames=False
CaseSensitiveGroupNames=False
Library=C:\idol\IDOLserver\IDOL\modules\user_ntsecurity
EnableLogging=False
DocumentSecurity=True
V4=True
SecurityFieldCSVs=username,domain
Domain=DOMAIN
DocumentSecurityType=NT_V4
```

```
[Documentum]
DocumentSecurity=True
SecurityFieldCSVs=username
DocumentSecurityType=Documentum_V4
CaseSensitiveUserNames=False
CaseSensitiveGroupNames=False
```

```
[Exchange]
DocumentSecurity=True
V4=False
SecurityFieldCSVs=username,domain
DocumentSecurityType=Exchange_V4
CaseSensitiveUserNames=False
```

```
CaseSensitiveGroupNames=False
```

```
[Netware]  
DocumentSecurity=True  
DocumentSecurityType=Netware_V4  
SecurityFieldCSVs=username  
CaseSensitiveUserNames=False  
CaseSensitiveGroupNames=False
```

[Viewing] Section

The [Viewing] section contains parameters that control the viewing of retrieved document content as formatted HTML. For example:

```
[Viewing]  
//whether caching of previous jobs is done case-sensitively  
CaseSensitiveURLs=True  
//Expire cached jobs older than this  
CacheExpirySeconds=86400  
//List of local directories containing documents that can be viewed.  
ViewLocalDirectoriesCSVs=  
//remove script tags  
StripScript=True  
//rewrite hyperlinks as IDOL actions  
OriginalBaseURL=False
```

Related Topics

- ["View Documents" on page 333](#)

Appendix B: Password Encryption

This appendix describes how to use the `Autpassword` command-line tool to encrypt passwords to use in configuration files.

- [Encrypt Passwords](#) 457

Encrypt Passwords

HPE recommends that you encrypt all passwords that you enter into a configuration file.

Create a Key File

A key file is required to use AES encryption. You can use this key file for password encryption, and you can use it in the `SecurityInfoKeys` configuration parameter to encrypt and decrypt IDOL Server security information.

To create a new key file

1. Open a command-line window and change directory to the IDOL Server installation folder.
2. At the command line, type:

```
autpassword -x -tAES -oKeyFile=./MyKeyFile.ky
```

A new key file is created with the name `MyKeyFile.ky`

Caution: To keep your passwords secure, you must protect the key file. Set the permissions on the key file so that only authorized users and processes can read it. IDOL Server must be able to read the key file to decrypt passwords, so do not move or rename it.

Encrypt a Password

The following procedure describes how to encrypt a password.

To encrypt a password

1. Open a command-line window and change directory to the IDOL Server installation folder.
2. At the command line, type:

```
autpassword -e -tEncryptionType [-oKeyFile] [-cFILE -sSECTION -pPARAMETER]  
PasswordString
```

where:

Option	Description
-t	The type of encryption to use:

Option	Description
<i>EncryptionType</i>	<ul style="list-style-type: none"> • Basic • AES <p>For example: <code>-tAES</code></p> <p>Note: AES is more secure than basic encryption.</p>
<code>-oKeyFile</code>	<p>AES encryption requires a key file. This option specifies the path and file name of a key file. The key file must contain 64 hexadecimal characters.</p> <p>For example: <code>-oKeyFile=./key.ky</code></p>
<code>-cFILE -sSECTION -pPARAMETER</code>	<p>(Optional) You can use these options to write the password directly into a configuration file. You must specify all three options.</p> <ul style="list-style-type: none"> • <code>-c</code>. The configuration file in which to write the encrypted password. • <code>-s</code>. The name of the section in the configuration file in which to write the password. • <code>-p</code>. The name of the parameter in which to write the encrypted password. <p>For example: <code>-c./Config.cfg -sMyTask -pPassword</code></p>
<i>PasswordString</i>	The password to encrypt.

For example:

```
autpassword -e -tBASIC MyPassword
```

```
autpassword -e -tAES -oKeyFile=./key.ky MyPassword
```

```
autpassword -e -tAES -oKeyFile=./key.ky -c./Config.cfg -sDefault -pPassword MyPassword
```

The password is returned, or written to the configuration file.

Decrypt a Password

The following procedure describes how to decrypt a password.

To decrypt a password

1. Open a command-line window and change directory to the IDOL Server installation folder.
2. At the command line, type:

```
autpassword -d -tEncryptionType [-oKeyFile] PasswordString
```

where:

Option	Description
<code>-t</code> <i>EncryptionType</i>	The type of encryption: <ul style="list-style-type: none">• Basic• AES For example: <code>-tAES</code>
<code>-oKeyFile</code>	AES encryption and decryption requires a key file. This option specifies the path and file name of the key file used to decrypt the password. For example: <code>-oKeyFile=./key.ky</code>
<i>PasswordString</i>	The password to decrypt.

For example:

