Oracle® Communications Indexing and Search Service

System Administrator's Guide

Release 1.0.5

E56605-03

August 2017

ORACLE®

Oracle Communications Indexing and Search Service System Administrator's Guide, Release 1.0.5

E56605-03

Contents

4 Best Practices for Indexing and Search Service and Oracle ZFS

5 Setting Up and Managing Indexing and Search Service Accounts

6 Managing and Monitoring the Indexing and Search Service Store

7 Troubleshooting Indexing and Search Service

8 Improving Indexing and Search Service Performance

Part II Administering a High-Availability System

9 Managing Indexing and Search Service High Availability

10 Managing the Watcher Service

16 Message Queue Consumer and Broker Log Messages

17 Index Service Log Messages

18 GlassFish Server Log Messages

Part V Indexing and Search Service Reference

19 Indexing and Search Service Command-Line Utilities

20 Indexing and Search Service Configuration Parameters

Preface

This guide explains how to administer Oracle Communications Indexing and Search Service and its accompanying software components.

Audience

This document is intended for system administrators whose responsibility includes Indexing and Search Service. This guide assumes you are familiar with the following topics:

- Oracle Communications Messaging Server
- Oracle GlassFish Server
- Oracle Directory Server Enterprise Edition and LDAP
- System administration and networking

Related Documents

For more information, see the following documents in the Indexing and Search Service documentation set:

- *Indexing and Search Service Installation and Configuration Guide*: Provides instructions for installing and configuring Indexing and Search Service.
- *Indexing and Search Service Release Notes*: Describes the new features, fixes, known issues, troubleshooting tips, and required third-party products and licensing.
- *Indexing and Search Service Security Guide*: Provides guidelines and recommendations for setting up Indexing and Search Service in a secure configuration.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Part I

Monitoring and Managing Indexing and Search Service

Part I contains the following chapters:

1

Indexing and Search Service System Administration Overview

This chapter provides an overview of Oracle Communications Indexing and Search Service, and describes the basic administration tasks and tools used to perform those tasks.

About Indexing and Search Service

Indexing and Search Service provides server-side indexing and search of Oracle Communications Messaging Server email content, including email attachments. Indexing and Search Service enables nearly instantaneous results from complex searches such as cross-folder searches. Because it is deployed as a separate service, the actual search effort is offloaded from Messaging Server. Additionally, any IMAP client that can communicate with Messaging Server can take advantage of this powerful search capability.

Overview of Indexing and Search Service Administration Tasks

An Indexing and Search Service administrator is responsible for the day-to-day tasks of maintaining and managing Indexing and Search Service and its users. The tasks also include managing Indexing and Search Service components, GlassFish Server, and potentially other Communications Suite components.

You perform the following tasks as an Indexing and Search Service administrator:

- Stopping and starting Indexing and Search Service
- Bootstrapping and managing user accounts
- Monitoring Indexing and Search Service
- Tuning Indexing and Search Service performance
- Troubleshooting Indexing and Search Service

About Indexing and Search Service Administration Tools

Indexing and Search Service is composed of an *indexing service* and a *search service*. The indexing service indexes email data in real time. Indexing is provided through web services, enabling you to index arbitrary data. Indexing and Search Service clients consume RESTful web services that provide the search capabilities. Components communicate over Java Messaging System (JMS) enabling distribution of components across multiple hosts. For more information about Indexing and Search Service

architecture and components, see *Indexing and Search Service Installation and Configuration Guide*.

Indexing and Search Service provides several command-line utilities for administering the server. For more information, see "Indexing and Search Service Command-Line Utilities".

Directory Placeholders Used in This Guide

Table 1–1 lists the placeholders that are used in this guide.

Table 1–1 Indexing and Search Service Directory Placeholders

Placeholder	Directory
IndexSearch_home	Specifies the installation location for the Indexing and Search Service software. The default is **/opt/sun/comms/jiss**.
GlassFish_home	Specifies the installation location for the Oracle GlassFish Server software. The default is **/opt/glassfish3/glassfish**.
MessagingServer_home	Specifies the installation location for the Messaging Server software. The default is **/opt/sun/comms/messaging64**.

2

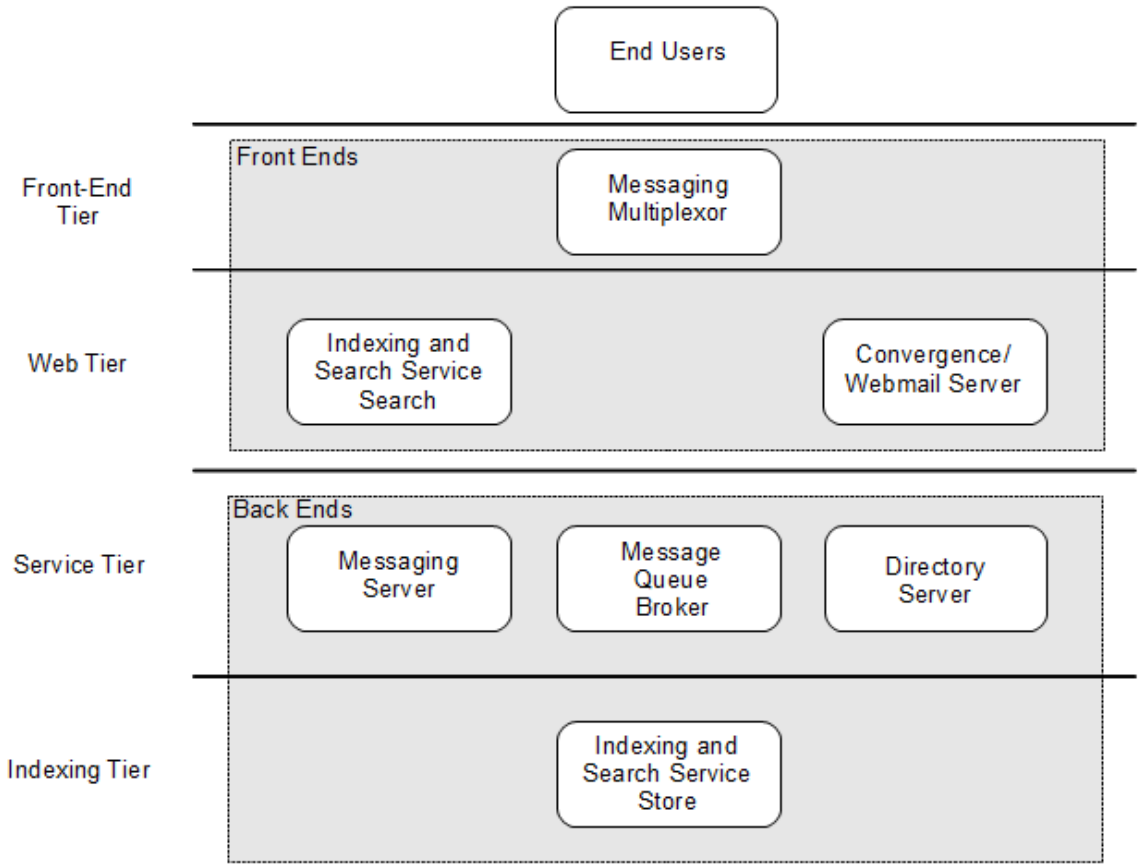
Indexing and Search Service Processes

This chapter discusses the different processes (also referred to as services) comprising Oracle Communications Indexing and Search Service and the components that it relies on.

Overview of Indexing and Search Service Software Architecture

Figure 2–1 shows the Indexing and Search Service logical tiered architecture.

Figure 2–1 Indexing and Search Service Logical Tiered View



This figure shows that an Indexing and Search Service deployment uses a logical architecture consisting of the following three tiers:

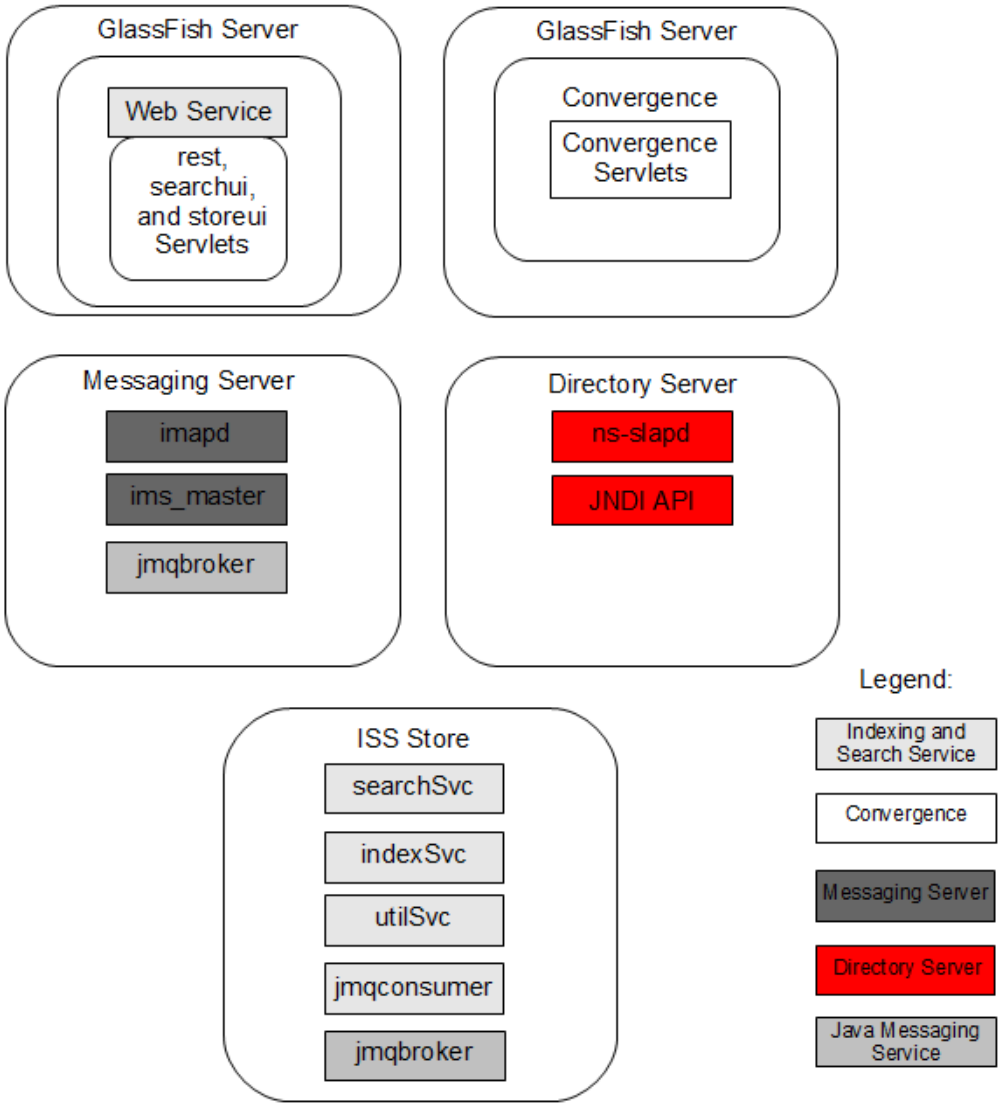
- Front-End/Web: Provides the Indexing and Search Service RESTful search servlet, Oracle Communications Convergence client, and Oracle Communications Message Server messaging multiplexor (MMP) front-ends
- Service: Provides the back-end Messaging Server (IMAP service), the Message Queue (JMQ) broker component of the Java Message Service, and LDAP services
- Indexing: Provides Indexing and Search Service store back ends (the "indexed" data for searching)

The Indexing and Search Service components communicate by using topics and queues on Java Message Service (JMS), enabling you to distribute components across multiple physical hosts. During the Indexing and Search Service initial configuration, you decide which Indexing and Search Service components (**web**, **jmq**, **index**, and so on) are configured on which hosts by running the Indexing and Search Service **setup** script. By default, all Indexing and Search Service components are configured on the host upon which Indexing and Search Service software is installed. For more information about the **setup** script, see *Indexing and Search Service Installation and Configuration Guide*.

Indexing and Search Service and Dependent Services

Figure 2–2 expands on Figure 2–1, showing a view of how the Indexing and Search Service services and other component services are distributed:

Figure 2–2 Indexing and Search Service Services View



From a deployment perspective, the groupings in this figure are pieces that could be deployed on one or multiple hosts, depending on your site's architecture.

Note: This figure shows that two Java Message Service JMQ brokers are present: one for Oracle Communications Messaging Server notifications and one for internal Indexing and Search Service communication. In a simple configuration, the JMQ broker for Messaging Server notifications is running on the Messaging Server host and the JMQ broker for inter-process Indexing and Search Service communication is running on the Indexing and Search Service host.

The following sections discuss each component's services.

Indexing and Search Service Web Services

Indexing and Search Service web services (**rest**, **store** and **searchui**) are deployed in an Oracle GlassFish Server web container. These services can be connected to directly by an end user or indirectly by services such as Convergence or Messaging Server's **imapd** process. The user/group directory in Oracle Directory Server Enterprise Edition (Directory Server) is used for authentication. These services communicate to the Indexing and Search Service back-end hosts by using Java Message Service.

Convergence

Indexing and Search Service content is searched by clients such as Convergence. Convergence is deployed in a GlassFish web container. Convergence contains service proxies for mail and Indexing and Search Service and other services, which communicate by using various protocols to Communications Suite services. For more information, see the topic on introduction to the Convergence service in *Convergence Installation and Configuration Guide*.

Messaging Server

Messaging Server consists of several processes, but for Indexing and Search Service, the most important ones are:

- **imapd**: Writes messages to and reads messages from the Messaging Server message store by using the IMAP protocol. Generates event notifications to Indexing and Search Service in response to user mailbox activity (for example, message submits, flag changes, expunged messages, new/renamed/deleted folders).
- **ims_master**: One of many channel programs that comprise the Messaging Server job controller. Delivers messages into the message store and generates event notifications for each new message delivery.

Directory Server

Indexing and Search Service requires Directory Server for LDAP information storage. Indexing and Search Service uses the Java Naming and Directory Interface (JNDI) API to enable communications between JMS and Directory Server. Indexing and Search Service uses the following properties to connect to Directory Server to know which name space to use:

```
java.naming.factory.initial
java.naming.provider.url
java.naming.security.authentication
java.naming.security.credentials
java.naming.security.principal
```

Indexing and Search Service uses JNDI to look up the values of the following objects, which are stored in LDAP:

```
com.sun.messaging.QueueConnectionFactory
com.sun.messaging.TopicConnectionFactory
com.sun.messaging.Queue
com.sun.messaging.Topic
```

Indexing and Search Service Store

Indexing and Search Service back-end services run as separate JVM instances. The core Indexing and Search Service services that are required for normal operation are:

- **indexSvc**: Bootstraps new users and indexes incoming content
- **searchSvc**: Searches index and returns results
- **jmqconsumer**: Listens to the JMQ on the message store and notifies **indexSvc** if new content requires indexing
- **utilSvc**: Provides utility services to Index Service

On an Oracle Solaris system, these services are managed through System Management Facility (SMF). The Indexing and Search Service command **svc_control.sh** starts and stops all services. The individual services can also be separately queried, enabled, disabled or restarted by using SMF, for example:

```
svcs | grep jiss
online 20:23:42 svc:/application/jiss-utilSvc:default
online 20:23:55 svc:/application/jiss-indexSvc:default
online 20:24:08 svc:/application/jiss-searchSvc:default
online 20:24:20 svc:/application/jiss-jmqconsumer:default
```

Log File Locations

Table 2–1 provides the default locations for the Indexing and Search Service log files by service.

Table 2–1 Indexing and Search Service Log File Locations

Service Name	Log Name and Default Location
Index Service	**/var/opt/sun/jiss/logs/iss-indexsvc.log.0**
JMQ Consumer	**/var/opt/sun/jiss/logs/iss-jmqconsumer.log.0**
JMQ broker	**/var/imq/instances/imqbroker/log/log.txt**
RESTful search servlet, StoreUI servlet, SearchUI servlet	**/opt/glasfish3/glassfish/domains/domain1/logs/server.log**
Search Service	**/var/opt/sun/jiss/logs/iss-searchsvc.log.0**
Util Service	**/var/opt/sun/jiss/logs/iss-utilsvc.log.0**

Service Interaction and Communication

From a high-level perspective, the Indexing and Search Service system operates as follows:

1. Content is searched by email clients.
   - Email clients such as Thunderbird or an IMAP-enabled mobile device get enhanced search through an IMAP SEARCH Indexing and Search Service gateway.
   - The Convergence client has an Indexing and Search Service plugin, which generates searches directly to Indexing and Search Service for attachment search, and enables fast cross folder search.
2. Messaging Server manages the content in the message store and provides message store event notifications to Indexing and Search Service.
3. GlassFish Server hosts Indexing and Search Service servlets.
   - These servlets consist of the RESTful search API (**rest**), thumbnail access (**store**), and a demo application (**searchui**).

4. A JMQ broker hosts the event notification destination to which Messaging Server is publishing, and search and index destinations used between Indexing and Search Service components.
   - Multiple brokers can be in use: one for Messaging Server JMQ notifications, and one for internal Indexing and Search Service communication.
5. Directory Server performs user authentication and JNDI lookups.
6. An Indexing and Search Service store instance stores the indexes, and:
   - Manages and serves up images from the thumbnail store
   - Parses email messages from the message store into indexing content
   - Receives and processes search request messages

The following section discusses the Indexing and Search Service components in more detail, and how information flows through the system.

Indexing and Search Service Data Flow

Refer to Figure 2–3 throughout this section. The numbered circles 1 through 4 correspond to the following logical architectural pieces in the following sections, and are indicated in the data flow descriptions such as (**1**), (**2**), and so on:

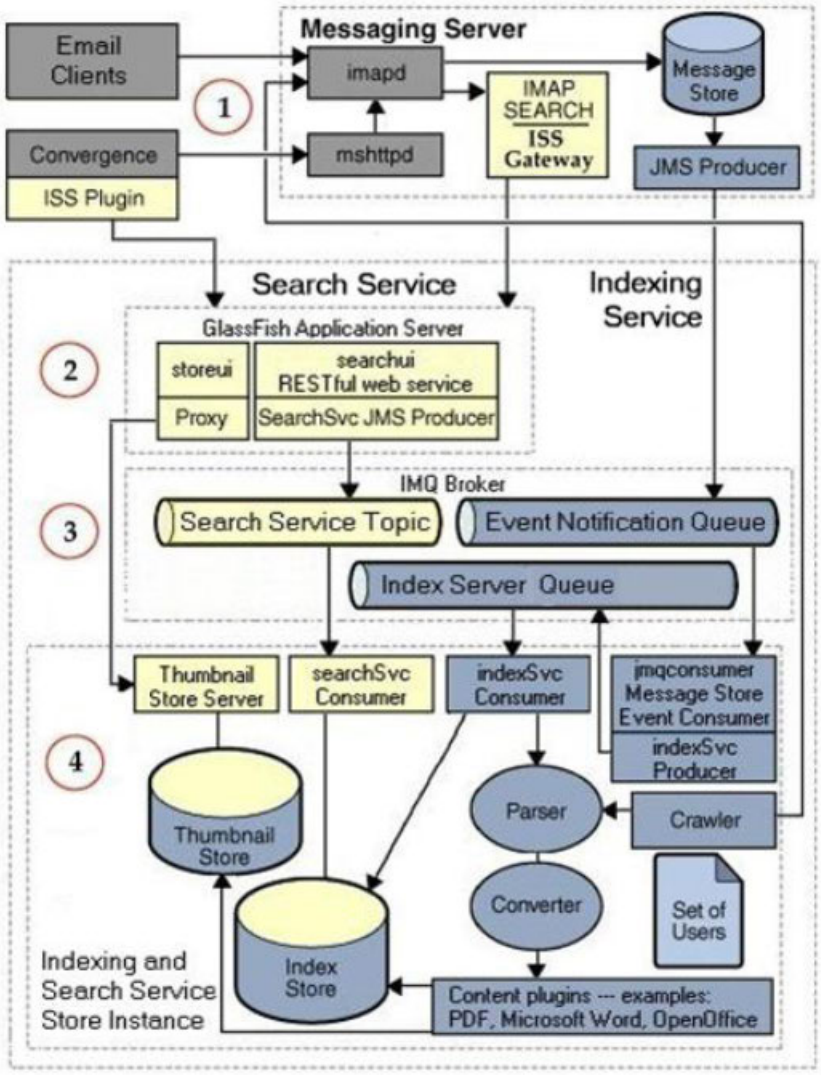
1. Client - Messaging Server

2. Web Tier

3. Service Tier

4. Indexing Tier

Figure 2–3 Indexing and Search Service Data Flow



Search Request

The Indexing and Search Service search request works as follows:

1. (**1**) IMAP clients request a search. For example, **x SEARCH BODY "test"**.
2. (**1**) The Indexing and Search Service Gateway diverts the search request to Indexing and Search Service.
3. (**2**) Indexing and Search Service RESTful web service processes search requests. For example, the following is a request received from Messaging Server:

   ```
   /rest/search?q=%2busername:user1%20%2bhostname:ms.example.com%20%2bfolder:
   %22INBOX%22%20%2bbody:test&c=2147483647&contentformat=simpleuid&format=atom
   ```

4. (**3**) Indexing and Search Service RESTful web service posts the search to the Search Service JMS Topic.
5. (**4**) The **searchSvc** service performs the search and returns the result over the JMS topic back to the RESTful web service.

If **searchSvc** is logged at the INFO level, a log entry similar to the following appears:

```
query number: 3, 3 total matching documents to query: +username:user1
+hostname:ms.example.com +folder:"INBOX" +body:test, search time was 46ms
```

6. (**2**) Indexing and Search Service RESTful web service sends the result back to the Messaging Server Indexing and Search Service Gateway.
7. (**1**) Messaging Server returns the result back to the IMAP client. For example, * **SEARCH 7 53 214** indicates that three matching emails were found: UIDs 7, 53 and 214.

Other clients, such as Convergence, issue search requests directly to Indexing and Search Service, which enables them to provide fast cross-folder search and use attachment search. In this case the flow is as follows:

1. (**1**) Client accesses the **Attachments** virtual folder inside the Convergence user interface.
2. (**2**) Indexing and Search Service RESTful web service manages search requests.

   For example, here is an example request from Convergence, as logged in the GlassFish Server log file:

   ```
   {{/rest/search?q=%2Busername%3Auser1%40example.com%20%2Bhostname%3Ams.example.
   com%20%2Battachment-type%3A(atdoc%20atppt%20atxls%20atodf%20atpdf%20athtml%20at
   plain%20atrtf%20atxml%20atimage%20ataudio%20atvideo%20atcompress%20atvcf%20
   atother%20atsencr%20atssign)&c=160&s=0&thumbnail=s&format=json&contentFormat=
   attachmentOnly&token=rB8SHWjWMA}}
   ```

3. (**3**) Indexing and Search Service RESTful web service posts the search to the Search Service JMS Topic.
4. (**4**) The **searchSvc** service performs the search and returns the result (to RESTful search service and then to Convergence).

   If **searchSvc** is logged at the INFO level, a log entry similar to the following appears:

   ```
   query number: 5, 160 total matching documents to query: +username:user1
   +hostname:ms.example.com +attachment-type:(atdoc atppt atxls atodf atpdf athtml
   atplain atrtf atxml atimage ataudio atvideo atcompress atvcf atother atsencr
   atssign), search time was 95ms
   ```

5. (**1**) The search result is displayed to the end user. If an attachment search had been requested, the client can then issue Indexing and Search Service thumbnail requests to resolve the thumbnail URLs.

Thumbnail Request

The Indexing and Search Service thumbnail request works as follows:

1. (**1**) Client receives the results of a search request.
2. (**2**) Client uses the received thumbnail URL to request the thumbnail. Log messages such as the following appear in the GlassFish Server log file for each downloaded thumbnail:

   ```
   [#|2012-02-14T00:38:29.384+0000|INFO|sun-appserver2.1.1|com.sun.comms.iss.store
   ui.StoreUIServlet|_ThreadID=74;_
   ThreadName=httpSSLWorkerThread-8070-0;|Processing file:
   /var/iss/attach//01/26/h1/u120026/f10/1195248462/00/53/tbn_small_2_
   ```

```
.jpg|#][#|2012-02-14T00:38:29.384+0000|INFO|sun-appserver2.1.1|com.sun.comms.is
s.storeui.StoreUIServlet|_ThreadID=74;_
ThreadName=httpSSLWorkerThread-8070-0;|LOG: ip:10.79.212.11 inst:iss1 auth:FORM
user:user1 mailhost:ms.example.com
pathinfo:/01/26/h1/u120026/f10/1195248462/00/53/tbn_small_2_.jpg|#]
```

3. (**2**) The Indexing and Search Service store servlet receives the request for the thumbnail and returns the result from disk or, for a multiple host deployment, issues a request to the Thumbnail Store Server.
4. (**4**) If applicable, the Thumbnail Store Server returns the thumbnail graphic.

Initial Indexing

The initial indexing of accounts (bootstrapping) works as follows:

1. (**4**) A user account is requested to be indexed by either the **issadmin.sh --bootstrap** command or the arrival of a message that indicates the account should be auto-provisioned (for example, the arrival of the INBOX Create event). For an auto-provisioned account, these messages appear in the JMQConsumer log file:

   ```
   logMessageHeaders INFO: Received event Create generated on Sat Feb 04 11:28:57
   PST 2012 with sequence number 7224 URL is
   imap://user100@ms.example.com/INBOX;UIDVALIDITY=1328383737/;UID=null
   indexEvent WARNING: Account user100@ms.example.com autoprovisioning begun
   setAccountState INFO: Set account user100@ms.example.com state to I
   scheduleEvent INFO: Queuing event BootStrap for user100@ms.example.com
    generated on Sat Feb 04 11:28:57 PST 2012, with sequence number 7224
   ```

2. (**4**) This command is delivered over the Index Server Queue to the **IndexSvc** service. This message is logged in JMQConsumer log file:

   ```
   run INFO: Account user100@ms.example.com is active on Sat Feb 04 11:28:58 PST
   2012, processing event BootStrap generated on Sat Feb 04 11:28:57 PST 2012 with
   sequence number 7224, time between generate and submit to index svc is 1956ms.
   ```

3. (**4**) A crawler within the Index Service connects to Messaging Server over the IMAP protocol and fetches all the folders and messages for the user.
4. (**4**) Each message is broken down into body parts and attached files.
5. (**4**) Each part is indexed into the index store.
6. (**4**) Thumbnail images are generated and stored if supported.
7. (**4**) The bootstrap is complete. This message is logged in the JMQConsumer log file:

   ```
   setAccountState INFO: Set account user100@ms.example.com state to A
   run INFO: Finished indexing event BootStrap for account user100@ms.example.com
   with sequence number 7224, time between submit to and return from index svc is
   1440ms.
   ```

Real-time Indexing

The real-time indexing of email content works as follows:

1. Each time a change is made to the Messaging Server message store, a JMQ notification message is posted by either the Messaging Server **imapd** process or **ims_master** process to the Event Notification Queue.

Changes include new mail (NewMsg, UpdateMsg), changed flags (ChangeFlag), folder operations (Rename, Create, Delete), expunged messages (ExpungeMsg), and so on.

2. The **jmqconsumer** service picks up these notifications and delivers them to the Index Service.
3. Each of these changes are applied to the Index and Thumbnail Store.
4. The following messages appear in the JMQConsumer log file when a new message arrives.

```
logMessageHeaders INFO: Received event NewMsg generated on Mon Feb 13 00:01:00
GMT 2012 with sequence number 4154 (size h:760 e:771 t:771 - Using text) URL is
imap://amam@ms.example.com/INBOX;UIDVALIDITY=1238618761/;UID=189538977
scheduleEvent INFO: Queuing event NewMsg for amam@ms.example.com generated on
Mon Feb 13 00:01:00 GMT 2012, with sequence number 4154
```

5. This message appears in the JMQConsumer log file when the event has been delivered to Index Service for indexing.

```
run INFO: Account amam@ms.example.com is active on Mon Feb 13 00:01:00 GMT
2012, processing event NewMsg generated on Mon Feb 13 00:01:00 GMT 2012 with
sequence number 4154, time between generate and submit to index svc is 390ms.
```

6. This message appears in the JMQConsumer log file when the message was successfully indexed.

```
run INFO: Finished indexing event NewMsg for account amam@ms.example.com with
sequence number 4154, time between submit to and return from index svc is
238ms.
```

7. The indexing of this event can also be tracked in the Index Service log file, which writes the following log entry in response to this event:

```
logmsg INFO: Received event NewMsg on account: amam@ms.example.com with
sequence number 4154, current backlog in queue: 0
updateCounts INFO: FolderCount.updateCounts: host=ms.example.com user=amam
updating folder: INBOX to 1476798/1476798/0 old mvalue:1476797 lastUid
value:null
```

3

Stopping and Starting Indexing and Search Service

This chapter describes how to stop and start Oracle Communications Indexing and Search Service services.

Stopping and Starting Indexing and Search Service

To stop and start Indexing and Search Service services, you use the **svc_control.sh** command.

Starting Indexing and Search Service Services

To start Indexing and Search Service services:

1. Log in as or become superuser (**root)**.
2. Run the **svc_control.sh** command:

```
IndexSearch_home/bin/svc_control.sh start
Starting indexSvc
Starting searchSvc
Starting jmqconsumer
```

Stopping Indexing and Search Service Services

To stop Indexing and Search Service services:

1. Log in as or become superuser (**root)**.
2. Run the **svc_control.sh** command:

```
IndexSearch_home/bin/svc_control.sh stop
Stopping jmqconsumer
Stopping searchSvc
Stopping indexSvc
```

4

Best Practices for Indexing and Search Service and Oracle ZFS

This chapter describes recommendations for configuring and administering Oracle ZFS with Oracle Communications Indexing and Search Service.

About Using ZFS

You can improve Indexing and Search Service performance and deployment scalability by using ZFS for the Indexing and Search Service index store files, attachment thumbnail store files, and log files. ZFS provides the following features that make it ideal for Indexing and Search Service:

- Snapshot backup
- Enables the use of less expensive SATA drives
- Built-in volume manager enables you to grow file systems dynamically

Before using ZFS to store the Indexing and Search Service files:

- Set up the I/O subsystem so that it performs well for small random read and write operations.
- Configure the ZFS storage pools (use the **zpool** command) with enough disks and virtual devices (**vdevs**) to provide sufficient I/O throughput.
- Use a mirrored storage pool configuration to improve read performance.
- If possible, use Solid State Drives (SSDs) as cache and log devices to improve read and write performance, respectively.

ZFS Configuration Recommendations for Indexing and Search Service

Follow these recommendations to configure ZFS and Indexing and Search Service:

1. Separate the Indexing and Search Service index store files, attachment thumbnail store files, and log files on different file systems.

   The index store's files contain user account information, along with each account's metadata and content representing the indexes of user account mail store content. The attachment thumbnail store's files contain thumbnail representations of user account mail store attachments. Each Indexing and Search Service component places log entries into its log file. For more information about the Indexing and Search Service store, see "Overview of the Indexing and Search Service Store Instance." For more information about Indexing and Search Service logging, see "Log Files".

Table 4–1 describes the configuration parameters, defined in the *IndexSearch_home*/**etc/jiss.conf** file, that specify the location of the Indexing and Search Service data.

Table 4–1 jiss.conf File Parameters

jiss.conf Parameter	Comments
iss.store.dir	Specifies location of the index store files.
iss.data.dir	Specifies location of the attachment thumbnail store files.
iss.log.dir	Specifies location of the Indexing and Search Service log files.

By separating index store, attachment thumbnail store, and log files onto different file systems, you enable your deployment to scale and to be responsive and resilient to variations in load. In addition, you can monitor each file system to ensure that it has sufficient I/O bandwidth.

The following example ZFS commands show how to create three separate file zpools, **indexpool**, **attachpool**, and **logpool**, for each of the Indexing and Search Service file systems **index**, **attach**, and **iss-logs**.

```
zpool create indexpool LUN
zpool set listsnapshots=on indexpool
zfs create indexpool/index
zfs set mountpoint=/var/opt/sun/comms/jiss/index indexpool/index
zpool create attachpool LUN
zpool set listsnapshots=on attachpool
zfs create attachpool/attach
zfs set mountpoint=/var/opt/sun/comms/jiss/attach attachpool/attach
zpool create logpool LUN
zfs create logpool/iss-logs
zfs set mountpoint=/var/opt/sun/comms/jiss/logs logpool/iss-logs
```

where:

LUN is a number (logical unit number) used to identify a logical unit, which is a device addressed by the SCSI protocol or Storage Area Network protocols which encapsulate SCSI, such as Fibre Channel or iSCSI.

The default ZFS recordsize is 128 Kbytes, which is the recommended recordsize for the Indexing and Search Service file systems.

Setting the recordsize of the Indexing and Search Service file systems ensures that any changes to their zpool's recordsize do not impact them, for example:

```
zfs get recordsize indexpool/index
NAME      PROPERTY    VALUE    SOURCE
indexpool  recordsize  128K     default
```

If the source is not local, then:

```
zfs set recordsize=128k indexpool/index

zfs get recordsize indexpool/index
NAME      PROPERTY    VALUE    SOURCE
indexpool  recordsize  128K     local
```

Do the same for the other Indexing and Search Service file systems.

2. Disable file access time record.

The index store does not use the file access time. By disabling file access time updates, you reduce unnecessary overhead. The following example ZFS commands disable file access time updates on Indexing and Search Service file systems **index**, **attach**, and **iss-logs**.

```
zfs set atime=off indexpool/index
zfs set atime=off attachpool/attach
zfs set atime=off logpool/iss-logs
```

3. Keep ZFS pool space under 80 percent utilization to maintain pool performance.

   ZFS pool performance can degrade when a pool is very full. As the pool approaches 100 percent full, more time is needed to find free space and it is more likely that the free space is available only in small chunks.

ZFS Backup Recommendations

Perform regular ZFS snapshot backups. You must back up the index store and attachment thumbnail store file systems atomically by using the **zfs snapshot -r** command. Then use the **zfs send** and **recv** commands, or an enterprise-level backup solution, to save the data.

Before you back up the index store and attachment thumbnail store, be sure to stop the **indexSvc** service.

The following example shows the use of **zfs snapshot** and **zfs send** and **recv** commands to back up two file systems.

```
zfs snapshot -r store@now
zfs send store/index@now | ssh host2 zfs recv store/index
zfs send store/attach@now | ssh host2 zfs recv store/attach
```

You can use **zfs send -i** to perform incremental backups. Destroy the snapshots when they are not needed. For example:

```
zfs destroy -r store@now
```

You use ZFS snapshots to back up and restore the entire index store files systems. You cannot back up and restore individual index accounts using the XFS Snapshot feature. However, you can use **issadmin.sh --export** command to back up individual accounts or groups of accounts and the **issadmin.sh --import** command to import index accounts. For more information, see "issadmin.sh."

5

Setting Up and Managing Indexing and Search Service Accounts

After installing Oracle Communications Indexing and Search Service, you must create, or "bootstrap" each account into the Indexing and Search Service store (also referred to as the index store). You can bootstrap accounts by either manually running the **issadmin.sh --bootstrap** command or by enabling account autoprovisioning. This chapter describes how to use each method, and the benefits of each.

Overview of the Indexing and Search Service Store Instance

The information in an Indexing and Search Service Store instance is organized into accounts. An account is uniquely identified by two strings, the user name and host (or domain) name. The account is the unit of security for content. Each account must have a password to control access to its data.

In the email environment, each account in the Indexing and Search Service store uses the same user name, host name, and password as the corresponding account in the Messaging Server message store from which its content is derived. Also, the Indexing and Search Service indexed data mirrors the folder directory structure of the email account.

Each account is assigned to one and only one group in the Indexing and Search Service store. Groups organize the underlying data files in the index. The data for all accounts assigned to a single group is indexed and stored together. This way of indexing and storing data reduces the overhead of managing multiple index directories for many accounts.

You can restrict a group to contain only one account. A group that can contain only one account is known as a *singleton group*. After you create a singleton group with an account, you cannot assign another account to that group. If the account in a singleton group is deleted from the Indexing and Search Service store, that group is no longer a singleton, until explicitly assigned again.

When you create accounts, you can use the *default allocation policy* or you can explicitly specify a particular group or singleton group to contain the account. The default allocation policy places accounts in the lowest numbered, previously defined group that does not already contain the maximum allowed number of accounts. The first group by default is **101**. The **iss.store.account.pergroup** configuration parameter defines the number of accounts that each group can contain (the default value is **1**). If all previously defined groups are full, the number of the next group created is the successor to the lowest numbered group after the first group that has no successor group, thus filling in gaps in the sequence of group numbers. (Groups with numbers less than the first group are not affected by the default allocation policy.)

The initial installation does not define any accounts in the Indexing and Search Service store instance. You create accounts during the indexing process, which is called *bootstrapping*. The bootstrapping process indexes the Messaging Server message store data and creates each account. For accounts that consist of very large numbers of emails, the initial indexing process can take up to several hours to complete. Currently, expect a bootstrapping rate of approximately 2000 email messages per minute.

It is difficult to estimate how much time the bootstrapping of any given account takes, so consider bootstrapping each account into a singleton group. After creating the account, check its size to see if it should be moved into another group with other accounts. See the **issadmin.sh --moveaccount** command for information about moving an account to another group. Bootstrapping accounts into singleton groups has the additional advantage of reducing the overhead on other accounts in the group that might slow the search process.

The indexing process sets the state of an Indexing and Search Service account based on its corresponding Messaging Server account. Table 5–1 describes Indexing and Search Service account states.

Table 5–1 Indexing and Search Service Account States

Account Flag	Account State	Description
A	Active	The account is searchable. Account state must be active to be searchable.
B	Bootstrapping	The account is being bootstrapped.
I	Inactive	The account is in maintenance mode and is not active.
X	Unknown	The account state is unknown. A new account is in this state after you run the **--createaccount** command.

You can manually change an account's state with the **issadmin.sh** command, for example, from Active to Inactive. An account being bootstrapped remains in the **B** (Bootstrapping) state. When bootstrapping has completed, the status changes to **A** (Active). An account whose status is Active is available for searching, and has real-time indexing events enabled.

The **issadmin.sh** command enables you to inspect and modify accounts, list account, group and folder information, modify the account state, create and delete accounts and folders, and verify the index against the Messaging Server store. For more information, see "issadmin.sh."

About Bootstrapping Indexing and Search Service Accounts

When you bootstrap user accounts, you enable Indexing and Search Service to index the users' email. You run the **issadmin.sh --bootstrap** command on the user accounts to be indexed.

Bootstrapping triggers the Indexing and Search Service Crawler program to connect by using the IMAP protocol to the Messaging Server message store. The Crawler obtains the list of folders for that user, walks through each folder, downloads the email, and adds it to the Indexing and Search Service store.