```
autpassword -d -tBASIC 9t3M3t7awt/J8A
```

```
autpassword -d -tAES -oKeyFile=./key.ky 9t3M3t7awt/J8A
```

The password is returned in plain text.

Appendix C: Languages and Language Files

This appendix provides reference information related to languages and language files.

- Supported Languages and Common Encodings 461
- Supported Encodings 505
- TermSize Parameter 507
- Per-Language Sentence-Breaking Files 508
- Stop Word Lists for Supported Languages 509

Supported Languages and Common Encodings

This section lists the languages that IDOL Server supports, and the most common encodings for each language.

You can set all IDOL Server Encodings settings to UTF8 or UCS2. The internal IDOL Server storage encoding is UTF8.

Acehnese

Acehnese		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	ACEHNESE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Afrikaans

Afrikaans		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	AFRIKAANS
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Albanian

Albanian		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	ALBANIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Amharic

Amharic		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	AMHARIC
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Arabic

Arabic ¹		
	Script:	Arabic
	[<i>MyLanguage</i>] section name:	ARABIC
	For encoding:	Set Encodings parameter to:
	Windows-CP1256	ARABIC
	ISO-8859-6	ARABIC_ISO
	UTF-8	UTF8

Armenian

Armenian		
----------	--	--

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	Script:	UTF8
	[<i>MyLanguage</i>] section name:	ARMENIAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Azeri

Azeri		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	AZERI
	For encoding:	Set Encodings parameter to:
	Windows-CP1251 KOI8-R ISO-8859-5 UTF-8	CYRILLIC CYRILLIC_KOI8 CYRILLIC_ISO UTF8

Basque

Basque		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	BASQUE
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

Belorussian

Belorussian		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	BELORUSSIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1251	CYRILLIC

	KOI8-R ISO-8859-5 UTF-8	CYRILLIC_KOI8 CYRILLIC_ISO UTF8
--	-------------------------------	---------------------------------------

Bengali

Bengali		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	BENGALI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Berber

Berber		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	BERBER
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Bihari

Bihari		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	BIHARI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Bikol

Bikol		
	Script:	UTF8

	[<i>MyLanguage</i>] section name:	BIKOL
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Bishnupriya

Bishnupriya		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	BISHNUPRIYA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Bosnian

Bosnian		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	BOSNIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1250 ISO-8859-2 UTF-8	EASTERNEUROPEAN EASTERNEUROPEAN_ISO UTF8

Breton

Breton		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	BRETON
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

Bulgarian

Bulgarian		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	BULGARIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1251 KOI8-R ISO-8859-5 UTF-8	CYRILLIC CYRILLIC_KOI8 CYRILLIC_ISO UTF8

Burmese

Burmese		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	BURMESE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Catalan

Catalan ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	CATALAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

Cebuano

Cebuano		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	CEBUANO
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Cherokee

Cherokee		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	CHEROKEE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Chinese Traditional

Chinese traditional ¹		
	Script:	Big-5
	[<i>MyLanguage</i>] section name:	CHINESE
	For encoding:	Set Encodings parameter to:
	Big-5 UTF-8	CHINESETRADITIONAL UTF8

Chinese Simplified

Chinese simplified ²		
	Script:	GB2312-80

¹The language has stemming embedded in sentence breaking.

²The language has stemming embedded in sentence breaking.

	[<i>MyLanguage</i>] section name:	CHINESE
	For encoding:	Set Encodings parameter to:
	gb2312	CHINESESIMPLIFIED
	UTF-8	UTF8

Chuvash

Chuvash		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	CHUVASH
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Croatian

Croatian		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	CROATIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1250	EASTERNEUROPEAN
	ISO-8859-2	EASTERNEUROPEAN_ISO
	UTF-8	UTF8

Czech

Czech ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	CZECH
	For encoding:	Set Encodings parameter to:
	Windows-CP1250	EASTERNEUROPEAN

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	ISO-8859-2 UTF-8	EASTERNEUROPEAN_ISO UTF8
--	---------------------	-----------------------------

Danish

Danish ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	DANISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

Divehi

Divehi		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	DIVEHI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Dutch

Dutch ²		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	DUTCH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

²A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

English

English ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	ENGLISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Erzya

Erzya		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	ERZYA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Esperanto

Esperanto		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	ESPERANTO
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Estonian

Estonian		
	Script:	Latin

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	[<i>MyLanguage</i>] section name:	ESTONIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1257 ISO-8859-4 UTF-8	NORTHERNEUROPEAN NORTHERNEUROPEAN_ISO UTF8

Ethiopic

Ethiopic		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	ETHIOPIC
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Faroese

Faroese		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	FAROESE
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

Finnish

Finnish ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	FINNISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	UTF-8	UTF8
--	-------	------

French

French ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	FRENCH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Frisian

Frisian		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	FRISIAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Gaelic

Gaelic		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	GAELIC
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

Galician

Galician		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	GALICIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Georgian

Georgian		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	GEORGIAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

German

German ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	GERMAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Gilaki

Gilaki		
--------	--	--

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	Script:	UTF8
	[<i>MyLanguage</i>] section name:	GILAKI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Greek

Greek ¹		
	Script:	Greek
	[<i>MyLanguage</i>] section name:	GREEK
	For encoding:	Set Encodings parameter to:
	Windows-CP1253	GREEK
	ISO-8859-7	GREEK_ISO
	UTF-8	UTF8

Greenlandic

Greenlandic		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	GREENLANDIC
	For encoding:	Set Encodings parameter to:
	Windows-CP1257	NORTHERNEUROPEAN
	ISO-8859-4	NORTHERNEUROPEAN_ISO
	UTF-8	UTF8

Guarani

Guarani		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	GUARANI

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Gujarati

Gujarati		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	GUJARATI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Haitian

Haitian		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	HAITIAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Hausa

Hausa		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	HAUSA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Hawaiian

Hawaiian		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	HAWAIIAN

	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Hebrew

Hebrew ¹		
	Script:	Hebrew
	[<i>MyLanguage</i>] section name:	HEBREW
	For encoding:	Set Encodings parameter to:
	Windows-CP1255	HEBREW
	ISO-8859-8	HEBREW_ISO
	UTF-8	UTF8

Hindi

Hindi		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	HINDI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Hungarian

Hungarian ²		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	HUNGARIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1250	EASTERNEUROPEAN

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set *Stemming* to *False* for this language.

²A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set *Stemming* to *False* for this language.

	ISO-8859-2	EASTERNEUROPEAN_ISO
	UTF-8	UTF8

Icelandic

Icelandic		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	ICELANDIC
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Igbo

Igbo		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	IGBO
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Ilokano

Ilokano		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	ILOKANO
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Indonesian

Indonesian		
	Script:	Latin

	[<i>MyLanguage</i>] section name:	INDONESIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Italian

Italian ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	ITALIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Japanese

Japanese ²		
	Script:	Japanese
	[<i>MyLanguage</i>] section name:	JAPANESE
	For encoding:	Set Encodings parameter to:
	Shift-JIS	SHIFTJIS
	EUC	EUC
	JIS	JIS
	UTF-8	UTF8

Javanese

Javanese		
	Script:	UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

²The language has stemming embedded in sentence breaking.

	[<i>MyLanguage</i>] section name:	JAVANESE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Kalmyk

Kalmyk		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	KALMYK
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Kannada

Kannada		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	KANNADA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Kapampangan

Kapampangan		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	KAPAMPANGAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Kazakh

Kazakh		
	Script:	Cyrillic

	[<i>MyLanguage</i>] section name:	KAZAKH
	For encoding:	Set Encodings parameter to:
	Windows-CP1251	CYRILLIC
	KOI8-R	CYRILLIC_KOI8
	ISO-8859-5	CYRILLIC_ISO
	UTF-8	UTF8

Khmer

Khmer		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	KHMER
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Kikongo

Kikongo		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	KIKONGO
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Kinyarwanda

Kinyarwanda		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	KINYARWANDA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Kirundi

Kirundi		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	KIRUNDI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Komi

Komi		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	KOMI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Korean

Korean ¹		
	Script:	Hangul
	[<i>MyLanguage</i>] section name:	KOREAN
	For encoding:	Set Encodings parameter to:
	KS C 5601-1987	KOREAN
	KS C 5601-1992	KOREAN
	UTF-8	UTF8

Kurdish

Kurdish		
	Script:	Latin

¹The language has stemming embedded in sentence breaking.

	[<i>MyLanguage</i>] section name:	KURDISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Kyrgyz

Kyrgyz		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	KYRGYZ
	For encoding:	Set Encodings parameter to:
	Windows-CP1251	CYRILLIC
	KOI8-R	CYRILLIC_KOI8
	ISO-8859-5	CYRILLIC_ISO
	UTF-8	UTF8

Lao

Lao		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	LAO
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Lappish

Lappish		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	LAPPISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1257	NORTHERNEUROPEAN
	ISO-8859-4	NORTHERNEUROPEAN_ISO

	UTF-8	UTF8
--	-------	------

Latin

Latin ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	LATIN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Latvian

Latvian		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	LATVIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1257	NORTHERNEUROPEAN
	ISO-8859-4	NORTHERNEUROPEAN_ISO
	UTF-8	UTF8

Lingala

Lingala		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	LINGALA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

Lithuanian

Lithuanian		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	LITHUANIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1257 ISO-8859-4 UTF-8	NORTHERNEUROPEAN NORTHERNEUROPEAN_ISO UTF8

Luxembourgish

Luxembourgish		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	LUXEMBOURGISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

Macedonian

Macedonian		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	MACEDONIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1251 KOI8-R ISO-8859-5 UTF-8	CYRILLIC CYRILLIC_KOI8 CYRILLIC_ISO UTF8

Malagasy

Malagasy		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	MALAGASY
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Malay

Malay		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	MALAY
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Malayalam

Malayalam		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	MALAYALAM
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Maltese

Maltese		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	MALTESE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Manipuri

Manipuri		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	MANIPURI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Maori

Maori		
	Script:	Latin1
	[<i>MyLanguage</i>] section name:	MAORI
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Marathi

Marathi		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	MARATHI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Mazandarani

Mazandarani		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	MAZANDARANI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Mirandese

Mirandese		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	MIRANDESE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Mongolian

Mongolian		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	MONGOLIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1251 KOI8-R ISO-8859-5 UTF-8	CYRILLIC CYRILLIC_KOI8 CYRILLIC_ISO UTF8

Nahuatl

Nahuatl		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	NAHUATL
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Navajo

Navajo		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	NAVAJO

	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Ndebele

Ndebele		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	NDEBELE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Nepali

Nepali		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	NEPALI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Newari

Newari		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	NEWARI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Norwegian

Norwegian ¹		
------------------------	--	--

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	Script:	Latin
	[<i>MyLanguage</i>] section name:	NORWEGIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Oriya

Oriya		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	ORIYA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Ossetian

Ossetian		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	OSSETIAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Panjabi

Panjabi		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	PANJABI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Papiamentu

Papiamentu		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	PAPIAMENTU
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Persian

Persian		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	PERSIAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Polish

Polish ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	POLISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1250	EASTERNEUROPEAN
	ISO-8859-2	EASTERNEUROPEAN_ISO
	UTF-8	UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

Portuguese

Portuguese ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	PORTUGUESE
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Pushto

Pushto		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	PUSHTO
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Quechua

Quechua		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	QUECHUA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Rhaeto-Romance

Rhaeto-Romance		
	Script:	UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	[<i>MyLanguage</i>] section name:	RHAETO-ROMANCE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Romanian

Romanian ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	ROMANIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1250 ISO-8859-2 UTF-8	EASTERNEUROPEAN EASTERNEUROPEAN_ISO UTF8

Russian

Russian ²		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	RUSSIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1251 KOI8-R ISO-8859-5 UTF-8	CYRILLIC CYRILLIC_KOI8 CYRILLIC_ISO UTF8

Sakha

Sakha		
-------	--	--

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

²A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SAKHA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Sami

Sami		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SAMI
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Sanskrit

Sanskrit		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SANSKRIT
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Serbian

Serbian		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	SERBIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1251 KOI8-R ISO-8859-5 UTF-8	CYRILLIC CYRILLIC_KOI8 CYRILLIC_ISO UTF8

Sesotho

Sesotho		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SESOTHO
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Sesotho sa Leboa

Sesotho sa Leboa		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SESOTHOALEBOA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Singhalese

Singhalese		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SINGHALESE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Siswant

Siswant		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SISWANT
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Slovak

Slovak ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	SLOVAK
	For encoding:	Set Encodings parameter to:
	Windows-CP1250 ISO-8859-2 UTF-8	EASTERNEUROPEAN EASTERNEUROPEAN_ISO UTF8

Slovenian

Slovenian		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	SLOVENIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1250 ISO-8859-2 UTF-8	EASTERNEUROPEAN EASTERNEUROPEAN_ISO UTF8

Somali

Somali		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	SOMALI
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

Sorbian

Sorbian		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	SORBIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1250 ISO-8859-2 UTF-8	EASTERNEUROPEAN EASTERNEUROPEAN_ISO UTF8

Spanish

Spanish ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	SPANISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1 UTF-8	ASCII UTF8

Sranan

Sranan		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SRANAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Sundanese

Sundanese		

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SUNDANESE
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Swahili

Swahili		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	SWAHILI
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Swedish

Swedish ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	SWEDISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Syriac

Syriac		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	SYRIAC
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

Tagalog

Tagalog		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	TAGALOG
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Tahitian

Tahitian		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TAHITIAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Tajik

Tajik		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	TAJIK
	For encoding:	Set Encodings parameter to:
	Windows-CP1251	CYRILLIC
	KOI8-R	CYRILLIC_KOI8
	ISO-8859-5	CYRILLIC_ISO
	UTF-8	UTF8

Tamil

Tamil		
-------	--	--

	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TAMIL
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Tatar

Tatar		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	TATAR
	For encoding:	Set Encodings parameter to:
	Windows-CP1251 KOI8-R ISO-8859-5 UTF-8	CYRILLIC CYRILLIC_KOI8 CYRILLIC_ISO UTF8

Telugu

Telugu		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TELUGU
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Thai

Thai		
	Script:	Thai
	[<i>MyLanguage</i>] section name:	THAI
	For encoding:	Set Encodings parameter to:
	Windows-CP874/ISO-8859-11 UTF-8	THAI UTF8

Tibetan

Tibetan		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TIBETAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Tokpisin

Tokpisin		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TOKPISIN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Tongan

Tongan		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TONGAN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Tsonga

Tsonga		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TSONGA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Tswana

Tswana		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TSWANA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Turkish

Turkish		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	TURKISH
	For encoding:	Set Encodings parameter to:
	Windows-CP1254/ISO-8859-9	TURKISH
	UTF-8	UTF8

Turkmen

Turkmen		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	TURKMEN
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Ukrainian

Ukrainian		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	UKRAINIAN
	For encoding:	Set Encodings parameter to:

	Windows-CP1251	CYRILLIC
	KOI8-R	CYRILLIC_KOI8
	ISO-8859-5	CYRILLIC_ISO
	UTF-8	UTF8

Urdu

Urdu		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	URDU
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Uyghur

Uyghur		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	UYGHUR
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Uzbek

Uzbek		
	Script:	Cyrillic
	[<i>MyLanguage</i>] section name:	UZBEK
	For encoding:	Set Encodings parameter to:
	Windows-CP1251	CYRILLIC
	KOI8-R	CYRILLIC_KOI8
	ISO-8859-5	CYRILLIC_ISO
	UTF-8	UTF8

Valencian

Valencian		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	VALENCIAN
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Venda

Venda		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	VENDA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Vietnamese

Vietnamese		
	Script:	Vietnamese
	[<i>MyLanguage</i>] section name:	VIETNAMESE
	For encoding:	Set Encodings parameter to:
	Windows-CP1258	VIETNAMESE
	UTF-8	UTF8

Waraywaray

Waraywaray		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	WARAYWARAY

	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Welsh

Welsh ¹		
	Script:	Latin
	[<i>MyLanguage</i>] section name:	WELSH
	For encoding:	Set Encodings parameter to:
	Windows-CP1252/ISO-8859-1	ASCII
	UTF-8	UTF8

Wolof

Wolof		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	WOLOF
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Xhosa

Xhosa		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	XHOSA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

¹A stemming algorithm is available for this language and is applied by default. If you do not want to apply stemming to this language, set `Stemming` to `False` for this language.

Yiddish

Yiddish		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	YIDDISH
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Yoruba

Yoruba		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	YORUBA
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Zulu

Zulu		
	Script:	UTF8
	[<i>MyLanguage</i>] section name:	ZULU
	For encoding:	Set Encodings parameter to:
	UTF-8	UTF8

Supported Encodings

The table below lists the encodings that IDOL supports. Not all encodings are valid for all supported languages. For a list of the most common supported encodings for each language, see "[Supported Languages and Common Encodings](#)" on page 461.

Supported Encodings

IDOL name	XML name	ISO name
ARABIC	Windows-1256	CP_1256

Supported Encodings , continued

IDOL name	XML name	ISO name
ARABIC_ISO	ISO-8859-6	ISO8859-6
ARABIC_MAC	x-mac-arabic	CP_10004
ASCII	ISO-8859-1	CP_ACP
ASCII_IBM	IBM850	CP_850
CHINESESIMPLIFIED	GBK	CP_936
CHINESETRADITIONAL	Big5	CP_950
CYRILLIC	Windows-1251	CP_1251
CYRILLIC_DOS	IBM866	CP_866
CYRILLIC_ISO	ISO-8859-5	ISO8859-5
CYRILLIC_KOI8	KOI8-R	CP_21866
EASTERNEUROPEAN	Windows-1250	CP_1250
EASTERNEUROPEAN_ISO	ISO-8859-2	ISO8859-2
EUC	EUC-JP	
GREEK	Windows-1253	CP_1253
GREEK_ISO	ISO-8859-7	ISO8859-7
HEBREW	Windows-1255	CP_1255
HEBREW_ISO	ISO-8859-8	ISO8859-8
JIS	JIS_Encoding	
KOREAN	KS_C_5601-1987	CP_949
LATIN3	ISO-8859-3	ISO8859-3
LATIN5	ISO-8859-9	ISO8859-9
LATIN6	ISO-8859-14	ISO8859-14
LATIN7	ISO-8859-13	ISO8859-13
LATIN9	ISO-8859-15	ISO8859-15
NORTHERNEUROPEAN	Windows-1257	CP_1257
NORTHERNEUROPEAN_ISO	ISO-8859-4	ISO8859-4
SHIFTJIS	Shift_JIS	CP_932

Supported Encodings , continued

IDOL name	XML name	ISO name
THAI	TIS-620	CP_874
TURKISH	Windows-1254	CP_1254
UCS2	ISO-10646-UCS-2	ISO-10646
UTF8	UTF-8	CP_UTF8
VIETNAMESE	Windows-1258	CP_1258
WESTERNEUROPEAN	Windows-1252	CP_1252

TermSize Parameter

The `TermSize` parameter in the `[Server]` section of the configuration file allows you to specify the maximum number of characters that any term in the IDOL Server data index can contain. The default value is 20.

For certain languages, this default term size might be too low. In this case, you can increase the value of the `TermSize` parameter.

Note: You cannot apply the `TermSize` parameter individually to a language.

The table below shows the `TermSize` value that HPE recommends for the specified languages. If you have content in more than one language, use the larger value.

Recommended `TermSize` value per language

Language	TermSize value
English and other European languages	20
Arabic	30
Chinese	30
Hebrew	30
Korean	30
Japanese	30
Thai	30
German	30
Greek	40

Per-Language Sentence-Breaking Files

For languages in which words are not delimited by spaces (Japanese, Chinese, Thai, and Korean), IDOL Server uses sentence-breaking libraries. In a default IDOL Server installation, these files are stored in the `IDOL/langfiles` directory.

If you run IDOL Server on a UNIX platform, specify the `LD_LIBRARY_PATH` to ensure that IDOL Server can find the sentence-breaking files that it requires.

The following tables list the files that the individual languages require.

- **Japanese**

NT	UNIX
japanesebreaking.dll	japanesebreaking.so
\jpn-cha\cforms.cha	/jpn-cha/cforms.cha
\jpn-cha\chadic.da	/jpn-cha/chadic.da
\jpn-cha\chadic.lex	/jpn-cha/chadic.lex
\jpn-cha\chasenrc	/jpn-cha/chasenrc
\jpn-cha\connect.cha	/jpn-cha/connect.cha
\jpn-cha\ctypes.cha	/jpn-cha/ctypes.cha
\jpn-cha\grammar.cha	/jpn-cha/grammar.cha
\jpn-cha\matrix.cha	/jpn-cha/matrix.cha
\jpn-cha\table.cha	/jpn-cha/table.cha
libchasen.dll	

- **Traditional Chinese**

NT	UNIX
chinesebreaking.dll	chinesebreaking.so
big5togb.txt	big5togb.txt
wordlist.txt	wordlist.txt
chineseconvlist.txt	chineseconvlist.txt

- **Simplified Chinese**

NT	UNIX
chinesebreaking.dll	chinesebreaking.so
big5togb.txt	big5togb.txt
wordlist.txt	wordlist.txt
chineseconvlist.txt	chineseconvlist.txt

- **Thai**

NT	UNIX
thaibreaking.dll	thaibreaking.so
thaidict.txt	thaidict.txt
thaiconvlist.txt	thaiconvlist.txt

- **Korean**

NT	UNIX
koreanbreaking.dll	koreanbreaking.so
main.dat	main.dat
prob.dat	prob.dat
main.fst	main.fst
prob.fst	prob.fst
pos.nam	pos.nam
tag.nam	tag.nam
tagout.nam	tagout.nam
connection.txt	connection.txt
StopPosNam.txt	StopPosNam.txt
TagName.txt	TagName.txt
koreanconvlist.txt	koreanconvlist.txt

Stop Word Lists for Supported Languages

A stop word list (stop list) is a list of common words that IDOL Server does not index. Words such as *the* or *a* occur too frequently to carry any significance, and IDOL Server does not require them to understand the concept of text. Using a stop list to remove these words can improve query results and performance, and save index space.

Each language that IDOL Server supports needs a stop list; if the IDOL Server installer does not include a stop word list for the language that you want to use, you can create one.

You can use a standard text editor to create or edit a stop list. Stop word lists are located in the IDOL Server `IDOL/langfiles` directory. For example, you might want to add any words that occur in most or all of your documents, and which you do not need to search for.

For all operations, IDOL Server recognizes words as stop words irrespective of the encoding they are in. For example, in Russian you can list a stop word in the UTF-8 encoding in the stop word list file, and IDOL Server recognizes it if it occurs in a document in KOI8 encoding.

For simplicity, HPE recommends that you type all the terms in the stop list in UTF-8 encoding. However, you can list the words in the stop list in any of the valid encodings for that language. For example, in Russian you can specify stop words in KOI8, UTF8, ISO, and so on.

You can specify words in uppercase or lowercase, and you can separate them with spaces or new lines.

Note: If necessary, you can use different encodings in the same stop list file. You need to specify each word only once; that is, you do not need to specify the same word in several different encodings.

For each encoding that you want to use, create a section in your stop list file. Give the section the same name as the language type that you are using (for example, `cyrillic_koi8`, `cyrillic_utf8`).

For example:

```
[cyrillic_utf8]
ДАЖЕ
ДЛЯ ДО
ЕЕ ЕГО
А БЕЗ БОЛЕЕ БЫ БЫЛs
```

Related Topics

- ["Supported Languages and Common Encodings" on page 461](#)

Appendix D: Manually Create IDX Files

To manually create an IDX file, you create a text file that contains the data that you want to index into IDOL Server, formatted in IDOL Server fields. The fields store the data in a format that IDOL Server can index.

- [IDX Format](#)511
- [Section a Document](#)513

IDX Format

IDOL Server fields in IDX files

#DREREFERENCE	Enter a unique reference string for the document. Usually this reference is a file name, URL, or a unique code number.
#DRETITLE	Enter the title of the document. You can enter multiple lines.
#DRECONTENT	Enter the content of the document. You can enter multiple lines. (This parameter is optional. However, if you do not enter #DRECONTENT, you must specify one or more #DREFIELD <i>NameN</i> fields. Otherwise, the document does not contain any content).
#DREFIELD <i>NameN</i>	Specify the name of each DREFIELD that you are defining, and enter an appropriate value for it. You must enclose the field value with quotation marks (""). For example, to index customer details: <pre>#DREFIELD surname1="Smith" #DREFIELD forename1="Peter" #DREFIELD title1="Mr." #DREFIELD surname2="Miller" #DREFIELD forename2="Susan" #DREFIELD title2="Dr."</pre> <p>If the document contains only one instance of the DREFIELD that you are defining, you do not need to add a qualifier to the name of the field. For example:</p> <pre>#DREFIELD company="Hewlett Packard Enterprise"</pre> <p>You can define the same DREFIELD with different values. For example:</p> <pre>#DREFIELD MyField="value1" #DREFIELD MyField="value2" #DREFIELD MyField="value3"</pre> <p>The field values can span multiple lines. For example:</p> <pre>#DREFIELD MyField="line1 line2"</pre>

IDOL Server fields in IDX files , continued

	<p>line3"</p> <p>#DREFIELD is optional. However, if you do not enter a DREFIELD <i>NameN</i> field, you must specify #DRECONTENT. Otherwise, the document does not contain any content.</p> <p>Note: DREFIELD names must not contain spaces, accents, or multibyte characters. If you use these text elements, IDOL Server removes them when it indexes the fields. You must also change any queries that reference field names that contain these elements to use the modified field name.</p>
#DREDATE	Enter the creation date of the document in the format that you specified for the DateFormatCSVs parameter in the IDOL Server configuration file. By default, this is yyyy/mm/dd.
#DREDBNAME	Enter the name of the database into which you want to index the document.
#DRESECTION <i>N</i>	<p>If you are indexing a large document and want to divide it into smaller sections, you can give each section a DRESECTION number to index the defined sections as individual documents into IDOL Server.</p> <p>If you divide a document into sections:</p> <ul style="list-style-type: none"> • the first section must be #DRESECTION 0. • the section numbers must be in numerical order. • apart from the #DRESECTION number and the #DRECONTENT, each section must contain the same IDOL Server field values. <p>(See "Section a Document" on the next page).</p>
#DREENDDOC	Indicates the end of the document. You must enter this delimiter.

Note: The text file must start with #DREREFERENCE and end with #DREENDDOC.

Example

This is an example of a text file that IDOL Server can index:

```
#DREREFERENCE 392348A0
#DREFIELD authorname1="Brown"
#DREFIELD authorname2="Edgar"
#DREFIELD title="Dr."
#DREDATE 1998/08/06
#DRETITLE
Jurassic Molecules
#DRECONTENT
Scientists announced last week the successful reproduction of a possible precursor
to all life on Earth. The molecules consist of a part of DNA and the molecular
"scissors" responsible for destroying messenger RNA in humans.
Using a technique called test tube evolution, scientists created a nucleic acid
```

enzyme, the first known enzyme that uses an amino acid to start chemical activity. Scientists hope that the creation of this molecule will lead to the elusive precursor. The precursor, by definition, will have to contain both the genetic code for replication and an enzyme to trigger self replication.

```
#DRETYPE text
#DREDBNAME Science
#DRESTORECONTENT y
#DREENDDOC
```

Tip: You can submit data for IDOL Server to index by using the wizard on the **Index** tab in the Console section of IDOL Admin. For more information, refer to the *IDOL Admin User Guide*.

Section a Document

If a document that you want to index contains more than 500 words, divide it into sections to make it more manageable for IDOL Server. If you want to index XML rather than IDX, you do not need to section your data because IDOL Server automatically applies sectioning to it.

Declare a separate document for each section that you split the original document into. Give each section a #DRESECTION number.

If you divide a document into sections:

- You must name the first section #DRESECTION 0.
- You must put the section numbers in numerical order.
- You must put the content of each section into the #DRECONTENT field.
- You must make each #DRECONTENT no more than 500 words long.
- You must give each section the same DRE field values, except for the #DRESECTION number and the #DRECONTENT.

Example Text File

The following IDX file example shows a document that has been divided into sections:

```
#DRREFERENCE 392348A0
#DREFIELD authorname1="Brown"
#DREFIELD authorname2="Edgar"
#DREFIELD title="Dr."
#DREDATE 1998/08/06
#DRETITLE
Jurassic Molecules
#DRESECTION 0
#DRECONTENT
Scientists announced last week the successful reproduction of a possible precursor to all life on Earth. The molecules consist of a part of DNA and the molecular "scissors" responsible for destroying messenger RNA in humans. Using a technique called test tube evolution, scientists created a nucleic acid enzyme, the first known enzyme that uses an amino acid to start chemical activity.
```

Scientists hope that the creation of this molecule will lead to the elusive precursor. The precursor, by definition, will have to contain both the genetic code for replication and an enzyme to trigger self replication.

At this point, no naturally occurring hybrid enzymes have been found. Scientists speculate that such enzymes may exist in nature and most certainly existed in Earth's early history.

#DRETYPE text

#DREDBNAME Science

#DRESTORECONTENT y

#DREENDDOC

#DREREERENCE 392348A0

#DREFIELD authorname1="Brown"

#DREFIELD authorname2="Edgar"

#DREFIELD title="Dr."

#DREDATE 1998/08/06

#DRETITLE

Jurassic Molecules

#DRESECTION 1

#DRECONTENT

Scientists have known for some time that the key ingredients for life are DNA, RNA, and proteins. An interesting chicken-egg dilemma has developed: which came first, RNA, DNA, or proteins? Many believe that a replicating RNA molecule is the likely precursor to all life on Earth.

RNA serves as both a genetic molecule and an enzyme in the body, which scientists believe strongly suggests the likelihood of an RNA precursor to all life. They speculate that RNA was first, followed by DNA, the much more stable of the two. It would serve as an efficient storehouse for the genetic code. Proteins, better catalysts than RNA, likely evolved later as well. At some point, the current three-based system developed from the initial one-based system of RNA.

Scientists hope that these scissors molecules may also have practical uses in medicine, since the molecules can efficiently shred specific DNA. Theoretically, it may be possible to tailor such a molecule to attack and shred harmful DNA from pathogenic organisms. These molecules could be made to be activated only in specific circumstances.

#DRETYPE text

#DREDBNAME Science

#DRESTORECONTENT y

#DREENDDOC

Appendix E: Category XML Format

This appendix describes the required structure of the XML file that you can use with IDOL Category to create categories.

- [Introduction](#) 515
- [XML Format](#) 515
- [Example Category XML Files](#) 523

Introduction

To use the IDOL Server Category component to use an XML file to create a category structure in IDOL:

- Import a hierarchy from an XML file, by running the `CategoryImportFromXML` action, as in this example:

```
action=CategoryImportFromXML&ImportFilename=MyCategory.xml&BuildNow=True
```

For information on importing category information by using the Category component, see ["Categorization" on page 143](#).

XML Format

This section lists and describes the tags that are allowed in the XML file from which you want to import a category structure.

You must include the `<autn:categories>` tag in the XML file; however, there are no required tags within `<autn:categories>`. If you include only the `<autn:categories>` tag, IDOL imports an empty category structure. To expand the category structure, use the `<autn:category>` tag and its children.

<autn:categories> (required)

The `<autn:categories>` tag marks the beginning of the XML categories that IDOL Server reads. When you use the `CategoryImportFromXML` action, IDOL Server reads the XML between the opening and closing `<autn:categories>` tags.

All categories (`<autn:category>`) are children of (`<autn:categories>`) when you export a category from IDOL to XML. You can then import the XML to IDOL. They should have the same format.

You must include an XML namespace in the tag. For example:

```
<autn:categories xmlns:autn="http://schemas.autonomy.com/aci">
```

Tag name	Number allowed	Required
<code>"<autn:category>"</code>	one or more	No

<autn:category>

The <autn:category> tag marks the limits of each category that you want to import in your XML file. You can include one or more <autn:category> tags inside the <autn:categories> tag, and <autn:category> tags can also contain child <autn:category> tags.

The following table lists the tags that are allowed in <autn:category>.

Tag name	Number allowed	Required
"<autn:name> (required)"	one	Yes
"<autn:id>"	one	No
"<autn:parent>"	one	No
"<autn:refersto>"	one	No
"<autn:trainingelement>"	one or more	No
"<autn:simplecat>"	one	No
"<autn:relevancecat>"	one	No
"<autn:details>"	one	No

<autn:name> (required)

The <autn:name> tag sets the name of the category. You must include one <autn:name> within each set of <autn:category> tags. For example:

```
<autn:name>UKpolitics</autn:name>
```

Tags allowed within <autn:name>: none

<autn:id>

The <autn:id> tag sets the IDOL category ID. If a category with this ID already exists in the server, IDOL uses the OnConflict ACI parameter to determine the action to take.

For further information, see the CategoryImportFromXML OnConflict parameter in the *IDOL Server Reference*.

<autn:parent>

The <autn:parent> tag identifies the ID of the parent category. This option is effective only if you set Flat to True in the CategoryImportFromXML action.

For further information, refer to the *IDOL Server Reference*.

<autn:refersto>

The <autn:refersto> tag identifies the ID of the category to which the new category refers. This information is used to create a category that refers to another, and which inherits its fields, training, and special documents. This option is effective only if you set Flat to True in the CategoryImportFromXML action.

For further information, refer to the *IDOL Server Reference*.

<autn:trainingelement>

The <autn:trainingelement> tag identifies the training element for a category. IDOL Server identifies concepts that belong to the category from this training set. You can include one or more <autn:trainingelement> tags in each set of <autn:category> tags.

The following table lists the tags that are allowed in <autn:trainingelement>.

Tag name	Number allowed	Required
One or more of these:		
"<autn:type>"	one	Yes
"<autn:content>"	one	See below
"<autn:language>"	one	No
"<autn:reference>"	one	See below
"<autn:docid>"	one	See below
"<autn:database>"	one	No

<autn:type>

The <autn:type> tag sets the type of training to be used by <autn:trainingelement>. Each <autn:trainingelement> should contain only one <autn:type> tag.

The following table describes the values that are valid for <autn:type>.

Options for <autn:type>	Description
TRAININGTEXT	Identifies the training type as text only.
BOOLEAN	Identifies the training type as Boolean. You must define the Boolean operator in

Options for <autn:type>	Description
	the <autn:content> tag. For example: <pre><autn:type>BOOLEAN</autn:type> <autn:content>(phone AND mobile)</autn:content></pre>
URLDOWNLOAD	Identifies text from a URL to use for training.
DREDOCUMENT	Identifies the training type as a document indexed in IDOL. The document is specified by either autn:reference or autn:docid.

<autn:content>

The <autn:content> tag specifies the actual training text or Boolean expression. This tag is required if "<autn:type>" is TRAININGTEXT, BOOLEAN, or URLDOWNLOAD.

For more information, see "<autn:type>" on the previous page.

<autn:language>

The <autn:language> tag specifies the language of the training text. This tag is optional.

<autn:reference>

The <autn:reference> tag specifies the IDOL DRREFERENCE to use for training. This tag is required if <autn:type> is DREDOCUMENT, and you do not set <autn:docid>.

Note: If the document is of the type DREDOCUMENT, you must set either <autn:reference> or <autn:docid>, but not both.

For more information, see "<autn:type>" on the previous page and "<autn:docid>" below.

<autn:docid>

The <autn:docid> tag specifies the IDOL DocID to use for training. This tag is required if <autn:type> is DREDOCUMENT, and you do not set <autn:reference>.

Note: If the document is of the type DREDOCUMENT, you must set either <autn:reference> or <autn:docid>, but not both.

For more information, see "<autn:type>" on the previous page and "<autn:reference>" above.

<autn:database>

The <autn:database> tag specifies the IDOL database in which the training document is located. You can set only one database for each <autn:trainingelement> tag. This tag is optional.

<autn:simplecat>

The <autn:simplecat> tag specifies whether the category is a simple category. For example:

```
<autn:simplecat>True</autn:simplecat>
```

Note: If you use this tag, do not set <autn:relevancecat>.

<autn:relevancescat>

The <autn:relevancescat> tag specifies whether the category is a relevance category. For example:

```
<autn:relevancescat>False</autn:relevancescat>
```

Note: If you use this tag, do not set <autn:simplecat>.

<autn:details>

The <autn:details> tag sets training details for the category. You can include one set of <autn:details> within each set of <autn:category> tags.

The following table lists the tags that are allowed in <autn:details>.

Tag name	Number allowed	Required
"<autn:generatedterms>" and "<autn:generatedweights>"	one	See below
"<autn:queryagenttnw>"	one	No
"<autn:modifiedterms>" and "<autn:modifiedweights>"	one	No
"<autn:exclusions>"	one	No
"<autn:inclusions>"	one	No
"<autn:fakeweights>"	one	No
"<autn:numresults>"	one	No
"<autn:threshold>"	one	No
"<autn:databases>"	one	No
"<autn:fieldtext>"	one	No
"<autn:taxonomyroot>"	one	No
"<autn:active>"	one	No
"<autn:role>"	one	No
"<autn:memberpermissions>"	one	No
"<autn:nonmemberpermissions>"	one	No
"<autn:simplecatdefaultcat>"	one	No
"<autn:relevantcat>"	one	No
"<autn:simplecatparam>"	one	No
"<autn:userfields>"	one	No

<autn:generatedterms>

The <autn:generatedterms> tag sets terms generated for a category from training. Do this only if you are editing an existing category from which you can take the terms. You can include one <autn:generatedterms> in each set of <autn:details> tags. For example:

```
<autn:generatedterms>LYMPH,MISDIAGNOS,PATHOLOGI</autn:generatedterms>
```

Note: If you specify terms for a category in the <autn:generatedterms> tag, you must enter a corresponding list of weights by using the <autn:generatedweights> tag.

The <autn:generatedterms> tag is required if you set <autn:queryagenttnw>. For more information, see "[<autn:queryagenttnw>](#)" below.

<autn:generatedweights>

The <autn:generatedweights> tag sets weights generated for the terms for the category from training. Do this only if you are editing an existing category from which you can take the weights. You can include one <autn:generatedweights> in each set of <autn:details> tags. For example:

```
<autn:generatedweights>5960,4035,4001</autn:generatedweights>
```

Note: If you specify weights for a category in the <autn:generatedweights> tag, you must enter a corresponding list of terms by using the <autn:generatedterms> tag.

The <autn:generatedweights> tag is required if you set <autn:queryagenttnw>. For more information, see "[<autn:queryagenttnw>](#)" below.

<autn:queryagenttnw>

The <autn:queryagenttnw> tag specifies the query string generated by terms and weights used to build the category. For example:

```
<autn:queryagenttnw>LYMPH~[5960] MISDIAGNOS~[4305] PATHOLOGI~[4001]  
</autn:queryagenttnw>
```

Note: You can use <autn:queryagenttnw> with either <autn:generatedterms> and <autn:generatedweights>, or <autn:modifiedterms> and <autn:modifiedweights>. If both sets exist, modified terms take precedence over generated ones.

The <autn:queryagenttnw> tag performs the same function as the IDOL CategoryBuild action. See "[Build Categories](#)" on page 151.

For more information, see "[<autn:generatedterms>](#)" above, "[<autn:generatedweights>](#)" above, "[<autn:modifiedterms>](#)" below, and "[<autn:modifiedweights>](#)" on the next page.

<autn:modifiedterms>

The <autn:modifiedterms> tag sets terms defined by the user. You can include one <autn:modifiedterms> within each set of <autn:details> tags. For example:

```
<autn:modifiedterms>LYMPH,MISDIAGNOS,PATHOLOGI</autn:modifiedterms>
```

For more information on changing the weights of terms in a category, see "[Change Category Term Weights](#)" on page 150.

Note: If you specify terms for a category in the <autn:modifiedterms> tag, you must enter a

corresponding list of weights by using the `<autn:modifiedweights>` tag.

<autn:modifiedweights>

The `<autn:modifiedweights>` tag sets weights for terms defined by the user. You can include one `<autn:modifiedweights>` within each set of `<autn:details>` tags. For example:

```
<autn:modifiedweights>5960,4035,4001</autn:modifiedweights>
```

For more information on changing the weights of terms in a category, see ["Change Category Term Weights" on page 150](#).

Note: If you specify weights for a category in the `<autn:modifiedweights>` tag, you must enter a corresponding list of terms by using the `<autn:modifiedterms>` tag.

<autn:exclusions>

A comma-separated list of documents to be excluded from category queries. For example:

```
<autn:exclusions>C:\temp\doc1.txt,C:\temp\doc2.txt</autn:exclusions>
```

For more information, see the `CategorySetSpecialDocs Exclusions` parameter in the *IDOL Server Reference*.

<autn:inclusions>

The `<autn:inclusions>` tag contains a comma-separated list of documents to be included in category queries. For example:

```
<autn:inclusions>C:\temp\docA.txt,C:\temp\docB.txt</autn:inclusions>
```

For more information, see the `CategorySetSpecialDocs Inclusions` parameter in the *IDOL Server Reference*.

Note: If you use this tag, you must set corresponding weights for the included terms with the `<autn:fakeweights>` tag.

<autn:fakeweights>

The `<autn:fakeweights>` tag contains a comma-separated list of weights for documents specified in `<autn:inclusions>`. Weights must correspond to the documents in number and order. For example:

```
<autn:fakeweights>800,2200</autn:fakeweights>
```

For more information, see the `CategorySetSpecialDocs Fakeweights` parameter in the *IDOL Server Reference*.

<autn:numresults>

The `<autn:numresults>` tag sets the maximum number of results that category queries can return. You can include one `<autn:numresults>` tag for each category within `<autn:details>` tags. For example:

```
<autn:numresults>10</autn:numresults>
```

<autn:threshold>

The `<autn:threshold>` tag sets the minimum relevance score that documents must possess to appear in category query results. You can include one `<autn:threshold>` tag for each category within

`<autn:details>` tags. For example:

```
<autn:threshold>400</autn:threshold>
```

<autn:databases>

The `<autn:databases>` tag sets the databases in which documents must exist to appear in category query results. You must separate multiple databases with plus symbols, commas, or spaces. For example:

```
<autn:databases>Archive,Minicar</autn:databases>
```

<autn:fieldtext>

The `<autn:fieldtext>` tag specifies the fields that result documents must contain, and the conditions that these fields have to meet for the documents to be returned as results.

For more information, see the `CategorySetFields Fieldtext` parameter in the *IDOL Server Reference*.

<autn:taxonomyroot>

The `<autn:taxonomyroot>` tag specifies whether the category is a taxonomy root. For example:

```
<autn:taxonomyroot>True</autn:taxonomyroot>
```

<autn:active>

The `<autn:active>` tag specifies whether the category is active. For example:

```
<autn:active>True</autn:active>
```

<autn:role>

The `<autn:role>` tag specifies the role or roles that you want to give access to the category. For example:

```
<autn:role>Usertype1,Usertype2</autn:role>
```

Use "`<autn:memberpermissions>`" and "`<autn:nonmemberpermissions>`" to set which category access permissions members and non-members of this role should have.

<autn:memberpermissions>

The `<autn:memberpermissions>` tag sets the category access permissions that you want role members to have. If you want to list multiple permissions, you must separate them with commas. For example:

```
<autn:memberpermissions>read,edit</autn:memberpermissions>
```

For more information, see the `CategorySetPermissions MemberPermissions` parameter in the *IDOL Server Reference*.

<autn:nonmemberpermissions>

The `<autn:nonmemberpermissions>` tag sets the category access permissions that you want role non-members to have. If you want to list multiple permissions, you must separate them with commas. For example:

```
<autn:nonmemberpermissions>read</autn:nonmemberpermissions>
```

For more information, refer to the `NonMemberPermissions` parameter for the `CategorySetPermissions` action in the *IDOL Server Reference*.

<autn:simplecatdefaultcat>

The `<autn:simplecatdefaultcat>` tag specifies which of the category's children is to be the default category for `CategorySimpleCategorize`. You must use the category ID value. For example:

```
<autn:simplecatdefaultcat>1230982349874568</autn:simplecatdefaultcat>
```

For more information, see the `CategorySetDetails SimpleCatDefaultCat` parameter in the *IDOL Server Reference*.

<autn:relevantcat>

The `<autn:relevantcat>` tag specifies which of the category's children is to be used as the relevant category. You must use the category ID value. For example:

```
<autn:relevantcat>324987602</autn:relevantcat>
```

For more information, see the `CategorySetDetails RelevantCat` parameter in the *IDOL Server Reference*.

<autn:simplecatparam>

The `<autn:simplecatparam>` tag sets a numeric factor that increases or decreases the probability of this category being chosen by the `CategorySimpleCategorize` action. For example:

```
<autn:simplecatparam>1.4</autn:simplecatparam>
```

For more information, see the `CategorySetDetails SimpleCatParam` parameter in the *IDOL Server Reference*.

<autn:userfields>

The `<autn:userfields>` tag sets fields and values defined by the user, as in this example:

```
<autn:userfields>  
<autn:acc-inbox-threshold>30</autn:acc-inbox-threshold>  
</autn:userfields>
```

Example Category XML Files

The following is an example XML file that includes only required elements:

```
<?xml version="1.0" encoding="UTF-8" ?>  
<autn:categories xmlns:autn="http://schemas.autonomy.com/aci/">  
  <autn:category>  
    <autn:name>MyCategory</autn:name>  
  </autn:category>  
</autn:categories>
```

The following is an example of a Category XML file that includes one category:

```
<?xml version="1.0" encoding="UTF-8" ?>  
<autn:categories xmlns:autn="http://schemas.autonomy.com/aci/">  
  <autn:category>
```

```
<autn:name>test</autn:name>
  <autn:trainingelement>
    <autn:type>TRAININGTEXT</autn:type>
    <autn:content>hotels paris food stay</autn:content>
    <autn:language>ENGLISH</autn:language>
  </autn:trainingelement>
  <autn:details>
    <autn:generatedterms>STAI,PARI,FOOD,HOTEL</autn:generatedterms>
    <autn:generatedweights>2352,2272,1884,417</autn:generatedweights>
    <autn:queryagenttnw>FOOD~[1884] HOTEL~[417] PARI~[2272]STAI~[2352]
    </autn:queryagenttnw>
    <autn:active>True</autn:active>
  </autn:details>
</autn:category>
</autn:categories>
```

Appendix F: GetStatus Action Response

This appendix describes the `GetStatus` action and the response that IDOL Server returns.

- [GetStatus Action](#) 525
- [IDOL Server GetStatus Response](#) 526
- [Example IDOL Server GetStatus Response](#) 533

GetStatus Action

The `GetStatus` action returns information about the status of IDOL Server and its components, as well as the configuration and content of the server.

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

where:

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

The `GetStatus` action is a diagnostic tool that you can use to check information about your server.

You can use the output from `GetStatus` to:

- view some of the IDOL Server configuration options without opening the configuration file.
- check that all IDOL Server components are running.
- monitor the number of documents in IDOL Server and see how close the server is to the configured document limit.
- monitor how many index actions IDOL Server has processed, and check the size of the index queue, for example to:
 - determine whether the server processes incoming index actions as fast as it receives them.
 - determine how many items the server has left to process, before performing maintenance or backups.
- check the results of the latest index validation.
- check language configuration and determine how many documents in the server use each configured language type.
- check document security configuration and determine how many documents in the server belong to each configured security type.
- check user security configuration.

Tip: You can also view status information by using the IDOL Admin interface. For more information, refer to the *IDOL Admin User Guide*.

Note: HPE recommends that you contact HPE Big Data Support before taking actions based on the information in the GetStatus action.

Related Topics

- ["Send Actions to IDOL Server" on page 29](#)

IDOL Server GetStatus Response

This section describes the XML tags that return in the response to a GetStatus action sent to the IDOL Proxy in a unified IDOL Server configuration that does not use a DAH or DIH.

For details about unified IDOL Server configurations, refer to the *IDOL Getting Started Guide*.

The GetStatus response from IDOL Proxy contains information from all its child components. Most tags result from the GetStatus response of the IDOL Server child components. However, the unified IDOL Server does not display all tags from child servers, and IDOL Proxy returns additional tags that none of the components return (such as, component status).

Tag	Description	Related configuration parameters
product	The component that returns the GetStatus data.	
version	The version of the component. Note: In the IDOL Server GetStatus response, this version is the version of the IDOL Proxy.	
indexport	The port used to send index actions to IDOL Server.	IndexPort
aciport	The port used to send ACI actions to IDOL Server.	Port
serviceport	The port used to send service actions to IDOL Server.	ServicePort
directory	The directory that contains IDOL Server.	
acithreads	The number of allowed ACI threads.	Threads
component	The status of each child component. This section contains a subsection for each component.	
status	The status of the component.	
aciport	The port used to send ACI actions to the component.	

Tag	Description	Related configuration parameters
indexport	<p>The port used to send index actions to the component.</p> <p>Note: This port applies only to indexing components, such as Content.</p>	
serviceport	<p>The port used to send service actions to the component.</p>	
licensed_languages	<p>A comma-separated list of the languages that your license allows this IDOL Server to use.</p>	
termsperdoc	<p>The configured number of best terms to generate for each document.</p>	TermsPerDoc
suggestterms	<p>The configured number of best terms to use in a Suggest action.</p>	SuggestTerms
documents	<p>The number of documents in IDOL Server that are available for search.</p>	
document_sections	<p>The number of document sections in IDOL Server.</p>	
committed_documents	<p>The number of documents that have been indexed into IDOL Server, including any deleted documents that have not been removed in a compaction.</p>	
deleted_sections	<p>The number of document sections for deleted sections in IDOL Server. This number represents the number of deleted sections since the last compaction operation. You can use it to check whether you need to compact the index to permanently delete these documents.</p>	
indexed_data_kb	<p>The amount of indexed data in the Content component.</p> <p>Note: This number does not specify the amount of disk space that IDOL Server is using. This value specifies the index size for licensing purposes.</p>	
full	<p>Whether the IDOL Server data index has reached the maximum size specified in</p>	MaxDocumentCount

Tag	Description	Related configuration parameters
	<p>the MaxDocumentCount or MaxIndexSizeKb configuration parameter.</p> <p>The DIH uses this tag when distributing index actions to IDOL Servers. You can configure DIH to stop indexing to full servers by using the RespectChildFullness DIH configuration parameter.</p>	MaxIndexSizeKb
full_ratio	How close the IDOL Server data index is to reaching the configured maximum number of documents, as determined by the MaxDocumentCount configuration parameter. If you have not set MaxDocumentCount, IDOL Server uses the MaxIndexSizeKb limit instead.	MaxDocumentCount MaxIndexSizeKb
terms	The number of terms in IDOL Server for which you can search.	
total_terms	The total number of terms in IDOL Server, including internal terms.	
term_hashes	The nearest power of 2 above the value of the DiskHash parameter.	DiskHash
record_size	The maximum size of terms in IDOL Server.	TermSize
max_occurrences	The highest number of documents in which any single term occurs.	
mindate	The earliest date of any document in IDOL Server (in AUTNDATE format).	
maxdate	The latest date of any document in IDOL Server (in AUTNDATE format).	
ref_fields	The number of reference fields that the documents in IDOL Server use.	
ref_hashes	The configured value of the RefHashes parameter.	RefHashes
indexqueue	Details about the IDOL Server index queue. See "Check Index Status" on page 65 .	

Tag	Description	Related configuration parameters
indexqueuereceived	The number of index actions that IDOL Server has received.	
indexqueuecompleted	The number of index actions that IDOL Server has completed.	
indexqueuequeued	The number of index actions remaining in the index queue.	
initialid	The index ID of the last DREINITIAL index action that the server received.	
termcache	Details about the IDOL Server term cache. See " [TermCache] Section " on page 451.	
used_kb	The current size of the IDOL Server term cache (in KB).	
num_terms	The number of terms currently stored in the IDOL Server term cache.	
limit_kb	The maximum configured size of the IDOL Server term cache (in KB).	
requests	The number of terms that have been requested from the cache.	
hits	The number of matches for terms in the cache that IDOL Server has received.	
hitrate	The rate at which IDOL Server has matched terms in the cache.	
indexcache	Details about the IDOL Server index cache. See " [IndexCache] Section " on page 440.	
used_kb	The current size of the IDOL Server index cache (in KB).	
num_terms	The number of terms currently stored in the IDOL Server index cache.	
limit_kb	The maximum configured size of the IDOL Server index cache (in KB).	IndexCacheMaxSize
num_blocks	The number of memory blocks allocated for IDOL Server indexing.	

Tag	Description	Related configuration parameters
fieldcodes	Details of fields in IDOL Server documents.	
base	The number of distinct field names in IDOL Server.	
total	The total number of field codes. If you have set the XMLFullStructure configuration parameter to True, this value is the total number of distinct occurrences of fields.	XMLFullStructure
databases	Details of the databases in IDOL Server. See "Create and Delete Databases" on page 383 .	
max_databases	The maximum allowed number of databases.	
num_databases	The total number of databases in IDOL Server.	NumDBs
active_databases	The number of active (not deleted) databases.	
database	Details of individual IDOL Server databases.	
name	The name of the database.	Name
documents	The number of documents in the database.	
sections	The number of document sections in the database.	
internal	Whether the database is configured as internal.	Internal
readonly	Whether the database is configured as read-only.	ReadOnly
expiry_hours	The expiration time (in hours) for documents in this database.	ExpiryTime
expiry_action	The expiration action to perform when documents expire from this database.	ExpireIntoDatabase
security_settings	Details of the configured security types in IDOL Server. See "Set up Security" on	

Tag	Description	Related configuration parameters
	page 353 .	
no_of_security_types	The total number of configured security types in IDOL Server.	
security_type	Details of individual configured document security types. See " [Security] Section " on page 448 .	[Security] section
name	The name of the security type. This value is the name of the configuration section where you define the security settings.	
documents	The number of documents that this security type applies to.	
sections	The number of document sections that this security type applies to.	
language_type_settings	Details of the configured language types in IDOL Server. See " Language Support " on page 99 .	
no_of_language_types	The number of configured language types in IDOL Server.	
language_type	Details of an individual language type. See " [LanguageTypes] Section " on page 441 .	[LanguageTypes] section
name	The name of the language type. This value is the name given to one encoding for a language, set in the Encodings configuration parameter.	Encodings
language	The language that applies to this language type. This value is the name of the language configuration section for this language type.	
encoding	The encoding that applies to this language type.	Encodings
documents	The number of documents with this language type.	
sections	The number of document sections with this language type.	

Tag	Description	Related configuration parameters
Validation	The results of index validation.	
result	<p>The result of validation on a particular index. There is a result tag for each type of index validation. This tag contains one of the following values:</p> <ul style="list-style-type: none"> • N/A. No index validation has run for the index type. • Success. Index validation has passed. • An error message. Indicates a validation failure. In this case you might need to regenerate a data index. 	ValidateDiskIndex ValidateNodeTable ValidateRefIndex ValidateUnstemmed ValidateSecIndex ValidateNumeric
type	<p>The type of index for the result. This can be one of the following values:</p> <ul style="list-style-type: none"> • diskindex • nodetable • refindex • unstemmed • secindex • numeric 	ValidateDiskIndex ValidateNodeTable ValidateRefIndex ValidateUnstemmed ValidateSecIndex ValidateNumeric
timestamp	The time of the last validation, in epoch seconds.	
duration	The duration (in seconds) of the last validation operation.	
autn:numusers	The number of users in IDOL Server.	
autn:maxusers	The maximum number of users allowed in IDOL Server. This value depends on your product license.	
securitytypes	User security information from the Community component.	[UserSecurity] section
defaultsecuritytype	The default user security type.	DefaultSecurityType
name	The name of the user security type.	
fields	The user security fields associated with the security type.	SecurityFieldCSVs
documentsecurity	Whether the security type uses the document security module.	DocumentSecurity

Tag	Description	Related configuration parameters
documentsecuritytype	The document security module that applies to the user security type.	DocumentSecurityType
autn:scheduledthreads	The number of threads available for scheduled tasks.	
autn:categories	The number of categories in IDOL Server.	

Example IDOL Server GetStatus Response

The following XML is an example response to the GetStatus action.