After initial bootstrapping of accounts, indexing of new messages in the Indexing and Search Service store begins when an email message change occurs in the Messaging Server message store. Email events that are significant for Indexing and Search Service include:

- Arrival of a new email message

- Deleting an email message
- Viewing (reading) an email message
- Setting an email message flag
- Creating a new folder
- Moving an email message to a new folder

These events generate Message Queue (JMQ) notifications containing the type of change. The Java Message Service (JMS) producer (the **jmqnotify** plugin) posts the notification message to the Event Notification Queue (the **imqbroker** that you configure on Messaging Server). On the Indexing and Search Service side, the JMQ consumers (Messaging Server event consumers) are listening to the Event Notification Queue. Events are tagged by the user, that is, the user who generated the event. Thus, the Indexing and Search Service store instance knows how to serve that particular user (knows which store instance that user is on), takes the message, and processes it.

When a user receives a new email message in the message store, an event notification is generated. This event notification attempts to fit the entire text of the email message into its payload (event message) so that Indexing and Search Service can then process all of the new message for indexing. Because the event message attempts to contain all the text of the email message, IMAP processing is conserved. Additionally, Messaging Server does not have to perform extra work on its end, because the email message is in memory when the event happens.

When you configure JMQ, you can set the size of the event message body. Currently, the Indexing and Search Service configuration instructions describe setting the message body at 256 Kbytes. When the message size is larger than the configured sized, the original message must be retrieved over IMAP.

If a user copies a message or sets a message flag, the event notification message contains all the information that Indexing and Search Service needs to update the Indexing and Search Service store. Indexing and Search Service does not need to download any more information to keep its store synchronized with the Messaging Server message store.

If the event notification is for a new email message that has arrived in a user's mailbox, Indexing and Search Service passes it to the Parser/Converter for processing. The message is separated into the fields that Indexing and Search Service indexes. Attachments are separated from the body text for processing by the Converter. As long as Indexing and Search Service has a converter for the attachment type, it extracts the "meaningful" text. This process is implemented by using a plugin architecture, but all plugins are internal to the Indexing and Search Service product. The Indexing and Search Service HTML converter indexes only text outside of HTML tags. That is, the HTML converter ignores HTML markup, and indexes only the content.

For PDF or OpenOffice attachments, the Indexing and Search Service converters translate the format to text content. Additionally, Indexing and Search Service discards stop words such as "the." Some attachments that Indexing and Search Service indexes are actually saved to the attachment store. The reason for this restriction is that some attachments do not have thumbnail images and so it does not make sense to store them. For example, Indexing and Search Service does not store thumbnail images for .txt and .xml attachments. Indexing and Search Service does support indexing Microsoft Office documents, including Word, Excel, PowerPoint, and Visio, in the attachment store.

About Email Attachments and Levels of Support

Indexing and Search Service supports many types of email attachments, including:

- doc (Microsoft Word), html, jpeg, odf (any OpenOffice file)
- other (uncategorized), pdf, plain (plain text)
- ppt (Microsoft Powerpoint), rtf, vcf, vsd (Microsoft Visio)
- xls (Microsoft Excel), xml
- iwork (Apple iWork, including Pages, Numbers, and Keynote); Indexing and Search Service also extracts attachments that come through with AppleSingle or AppleDouble encoding
- audio, compress, image, video

Indexing and Search Service bases its categorization decision upon a combined analysis of the advertised MIME type, the attachment name's extension, and the actual software bits. Indexing and Search Service places unrecognized types into the "other" group. The doc, ppt, vsd, and xls designations are shorthand that includes the XML Office 2007 type files. That is, Indexing and Search Service categorizes docx files as doc.

Indexing and Search Service provides three levels of "support" for an attachment type. At the first level of support, Indexing and Search Service categorizes the attachment by file type and makes it searchable by type or by its advertised file name. The audio, compress, image, and video types are at this level (these files are not unpacked nor does Indexing and Search Service attempt to extract any text from them other than their name). At the second level of support, Indexing and Search Service does extract the text from the file and indexes it. Types at this level includes doc, html, plain, ppt, rtf, vcf, vsd, xls, and xml. At the third level of support, Indexing and Search Service extracts the text from the file and extracts or generates a file-specific thumbnail image. These include jpeg, odf, and pdf, and iwork. By default, Indexing and Search Service turns off thumbnails for these types.

Currently, Indexing and Search Service does not enable you to customize other format types for inclusion in its indexing service. If you do have an attachment that is in one of the supported formats, such as PDF, XML, plain text, and so on, then Indexing and Search Service extracts the text.

Manually Bootstrapping Indexing and Search Service Accounts

After installing Indexing and Search Service, you must "bootstrap" each account into the Indexing and Search Service store by running the *IndexSearch_home*/**bin/issadmin.sh --bootstrap** command. You can bootstrap accounts either manually or automatically. This section describes how to manually bootstrap accounts. For information on automatically bootstrapping accounts, see "Autoprovisioning Accounts".

Manually bootstrapping accounts gives you more control over how you allocate accounts. When you create an account by using the **issadmin.sh** command, you can select either the group or singleton status for an account, by using the **--group** and **--singleton** options. You can also use the **--accountlist** option to specify a file containing account information for one or more accounts used to repeat the command action over multiple accounts.

Bootstrapping an Account by Using the Default Allocation Policy

To bootstrap an Indexing and Search Service account:

1. Log in as or become superuser (**root**), or log in as the Indexing and Search Service user.
2. Run the **issadmin.sh --bootstrap** command with the **--user** and **--host** options, for example:

```
cd IndexSearch_home/bin
issadmin.sh --bootstrap --user user2 --host mailhost.example.com --runoptimizer
true
```

By default, the **issadmin.sh** command uses the system administration information configured in the **mail.imap.admin.username** parameter (from the *IndexSearch_home*/**etc/jiss.conf** file) and **mail.imap.admin.password** parameter (from the *IndexSearch_home*/**etc/jiss_passwd.conf** file) to establish access to the account on the Messaging Server message store.

If you specify only **--user** and **--host** options, the Indexing and Search Service store assigns the account being bootstrapped to a group by using the default allocation policy. For more information about the default allocation policy, see "Overview of the Indexing and Search Service Store Instance".

Bootstrapping an Account by Allocating to a Specific Singleton Group

The default allocation policy is convenient for a deployment that consists of small numbers of small accounts. For more control over account allocation, create accounts beforehand by using the **issadmin.sh --createaccount** option. Alternatively, you can use the **issadmin.sh --bootstrap** command with the **--group** option and the **--singleton** options to direct the bootstrap to use a specific group number and to require that the account be created as the only account in the group, respectively.

To bootstrap an Indexing and Search Service account to a specific singleton group:

1. Log in as or become superuser (**root**), or log in as the Indexing and Search Service user.
2. Run the **issadmin.sh --bootstrap** command with the **--group** and **--singleton** options, for example:

```
cd IndexSearch_home/bin
issadmin.sh --bootstrap --user user2 --host mailhost.example.com --group group2
--singleton --runoptimizer true
```

Bootstrapping an Account by Using the --accountlist Option

The **--accountlist** *file* option enables you to specify a file containing account information for one or more accounts used to repeat the command action over multiple accounts. In this case, when used with the **--bootstrap** option, it executes the command over a list of accounts, all within one executable command.

1. Log in as or become superuser (**root**), or log in as the Indexing and Search Service user.
2. Run the **issadmin.sh --bootstrap** command with the **--accountlist** *file* option, for example:

```
cd IndexSearch_home/bin
issadmin.sh --bootstrap --accountlist /tmp/users.conf --host
mailhost.example.com --threads 10
```

Removing an Account from Active State

To remove an account after it has been bootstrapped from the Active state for maintenance (such as deleting or modifying it):

```
issadmin.sh --setstate I --host host --user user
```

The account state is set to **I** for Inactive and is disabled from searching and real-time updates.

Bootstrapping Examples

The following examples illustrate different ways to construct the account file used by **issadmin.sh --accountlist** *file* commands. Each non-comment line in the **--accountlist** *file* must conform to the following syntax:

```
;groupnum;state;singleton;detail;hostname;username
```

where:

- *groupnum* is the number of the Indexing and Search Service group to which to assign the account
- *state* is the account state, one of a (active), i (inactive), or b (bootstrapping)
- *singleton* is the group to contain a single account
- *detail* overrides the value in the **--detail** option when used with the **issadmin.sh --checkaccount**, **--checkfolder**, and **--checkstore** options
- *hostname* is the name of the Indexing and Search Service index store host
- *username* is the email account of the user being bootstrapped

Only *username* is required in the **--accountlist** *file*, the other entries are optional. Such a file looks like this:

```
user1
user2
user3
```

When using only the *username* in the **--accountlist** *file*, the system bootstraps the users according to the value of the **issadmin.sh --setdefaulthostname** *host* command.

Bootstrapping Two Accounts

The following command bootstraps two accounts:

```
issadmin.sh --bootstrap --accountlist twoaccounts
```

where the file **twoaccounts** contains the following information:

```
# simple file of two accounts
;;;;;bco108.example.com;test@example.com
;;;;;bco108.example.com;admin
```

If the accounts were not created previously, then the command using the **twoaccounts** file creates them in the next available groups by using the default allocation policy. After the bootstrap, the same file can be used to check the accounts with the following commands:

```
issadmin.sh --accountinfo --accountlist twoaccounts
```

```
issadmin.sh --checkaccount --accountlist twoaccounts
```

Bootstrapping Five Accounts

In this example, the file **fiveaccounts** contains the following information, with two of the five accounts to be bootstrapped into specific groups:

```
# five accounts, two with specific group numbers
;;;;;bco108.example.com;test@example.com
;;;;;bco108.example.com;admin
;110;;;;bco108.example.com;bill
;111;;;;bco108.example.com;charles
;;;;;bco108.example.com;greg
```

Assuming none of these accounts has been created beforehand, the following command bootstraps them all, placing the first two and last accounts according to the default allocation policy, and the third and fourth in specific groups.

```
issadmin.sh --bootstrap --accountlist fiveaccounts
```

> **Note:** In this example, the order in which the accounts are bootstrapped is the order they appear in the file because the **--threads** *N* option was not specified (meaning use a single thread). The order in which accounts are bootstrapped could differ if you run the command using multiple threads. Thus, it is possible that the groups selected by the default allocation policy do not correspond to the order of the accounts in the file.

Setting Account State with Multiple Accounts

In this example, the file **fiveaccounts** contains the following information:

```
# fiveaccounts, two with state specified
;;;;;bco108.example.com;test@example.com
;;A;;;bco108.example.com;admin
;110;I;;;bco108.example.com;bill
;111;;;;bco108.example.com;charles
;;;;;bco108.example.com;greg
```

Assuming that all of these accounts have been created beforehand, the following command changes the state to Active of all accounts except **bill**. In the **fiveaccounts** file, **bill** is set to I (Inactive).

```
issadmin.sh --setstate A --accountlist fiveaccounts
```

The **--setstate** command does not use the group number, and the "state" value in the file overrides the value specified on the command line for account **bill**.

This file could also be used in the previous example, because the "state" field does not affect the behavior of the **--booostrap** command.

Autoprovisioning Accounts

This section describes how to use autoprovisioning to add new accounts to the Indexing and Search Service store automatically, how to selectively autoprovision accounts, and how to check for and recover from autoprovisioning errors.

Overview of Autoprovisioning

By using autoprovisioning, you make account creation less prone to error than when manually bootstrapping accounts. However, unlike the **issadmin.sh** command, which gives you control over the group or singleton status for an account, autoprovisioning simply creates the account in the next available index group in the Indexing and Search Service store by using the default allocation policy. For more information about the default allocation policy, see "Overview of the Indexing and Search Service Store Instance".

Caution: Before using autoprovisioning, ensure that you understand how it works to avoid creating undesirable effects on the Indexing and Search Service store.

When autoprovisioning is enabled, the system creates a new Indexing and Search Service account as follows:

1. An administrator creates a new email account in the Messaging Server message store.
2. When activity first occurs for the email account, indicating that the account is active, Messaging Server generates a JMQ notification. For example, when mail first arrives for this email account, the standard email folders such as **INBOX** are created in the message store, which in turn creates a notification.

Note: Autoprovisioning only applies when Indexing and Search Service sees a **create** event on the **INBOX**. Autoprovisioning does not occur if you restore an account with the Messaging Server **imsrestore** command.

---

3. The Indexing and Search Service **jmqconsumer** process, which is listening to the Messaging Server's Message Queue, receives the event notification and triggers autoprovisioning to occur for the account.
4. The Indexing and Search Service account is created in the next available index group in the Indexing and Search Service store by using the default allocation policy.

Note: If you maintain a list of your Indexing and Search Service accounts, understand that this list can become out of synchronization due the nature of autoprovisioning. When autoprovisioning is enabled, the system can create new accounts in the Indexing and Search Service store at any time. Thus, your list can become incomplete. Your list can also become incomplete whenever an account is explicitly deleted from the store. Ensure that your list is current by using the **issadmin.sh --listaccountlistfile** command to obtain all accounts currently in the Indexing and Search Service store. Use the **--accountlist** option to enhance the formatting of the account output.

Enabling Autoprovisioning

Autoprovisioning is controlled at the global level through the **iss.autoprovision.enabled** configuration parameter. Once you set the

iss.autoprovision.enabled parameter to **true**, when you add new email accounts and those accounts become active, Indexing and Search Service automatically provisions accounts for them in the Indexing and Search Service store.

To enable Indexing and Search Service autoprovisioning:

1. Edit the *IndexSearch_home***/etc/jiss.conf** file and set the **iss.autoprovision.enabled** configuration parameter to **true**.
2. Refresh the Indexing and Search Service configuration for the change to take place.

   ```
   issadmin.sh --refresh
   ```

Selectively Autoprovisioning Accounts

When autoprovisioning is enabled, the default setting creates accounts for which the trigger event is received. However, for a variety of reasons, you might want to restrict the set of accounts for which autoprovisioning creates an account. For example, in an Indexing and Search Service deployment of multiple hosts, you might want to autoprovision accounts across your hosts for load balancing purposes. Also, you do not want to create duplicate accounts on each host.

To restrict how to automatically provisions accounts across multiple hosts, use the following account selection configuration parameters:

- **iss.autoprovision.host.include.list**
- **iss.autoprovision.user.include.list**
- **iss.autoprovision.host.exclude.list**
- **iss.autoprovision.user.exclude.list**

Each **iss.autoprovision*.** parameter takes a value specified as a list of regular expressions. These parameters control which host names and user names are included and excluded from autoprovisioning on each Indexing and Search Service instance.

To choose which accounts are autoprovisioned for Indexing and Search Service:

1. Edit the *IndexSearch_home***/etc/jiss.conf** file and set the **iss.autoprovision*.** configuration parameters with the appropriate regular expression.

   Indexing and Search Service employs the following rules when evaluating these expressions:

   - Each expression is compared to the name of the account before it is autoprovisioned.
   - The host name of the account must match one of the "**host.include**" expressions, and none of the "**host.exclude**" expressions, to be autoprovisioned.
   - Similarly, the user name must match one of the "**user.include**" expressions, and none of the "**user.exclude**" expressions, to be autoprovisioned.
   - Each "**include**" expression is checked before any "**exclude**" expressions. Thus, if no match of any "**include**" expression occurs, the account is not autoprovisioned, and the "**exclude**" checks are unneeded.
   - The expression lists are applied left to right, so place expressions with more general patterns before expressions with more specific matching.

2. Refresh the Indexing and Search Service configuration for the change to take place.

   ```
   issadmin.sh --refresh
   ```

Using Account Selection Parameters

You use the **include** and **exclude** configuration parameters to restrict the set of accounts to be autoprovisioned on the Indexing and Search Service instance. If, for example, you want to avoid autoprovisioning all accounts in a domain named **abc.com**, you could set the following configuration parameter in each Indexing and Search Service instance configuration file for each instance where the domain should be excluded:

iss.autoprovision.user.exclude.list = .+@abc.com

Many more include and select configurations are possible. For more information about writing regular expressions, see the Java Class Pattern page at:

http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html

Simple Autoprovisioning Example

Consider a deployment of two Messaging Server hosts named **hostAM** and **hostNZ**, and two Indexing and Search Service instances named **issAM** and **issNZ**. Assume that all existing users whose names start with the letters "a" through "m" are hosted on **hostAM** and bootstrapped on **issAM**. All users whose names start with letters "n" through "z" are hosted on **hostNZ** and bootstrapped on **issNZ**. Further, assume that **issAM** is configured only to receive event notifications from **hostAM** and **issNZ** is configured to receive only event notifications from **hostNZ**.

To configure autoprovisioning for this example:

1. Set the **iss.autoprovision.enabled** parameter to **true** in the **jiss.conf** file on both **issAM** and **issNZ**.
2. Refresh the Indexing and Search Service configuration for the change to take place.

```
issadmin.sh --refresh
```

If the **iss.autoprovision.enabled** parameter were enabled on only one Indexing and Search Service instance, then only new accounts directed to that instance would be autoprovisioned. The default values for the other autoprovisioning parameters enable any account to be autoprovisioned, so no additional selection regular expressions need to be added.

Every new account added to Messaging Server hosts **hostAM** or **hostNZ** is autoprovisioned only in the corresponding Indexing and Search Service store because the event notifications sent for each account are received in only one Indexing and Search Service instance.

Complex Autoprovisioning Example

The previous example contains a fairly simple configuration. Now consider a more complicated approach in which both Messaging Servers hosts send event notifications to both Indexing and Search Service instances. In this example, the selection parameters are configured to distribute new accounts to either Indexing and Search Service instance for autoprovisioning.

Note: If an account is not created manually or autoprovisioned on an Indexing and Search Service instance, any events received for that account are ignored.

In this case, the selection criteria for which accounts to autoprovision on each Indexing and Search Service instance must partition the set of all accounts such that each account is indexed on exactly one of the two Indexing and Search Service instances. One way to partition this example is as follows:

- Set the following parameter on host **issAM**:

  iss.autoprovision.user.include.list = [a-m].+

- Set the following parameter on host **issNZ**:

  iss.autoprovision.user.include.list = [n-z].+

This configuration enables only account names starting with letters "a" through "m" to be autoprovisioned on instance **issAM**, and only those starting with letters "n" through "z" to be autoprovisioned on instance **issNZ**. Every account matches on exactly one instance, and all possible accounts are included somewhere.

Now consider how, over time, instance **issAM** has many more accounts than **issNZ**. To help balance the load, you decide to autoprovision new accounts starting with the letters "d" and "j" on **issNZ** from now on. The configuration parameters can be changed as follows:

- Set the following parameters on host **issAM**:

  iss.autoprovision.user.include.list = [a-m].+

  iss.autoprovision.exclude.include.list = d.+, j.+

- Set the following parameter on host **issNZ**:

  iss.autoprovision.user.include.list = [n-z].+, d.+, j.+

Several other possibilities exist to achieve this change with regular expressions. This example shows the utility of the **exclude** parameter. This configuration change does not affect the location of previously indexed accounts, just new ones autoprovisioned after the configuration is refreshed. Both Indexing and Search Service instances eventually contain accounts starting with the letters "d" and "j" as new accounts are autoprovisioned.

About Error Handling in Autoprovisioning

This section describes the ways in which Indexing and Search Service checks for autoprovisioning errors.

Errors in Regular Expressions

Indexing and Search Service checks the autoprovisioning parameters that specify regular expression lists for proper syntax when the configuration is refreshed. If any of the expressions are malformed, an error message is logged, and that element of the list is omitted, allowing the remaining expressions to be used to include or exclude accounts. This evaluation could result in unintentionally autoprovisioning or excluding some user or host names, but does enable the processing to continue.

However, while each of the regular expressions may be well formed, account names or hosts may not be matched as intended. No checking is performed so that all accounts either match any include or exclude criteria. Thus, you must ensure that all account names and hosts for which autoprovisioning is intended match some include criteria, and only accounts to be excluded match the exclude criteria. If a trigger event for an account arrives, and the account fails to match any include criteria or matches any exclude criteria, then the account is not autoprovisioned, and a WARNING level message is logged. If you find that accounts are not being autoprovisioned as

expected, these message can help you understand whether the selection parameters are wrong.

Time Out Errors

Because autoprovisioning requires creating then bootstrapping an account, this processing is expensive relative to most other events, such as changing flags, and so on, and requires more synchronization than usual. Normally autoprovisioning completes in a few seconds. However, when the rate of event processing is high, an attempt to autoprovision a new account might time out, and not complete, to avoid delaying other event processing.

This time out failure might leave the account in an unfinished state, preventing new events processing for the account. Indexing and Search Service attempts to retry the autoprovisioning that timed out automatically in a separate thread, to limit interference with other processing. This retry might be repeated a few times if the load is high. In general, this retry enables the autoprovisioning to complete. A retry needs some added minutes to complete after the original time out occurred. A WARNING message is logged when a time out occurs, and additional log messages show the progress of any retry.

Autoprovisioning Best Practices

The previous examples illustrate the use of the selection configuration parameters. In practice, deployments likely have more than two Messaging Server hosts and Indexing and Search Service instances, and thus, they require a more complex autoprovisioning configuration.

Follow these practices for autoprovisioning:

- Keep the selection criteria as simple as practical to ensure that you cover the full range of account names and hosts.
- If you disable autoprovisioning on any specific instance, check that any accounts it had been autoprovisioning are reflected in the configuration of other instances.
- Because the **jiss.conf** configuration file for each Indexing and Search Service instance is local to the instance, keep a copy of the autoprovisioning parameters for all instances in a global file so that you can understand and track changes for the selection criteria for new accounts.

Administering Periodic Automatic Synchronization

This section explains how to control periodic automatic synchronization, which is how Indexing and Search Service corrects accounts that have become out-of-sync with the message store. Administering periodic automatic synchronization involves:

- Enabling Periodic Autosync
- Selecting Appropriate Periodic Autosync Configuration Values
- Tuning Periodic Autosync
- Monitoring Autosync Progress
- Automatically Bootstrapping Missing Accounts
- Enabling Periodic Autobootstrap
- Selecting Conditions for Events to Track in Periodic Autobootstrap
- Tuning Periodic Autobootstrap

Overview of Periodic Automatic Synchronization

As the Messaging Server message store changes, it sends event notifications to Indexing and Search Service so that the index information for each account can be updated. At times, changes to the message store might not get reflected in the Indexing and Search Service store. For example, a folder in an account might be reconstructed in the message store, changing the assignment of UID numbers to emails, without notification to Indexing and Search Service. While Indexing and Search Service can detect the UID validity mismatch during subsequent event processing, and re-bootstraps the folder to synchronize it, a long time can pass before an event triggers the correction. Indeed, there is no guarantee any event involving the reconstructed folder is ever received, so the mismatch remains uncorrected, and searches on such a folder continually produce invalid results. Also, under very heavy load, notification messages can be delayed or lost, causing the index to become out of synchronization with the message store.

To correct the index after such unseen changes to the message store, Indexing and Search Service provides a periodic check of each account against the message store. If any account is found to be not synchronized with the message store, Indexing and Search Service performs the necessary **--sync** or **--bootstrap** commands to correct the index automatically. Indexing and Search Service refers to this process as *periodic autosync*.

Enabling Periodic Autosync

Periodic autosync is controlled at the global level through the **iss.indexsvc.periodic.autosync.enabled** configuration parameter. Once the **iss.indexsvc.periodic.autosync.enabled** parameter has been set to **true**, Indexing and Search Service creates a list of all known active accounts in the index, and orders them in a list to be processed. During each time period, as specified by the configuration parameter **iss.indexsvc.periodic.autosync.interval**, a set of accounts is removed from the list and checked against the message store then synchronized if needed. The number of accounts in the set is determined by the value of the configuration parameter **iss.indexsvc.periodic.autosync.count**. After enough periods have completed to exhaust the list of accounts, the list is recreated and the periodic autosync continues again. In this way all accounts are eventually checked and synchronized over a period under control of the parameter choices.

To enable periodic autosync:

1. Edit the **jiss.conf** file and set the **iss.indexsvc.periodic.autosync.enabled** configuration parameter to **true**.
2. (Optional) Set the **iss.indexsvc.periodic.autosync.interval** configuration parameter to specify the number of seconds to wait from the start of one period until the start of the next autosync period. The default is 300 seconds.
3. (Optional) Set the **iss.indexsvc.periodic.autosync.count** configuration parameter to specify the number of accounts to process from the list of accounts yet to be synced during each period. The default is 1000.
4. Refresh the Indexing and Search Service configuration for the change to take place.

```
issadmin.sh --refresh
```

5. For tuning suggestions, see "Tuning Periodic Autosync."

Selecting Appropriate Periodic Autosync Configuration Values

The periodic autosync of accounts imposes overhead on both the Indexing and Search Service and Messaging Server hosts. Choose values for the autosync configuration parameters to balance the trade-off between rapid cycling through each account and the cost of load on the system, which can increase response time. The total time to check every active account in the Indexing and Search Service store can be estimated by dividing the number of such accounts by the value of the "count" parameter and multiplying by the "interval". (This assumes that "count" accounts can always be checked or synchronized in "interval" seconds.) For example, if there are 10000 active accounts, and "count" is 100 and "interval" is 600 seconds (10 minutes), all accounts are checked in approximately 16.7 hours (10000/100 * 600 = 60,000 seconds, or 16.7 hours).

Autosync processing time depends on many factors, including:

- How many accounts are checked in a group
- How many emails are contained in each account
- How many accounts are found to have problems that must be synchronized (if any)

You cannot tell in advance what load this imposes on either the Indexing and Search Service or Messaging Server hosts. As the "count" value is increased and the "interval" value is reduced, the elapsed time to check all accounts decreases, but the average overhead imposed on the servers increases.

Therefore, set the initial values for "count" and "interval" so that the rate of processing is no faster than about 1 account every 5 seconds. (The rate in the preceding example shows 100/600 or about 1 every 6 seconds.) By running at such a conservative rate for a while, the overhead on the Indexing and Search Service and Messaging Server hosts can be monitored, and after some experience, faster rates can be attempted until the appropriate balance can be established.

The periodic autosync feature writes INFO messages at regular times to the log file. You can use this information to see how many accounts have been checked, how many were valid (that is, did not need to be synchronized), and how many accounts were synchronized because of a problem. From this information and the amount of idle time between periods (seen in the log as how long the autosync sleeps after a period), you can adjust the "count" and "interval" to the desired frequency.

Periodic Autosync Configuration Examples

The following examples show the factors involved in configuring periodic autosync.

Large Interval, Small Count, Significant Idle Time

The previous example (10000 accounts, 100 checked per period, 600 seconds between periods) demonstrates a relatively slow rate of processing that should have low impact on both the index and message stores. If the time it takes to check the accounts in each period is short compared to the interval between periods, then the load from the autosync is "pulsed," that is, it becomes higher for a period (while the accounts are being checked) and lower when the period is done and is waiting for the next one to begin. If the load during the working phase of the period is within acceptable bounds for your deployment, the "interval" value can be reduced to shorten the wait. Reducing the "interval" value could reduce the overall elapsed time significantly, and maintain a steadier load on the hosts.

Small Interval, Large Count, No Idle Time

A more aggressive configuration (10000 accounts, 500 checked per period, 300 seconds between periods) demonstrates how the load might increase. If all the "count" accounts cannot be checked within the "interval," then the autosync runs continuously with no wait time between periods. Running autosync continuously might be acceptable if your deployment has enough resources to handle the load. Under these conditions all accounts are checked in 10000/500 * 300 = 6000 seconds or 1.7 hours compared to 16.7 with the previous example if the 500 accounts can be checked within 300 seconds. Checking all accounts within this shorter time frame might be possible under some conditions, such as many accounts with relatively small numbers of emails, or during periods of relatively low activity so the contents of the accounts do not change much. With such a configuration, the load on the system is much more sensitive to arrival of new email or heavy user search activity.

Tuning Periodic Autosync

This section describes ways in which you can tune and monitor periodic autosync.

Controlling the Number of Threads for Periodic Autosync

The number of threads used to run the checks affects the overhead and speed of autosync processing. To set the number of threads used to run the checks, use the **iss.indexsvc.periodic.autosync.thread.count** configuration parameter. The default value is 10 threads. By decreasing the number of threads used, the elapsed time of processing within the period tends to increase, spreading the load on both the Indexing and Search Service and Messaging Server hosts over a longer period. Increasing the number of threads increases the amount of work done in parallel, which tends to increase the instantaneous load, but decreases the elapsed time to complete all the checks in the period.

Controlling the Level of Account Checking

The **--checkaccount** and **--sync** options that implement autosync processing can be modified by the **iss.indexsvc.periodic.autosync.flags.enabled** parameter to perform more or less detailed checking of accounts. When this parameter has the value **true**, autosync behaves as if the **--detail** *flags* option has been specified for each account. Setting this parameter to **true** provides more detailed checking (that is, detects more out-of-sync problems) at the cost of additional overhead to both Indexing and Search Service and Messaging Server. If the overhead is excessive, set the value of this parameter to **false** to reduce the cost of using autosync. By default, the value for this parameter is **true**.

Monitoring Autosync Progress

The Index Service writes INFO level log messages that describe the autosync progress. Whenever the cycle begins, the list of all accounts participating in the autosync is generated, and messages such as the following are logged:

```
Sat Jul 07 18:39:22 PDT 2012 com.sun.comms.iss.store.AutoSyncManager
findAllCandidates INFO: findAllCandidates: time to create list of 53995
candidates : 339ms

Sat Jul 07 18:39:22 PDT 2012 com.sun.comms.iss.store.AutoSyncManager
findAllCandidates INFO: findAllCandidates: current number of accounts found by
state: total: 60000 active: 53995 inactive: 3 bootstrap: 2 unknown: 6000
```

The breakdown of accounts by account state can help identify problem accounts, such as those left in the bootstrap, inactive, or unknown state.

When a period of work begins, Indexing and Search Service logs how many accounts it is processing as a group. For example:

```
Sat Jul 07 23:14:53 PDT 2012 com.sun.comms.iss.indexapi.IndexService
runAutoSyncLoop INFO: Periodic autosync summary: processing 250 accounts every 300
seconds will complete 46745 accounts currently queued in about 56100 seconds
(15.5 hours)
```

If the time estimate for completing all the active accounts is too large, consider changing the configuration parameters **iss.indexsvc.periodic.autosync.interval** or **iss.indexsvc.periodic.autosync.count** to speed up the cycle.

As the next period of work begins processing, the names of all accounts to be checked in the period are logged. For example:

```
Sat Jul 07 23:14:53 PDT 2012 com.sun.comms.iss.indexapi.IndexService runAuto INFO:
250 autosync candidates found: all users share host sc11.example.com unless
explicitly present:

candidate: 3010340@colorone.org, 3010346@colorone.org, 3010348@colorone.org,
3010313@colorone.org, 3010324@colorone.org, 3010350@colorone.org,
3010353@colorone.org, 3010344@colorone.org, 3010357@colorone.org,
3010322@colorone.org

candidate: 3010351@colorone.org, 3010364@colorone.org, 3010363@colorone.org,
3010362@colorone.org, 3010367@colorone.org, 3010360@colorone.org,
3010342@colorone.org, 3010372@colorone.org, 3010347@colorone.org,
3010358@colorone.org

...

candidate: 3010666@colorone.org, 3010679@colorone.org, 3010676@colorone.org,
3010661@colorone.org, 3010678@colorone.org, 3010674@colorone.org,
3010686@colorone.org, 3010685@colorone.org, 3010688@colorone.org,
3010691@colorone.org
```

Additional log messages appear as each individual account is checked or synchronized. Every 10 periods, a summary of progress since the beginning is logged. For example:

```
Sat Jul 07 22:43:49 PDT 2012 com.sun.comms.iss.store.AutoSyncManager
logAutoSyncStats INFO: statistics: number of accounts checked: 6250 number of
accounts OK: 5451 number of accounts synced: 799 number of accounts skipped
because another command was running: 0 number of accounts skipped because folder
not found: 0 number of accounts skipped because authentication failed: 0 number of
accounts failed for other reasons: 0
```

These totals continue to accumulate over multiple cycles of active accounts. If the number of accounts synchronized is low relative to the number of accounts that are "OK," consider modifying the configuration parameters to lengthen the time to complete the full cycle of accounts to reduce the overhead due to autosync.

Automatically Bootstrapping Missing Accounts

Similar to the situation for active accounts described in "Overview of Periodic Automatic Synchronization", accounts which have not been bootstrapped or autoprovisioned might cause event notifications of changes to such accounts to fail. To

correct these accounts, you can use the *autobootstrapping* feature, which is useful if you want your deployment to index uninitialized accounts under such conditions.

Enabling Periodic Autobootstrap

Periodic autobootstrap is controlled at the global level through the **iss.indexsvc.periodic.autobootstrap.enabled** configuration parameter. When **iss.indexsvc.periodic.autobootstrap.enabled** is set to **true**, Indexing and Search Service tracks errors that occur in event notification events to create a list of accounts that should be bootstrapped. Errors are counted per account. Any event that fails because the account cannot be found is counted, and if enough such events occur, then the account is added to the list to be automatically bootstrapped. To avoid excessive load from the bootstrap process, accounts on this list are scheduled periodically to be bootstrapped at regular intervals, as specified by the **iss.indexsvc.periodic.autobootstrap.interval** configuration parameter. The number of accounts on the list bootstrapped during each period is determined by the value of the **iss.indexsvc.periodic.autobootstrap.count** configuration parameter. Once an account has been bootstrapped it is removed from the list, and event errors caused because its index was missing should no longer occur.

To enable periodic autobootstrap:

1. Edit the **jiss.conf** file and set the **iss.indexsvc.periodic.autobootstrap.enabled** configuration parameter to **true**.
2. (Optional) Set the **iss.indexsvc.periodic.autobootstrap.interval** configuration parameter to specify the number of seconds to wait from the start of one period until the start of the next autobootstrap period. The default is 300 seconds.
3. (Optional) Set the **iss.indexsvc.periodic.autobootstrap.count** configuration parameter to specify the number of accounts to process from the list of accounts yet to be bootstrapped during each period. The default is 500.
4. Refresh the Indexing and Search Service configuration for the change to take place.

   ```
   issadmin.sh --refresh
   ```

Selecting Conditions for Events to Track in Periodic Autobootstrap

You can choose which error conditions are used to select accounts for autobootstrapping, based on the kind and number of events that fail. To control what kind of events can trigger automatic bootstrapping of accounts, the configuration parameter **iss.indexsvc.periodic.autobootstrap.triggerlist** enables you to specify that all, none, or some specific kinds of event errors should be counted toward autobootstrapping. The default value of this parameter is **ALL** or empty, indicating all event errors are counted to trigger autobootstrapping. If the value is **NONE**, then no event errors are counted. You can also specify a comma-delimited list of specific event names, taken from the following fixed set:

NewMsg, **UpdateMsg**, **ChangeFlag**, **Copy**, **Create**, **Delete**, **ExpungeMsg**, **Rename**

Only event errors for any of the event kinds specified are counted. (All values specified in this parameter are treated as case insensitive.)

Once the kind of event errors used to trigger automatic bootstrapping has been established, the count of how many events before the account is submitted for autobootstrapping is determined by the value of the configuration parameter **iss.indexsvc.periodic.autobootstrap.triggercount**. The default value is **1**.

Tuning Periodic Autobootstrap

This section describes ways in which you can tune and monitor periodic autobootstrap.

Controlling the Number of Threads for Autobootstrap

The number of threads used to run the bootstrap on multiple accounts affects the overhead and speed of autobootstrap processing. To set the number of threads used to run the bootstraps, use the **iss.indexsvc.periodic.autobootstrap.thread.count** configuration parameter. The default value is 10 threads. By decreasing the number of threads used, the elapsed time of processing within the period tends to increase, spreading the load on both the Indexing and Search Service and Messaging Server hosts over a longer period. Increasing the number of threads increases the amount of work done in parallel, which tends to increase the instantaneous load, but decreases the elapsed time to complete all the bootstraps in the period.

Managing the Autobootstrap Account List

Both the periodic autosync and autobootstrap features maintain a list of accounts to be processed. Unlike the autosync account list which is volatile, the autobootstrap account list is preserved across service restart. Thus, the accounts can continue to be bootstrapped when the server starts again. You can also manage the autobootstrap processing dynamically as it progresses. You do so by running **issadmin.sh** commands to list, delete, and add accounts to the list. See the **issadmin.sh --autobootaccounts**, **--deleteautobootlist**, **--setautobootlist**, and **--unsetautobootlist** commands in "Indexing and Search Service Command-Line Utilities" for how to review and modify the list of accounts to be autobootstrapped.

Monitoring Autobootstrap Progress

The Index Service writes INFO level log messages that describe the progress of the autobootstrap. Whenever a new work period begins, the list of all accounts participating in the autobootstrap at that point is generated, and messages such as the following are written to the log file:

```
Thu Jul 05 11:51:27 PDT 2012 com.sun.comms.iss.store.AutoSyncManager
selectPeriodicBootstrapCandidates INFO: selectPeriodicBootstrapCandidates: 134
accounts checked, 125 accounts selected

Thu Jul 05 11:51:27 PDT 2012 com.sun.comms.iss.indexapi.IndexService
runAutoBootLoop INFO: 5500 accounts currently awaiting bootstrap
```

This example shows that the number of accounts checked (134) is greater than the number selected (125). Possible reasons could be that some accounts in the original list were found to exist already, and so were not bootstrapped, or that an account had been added to the list multiple times. This situation occurs because accounts can be created through autoprovisioning or manually while waiting to be autobootstrapped.

The specific accounts to be bootstrapped in this period are then written to the log file as follows:

```
Thu Jul 05 11:51:27 PDT 2012 com.sun.comms.iss.indexapi.IndexService runAuto INFO:
125 autoboot candidates found: all users share host sc11.example.com unless
explicitly present:

candidate: 3026464@colorone.org, 3016948@colorone.org, 3005936@colorone.org,
3026633@colorone.org, 3048656@colorone.org, 3026604@colorone.org,
3059277@colorone.org, 3016072@colorone.org, 3048919@colorone.org,
```

```
3016203@colorone.org

candidate: 3048186@colorone.org, 3026696@colorone.org, 3026869@colorone.org,
3059255@colorone.org, 3005852@colorone.org, 3005295@colorone.org,
3005727@colorone.org, 3026976@colorone.org, 3059929@colorone.org,
3059432@colorone.org

...

candidate: 3016352@colorone.org, 3048236@colorone.org, 3037843@colorone.org,
3005418@colorone.org, 3016549@colorone.org, 3048754@colorone.org,
3037213@colorone.org, 3005932@colorone.org, 3016985@colorone.org,
3059413@colorone.org

candidate: 3059013@colorone.org, 3005853@colorone.org, 3016487@colorone.org,
3048117@colorone.org, 3059800@colorone.org
```

Additional log messages appear as each individual account is bootstrapped. When all the accounts in the group have been bootstrapped, bootstrapping starts on another group after the configured interval has elapsed, or starts immediately if the time to complete the group exceeds the interval.