```
<autnresponse>
  <action>GETSTATUS</action>
  <response>SUCCESS</response>
  <responsedata>
    <product>IDOL Server</product>
    <version>10.4.0</version>
    <build>1076091</build>
    <indexport>9001</indexport>
    <queryport>-1</queryport>
    <aciport>9000</aciport>
    <serviceport>9002</serviceport>
    <directory>C:\Program Files\IDOL\IDOLServer\IDOL</directory>
    <acithreads>4</acithreads>
    <component>
      <content>
        <status>RUNNING</status>
        <aciport>9010</aciport>
        <indexport>9011</indexport>
        <serviceport>9012</serviceport>
      </content>
      <community>
        <status>RUNNING</status>
        <aciport>9030</aciport>
        <serviceport>9031</serviceport>
      </community>
      <category>
        <status>RUNNING</status>
        <aciport>9020</aciport>
        <serviceport>9021</serviceport>
      </category>
      <agentstore>
        <status>RUNNING</status>
        <aciport>9050</aciport>
        <indexport>9051</indexport>
      </agentstore>
    </component>
  </responsedata>
</autnresponse>
```

```
    <queryport>9052</queryport>
    <serviceport>9053</serviceport>
  </agentstore>
  <view>
    <status>RUNNING</status>
    <aciport>9080</aciport>
    <serviceport>9081</serviceport>
  </view>
</component>
<licensed_languages>UNLIMITED</licensed_languages>
<querythreads>0</querythreads>
<termsperdoc>50</termsperdoc>
<suggestterms>40</suggestterms>
<documents>1086404</documents>
<document_sections>1368042</document_sections>
<committed_documents>1370830</committed_documents>
<deleted_sections>0</deleted_sections>
<indexed_data_kb>82643321</indexed_data_kb>
<full>>false</full>
<full_ratio>0.69</full_ratio>
<terms>3675990</terms>
<total_terms>3872552</total_terms>
<term_hashes>16777216</term_hashes>
<record_size>53</record_size>
<max_occurrences>611200</max_occurrences>
<mindate>1059260400</mindate>
<maxdate>1437484356</maxdate>
<mindatestring>23:00:00 26/07/2003</mindatestring>
<maxdatestring>13:12:36 21/07/2015</maxdatestring>
<ref_fields>1</ref_fields>
<ref_hashes>10000000</ref_hashes>
<indexqueue>
  <indexqueuereceived>78</indexqueuereceived>
  <indexqueuecompleted>73</indexqueuecompleted>
  <indexqueuequeued>5</indexqueuequeued>
  <initialid>0</initialid>
</indexqueue>
<termcache>
  <used_kb>3867</used_kb>
  <num_terms>29</num_terms>
  <limit_kb>102400</limit_kb>
  <requests>41</requests>
  <hits>2</hits>
  <hitrate>4</hitrate>
</termcache>
<indexcache>
  <used_kb>6303</used_kb>
  <num_terms>9873</num_terms>
  <limit_kb>102400</limit_kb>
```

```
<num_blocks>1</num_blocks>
</indexcache>
<fieldcodes>
  <base>0</base>
  <total>77</total>
</fieldcodes>
<databases>
  <max_databases>65534</max_databases>
  <num_databases>3</num_databases>
  <active_databases>3</active_databases>
  <database>
    <name>News</name>
    <documents>1085127</documents>
    <sections>1366685</sections>
    <internal>>false</internal>
    <readonly>>false</readonly>
    <expiry_hours> 4320 </expiry_hours>
    <expiry_action> move to Archive </expiry_action>
  </database>
  <database>
    <name>Archive</name>
    <documents>1006</documents>
    <sections>1086</sections>
    <internal>>false</internal>
    <readonly>>true</readonly>
    <expiry_hours>No Information Available</expiry_hours>
    <expiry_action>No Information Available</expiry_action>
  </database>
  <database>
    <name>World</name>
    <documents>262</documents>
    <sections>262</sections>
    <internal>>false</internal>
    <readonly>>false</readonly>
    <expiry_hours>No Information Available</expiry_hours>
    <expiry_action>No Information Available</expiry_action>
  </database>
</databases>
<security_settings>
  <no_of_security_types>5</no_of_security_types>
  <security_type>
    <name>NT_V4</name>
    <documents>0</documents>
    <sections>0</sections>
  </security_type>
  <security_type>
    <name>Netware_V4</name>
    <documents>0</documents>
    <sections>0</sections>
```

```
</security_type>
<security_type>
  <name>Notes_V4</name>
  <documents>0</documents>
  <sections>0</sections>
</security_type>
<security_type>
  <name>Exchange_V4</name>
  <documents>0</documents>
  <sections>0</sections>
</security_type>
<security_type>
  <name>Documentum_V4</name>
  <documents>0</documents>
  <sections>0</sections>
</security_type>
</security_settings>
<language_type_settings>
  <no_of_language_types>1</no_of_language_types>
  <language_type>
    <name>englishUTF8</name>
    <language>ENGLISH</language>
    <encoding>UTF8</encoding>
    <documents>1086404</documents>
    <sections>1368042</sections>
  </language_type>
</language_type_settings>
<validation>
  <result type="diskindex">N/A</result>
  <result type="nodetable">N/A</result>
  <result type="refindex">N/A</result>
  <result type="unstemmed">N/A</result>
  <result type="secindex">N/A</result>
  <result type="numeric">N/A</result>
</validation>
<autn:numusers>25</autn:numusers>
<autn:cachedusers>0</autn:cachedusers>
<autn:usersize>0</autn:usersize>
<autn:maxusers>5000</autn:maxusers>
<autn:numterms>0</autn:numterms>
<autn:termsize>0</autn:termsize>
<autn:maxterms>0</autn:maxterms>
<autn:termoffset>0</autn:termoffset>
<autn:storeversion>0</autn:storeversion>
<securitytypes>
  <defaultsecuritytype>autonomy</defaultsecuritytype>
  <securitytype>
    <name>autonomy</name>
    <documentsecurity>>false</documentsecurity>
```