Replicating Autobootstrap Events to a Standby Server

You can configure Indexing and Search Service to use a standby Indexing and Search Service server for autobootstrapping accounts. The bootstrap event on the standby server is delayed so that it only runs when the main Indexing and Search Service server is done bootstrapping the user.

To replicate autobootstrap events to a standby server:

1. Set up the Indexing and Search Service standby server with the same configuration as the main server, including:
   - Same **iss.imq.user** and **iss.imq.password**
   - Same **instance.name** and the derived parameter **iss.indexsvc.dst.name**
2. On the main Indexing and Search Service server, set the following configuration parameter:

   `iss.bootstrap.standby = host:port`

   where:

   host is the host name of the standby Indexing and Search Service server

   port is the port number on which the Java Message Queue server is listening (default is 7676)
3. If you are not using the default instance names, then on the main Indexing and Search Service server, set the following configuration parameter:

   `iss.bootstrap.queue = Index`*`instance.name`*

   where:

   Index*instance.name* is the destination queue name on the standby instance (can be related to the host name, but no dashes are allowed)

Automatically Removing Orphaned Accounts

This section describes how you configure Indexing and Search Service to automatically remove Indexing and Search Service accounts that no longer have a corresponding Messaging Server account.

Overview of Automatically Removing Orphaned Accounts

An Indexing and Search Service account that no longer appears to have a corresponding Messaging Server account is "orphaned." This situation can occur, for example, when you move Messaging Server user accounts between message stores, for load balancing purposes, without updating Indexing and Search Service. The corresponding Indexing and Search Service accounts are orphaned because they are not automatically moved with the Messaging Server accounts. (Refer to the use of the **issrehostuser.sh** scripts to move the Indexing and Search Service account when the Messaging Server **rehostuser** command is used.) Thus, the Indexing and Search Service account remains in its original location, no longer synchronized with the Messaging Server account that moved.

Before Indexing and Search Service 1.0.5.18.0, correcting an orphaned account required manually updating the Indexing and Search Service store by moving the account (by using the **issadmin.sh --export** and **--import** commands), or by deleting the old account then bootstrapping it on the new store. Indexing and Search Service 1.0.5.18.0 and greater provides an automated, configurable mechanism to remove the orphaned account from the proper Indexing and Search Service store. Autoprovisioning could then be used to index the account on its new host (with either **autobootstrap**, as described in "Enabling Periodic Autobootstrap" or with **autosync**, as described in "Enabling Periodic Autosync").

To remove orphaned accounts automatically, you must set both the global autosync parameter, **iss.indexsvc.periodic.autosync.enabled**, and the **iss.indexsvc.periodic.autodelete.enabled** parameter to **true**. The **iss.indexsvc.periodic.autosync.enabled** parameter enables Indexing and Search Service to detect orphaned accounts. The Indexing and Search Service autosync process scans for accounts that are not in the Active state, as they are potentially orphaned. The accounts are placed on a list of candidates to check for autodelete. The **iss.indexsvc.periodic.autodelete.enabled** parameter enables Indexing and Search Service to remove candidate accounts from the store periodically.

This list of potentially orphaned accounts to be deleted is processed periodically, based on the value of the configuration parameter **iss.indexsvc.periodic.autodelete.interval**. During each autodelete period, the list of accounts to be deleted is scanned. If all conditions for autodelete are satisfied for an account, that account is deleted from the store. Conditions can be set for autodelete and delay time to allow for administrator intervention.

During an autodelete scan, each candidate account is checked to confirm that it cannot be found in Messaging Server. If the account cannot be found, then the last transaction date of the account is compared to the value in the configuration parameter **iss.indexsvc.periodic.autodelete.purge.interval** to determine if the account has been recently modified. If the account has been modified more recently than the purge interval, then the account is returned to the candidate list to be checked again during the next scan. If the account has not been modified, it is deleted from the Indexing and Search Service store.

Enabling Automatic Removal of Orphaned Accounts

The configuration parameter **iss.indexsvc.periodic.autosync.enabled** controls whether Indexing and Search Service automatically deletes orphaned accounts. To enable the autodelete feature, the value for **iss.indexsvc.periodic.autosync.enabled** must be **true**. When the value is **false**, the autodelete feature is disabled.

To automatically remove orphaned accounts:

1. In the *IndexSearch_home***/etc/jiss.conf** file, verify that the value of the **iss.indexsvc.periodic.autosync.enabled** parameter is set to **true** (default is **false**).
2. In the *IndexSearch_home***/etc/jiss.conf** file, verify that the value of the **iss.indexsvc.periodic.autodelete.enabled** parameter is set to **true** (the default).
3. Use the **iss.indexsvc.periodic.autodelete.interval** parameter to specify the interval in seconds between autodelete scans. During each autodelete scan, Indexing and Search Service reviews the list of accounts to be deleted. If all conditions for autodelete are met, Indexing and Search Service deletes an account from the store. By adjusting the conditions for autodelete and the time between scans, you can allow extra time to make additional changes to the accounts that are candidates for autodelete.
4. Use the **iss.indexsvc.periodic.autodelete.purge.interval** to specify the purge interval in seconds. This interval is the time that must have elapsed since the last modification to an account.
5. Refresh the Indexing and Search Service configuration for your changes to take effect:

   ```
   issadmin.sh --refresh
   ```

Disabling Automatic Removal of Orphaned Accounts

If you set either the **iss.indexsvc.periodic.autosync.enabled** parameter or **iss.indexsvc.periodic.autodelete.enabled** parameter to **false**, then the autodelete feature is disabled. Setting **iss.indexsvc.periodic.autodelete.enabled** to **false** disables only the autodelete processing, not the candidate detection. Setting **iss.indexsvc.periodic.autosync.enabled** to **false** disables all autosync processing, including the candidate detection, so no more accounts are autodeleted.

To disable automatic removal of orphaned accounts:

1. In the *IndexSearch_home***/etc/jiss.conf** file, verify that the value of the **iss.indexsvc.periodic.autodelete.enabled** parameter or the **iss.indexsvc.periodic.autosync.enabled** parameter is set to **false**, depending on how you want to disable autodelete processing.
2. Refresh the Indexing and Search Service configuration for your changes to take effect:

   ```
   issadmin.sh --refresh
   ```

Managing Out-of-Sync State Information

This section describes ways to manage accounts that have gotten out-of-sync with their Messaging Server stores.

Integrating JMQConsumer Information with Autosync

The autosync feature integrates information from JMQConsumer to decide how to check and synchronize accounts. The time required to completely synchronization out-of-sync accounts is reduced in the following ways:

- JMQConsumer might drop an event for an account when the size of its events backlog reaches the maximum allowed, or when other internal data limits are reached. When an event is dropped for an account, autosync is notified and the account is identified to be out-of-sync. The result is that the account can be synchronized without the initial **--checkaccount** step.
- The **--checkaccount** command used by autosync halts when any folder is first detected to be out-of-sync. Halting **--checkaccount** avoids the overhead involved in checking the rest of the account before starting the sync process.

An event may be skipped in JMQConsumer when the processing of the account detects that the event has already been incorporated, for example, by a recent **--sync** or **--bootstrap** command. Skipping events that are already incorporated avoids unnecessary overhead and helps reduce the event backlog quickly.

The autosync feature sends information about an account synchronization to JMQConsumer, including the time of the synchronization and the NEXTUID value for each folder, which indicates when and how the folder last changed. This information enables JMQConsumer to skip any events in the account backlog that occurred before the synchronization, because these have already been incorporated into the index for the account.

Output from the periodic autosync in the Index service log reflects the current rate of progress more accurately. The estimated time for completing the accounts remaining on the autosync candidates list is based on the actual time required for the previous period instead of the configured interval (which may be much less than the actual time).

Statistics for periodic autosync are added to the Index server statistics file. Values tracked include:

- Number of candidates added (from JMQConsumer dropping events)
- Number of accounts checked and synchronized
- Number of candidate accounts that were skipped instead of being checked and synchronized
- Various types of failures in the synchronization process

These statistics are updated periodically with the other Indexing and Search Service statistics to provide additional insight into how often accounts need to be synchronized.

Accessing Account Event Backlog Information

Information about the backlog of unprocessed events held in the JMQConsumer service includes more details per account, a more detailed periodic summary, and a command-line query feature for quickly selecting details about specific subsets of the backlog information.

Logging Periodic Account Event Backlog Information

The periodic backlog queue checking in JMQConsumer runs in an independent thread that loops at regular intervals instead of running in the same thread that reads new

messages from the broker. The time interval between sampling is regular, overhead is reduced, and the reading thread is not blocked while it processes the event backlog.

The output from the periodic checking of the event queue backlog in the JMQConsumer service is redirected into separate statistics log files, similar to how statistics are generated in the Indexing and Search Service logs. The information recorded includes details about the backlog. The accounts listed are grouped by state, so that all accounts in the Active state are listed together, as are those in the Inactive, Bootstrap, and Unknown states. For individual accounts, counts are kept of the number of events dropped and skipped.

The summary of the backlog includes the number of dropped events, with **ChangeFlag** events and others counted separately. Totals for the number of events received, finished, failed, and skipped, and the number of events received for which there was no account found, are logged periodically, including the instantaneous rate per second for each statistic.

Using issadmin.sh to Query Account Event Backlog Information

The periodic logging of the backlog displays the state of all accounts for which there are events waiting to be processed at regular intervals.

Alternatively, you can use the **issadmin.sh --listbacklog** command to dynamically query specific accounts that require follow-up. With no other arguments, the **issadmin.sh** command displays information for the entire backlog that is similar to what is periodically logged. To sort specific data from the large backlog output, you can use the following selection mechanisms:

- To select by *account name*: Use the **--user** and **--host** options to select a single account, or the **--accountlist** option with a file containing multiple account names. (Although the **--threads** and **--continueonerror** options can be specified with the **--accountlist** option, these options have no effect when used with the **--listbacklog** command.)
- To select by *state of the account*: Use the **--selectstate** option to specify a comma-delimited list of states from the set A, B, I, and X, representing Active, Bootstrap, Inactive, and Unknown respectively. The output shows the counts of events that were skipped and dropped for each account.
- To select by *age of events*: Use the **--selecttime** *H*[:*M*] option to specify the number of hours and minutes that an event must have been in the backlog before it is considered a match. Use this option to find accounts with older events while ignoring accounts whose events have been queued recently. To specify a time less than one hour, use zero for the number of hours.

Use these selection mechanisms together to return results that contain only accounts with events in their backlog that satisfy all the selection criteria.

Note: You can only use the **--listbacklog** command when the services are running.

Rehosting and Deleting Accounts

When rehosting accounts between Messaging Server hosts, the source account is deleted from one Messaging Server message store and created in another. The **issrehost*** scripts in the Indexing and Search Service **examples** directory are provided to automate the corresponding process of rehosting the Indexing and Search Service index between Indexing and Search Service stores. The **issrehost** process includes

deleting the old account in the index and creating the new one. These scripts properly sequence the steps to ensure that the Indexing and Search Service account information is created.

Neither the **rehostuser** nor **imsrestore** Messaging Server commands cause an event notification that triggers autoprovisioning. When you use these commands, take care to explicitly create any new Indexing and Search Service accounts on the destination Indexing and Search Service store by using the steps in the **issrehost*** scripts, or by using the **issadmin.sh --createaccount** and **--bootstrap** commands to index such accounts.

Whenever an account is removed from the Messaging Server message store (by using the **mboxutil -d** command), the data in all folders is deleted, and event notifications so generated cause the index data in Indexing and Search Service also to be deleted. However, the account itself is not deleted from Indexing and Search Service, just all the data. Use the **issadmin.sh --deleteaccount** command to remove the account from the Indexing and Search Service store afterward.

Sometimes, you might need to delete an Indexing and Search Service account during maintenance (for example, to rebootstrap an account that has become corrupted or that cannot be synchronized by using the **--sync** command). If the account is also being deleted or restored on the Messaging Server message store during this period, the autoprovisioning feature might be triggered if the INBOX gets recreated. To avoid unwanted interference in such cases, create the Indexing and Search Service account explicitly again after you delete it by using the **issadmin.sh --createaccount** command before you attempt other maintenance on the account.

6

Managing and Monitoring the Indexing and Search Service Store

This chapter describes how to administer the Oracle Communications Indexing and Search Service store.

About Account Placement in the Indexing and Search Service Store

When you create an Indexing and Search Service account, it is uniquely assigned to a single Indexing and Search Service store instance, which contains all the account's index and data information. To manage the size of the store and location of accounts, you must be able to move an account from one store instance to another. This process consists of two steps:

1. Exporting the information from its current store instance
2. Importing the exported information into another store instance

The export step creates a snapshot of the account information in a file structure separate from the store instance. This structure can be used as an archive and as a backup. An exported account is therefore effectively a recovery from that backup.

Terminology and Conventions

The following terms apply to the process of exporting and importing accounts:

- **Account.** Set of user information uniquely identified by "user" and "host" strings.
- **Account snapshot.** All information for an account captured at a given time by the **issadmin.sh --export** command.
- **Snapshot repository.** Directory containing zero or more account snapshots. If you do not specify a snapshot repository by using the **--eximpath** option to the **issadmin.sh** command, the system uses the default snapshot repository to save snapshots.

Exporting and Importing Accounts Restrictions

The following restrictions are applicable to exporting and importing Indexing and Search Service accounts:

- When exporting and importing an account, ensure that the account state is set to Inactive.

  This setting ensures the validity of information while the export and import is occurring.

- You can only export or import one account per command.

  However, you can use the **--accountlist** *file* option to export or import multiple accounts in a single command, if the values specified for all the other command line options used are the same for all accounts in the list (for example, options like **--rehost** and **--eximpath**).

- During an export and import, you can move the account but not rename it.

  The user name must not change during the export and import process. To move the account to a different host, run the **issadmin.sh** command with the **--rehost** option. If the user name changes, the index records cannot be reused. The user name is treated as a new account, which you must then bootstrap.

- The administrator is responsible for managing the snapshot repository contents.

  Indexing and Search Service does not contain any automatic process for deleting old export snapshots. As you add new snapshots, disk space is used until you perform a cleanup of old snapshots. The **eximsummarylist** summary file, located in the snapshot base directory, can also be cleaned up by simply editing it. The **eximsummarylist** file contains information for all the users in one snapshot directory, and a roll-up of all the export commands (as opposed to being overridden from a new **issadmin.sh --export** command). If you do edit **eximsummarylist**, ensure that you do not invalidate the basic structure, because locating the proper snapshot to **--import** depends on its integrity. This file also serves as a log record of each **--export** that occurred in the snapshot repository.

Exporting and Importing Accounts Example

In the following example, the user mailbox is initially on **mailhostA** and the Indexing and Search Service account is on **isshostA**. You are moving the mailbox to **mailhostB** and the Indexing and Search Service account to **isshostB**.

1. From **isshostA**, set the account to Inactive and export the account.

   ```
   (isshostA)cd IndexSearch_home/bin
   (isshostA)issadmin.sh --user userA --host mailhostA.example.com --setstate I
   (isshostA)issadmin.sh --export --eximpath /export/archive --user userA --host
   mailhostA.example.com
   ```

2. Move the mailbox from **mailhostA** to **mailhostB**.

   ```
   (mailhostA)cd MessagingServer_home/bin
   (mailhostA)rehostuser -u userA -d mailhostB.example.com
   ```

3. From **isshostB**, create the Indexing and Search Service account.

   ```
   (isshostB)cd IndexSearch_home/bin
   (isshostB)/issadmin.sh --createaccount --userA --host mailhostB.example.com
   ```

4. Rehost the account from **isshostA** to **isshostB**.

   ```
   (isshostA)issadmin.sh --import --eximpath
   /net/isshostA.example.com/export/archive --user userA --host
   mailhostB.example.com --rehost mailhostA.example.com
   ```

5. From **isshostB**, check the account.

   ```
   (isshostB)issadmin.sh --checkaccount --user userA --host mailhostB.example.com
   ```

6. If the **checkaccount** command did not return "in sync," perform a manual synchronization.

```
(isshostB)issadmin.sh --checkaccount --sync --user userA --host
mailhostB.example.com
```

7. Set the account to Active.

```
(isshostB)issadmin.sh --setstate A --user userA --host mailhostB.example.com
```

8. (Optional) Delete the account on **mailhostA**.

```
(isshostA)issadmin.sh --deleteaccount --user userA --host mailhostA.example.com
```

Each account snapshot consists of a directory in the snapshot repository. This directory contains directories for the meta index, content index, dIndex records (a modified subset for this account), data store files, and a plain-text summary file describing the snapshot. The snapshot also appears as part of the overall **exisummarylist** file. This summary identifies a specific account within the snapshot repository.

A snapshot repository can contain any number of such account snapshot directories. It also contains a plain-text summary file (**eximsummarylist**) that is used to navigate multiple snapshot directories in the repository.

The **--eximpath** option can specify any available directory. If you do not specify a directory, the **--export and --import** options use **iss.store.dir/snapshots** as the default. You can override the default setting by using the **--setdefaulteximpath** option to the **issadmin.sh** command.

Rehosting Users Automatically

The process of rehosting a user from one Messaging Server instance to another is simple and automated. The **issrehostuser.sh** script and the **rehostuser** command work together to move the account from its Indexing and Search Service instance to a new one while at the same time the account is moved to its new Messaging Server instance. For more information about using the **rehostuser** command, see the Messaging Server documentation. See "issrehostuser.sh" for syntax description and options.

Monitoring the Indexing and Search Service Store

This section describes how to monitor the status of the Indexing and Search Service store by using the **isssadmin.sh** command with options such as **--listaccounts**, **--listfolders**, and **--checkaccounts**. See "issadmin.sh" for more information about these options.

Listing Information for All Accounts

To list the information of all accounts in the Indexing and Search Service store, including any accounts currently being bootstrapped, run the following command:

```
issadmin.sh --listaccounts
```

Verifying Account Bootstrapping

To verify that accounts have been bootstrapped:

1. View the files in the log directory (defined by **iss.log.dir** in the *IndexSearch_home*/**etc/jiss.conf** file) to check for any problems while bootstrapping the account.
2. Run either **issadmin.sh--checkaccount** or **issadmin.sh--checkfolder** to check the account for problems in specific emails that might have occurred during bootstrapping.

These commands compare the data in the index against the current state of the Messaging Server store for that account and list any problems found in the index. For serious problems, you can delete the account (or just a single folder) and bootstrap it again to correct inconsistencies. Sometimes the bootstrapping does not process attachment data completely, causing incomplete search results, but the rest of the account data can be searched.

Gathering Periodic Statistics

Log files contain error and informational messages about the operation of the various Indexing and Search Service processes. The **indexSvc**, **searchSvc**, and **jmqconsumer** processes also log statistics about their progress since the server was last restarted in separately configured log files. These log files provide insight into overall service status. You can use these log files to detect use patterns and performance issues more easily than analyzing the individually logged informational messages.

Generating a Statistic Snapshot

To generate a statistic snapshot for the **indexSvc** and **jmqconsumer** processes, use the **issadmin.sh** command with the **--liststats** and **--listbacklog** commands, respectively. For more information about these options, see "issadmin.sh".

Index Service Statistics: Output of liststats Option

The following example output shows the kinds of information that the **--liststats** option can provide on a running **indexSvc** service. The output shows the general format of statistics that are periodically recorded in a log file. The first line indicates that the frequency is about every 10 minutes (600 seconds, the default). You can vary this frequency to between 15 and 60000 seconds by using the **iss.indexsvc.statisticsinterval** configuration parameter. The second line describes the various columns that appear in each line of the data that follows.

For details about specific lines of information, see the Notes that follow the output.

Following the "autosync" lines are summary statistics for the Buffer Manager used within Lucene to reuse various sizes of data buffers to reduce memory pressure. The two sets of buffers are configured and behave independently of each other. If either the "too big buffer created" or "too big ByteB created" fields are not zero, additional information is printed indicating what size buffer was requested that could not be cached. Such buffers are created and released to the heap (not cached or reused), and, if they occur frequently, might be a source of large memory overhead (inducing garbage collection).

```
Fri Jul 06 09:06:59 GMT+00:00 2012 common.LogUtils.indexStats runLoop INFO:     607290  ms since
prior sample, server has been running for 1.1 hours)
    count  description                prior count     delta  percentage  rate/sec
  2314883  basic docs created            2259890     54993         2        91        (Note  1)
       14  document types created              14
 table of doctypes:
   950418 xemailxmetax                    935619     14799         1        24        (Note  2)
   941960 xemailxcontentx                 928396     13564         1        22        (Note  3)
    93150 xemailxattachx                   91877      1273         1         2        (Note  4)
   167272 affpd                           157615      9657         6        16        (Note  5)
    92335 afd                              83389      8946        10        14        (Note  6)
    22761 amd                              20086      2675        13         4        (Note  7)
    31430 atdd                             28139      3291        11         5        (Note  8)
    10390 asd                               9731       659         6         1        (Note  9)
        0 admd
```

```
      1 sdc                                1                                    (Note 10)
      2 sda                                2                                    (Note 10)
   3442 sdb                             3356       86         2         0      (Note 10)
      1 dd                                 1                                    (Note 11)
   1721 agd                             1678       43         2         0      (Note 12)

 186300  attachment type fields create  183754     2546         1         4      (Note 13)
     21  attachment types created          21
table of attachment types:
  57924 pdf                            57402      522         0         0
   2474 plain                           1270     1204        94         1
     14 audio                              2       12       600         0
    342 odf                              206      136        66         0
   1872 ssign                           1038      834        80         1
   2726 image                           1390     1336        96         2
    232 html                             178       54        30         0
    314 compress                         108      206       190         0
   1800 vcf                             1270      530        41         0
 112376 jpeg                           110352     2024         1         3
     40 xml                               30       10        33         0
    898 other                            712      186        26         0
     32 ppt                               18       14        77         0
     68 doc                               36       32        88         0
   4944 pgpsign                         3848     1096        28         1
     30 rtf                               26        4        15         0
     48 video                             22       26       118         0
     28 xls                               16       12        75         0
     10 applefile                          2        8       400         0
     24 iwork                              6       18       300         0
     52 vsd                                4       48      1200         0

 651138  single doc searches           627338    23800         3        39      (Note 14)
 158960  single doc search found none  150370     8590         5        14      (Note 15)
  22943  single map doc searches        22753      190         0         0      (Note 16)
 221513  account docs from cache       207014    14499         7        24      (Note 17)
 141091  folder flag docs from cache   127189    13902        10        23      (Note 18)
1641865  total docs cached             1608713    33152         2        55      (Note 19)
 140844   account                      138145     2699         1         4
 864760   folder                       855814     8946         1        14
      0   folder flags (obsolete)
 636261   folder flags partial         614754    21507         3        35
      0   account map
 441613  total docs removed from cache 441560       53         0         0      (Note 20)
    485   account                         459       26         5         0
    485   folder                          459       26         5         0
      0   folder flags (obsolete)
 440643   folder flags partial         440642        1         0         0
      0   account map
 238638  size of account doc cache     238556       82         0         0
1560726  size of folder doc cache     1560686       40         0         0
 108530  size of flags partial cache   93945    14585        15        24
  46266  # folder flag sets cached     40330     5936        14         9
      0  size of account map doc cache
 125886  calls to closeAccess          118205     7681         6        12      (Note 21)
  35323  calls to delayCloseAccess      31590     3733        11         6      (Note 22)
   9716  closeAccess skipped             9161      555         6         0      (Note 23)
 158610  closeWriters done             148836     9774         6        16      (Note 24)
 119321  groupNumberCache size         119278       43         0         0      (Note 25)
 119246  groupAccountGroupCache size   119207       39         0         0      (Note 26)
      0  group Cache remote updates                                            (Note 27)
```

```
       0  group Cache entries cleared                                          (Note 28)
  379680  # IndexReaders opened          354571     25109         7        41  (Note 29)
      11    # opens took over  1 sec         11
       4    # opens took over 10 secs         4
  148684  # dIndex IndexReaders opened   140264      8420         6        14  (Note 30)
   56756    # opens took over  1 sec      51016      5740        11         9
    6214    # opens took over 10 secs      5772       442         7         0
table of event types:
  231875  Total events since index init  223109      8766         3        14  (Note 31)
   36294   NewMsg events                  32894      3400        10         5
   20033   UpdateMsg events               17938      2095        11         3
       0   Rename events
       0   Delete events
       0   Create events
    8788   Copy events                     7486      1302        17         2
   13206   ExpungeMsg events              11283      1923        17         3
   31579   ChangeFlag events              31579
    1969   Bootstrap events                1923        46         2         0
       0   SetAcl events
       0   Login events
       0   Logout events
       0   unknown events
autosync:
   19437  candidates added                16739      2698        16         4  (Note 32)
   10000  accounts checked                 9400       600         6         0
    8690  accounts found OK                8687         3         0         0
    1310  accounts needed sync              713       597        83         0
     993  account checking skipped          396       597       150         0
       2  sync fail: other command            2
       0  sync fail: missing folder
       0  sync fail: authentication
       0  sync fail: other reasons
autobootstrap:                                                               (Note 33)
       0  candidates reboot by age
       0  candidates from checkstore
     500  accounts bootstrapped
simple buffer cache limits:     tiny  4096  small 16384  medium  6144  large  64  xlarge    16
simple buffer cache sizes now:  tiny   106  small  4661  medium  5511  large  33  xlarge    11
simple buffer sizes (in kb):    tiny   128b small     2  medium    16  large 128  xlarge  1024
   4894m  reused buffers                  4610m 283790246         6    472924
   2517m    tiny buffer reused            2375m 141769185         5    236252
   2343m    small buffer reused           2203m 139914905         6    233162
25007652    medium buffer reused        23564990   1442662         6      2404
 8265133    large buffer reused          7602221    662912         8      1104
   17208    xlarge buffer reused           16626       582         3         0
   32186  created buffers                  30709      1477         4         2
     156    tiny buffer created              156
    5009    small buffer created            5009
   26977    medium buffer created           25500      1477         5         2
      33    large buffer created               33
      11    xlarge buffer created              11
       0  too big buffer created
   4894m  requested buffers                4610m 283791724         6    472927
   2517m    tiny buffer requests          2375m 141769186         5    236252
   2343m    small buffer requests         2203m 139914905         6    233162
25034437    medium buffer requests      23590298   1444139         6      2406
 8265158    large buffer requests        7602246    662912         8      1104
   17215    xlarge buffer requests         16633       582         3         0
   4894m  cached buffers                  4610m 283792417         6    472928
   2517m    tiny buffer cached            2375m 141769186         5    236252
```

```
   2343m    small buffer cached           2203m 139915043         6    233162
25012971    medium buffer cached        23568277   1444694         6      2407
 8265158    large buffer cached          7602246    662912         8      1104
   17215    xlarge buffer cached           16633       582         3         0
   20473  not cached buffers               18857      1616         8         2
       0    tiny buffer not cached
       0    small buffer not cached
   20473    medium buffer not cached       18857      1616         8         2
       0    large buffer not cached
       0    xlarge buffer not cached
       0  unknown size buffer not cache
ByteBuffer cache limits:     tiny  4096  small 16384  medium  6144  large    64 xlarge    32
ByteBuffer cache sizes now:  tiny   128  small  4639  medium  5476  large    32 xlarge    20
ByteBuffer sizes (in kb):    tiny   128b small     2  medium    16  large   128 xlarge  1024
41008324  reused ByteB                  38397306   2611018         6      4351
 1271295    tiny ByteB reused            1179606     91689         7       152
16413900    small ByteB reused          15250954   1162946         7      1938
23130463    medium ByteB reused         21785860   1344603         6      2240
  176798    large ByteB reused            165554     11244         6        18
   15868    xlarge ByteB reused            15332       536         3         0
   33957  created ByteB                    32525      1432         4         2
     128    tiny ByteB created               128
    4987    small ByteB created             4987
   28790    medium ByteB created            27358      1432         5         2
      32    large ByteB created                32
      20    xlarge ByteB created               20
       0  too big ByteB created
41041241  requested ByteB               38428791   2612450         6      4353
 1271295    tiny ByteB requests          1179606     91689         7       152
16418375    small ByteB requests        15255429   1162946         7      1938
23158869    medium ByteB requests       21812834   1346035         6      2243
  176822    large ByteB requests          165578     11244         6        18
   15880    xlarge ByteB requests          15344       536         3         0
41017579  cached ByteB                  38404592   2612987         6      4354
 1271295    tiny ByteB cached            1179606     91689         7       152
16418027    small ByteB cached          15254979   1163048         7      1938
23135555    medium ByteB cached         21789085   1346470         6      2243
  176822    large ByteB cached            165578     11244         6        18
   15880    xlarge ByteB cached            15344       536         3         0
   22355  not cached ByteB                 20723      1632         7         2
       0    tiny ByteB not cached
       0    small ByteB not cached
   22355    medium ByteB not cached        20723      1632         7         2
       0    large ByteB not cached
       0    xlarge ByteB not cached
       0  unknown size ByteB not cached
Tue Oct 15 13:44:21 PDT 2013 common.LogUtils.indexStats runLoop INFO: net change
 to AddDocsThreadsList size: 2 had been 0
Tue Oct 15 13:44:21 PDT 2013 common.LogUtils.indexStats runLoop INFO: net change
 to SingleFolderList size: 2 had been 0
Tue Oct 15 13:44:21 PDT 2013 common.LogUtils.indexStats runLoop INFO: net change
 to FolderThreadsList size: 4 had been 1
Tue Oct 15 13:44:21 PDT 2013 common.LogUtils.indexStats runLoop INFO: statistics
 sleeping for 600 seconds
```

Notes:

1. Number of Lucene documents created since the server started. Following this line is the number of different type of documents, which are then formatted in a table.

2. Number of meta data documents created. Each email must have one meta document. Therefore, this number represents the number of emails processed since the server started.
3. Number of content documents created. The body of an email is recorded here. This number is usually less than or equal to the number of meta documents.
4. Number of attachment documents created. Each email attachment causes one such document to be created. Usually this count is much smaller than the previous two counts.
5. Number of account folder flag partial documents created. Flag information for each email in each folder of an account is grouped in these documents. Updated whenever email flag values change.
6. Number of account folder documents created. Each folder in each account requires one such document to record details such as name, current number of emails, and so on.
7. Number of account map documents created. This number is the basic account information record, containing identification data (host, user) plus account structure (number of folders, and so on).
8. Number of account transition date documents created. Used to track the time of major account state transitions (such as creation of new emails in account, and so on).
9. Number of account state documents created. Used to track each account state transition.
10. Number of store summary documents created. Three separate document types are created (a,b,c) to represent the data because different parts are usually updated at different frequencies depending on what processing occurs. Data includes global counts, fixed configuration parameters, and so on.
11. Dummy used for initializing empty index directories; always exactly one.
12. Number of account group documents created. Updated when accounts are assigned, removed, or reassigned to an index group.
13. Number of attachment type fields created. Content and attachment documents use these fields to identify the kind of attachments in the email. The table shows how many attachments of each kind were processed. Usually it is exactly twice the number of **xemailxattachx** documents created.
14. Number of times a particular document was searched for (such as **affpd** and **amd** doctypes).
15. Number of times a particular document was searched for and not found. This situation is expected at various times, such as when creating a new account, where the account record must be checked not already to exist, and so is expected not to be found before it is created.
16. Number of times a single document was searched for (usually **afd** and **amd**). It uses a different interface, so a restricted subset of the counts in the previous two lines.
17. Number of times an account document was found in the memory cache when looked up, thus avoiding the disk I/O overhead.
18. Number of times a folder document was found in the memory cache when looked up, thus avoiding the disk I/O overhead.

19. Number of documents added to the in-memory cache. Various kinds of documents are cached to avoid I/O overhead because they are frequently accessed and infrequently changed. Account, folder, and flag documents are counted separately.
20. Number of documents removed from in-memory cache. Cached documents are removed when data is stale, when it might also be necessary to update the index on disk.
21. Number of times access to an account was released to ensure that the account is no longer modified, and any index writers can be closed.
22. Number of times release of access to an account was delayed, causing a delay in the close of any open index writers. This process is useful when several real-time events for an account occur in quick succession, since the rapid close and reopen of the index is unnecessary overhead and a potentially large throughput bottleneck.
23. Number of times a **delayedClose** was skipped (for example, no close done). This situation occurs when real-time events occur so rapidly that a new event requiring the same index writer arrives before the previous delayed close has completed. Thus this number measures how often the close and reopen was avoided between subsequent events, which can speed up processing at the cost of delaying the disk index update.
24. Number of times any index writer was actually closed. Counts the close of the meta and content index writer for every account; indicates how frequently index directories on disk are being updated.
25. Number of group number to account associations cached. Assignment of any account to a group occurs when it is created or moved, and does not change often so this information is cached in memory to speed up lookup (avoids disk I/O). This line is primarily interesting only when creating, moving, importing, or deleting accounts to indicate progress.
26. Number of group documents currently cached. Group information is cached in memory because it changes infrequently after accounts have been bootstrapped and assigned into groups. This line is primarily interesting only when creating, moving, importing, or deleting accounts to indicate progress.
27. Number of times a group notification was sent to remote servers (such as index service to search service). This situation happens when major changes to the in-memory group cache occurs to signal remote servers to update their in-memory group caches. (This feature is disabled by default. See the **iss.store.groupcache.enabled** parameter for more information.)
28. Number of group cache entries sent in group notifications to remote servers. This number is a cumulative total of all group entries that are removed from the group cache. This feature is disabled by default. See the **iss.store.groupcache.enabled** parameter for more information.
29. Number of times any **IndexReader** was opened.
30. Number of times an **IndexReader** was opened for the **dIndex** directory. Included in the total for any **IndexReader** opened.
31. Total number of events processed by the index service since the store instance was created. Below this total is the break down of the events by type processed since the service was last started. These numbers restart at zero if the service is restarted, but the total accumulates across server restart.
32. Total number of accounts added to the autosync candidates list after the initial full candidate list was created. These are accounts where events were dropped or lost,

which caused the system to add the account to the front of the list to be corrected soon.

33. Number of users scheduled for rebootstrapping due to account age (the default is one year) and the total number of autobootstrapped accounts in the statistics period.

Understanding Search Service Statistics

The following example output shows the kind of information that is logged for the **searchSvc** service. The example below shows the general format of statistics that are periodically recorded in a log file. These numbers are totals accumulated since the service was started. The first line indicates that the frequency is about every 30 minutes (1800 seconds, the default). You can vary this frequency to between 15 and 180000 seconds by using the **iss.searchsvc.statisticsinterval** configuration parameter. The second line describes the various columns that appear in each line of the data that follows.

For details about specific lines of information, see the Notes that follow the output.

Following the "search time average" lines are summary statistics for the Buffer Manager used within Lucene to reuse various sizes of data buffers to reduce memory pressure. The two sets of buffers are configured and behave independently of each other. If either the "too big buffer created" or "too big ByteB created" fields are not zero, additional information is printed indicating what size buffer was requested that could not be cached. Such buffers are created and released to the heap (not cached/reused), and, if they occur frequently, might be a source of large memory overhead (inducing garbage collection).

```
Tue Oct 15 13:44:11 PDT 2013 common.LogUtils.searchStats runLoop INFO:    1800298  ms since prior
sample, server has been running for 2.0 hours
    count  description                prior count     delta  percentage  rate/sec
    54430  single doc searches              35200     19230        54        10        (Note  1)
        0  single doc search found none
    54430  single map doc searches          35200     19230        54        10        (Note  2)
        0  account docs from cache
        0  folder flag docs from cache
    54395  calls to closeAccess             35174     19221        54        10        (Note  3)
        0  calls to delayCloseAccess                                                    (Note  4)
       0  closeAccess skipped                                                      (Note  5)
       0  closeWriters done                                                     (Note  6)
    17730  groupNumberCache size            13219      4511        34         2        (Note  7)
    17730  groupAccountGroupCache size      13219      4511        34         2        (Note  8)
        0  group Cache remote updates                                                   (Note  9)
        0  group Cache entries cleared                                                  (Note 10)
   158711  # IndexReaders opened           102557     56154        54        31        (Note 11)
        5    # opens took over  1 sec           2         3       150         0
        0    # opens took over 10 secs
    54451  # dIndex IndexReaders opened     35212     19239        54        10        (Note 12)
     1689    # opens took over  1 sec         368      1321       358         0
        0    # opens took over 10 secs
 search statistics:
    54451  total queries                    35211     19240        54        10        (Note 13)
        0   query failed
        0   query parse failed
        0   query parse token failed
        0   query rewritten

    35327  total number of folder terms     21267     14060        66         7        (Note 14)
 table of search folder values:                                                          (Note 15)
```

```
   35327 "INBOX"                          21267     14060        66         7

 table of search field values:                                                (Note 16)
   19124 attachment-type                  13944      5180        37         2
   35327 folder                           21267     14060        66         7
   24668 subject                          17570      7098        40         3
   10659 body                              3697      6962       188         3
 search result counts:                                                        (Note 17)
   20098  zero results                    12303      7795        63         4
   17897  1-10 results                    11777      6120        51         3
   11127  11-100 results                   7535      3592        47         1
    4760  101-1000 results                 3234      1526        47         0
     515  more than 1000 results           326       189        57         0
 search time counts:                                                          (Note 18)
   47516 less than 1 second               32447     15069        46         8
    6830 1 less than 5 seconds             2725      4105       150         2
      48 5 less than 10 seconds               2        46      2300         0
       3 10 less than 20 seconds              1         2       200         0
       0 20 less than 30 seconds
       0 30 seconds or over
 search time totals (ms):                                                     (Note 19)
 20830416 less than 1 second            13658003   7172413        52      3984
 11131942 1 less than 5 seconds          3751215   7380727       196      4099
   290501 5 less than 10 seconds           12692    277809      2188       154
    39714 10 less than 20 seconds          11836     27878       235        15
       0 20 less than 30 seconds
       0 30 seconds or over
 search time average (ms):                                                    (Note 20)
     438 less than 1 second                 420        18         4         0
    1629 1 less than 5 seconds             1376       253        18         0
    6052 5 less than 10 seconds            6346
   13238 10 less than 20 seconds           11836      1402        11         0
       0 20 less than 30 seconds
       0 30 seconds or over
 simple buffer cache limits:     tiny  4096  small 16384  medium  6144  large   64  xlarge    16
 simple buffer cache sizes now:  tiny    64  small  1090  medium  3295  large    8  xlarge     4
 simple buffer sizes (in kb):    tiny   128b small     2  medium    16  large  128  xlarge  1024
    1199m  reused buffers               756545319 442563958        58    245828
579180081    tiny buffer reused         369498075 209682006        56    116470
593141149    small buffer reused        371354650 221786499        59    123194
 24863511    medium buffer reused        14524620  10338891        71      5742
  1924536    large buffer reused          1167974    756562        64       420
        0    xlarge buffer reused
     4586  created buffers                  2320      2266        97         1
       64    tiny buffer created              64
     1117    small buffer created           1117
     3393    medium buffer created          1127      2266       201         1
        8    large buffer created              8
        4    xlarge buffer created             4
        0  too big buffer created
    1199m  requested buffers            756547115 442566224        58    245829
579180081    tiny buffer requests       369498075 209682006        56    116470
593142010    small buffer requests      371355511 221786499        59    123194
 24866712    medium buffer requests      14525555  10341157        71      5744
  1924536    large buffer requests        1167974    756562        64       420
        0    xlarge buffer requests
    1199m  cached buffers               756546029 442567185        58    245829
579180081    tiny buffer cached         369498075 209682006        56    116470
593141983    small buffer cached        371355268 221786715        59    123194
 24866614    medium buffer cached        14524712  10341902        71      5744
```

```
 1924536    large buffer cached          1167974    756562        64       420
       0    xlarge buffer cached
       0  not cached buffers
       0    tiny buffer not cached
       0    small buffer not cached
       0    medium buffer not cached
       0    large buffer not cached
       0    xlarge buffer not cached
       0  unknown size buffer not cache
ByteBuffer cache limits:     tiny  4096  small 16384  medium  6144  large    64  xlarge    32
ByteBuffer cache sizes now:  tiny   126  small  1087  medium  3295  large     8  xlarge     8
ByteBuffer sizes (in kb):    tiny   128b small     2  medium    16  large   128  xlarge  1024
27492648  reused ByteB                16168752  11323896        70      6290
  198471    tiny ByteB reused           123644     74827        60        41
 2503419    small ByteB reused         1591446    911973        57       506
24746464    medium ByteB reused       14426276  10320188        71      5732
   44294    large ByteB reused           27386     16908        61         9
       0    xlarge ByteB reused
    4653  created ByteB                   2386      2267        95         1
     128    tiny ByteB created              128
    1115    small ByteB created            1115
    3394    medium ByteB created           1127      2267       201         1
       8    large ByteB created               8
       8    xlarge ByteB created              8
       0  too big ByteB created
27496261  requested ByteB             16170098  11326163        70      6291
  198471    tiny ByteB requests         123644     74827        60        41
 2504022    small ByteB requests       1592049    911973        57       506
24749474    medium ByteB requests     14427019  10322455        71      5733
   44294    large ByteB requests         27386     16908        61         9
       0    xlarge ByteB requests
27496132  cached ByteB                16169016  11327116        70      6291
  198469    tiny ByteB cached           123644     74825        60        41
 2503994    small ByteB cached         1591810    912184        57       506
24749375    medium ByteB cached       14426176  10323199        71      5734
   44294    large ByteB cached           27386     16908        61         9
       0    xlarge ByteB cached
       0  not cached ByteB
       0    tiny ByteB not cached
       0    small ByteB not cached
       0    medium ByteB not cached
       0    large ByteB not cached
       0    xlarge ByteB not cached
       0  unknown size ByteB not cached
Tue Oct 15 13:44:11 PDT 2013 common.LogUtils.searchStats runLoop INFO: statistics sleeping for 1800
seconds
```