```
</securitytype>
<securitytype>
  <name>notes</name>
  <fields>username</fields>
  <documentsecurity>true</documentsecurity>
  <documentsecuritytype>Notes_V4</documentsecuritytype>
</securitytype>
<securitytype>
  <name>ldap</name>
  <documentsecurity>>false</documentsecurity>
</securitytype>
<securitytype>
  <name>documentum</name>
  <fields>username</fields>
  <documentsecurity>true</documentsecurity>
  <documentsecuritytype>Documentum_V4</documentsecuritytype>
</securitytype>
<securitytype>
  <name>exchange</name>
  <fields>username, domain</fields>
  <documentsecurity>true</documentsecurity>
  <documentsecuritytype>Exchange_V4</documentsecuritytype>
</securitytype>
<securitytype>
  <name>netware</name>
  <fields>username</fields>
  <documentsecurity>true</documentsecurity>
  <documentsecuritytype>Netware_V4</documentsecuritytype>
</securitytype>
</securitytypes>
<autn:schedulethreads>2</autn:schedulethreads>
<autn:categories>5953</autn:categories>
</respondedata>
</autnresponse>
```


Appendix G: Error Codes

This appendix lists IDOL error codes.

- [Community Error Codes](#) 539

Community Error Codes

The following table describes common error codes for the Community component.

Error code	Description
-2147483391	Error: Bad Parameter
-2147483390	Error: Out Of Memory
-2147483388	Error: File Error
-2147438079	Error: User Exists
-2147438053	Error: User Not Found
-2147438078	Error: Too Many Users
-2147438077	Error: Agent Exists
-2147438076	Error: Agent Not Found
-2147438075	Error: Data Index Error, Data Index not found
-2147438074	Error: Agent Index Error
-2147438073	Error: Too Many Agents
-2147438072	Error: Field Not Found
-2147438071	Error: Role Exists
-2147438070	Error: Role Not Found
-2147438069	Error: Privilege Exists
-2147438068	Error: Privilege Not Found
-2147438067	Error: Profile Not Found
-2147438065	Error: Numeric terms must be specified alone
-2147438064	Error: Data Index Error, no terms found

Glossary

A

Access Control List.

See ACL.

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.

Adaptive Probabilistic Concept Modelling

See APCM.

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. See Also: explicit agent, implicit agent

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 that legacy technologies use to categorize documents). You can then query IDOL Server with text and an AgentBoolean field to return categories whose Boolean agent matches this text.

ALD

Automatic Language Detection. The process of automatically detecting the language of a particular document, and indexing it into IDOL Server according to the rules for the detected language.

alerting

An automatic process for alerting users, by e-mail, text, or message, when new content is added to IDOL Server that matches their agents or profiles. See Also: mailing.

APCM

Adaptive Probabilistic Concept Modelling. A technique whereby terms are given a weight according to their statistical importance in IDOL Server. Terms can have a weight between 0 and 255.

AQG

Automatic Query Guidance. A set of operations that use the results from query summaries. AQG includes dynamic thesaurus generation, automatic query disambiguation, query refinement, and rapid

clustering of a results set. See Also: query summary.

authentication

The process of checking user credentials (user names, passwords, and PIN codes) against an IDOL Server or external security repository. The authentication process identifies a user, and allows IDOL Server to confirm their access permissions for different documents.

autnrank

An internal IDOL Server document rank, which determines the order in which two or more documents return in a results list when the relevance or other sort option is equal.

Automatic Language Detection

See ALD.

Automatic Query Guidance

See AQG.

Autonomy Content Infrastructure

See ACI.

C

category

A set of criteria that define a particular topic, which you can use to categorize documents that contain content relevant to the topic.

Category component

The IDOL Server component that manages categorization and clustering.

Category index

An IDOL Server index that stores categories.

category training

Text or documents that define a topic or subject for a particular category. When IDOL Server categorizes documents, it matches

document content to similar category training.

cluster

A hierarchically agglomerated collection of data that has been extracted from snapshots. Each cluster represents a concept area that contains a set of items, which share common properties. Clustering data allows you to make trends and developments in data visible.

combine

A query operation that combines two or more query results into a specified smaller number of results. The most usual case is to combine two or more sections of the same document as a single query result. It can also combine results by a reference or metadata field value.

community

All the people in a user network neighborhood. It allows users to find other people in the community who have been looking at similar documents, or have agents that are similar to their agents.

Community component

The IDOL Server component that manages users and communities.

concept summary

A brief summary of each result document that returns for a query. The concept summary displays a few sentences that are typical of the result content (these sentences can be from different parts of the result document).

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.

context summary

A conceptual summary of the result document that is biased by the terms of the query. A context summary includes sentences that are particularly relevant to the terms in the query (these sentences can be from different parts of the result document).

crawling

The process that Connectors and Web crawlers use to retrieve content from Web resources, by recursively following hyperlinks from an initial page. See Also: spidering

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 to store data 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.

default user

The default user role in IDOL Server. A default user has only the privileges that have been allocated to this default role.

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.

Distributed Index Handler

See DIH.

E

Education

The process of extracting entities (patterns of text) from documents.

entity

In Education, an entity is a word, phrase, or block of information that the Education

component can match and extract from documents. An entity can be a specific text string, such as a name, or it can be a pattern of text such as an address or phone number. You define the pattern in a grammar, which Education uses to find the entities in documents.

expertise location

An IDOL Server operation to find groups of users with a particular set of expertise or interests.

explicit agent

An IDOL agent that users explicitly create from themselves. See Also: agent, implicit agent

Extensible Markup Language

See XML.

extraction

The process of extracting text, metadata, and subfiles from documents. IDOL KeyView performs this extraction process in IDOL.

F

faceted search

See: parametric search

fetch

The process of downloading documents from the repository in which they are stored (such as a local folder, Web site, Lotus Domino server, and so on), importing them to IDX format, and indexing them into IDOL Server.

fetch task

A group of settings that instruct a Connector how to retrieve data from a repository. Connectors can run fetch tasks automatically, or in response to an action.

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

G

grammar

In Education, a grammar is a pattern that defines an entity. See Also: Education, entity

H

hyperlink

The ability for IDOL Server to connect related documents to results, by using suggestions. See Also: suggest

I

identifier

An encoded value that identifies the source of a document in IDOL Server. Connectors and CFS add identifiers to every document that they create for indexing into IDOL Server. They store this value in the AUTN_IDENTIFIER field.

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.

IDOL Search Optimizer

An IDOL application that allows you to manage IDOL Server content.

IDX

A structured file format that can be indexed into IDOL Server. You can use a connector to import files into this format, or you can manually create IDX files.

implicit agent

An IDOL agent that is created as part of a user profile. When you profile a user, IDOL Server creates these agents for a user, according to the documents and search results that the user views. See Also: agent, explicit agent

importing

After a document has been downloaded from the repository in which it is stored, it is imported to an IDX or XML file format. This process is called importing.

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 and manipulate the data index.

index fields

Fields that IDOL Server processes linguistically when it stores them. Store fields that contain text that 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 (such as 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.

Intelligent Query Logic.

See IQL.

IQL

Intelligent Query Logic. Functionality that allows you to set up rules to return a particular set of documents, or to run a secondary query in response to an initial keyword or conceptual query.

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.

Lua

An embedded scripting language that you can use to write custom scripts to expand certain IDOL functionality.

M

mailing

An automatic process for sending an email to users when new content is added to IDOL Server that matches their agents or profiles. See Also: alerting

mapped security

A security setup where IDOL Connectors index documents into IDOL Server with an encrypted access control list, which IDOL Server uses to match user permissions for the document. With this method, IDOL Server does not need to check the

original data repository to check the security information every time a user attempts to access the document. Compare With: unmapped security. See Also: access control list.

meaning based computing

The ability for computers to act on the meaning of content. This includes conceptual searching, and also workflows that automatically process documents according to their content.

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 search

A type of query that returns a list of all possible values of a specified field for documents that match a particular standard query. You can use the values to find matching documents with a particular property. This process is also known as faceted search. Compare With: FieldText

PIN code

Personal Identification Number security feature used in addition to a user ID and password.

primary domain controller

A server computer in a Microsoft Windows domain that controls various computer resources. See also: active directory, LDAP.

privilege

Role-based capabilities that determine, for example, whether a user is allowed to

access specific data.

profile

Information about a user that is based on the concepts in documents that the user reads. Every time a user opens a document, IDOL Server updates their profile. This process allows the administrator to bring new documents that match the interests in a user profile to the attention of the users.

promotions

Targeted content that you want to display to users but that is not included in the search results, such as advertisements.

Q

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 summary

A query operation that determines the important topics and phrases in a set of documents. Query summaries are used in Automatic Query Guidance (AQG). See Also: AQG.

quick summary

A brief summary of each result document that returns for a query. The quick summary displays the first few sentences of the result document.

R

reference

A string that is used to identify a document. This 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. It uses them at query time 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.

retrain

The process used to increase the accuracy of agents by indicating which of the results that return to you are most relevant to your query. The retrained agent then returns more relevant results.

role

A set of privileges that an administrator can allocate to an IDOL Server user.

S

section breaking

The process of separating a document into sections for indexing. The number of sections that a document is split into increases proportionally with the size of the document. This process ensures that when you, for example, query for text that is relevant to a specific part of a book, IDOL Server can find the appropriate section and return it (if the book was not indexed in sections, IDOL Server might not find the text you search for, because it might not be conceptually relevant to the entire book).

security

Security includes anything that makes sure that only authorized users can access or perform actions on data. It includes making sure that only permitted users can view and retrieve documents, user authentication, and secure communications.

seed

In clustering with snapshots, a seed is a potential cluster. It contains a document, and suggested conceptually similar documents from the IDOL Server index.

sentiment analysis

A form of Education that identifies positive and negative sentiment in text.

snapshot

Internal raw data from which you can extract clusters. You can thus generate cluster information and spectrographs.

soundex

An IDOL Server query type that allows you to search for a term by using a phonetic spelling.

spectrograph

A graphical representation of the results of clustering. The spectrograph displays clusters of documents, and the similarities between different clusters.

spidering

The process that Connectors use to retrieve content from Web resources, by recursively following hyperlinks from an initial page.

stemming

The process of extracting the morphological root of a word. In languages, some words have a common morphological root. HPE provides stemming algorithms that reduce words to this form. This process allows IDOL Server to match concepts regardless of

the grammatical use of words. In English, for example, the words 'help', 'helpful', 'helping', and 'helped' all reduce to their stem 'help' without significant loss of meaning. HPE provides as standard a set of stemming algorithms for the most commonly used languages. IDOL Server applies stemming after stop words have been discarded, both at index time (when content is stored in IDOL Server), and at query time (query text is stopped and stemmed before it is matched).

stop word

A very common word that occurs too frequently to be useful for searching. Stop words include articles (for example, the) and prepositions (for example, to or from). Stop words are language-specific. You can use a stop word list in IDOL Server to allow it to discard these words at index and query time to save index space and improve retrieval performance.

stop word list

Also called stop list. A list (located in the IDOL Server langfiles directory) that contains common words (stop words) that IDOL Server does not store. Words such as the, and, or a occur too frequently to carry any significance, and IDOL Server does not require them to understand the concept of the text.

stopping

The process of removing the words listed in the stop word list from documents before they are stored in IDOL Server, and from query text before it is matched against IDOL Server content.

stored state

A set of query results, which is stored in IDOL Server for you to re-use later in other operations. When you store a state, IDOL Server provides a state token, which you can use to retrieve the stored state.

subfile

A file that has been extracted from a container (such as a .ZIP archive)

suggest

A type of query that returns documents that contain similar concepts to a particular document, rather than matching a particular query string. See Also: query

summary

A few sentences or paragraphs that describe what a document is about. IDOL can automatically create summaries from document content.

synonym file

A file that allows IDOL Server to handle synonym queries. A synonym query returns results which are conceptually similar to the query terms, and conceptually similar to the synonyms that are available for the query terms. A synonym file contains comma-separated lists of synonym strings for words. You can specify lists for each language type that you have set up in IDOL Server in this file.

synonym search

A type of query that returns documents that contain synonyms for a particular search term, as well as documents that contain the term. See Also: query, synonym file

T**tagging**

The process of adding extra information to documents. The tag might be a category, or entities returned from Education. Tagging usually adds a field to a document, which you can use to search by the name of a tag.

taxonomy

An automatically created hierarchical structure of clusters or other information. A

taxonomy provides you with an overview of the information landscape, and an insight into specific areas of the information.

term

The basic entity that IDOL Server indexes (for example, a word in a document after it has been stemmed).

Terms and Weights.

See TNW.

TNW

Terms and Weights. These values are used in categorization to define the most important terms that define a category topic.

training

Text, documents, and query syntax used to define the topic that an agent or category must match.

U**Universal Query Language.**

See UQL.

unmapped security

A security setup where IDOL Server checks the security entitlement of a user against the original data repositories in real time when the user attempts to access a document. With this method, IDOL Server always has the current security information, but the response can be slow because of the additional connection to the repository. Compare With: mapped security

UQL

Universal Query Language. A name for the IDOL Server query syntax, which you can use for keyword, conceptual, Boolean, and Wildcard searches.

V

View

An IDOL component that converts files in a repository to HTML formats for viewing in a Web browser.

W

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 IDOL Server Administration Guide (IDOL 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!