Notes:

1. Number of times a particular document was searched for (such as **affpd** and **amd** doctypes). Refer to corresponding **IndexServer** statistics for more detail.
2. Number of times a single document was searched for (usually **afd** and **amd**). It uses a different interface, so a restricted subset of the counts in the previous two lines.
3. Number of times access to an account was released. This process ensures that the account is no longer modified, so any index writers can be closed. Search does not write to any accounts.

4. Number of times release of access to an account was delayed, causing a delay in the close of any open index writers. Since search never writes to any accounts, this value should always be zero.
5. Number of times a **delayedClose** was skipped (for example, no close done). Because search never writes to any accounts, this value should always be zero.
6. Number of times any index writer was actually closed. Because search never writes to any accounts, this value should always be zero.
7. Number of group number to account associations cached. Assignment of any account to a group occurs when it is created or moved, and does not change often so this information is cached in memory to speed up lookup (avoids disk I/O). This line is primarily interesting only when creating, moving, importing, or deleting accounts to indicate progress.
8. Number of group documents currently cached. Group information is cached in memory because it changes infrequently after accounts have been bootstrapped and assigned into groups. This line is primarily interesting only when creating, moving, importing, or deleting accounts to indicate progress.
9. Number of times a group notification was sent to remote servers (such as index service to search service). Refer to corresponding **IndexServer** statistic for details.
10. Number of group cache entries sent in group notifications to remote servers. Refer to corresponding **IndexServer** statistic for details.
11. Number of times any **IndexReader** was opened.
12. Number of times an **IndexReader** was opened for the dIndex directory. Included in the total for any **IndexReader** opened.
13. Number of search queries received. The lines following this number show the counts of failures by various types, and the number of times a search query was rewritten into a different form for easier processing. (The rewritten count is of interest primarily to developers.) Note the number of successful searches is found by subtracting the failure counts from the total received.
14. Number of search queries that contained any folder term. Most searches specify a folder to search. Those searches that do not specify a folder search the entire account, across all folders.
15. Table of counts of searches with folder terms, grouped by name of folder. This table will have a line for each unique folder name searched. In this example only the **INBOX** folder was ever searched.
16. Table of counts of searches with each kind of term. The hostname and username terms are not counted because every search query must contain these terms.
17. Table of counts of searches based on how many results were returned.
18. Table of counts of searches based on how long each search took.
19. Table of total times taken by searches based on how long each took. Times are in milliseconds. This table shows how long the search server spends processing these types of search query.
20. Table of average times taken by searches based on how long each took. Times are in milliseconds. This table shows how long a typical search in each category took.

JMQConsumer Statistics: Output of listbacklog Option

The following example output shows the kind of information that the **--listbacklog** command option can provide from a running JMQConsumer service. The example shows the general format of statistics that are periodically recorded in a log file. These numbers are totals accumulated since the service was started.

The first two lines report how often the service runs (about every 30 minutes (1800 seconds)), and how long the server has been running.

The next set of lines list the counts of backlogged events for each account with greater than zero events queued, grouped by account state. The number of lines in this set varies depending on how many accounts have events backlogged. Many of the lines indicating "Active" and "Inactive" accounts have been omitted in this example because they are so numerous. The summary of this information follows, starting with the totals for all events backlogged and for all users, with these totals then divided by account state.

After this summary, a table of counts is displayed, listing the number of events received, finished, failed, skipped, or dropped since the service started. Finally, the number of accounts whose state changed since the previous statistics were generated is logged, with the time required to generate these statistics.

For details about specific lines of information, see the Notes that follow the output.

```
Tue Oct 15 13:14:13 PDT 2013 common.LogUtils.jmqStats run INFO: queue check sleeping for 1800
seconds
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats runQueueCheck INFO: runQueueCheck: started,
server has been running for 2.5 hours
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: All users share host
sc11b.example.com unless explicitly present in following messages:
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Active:
3025193@colorone.org:2 3099348@colorone.org:1 3076509@colorone.org:1 3114789@colorone.org:1
3014263@colorone.org:2 3063475@colorone.org:1
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Active:
3100179@colorone.org:2 3021434@colorone.org:1 3026333@colorone.org:1 3087690@colorone.org:1
3050238@colorone.org:2 3049006@colorone.org:1
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Active:
3025356@colorone.org:1 3012582@colorone.org:1 3034605@colorone.org:1 3062376@colorone.org:5
3071166@colorone.org:1 3075015@colorone.org:1
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Active:
3080474@colorone.org:2 3059811@colorone.org:1 3080156@colorone.org:1 3088724@colorone.org:3
3012326@colorone.org:1 3010701@colorone.org:1
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Active:
3029686@colorone.org:3 3036588@colorone.org:1 3118407@colorone.org:1 3090770@colorone.org:5
3112683@colorone.org:1 3112397@colorone.org:2
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Active:
3098379@colorone.org:1 3109379@colorone.org:1 3075342@colorone.org:1 3057362@colorone.org:1
3005288@colorone.org:1 3003412@colorone.org:1
...
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Inactive:
3091024@colorone.org:1 3013443@colorone.org:1 3066022@colorone.org:2 3118394@colorone.org:1
3039528@colorone.org:1 3107080@colorone.org:3
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Inactive:
3013876@colorone.org:3 3031089@colorone.org:2 3116432@colorone.org:3 3016384@colorone.org:2
3097988@colorone.org:1 3031007@colorone.org:1
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Inactive:
3056847@colorone.org:3 3065067@colorone.org:2 3116299@colorone.org:1

Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO: Bootstrap:
3099295@colorone.org:5 3107340@colorone.org:3 3030422@colorone.org:2 3060000@colorone.org:13
```

```
3116786@colorone.org:2 3045641@colorone.org:2

Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO:  Total: 28755 Users: 18556
Total active: 28577 Active Users: 18469 Total inactive: 151 Inactive Users: 81 Total bootstrap: 27
Bootstrap Users: 6 Total Unknown: 0 Unknown Users: 0
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats printQueues INFO:
    count  description                prior count     delta  percentage  rate/sec
   203129 events Received                 132351     70778        53        39          (Note 1)
        0 events unsupported
   109975 events Finished                  84252     25723        30        14          (Note 2)
        0 events failed/Not Finished
      488 events with no account              407        81        19         0          (Note 3)
      285 events skipped from queues          227        58        25         0          (Note 4)
        0 events dropped: noops
    63582 events dropped: changeflags       29130     34452       118        19
        0 events dropped: others
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats call INFO: 0 account states changed, took
460ms
Tue Oct 15 13:44:13 PDT 2013 common.LogUtils.jmqStats runQueueCheck INFO: runQueueCheck: done
```

Notes:

1. Number of events read. This number is the total of all events processed, dropped, skipped, or backlogged since the service started.
2. Number of events successfully processed by the Index Service.
3. Number of events read for which no account is defined in the Index Service. These events are ignored.
4. Number of events skipped. These are events previously in the backlog that were not processed for a variety of reasons. (This number is different from the dropped events which were never backlogged.)

Using a Firewall Between Indexing and Search Service and Messaging Server

If you deploy a firewall between your Messaging Server and Indexing and Search Service hosts, you must open an additional port on the firewall for Message Queue (JMQ). That is, JMQ requires two ports. For more information, see "Connecting Through a Firewall" in *Sun GlassFish Message Queue Administration Guide* at:

http://docs.oracle.com/cd/E19226-01/821-0027/gcuhq/index.html

Using Solaris Management Facility to Manage Indexing and Search Service

The Service Management Facility (SMF) enables you to manage the operating system and application services, such as Indexing and Search Service. SMF replaces the legacy **init** scripting start-up mechanism common in prior releases of Oracle Solaris. SMF improves the availability of a system by ensuring that essential system and application services run continuously even when hardware or software failures occur.

The following steps describe how to configure the **iss.user** to be able to manage Indexing and Search Service through SMF. In this example, the **iss.user** is defined to be **jiss**.

1. Log in as **root**.

2. Set up Indexing and Search Service so that its services are managed by the user **jiss** by adding the following line to the **/etc/security/auth_attr** file.

```
solaris.smf.manage.iss:::Manage ISS Service States::
```

3. Run the following command to add the user privilege.

```
usermod -A "solaris.smf.manage.iss" jiss
```

4. Run the **svccfg** command to configure the **utilSvc** service then refresh the configuration.

```
svccfg -s jiss-utilSvc setprop general/action_authorization=astring:
'solaris.smf.manage.iss'
svccfg -s jiss-utilSvc setprop general/value_authorization=astring:
'solaris.smf.manage.iss'
svcadm refresh jiss-utilSvc
```

5. Run the **svccfg** command to configure the **indexSvc** service then refresh the configuration.

```
svccfg -s jiss-indexSvc setprop general/action_authorization=astring:
'solaris.smf.manage.iss'
svccfg -s jiss-indexSvc setprop general/value_authorization=astring:
'solaris.smf.manage.iss'
svcadm refresh jiss-indexSvc
```

6. Run the **svccfg** command to configure the **searchSvc** service then refresh the configuration.

```
svccfg -s jiss-searchSvc setprop general/action_authorization=astring:
'solaris.smf.manage.iss'
svccfg -s jiss-searchSvc setprop general/value_authorization=astring:
'solaris.smf.manage.iss'
svcadm refresh jiss-searchSvc
```

7. Run the **svccfg** command to configure the **jmqconsumer** service then refresh the configuration.

```
svccfg -s jiss-jmqconsumer setprop general/action_authorization=astring:
'solaris.smf.manage.iss'
svccfg -s jiss-jmqconsumer setprop general/value_authorization=astring:
'solaris.smf.manage.iss'
svcadm refresh jiss-jmqconsumer
```

8. Verify the configuration.

```
su - jiss
/usr/sbin/svcadm restart jiss-jmqconsumer
/usr/sbin/svcadm restart jiss-searchSvc
/usr/sbin/svcadm restart jiss-indexSvc
/usr/sbin/svcadm restart jiss-utilSvc
```

Improving Global Directory Index Performance

The Global Directory Index, also referred to as the dIndex, controls the Indexing and Search Service store. This section describes how to improve dIndex performance.

Overview of the Global Directory Index

On average, the dIndex directory of a typical Indexing and Search Service store instance containing 150,000 accounts requires about 400 MB of disk space. Read, write, and especially segment merge operations on such large index directories are memory intensive and put a heavy workload on the CPU. Processing is also slowed due to how disk writes are serialized, even on updates to separate accounts. To reduce processing overhead as the dIndex grows over time, Indexing and Search Service 1.0.5.18.0 and greater uses an alternate form of the dIndex, which distributes data for different accounts across multiple, smaller directories. This data distribution enables quicker access in parallel for independent account transactions. Using this dIndex alternative form, most dIndex data reads and updates require a smaller memory footprint, and fewer transactions need to be serialized.

The first implementation of the dIndex is known as "format 1." The form of the dIndex introduced in Indexing and Search Service 1.0.5.18.0 is known as "format 2." Both dIndex format 1 and 2 are fully functional, and either form can be used for any Indexing and Search Service instance independent of any other instance in a multiple Indexing and Search Service instance deployment. However, you can only use the format 2 dIndex with software installed from Indexing and Search Service 1.0.5.18.0 or greater.

Format 1 remains the default form of the dIndex. To use format 2, set the **iss.store.partitions.count** configuration parameter to a value greater than zero. You can only change this parameter when Indexing and Search Service services are stopped. Thus, this parameter is not refreshable. Once you have defined the value of this parameter for an instance, you can only change the format of the dIndex by using the **issadmin.sh --converttoformat** command.

You can set the partition count value in the **jiss.conf** file before creating a new Indexing and Search Service instance, and the Indexing and Search Service store can be populated by bootstrapping as before by using the value supplied. Once the dIndex has been created, usually the first time the Index Service has been started, the value is fixed in the dIndex, and can only be subsequently modified by using the **--converttoformat** command. You must convert any Indexing and Search Service store instances created before Indexing and Search Service 1.0.5.18.0 to use this feature.

Choosing an Appropriate Partition Count

The value of the **iss.store.partitions.count** parameter defines the number of dIndex directories created for the store. If the count is zero, only the format 1 dIndex is created. For any value *N* greater than zero, the format 2 dIndex is created. *N* additional directories are created in the same directory as the dIndex named **dIndex***XX*, where *XX* takes the value 00 thru *N*-1. When the **--converttoformat 2** command is used to convert an existing format 1 dIndex, account groups are distributed across the *N* **dIndex***XX* directories more or less evenly to balance the load. Once assigned to a specific **dIndex***XX* directory, the group does not move, but any account may be moved to a different group (as with the format 1 implementation).

Information for all accounts in each group is distributed to the same **dIndex***XX* directory. The amount of information for each account might vary greatly, due to the amount of content and structure of the account (for example, how many emails and how many folders the account contains). If the distribution of accounts is unbalanced, you can move accounts between account groups after the **--converttoformat 2** command completes by using the **--moveaccount** command. Each **dIndex***XX* directory is independently backed up in the same manner as the **dIndex** directory. (Thus, if you need a backup to recover from data corruption in the dIndex, you only need to

substitute backup files individually for those partitions exhibiting problems, limiting potential data loss.)

The choice of value for the **iss.store.partitions.count** parameter depends on the number of accounts, account groups, and amount of data in the dIndex and account group directories. Theoretically, the larger the partition count, the more potential parallel activity between accounts is possible. The memory needed to process each individual account should typically be less because only a fraction of the dIndex must be referenced for each transaction. However, if more transactions run in parallel, the cumulative memory use at any given time could be greater than without partitioning. For a format 1 dIndex size up to about 0.5 Gbyte, a partition count value of 5 to 10 typically reduces the average **dIndex***XX* size enough to improve performance. For larger format 1 dIndex sizes up to 1 Gbyte and more, higher partition count values might be needed.

> **Note:** Limited tests have shown performance improvement plateaus rapidly after partition count 5 for a format 1 dIndex size of about 0.5 Gbyte. Size is not the only factor, so it is not possible to predict accurately how your performance is affected.

The partition number for any group is displayed in the **--accountinfo** output (on the line containing the meta and content index sizes) whenever the partition count is not zero. The **--converttoformat** command takes up to about 10 minutes to run for most Indexing and Search Service store instances up to about 1 Gbyte of data, depending on your specific configuration and hardware.

Changing Partition Count

Changing the partition count for a format 2 dIndex involves converting back to format 1, updating the **iss.store.partitions.count** parameter, then converting back to format 2 using a different partition count. To avoid confusion, remove old backup files that are no longer valid for the new partition count.

To change the partition count:

1. Convert the dIndex back to format 1 by editing the **jiss.conf** file and setting the **iss.store.partitions.count** parameter to 0.
2. Run the **--converttoformat 1** command.

   ```
   issadmin.sh --converttoformat 1
   ```

3. Convert the dIndex back to format 2 by editing **jiss.conf** and setting **iss.store.partitions.count** to a non zero partition.
4. Run the **--converttoformat 2** command.

   ```
   issadmin.sh --converttoformat 2
   ```

5. To check for backup files that are no longer valid, change to the directory specified by the **iss.store.dir** configuration parameter.
6. Remove unneeded backup files.

   ```
   rm -rf dIndexNN.backup*
   ```

7

Troubleshooting Indexing and Search Service

This chapter describes how to troubleshoot your Oracle Communications Indexing and Search Service deployment.

Log Files

This section explains how to use Oracle Communications Messaging Server and Indexing and Search Service log files to diagnose problems.

Messaging Server IMAP Logs

Because Indexing and Search Service does not see the original IMAP commands, you must use Messaging Server logs to view this information. You can use the Messaging Server telemetry logs to check on some IMAP searches, for example ESEARCH searches. For more information about telemetry logs, see the topic on checking user IMAP, POP, and Webmail sessions by using telemetry in *Messaging Server System Administrator's Guide*.

You can then use the host name, user name, and folder information in the Indexing and Search Service logs to match up with the corresponding IMAP command in the Messaging Server logs.

Location of the Messaging Server IMAP log: *MessagingServer_home***/logs/imap**

When to use: If bootstrapping of accounts is failing, authentication might be a problem. Check the Messaging Server **imap** log. If your read-only message store administrative user (**mail.imap.admin.username** in the **/opt/sun/comms/jiss/etc/jiss.conf** file) is the problem, you might see errors similar to the following:

```
[Account Information: connect [172.20.241.110:56831]
[Account Notice: badlogin: [172.20.241.110:56831] plain User not found
```

If the password of that user is incorrect (**mail.imap.admin.password** in the **/opt/sun/comms/jiss/etc/jiss.conf** file), you might see errors similar to the following:

```
[Account Information: connect [172.20.241.110:56854]
[Account Notice: [172.20.241.110:56854] Password verification failed
[Account Notice: badlogin: [172.20.241.110:56854] plain User not found
```

Also, ensure that this user is specified in **store.indexeradmins** (to give read-only message store administrative rights) and that you have restarted the **imapd** process, to pick up this change. Failure to do so causes errors similar to the following:

```
WARNING: Caught IssException: com.sun.comms.iss.common.IssException:
BootStrap of account, User: testuser1 Host: mailhost.example.com
```

```
Not authorized to login as specified user
```

You must add this read-only message store administrative user manually (that is, it is not automatically created) during the Indexing and Search Service installation process. For more information, see *Indexing and Search Service Installation and Configuration Guide*.

Service Management Facility (SMF) Logs

Location: **/var/svc/log/*jiss***

When to use: If you receive the following error when trying to start Indexing and Search Service:

```
# /opt/sun/comms/jiss/bin/svc_control.sh start
Starting utilSvc
Starting indexSvc
svcadm: Instance "svc:/application/jiss-indexSvc:default" is in maintenance state.
```

To determine the cause for **indexSvc** entering the maintenance state, check the **/var/svc/log/application-jiss-indexSvc:default.log** log file. There can be several possible causes, one of which can be that the Message Queue broker might not be running.

After determining the cause, fix the underlying issue (for example, start the Message Queue (JMQ) broker), and run the **svc_control.sh stop** command (to clear the maintenance state), followed by the **svc_control.sh start** command again.

Messaging Server and Indexing and Search Service IMQ Broker Logs

Location: **/var/imq/instances/imqbroker/log/log.txt**

When to use: To investigate problems, such as authentication failures, from the IMQ brokers on either the Messaging Server or the Indexing and Search Service system.

For more information, see "Message Queue Consumer and Broker Log Messages".

Indexing and Search Service GlassFish Server Logs

Location: *GlassFish_home***/appserver/domains/domain1/log/server.log**

When to use: Verify that Oracle GlassFish Server is receiving, authenticating, and servicing requests.

If you see a query from Messaging Server that is refused, ensure that the originating IP address is correctly specified in the **mail.server.ip** parameter in the **jiss.conf** file. If you do not see a query as expected from Messaging Server in the Indexing and Search Service GlassFish Server log, check the settings of the **service.imap.indexer.*configutil** parameters. If those settings are correct, try issuing a query directly from the Messaging Server and check the Indexing and Search Service GlassFish Server log again, for example:

```
# telnet isshost.example.com 8080
Trying 10.10.10.10...
Connected to isshost.example.com.
Escape character is '^]'.
GET
/rest/search?q=%20%2busername:user1%20%2Bhostname:mailhost.example.com&contentform
at=simpleuid&format=atom HTTP/1.0

HTTP/1.1 200 OK
```

```
X-Powered-By: Servlet/2.5
Server: Sun GlassFish Enterprise Server v2.1.1 Patch16
Content-Language: *
Accept-Ranges: bytes
Content-Type: text/xml;charset=UTF-8
Content-Length: 1781
Date: Thu, 14 Nov 2013 22:24:27 GMT
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">
<title>isshost.example.com search: +username:user1
+hostname:mailhost.example.com</title>
<link>http://isshost.example.com/rest/search?q=+username:user1%20+hostname:isshost
.example.com&amp;format=atom</link>
<updated>Thu Nov 14 22:24:27 UTC 2013</updated>
<author><name>Oracle, Inc.</name></author><id>urn:uuid:9999999999999</id>
<opensearch:totalResults>39</opensearch:totalResults>
<opensearch:startIndex>0</opensearch:startIndex>
<opensearch:itemsPerPage>10</opensearch:itemsPerPage>
<opensearch:Query role="request" searchTerms="+username:user1
+hostname:isshost.example.com" searchPage="-1" />
<link rel="search" type="application/opensearchdescription+xml"
href="http://isshost.example.com" />
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>1</id>
</entry>
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>2</id>
</entry>
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>3</id>
</entry>
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>4</id>
</entry>
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>5</id>
</entry>
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>6</id>
</entry>
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>7</id>
</entry>
```

```
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>8</id>
</entry>
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>9</id>
</entry>
<entry>
<folder>Trash</folder>
<uidvalidity>1195182036</uidvalidity>
<id>10</id>
</entry>
</feed>
Connection to isshost.example.com closed by foreign host.
```

For more information, see "Index Service Log Messages".

Indexing and Search Service Messaging Server JMQ Event Consumer Logs

Location: **/var/opt/sun/comms/jiss/logs/iss-jmqconsumer.log.0**

When to use: Verify the status of the Messaging Server event consumer (**jmqconsumer**) to ensure that real-time updates are working (that is, arrival of new mail, deletion of mail, new folder creation, and so on).

The mail store JMQ broker is only accessed by **jmqconsumer**. JMQ broker connection problems with the Messaging Server message store appear in this log file. The following messages show that the JMQ broker is functioning:

```
Mar 24, 2009 10:55:22 AM com.sun.comms.iss.jmqconsumer.JMQConsumer$AsynchConsumer
run
 INFO: Listening to Queue INDEXMS on mailhost.example.com:7676
Mar 24, 2009 12:43:28 PM com.sun.comms.iss.jmqconsumer.JMQConsumer$AsynchConsumer
run
 INFO: Listening to Queue INDEXMS on mailhost.example.com:7676
Mar 24, 2009 12:44:07 PM com.sun.comms.iss.common.AccountStateMgrImpl
setAccountState
 INFO: Set account user1@mailhost.example.com state to B
Mar 24, 2009 1:12:41 PM com.sun.comms.iss.common.AccountStateMgrImpl
setAccountState
 INFO: Set account user1@mailhost.example.com state to A
```

For more information, see "Message Queue Consumer and Broker Log Messages".

Indexing and Search Service Messaging Server JMQ Consumer Statistics Log

Location: **/var/opt/sun/comms/jiss/logs/iss-jmqconsumer-stats.log.0**

When to use: To check JMQ event statistics related to performance and patterns of service over long periods of time.

For more information, see "JMQConsumer Statistics: Output of listbacklog Option".

Indexing and Search Service Index Service Log

Location: **/var/opt/sun/comms/jiss/logs/iss-indexsvc.log.0**

When to use: To check issues with indexing during bootstrapping or during the indexing of new real-time events from Messaging Server.

For more information, see "Index Service Log Messages".

Indexing and Search Service Index Service Statistics Log

Location: **/var/opt/sun/comms/jiss/logs/iss-indexsvc-stats.log.0**

When to use: To check indexing statistics related to performance and patterns of service over long periods of time.

For more information, see "Index Service Statistics: Output of liststats Option".

Indexing and Search Service Search Service Log

Location: **/var/opt/sun/comms/jiss/logs/iss-searchsvc.log.0**

When to use: To check on the arrival and servicing of search requests.

For more information, see "Index Service Log Messages".

Indexing and Search Service Search Service Statistics Log

Location: **/var/opt/sun/comms/jiss/logs/iss-searchsvc-stats.log.0**

When to use: To check search statistics related to performance and patterns of search requests over long periods of time.

For more information, see "Index Service Statistics: Output of liststats Option".

Indexing and Search Service Utility Service Log

Location: **/var/opt/sun/comms/jiss/logs/iss-utilsvc.log.0**

When to use: To check on the arrival and servicing of utility requests.

For more information, see "Index Service Log Messages".

Using Command-Line Tools to Diagnose Problems

Indexing and Search Service provides command-line tools to manage the Indexing and Search Service store. The primary tool is the **issadmin.sh** script. Other tools listed in this section are intended for specific problems. For more information, see "Indexing and Search Service Command-Line Utilities".

Using checkIndex.sh

When to use: If an individual index directory is corrupted. Checks for consistency and corrects some types of errors at the Lucene data record level.

Using lucli.sh

When to use: To inspect internal database structure at the Lucene data record level. This script runs from the command line and supports commands to search and display records in an individual index directory. Its features are similar to those of the **luke.sh** command.

Using luke.sh

When to use: To inspect internal database structure at the Lucene data record level. This script runs a graphical user interface (GUI) interface that corresponds to **lucli.sh**. To enable this script, you must manually download the jar files.

Using mergeIndex.sh

When to use: To manipulate index directories directly at the Lucene file level. This script is rarely used, because it requires detailed knowledge of internal store structure.

Using searchRun.sh

When to use: To search accounts by using the command-line interface. Can be used to check if an account contains the expected data from search queries that have failed.

Using issadmin.sh

When to use: To diagnose general problems, to recreate lost data from single or multiple accounts, and to list, create, delete, check, and sync accounts and folders.

When you suspect a problem with an account, for example, after finding an error message in the log files, you can print a summary of the account structure by using the **--accountinfo** option to the **issadmin.sh** command.

Using the --accountinfo Option Output

Output from the **--accountinfo** option includes:

- Name of the account
- Group to which the account is assigned
- Disk space used by the account's group
- Account state
- Time that the account was created in the Indexing and Search Service store
- Time that the state of the account was last changed
- Counts of the number of emails in each folder and total for the account
- Specific information about each folder by name (including the number of emails found containing attachments as indicated by the **at:** field, number of emails that produced attachment store files, typically thumbnails used for client display or content files under some configurations, and other information)

Folder names are case sensitive and should display as seen in the email client. A name that appears as a string of question marks usually indicates a problem with displaying other character sets (such as Korean or Chinese) on your terminal. Ensure that your terminal is configured to display UTF-8, for example:

```
# LANG=en_US.UTF-8
# export LANG
```

In addition to the information for the specific account, the header of the output displays some global characteristics of the store. The total size, total search, and total index values are always zero. Most of the other values are easy to understand. If any appear inconsistent (such as the number of accounts do not match the number you created less those you deleted), then investigate using the **--listbrief**, or **--listaccounts**, or both options. (These commands are more expensive and can take several minutes to

complete when the store contains many accounts.) The dIndex memory locked field should usually be **false**, unless at some point you used the **--lockmemoryindex** option. Take special care using this option, because it can cause unexpected behavior in the store and search.

The email counts can have two forms: a single integer, or a pair of integers separated by a slash ("/"). The single integer form is the number of emails currently in the folder. The first integer of a pair also indicates the number of emails currently in the folder. The second integer might be smaller or larger than the first. Its presence indicates some emails were copied to or from the folder, and it is the number of emails whose copied contents is still associated with the folder. Immediately after an account is bootstrapped, **--accountinfo** should not show any folder with a count using the pair form. A folder with a count using the pair form indicates a problem with the bootstrap procedure. (After a user has been manipulating the contents of folder of an account, the pair form could be due to normal copying or moving message between folders.) If you suspect a problem, use the **--checkfolder** option or **--checkaccount** option to verify that there is no problem with the content of the account.

Using the --checkaccount Option Output

Output from the **--checkaccount** option shows whether the account information in the index matches (or is "in sync") with the information in the Messaging Server message store that the index depends upon. (The **--checkfolder** option performs the same kind of checking on a single folder.) If you suspect that an account has a problem, the **--checkaccount** option output indicates which folders are not synchronized. If the number of emails in any folder does not match the Messaging Server message store count, use the **--sync** option with the **--checkaccount** option to perform on-demand account update. This process usually synchronizes the account within minutes.

The **--checkaccount** option updates the number of emails in the index to match the Messaging Server message store. However, it does not perform a detailed verification of all information in the index against the Messaging Server message store, as that process is more time and resource intensive. Add the **--detail** modifier to the **--checkaccount** option to perform a detailed check. The **flags** level also checks for any email flag information that is out of sync, and when used with the **--sync** option, it corrects such problems. When you specify a level of **status** or **full**, the **--detail** modifier likely produces much more output. For example, you would see any indexing problems found with individual emails in the account or folder. Many of these **status** messages result from attachments containing data that cannot be interpreted properly. Sometimes the data format is inconsistent or other limitations on the indexing conversion process occur. These messages cannot be avoided when using the **--sync** option. These messages represent limitations on the kind of data that can be indexed and searched and might explain why some search queries are not able to return results as expected.

However, the output from the **--detail status** modifier can indicate other problems with the indexed data that can be corrected. If problems persist with the account after using **--sync**, examine them for clues about what might be wrong.

When the --sync Option Does Not Synchronize an Account

You can correct many problems with accounts by using the **--sync** option. If **--sync** fails to correct the account, there might be an internal consistency problem which requires other approaches. If the problem appears to be local to a specific folder, then using the **--deletefolder** option on the problem folder, followed by a bootstrap (using the **--bootstrap --folder** option to **issadmin.sh**), might correct it. Each folder can be corrected in turn, or, in the worse case, you must use the **--deleteaccount** option and

rebootstrap the entire account. If you must perform such repairs, start by using the **--setstate X** option to ensure the account index is offline from the real-time event updates while you delete and rebootstrap folders or the account.

Details of the Results of the --checkstore Option

Output from the **--checkstore** option describes the overall state of the index store, in contrast to the **--checkaccount** and **--checkfolder** commands, which describe information about the structure of individual accounts. There are two parts to this information: one part is generated by the **--detail dindex** modifier and one by the **--detail store** modifier. Both parts appear when the **--detail full** modifier is used.

The output from using the **--detail dindex** modifier contains information about the master **dIndex** directory alone. The command checks for the records describing accounts and groups, looking for duplicate or missing records, and comparing their information for internal consistency between records.

The output from using the **--detail store** modifier contains information about the index store that holds the individual account group index directories. The command checks the information in the dIndex against what it finds in the index store directories for extra or missing group index directories, comparing the **dIndex** and store directories for consistency.

The **--detail verbose** modifier provides control over the message output generated by the **--checkstore** command. If you specify this option, the output may be quite verbose, as it generates explicit messages for each account or index group it checks. By default the message output is smaller. Use the **--detail verbose** option only if the default messages do not provide enough detail.

If you expect the output from the **--checkstore** command to be quite verbose, sometimes the **--altoutput** *file* option is required to avoid problems when using **--checkstore** with a store containing many accounts.

Only run the **--checkstore** command while the services are stopped. The **--checkstore** command checks dIndex consistency against the rest of the store, and so you must avoid activity that could modify dIndex. Depending on the store size and complexity, the **--checkstore** command with the **--full** option could take up to 40 minutes. Schedule such work when you can shut down the services.

The **--sync** option can be used with the **--checkstore --detail dindex** options to correct inconsistencies found in the dIndex. The **--sync** option might update the **dIndex**, and so is not permitted while the services are running. After running with **--sync**, check the output of the **--checkstore** command again. Some problems might not have been corrected by **--sync**, and require further intervention.

The **--checkstore --sync** command prompts you if you also specify the **--prompt** option, enabling you to step through the process. You can disable the prompting when the command begins. The **--sync** option can correct many inconsistencies, such as duplicate account document, duplicate folder documents, missing or orphaned account documents, and so forth. Use the **--sync** option with the **--detail full** option in most circumstances, to detect and correct as many problems as possible at once. Using both options together takes longer to complete, but it avoids partially correcting the store, which can lead to inconsistencies that are much harder to detect and correct.

When the **--checkstore --sync** command detects and corrects a problem, typically one or more records for accounts are removed from the dIndex and meta and content index directories. The result is that accounts are effectively deleted from the store. You must reboostrap such accounts after the **--sync** is complete. Account names that have been deleted by the **--sync** option are appended to the

iss.store.dir/**store/checkstore.accounts** file. This file has the same format as that used by the **--accountlist** *FILE* option. This file is automatically read when the **IndexService** restarts and the accounts in it are added to the autobootstrap queue to be recreated, as long as the value of the configuration parameter **iss.indexsvc.periodic.autobootstrap.enabled** is **true**. Once the accounts are added to the autobootstrap queue, they are deleted from this file. The system then manages the accounts as part of the autobootstrap queue.

Interrupting Commands

Commands such as **issadmin.sh** invoke services in the **IndexService** that is a separate running process. (You can also run the **issadmin.sh** command when the **IndexService** is not running.) When these commands are executed, they submit tasks to the **IndexService**, which performs the work, then they wait for the response. If you interrupt one of the commands (by using either Control-C at the command line or the **kill** command), the submitting process is stopped, but not the services that are active in the **IndexService**. These services continue to run to completion even though you have killed the submitting process. Some commands, like bootstrapping a large account or various uses of the **--accountlist** option, also can take a long time to complete, and you might need to interrupt these requests.

To stop a service in the **IndexService** after you interrupt a command, use the **--listactiveservices** option to determine which services are still active in the **IndexService**. Typical output for a host running Indexing and Search Service after interrupting a **--checkaccount --sync** command might resemble the following:

```
issadmin.sh --listactiveservices
Fri Aug 24 00:29:16 GMT 2012
active index services:
a:28323:checkaccount:sync:mailhost.example.com:jennifer:javaone2009:edoc:0:1:Fri_
Aug_24_00_28_45_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:2000:2001:
Fri_Aug_24_00_29_13_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:3000:3001:
Fri_Aug_24_00_29_13_GMT_2012:w
a:Autosync:checkaccount:sync:mailhost.example.com:amam:INBOX:Fri_Aug_24_00_00_12_
GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:5000:5001:
Fri_Aug_24_00_29_13_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:1000:1001:
Fri_Aug_24_00_29_13_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:Fri_Aug_24_00_
28_45_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:0:1:Fri_
Aug_24_00_29_12_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:javaone2009:Fri_Aug_24_00_
28_45_GMT_2012:w
a:28323:checkaccount:1:jennifer:mailhost.example.com:Fri_Aug_24_00_28_44_GMT_
2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:4000:4001:
Fri_Aug_24_00_29_13_GMT_2012:w
a:Autosync:checkaccount:sync:mailhost.example.com:amam:INBOX:edoc:0:189816754:Fri_
Aug_24_00_11_44_GMT_2012:w
Fri Aug 24 00:29:23 GMT 2012
```

This example shows that many threads are still active: Each line of output is a separate thread. Names that end with ":w" indicate that these threads are potentially writing to the index. (The a: indicates an **issadmin.sh** command. The number is a process ID used to identify all threads created under the same **issadmin.sh** command. Other

fields are command specific. Threads created from the autosync and other features are also listed.)

By using the **--stopservices** command, you can stop any one of the named threads. The following command stops a single thread:

```
issadmin.sh --stopservice
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:Fri_Aug_24_00_
28_45_GMT_2012:w
```

However, usually you would want to stop all threads in a group. The following example stops all threads whose names start with a:28323:

```
issadmin.sh --stopservice a:28323:
```

Any string that ends with a colon (:) can be used as a prefix/wildcard.

The **--stopservice** command generates the following output:

```
issadmin.sh --stopservice a:28323:
Fri Aug 24 00:29:43 GMT 2012
stop of a:28323: :
a:28323:checkaccount:sync:mailhost.example.com:jennifer:javaone2009:edoc:0:1:Fri_
Aug_24_00_28_45_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:2000:2001:
Fri_Aug_24_00_29_13_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:3000:3001:
Fri_Aug_24_00_29_13_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:5000:5001:
Fri_Aug_24_00_29_13_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:1000:1001:
Fri_Aug_24_00_29_13_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:Fri_Aug_24_00_
28_45_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:0:1:Fri_
Aug_24_00_29_12_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:javaone2009:Fri_Aug_24_00_
28_45_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:1:jennifer:mailhost.example.com:Fri_Aug_24_00_28_44_GMT_
2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:4000:4001:
Fri_Aug_24_00_29_13_GMT_2012:w writing service not canceled; marked to stop
Fri Aug 24 00:29:53 GMT 2012
```

Threads that are "done" stop immediately. To avoid corrupting the index data, threads marked with ":w" must stop when they next reach a point where the index is no longer being written. This process can take a bit longer, but the threads usually stop fairly quickly.

However, because the **--sync** command continues to run while you are using these commands, more threads might be created by threads that have not stopped yet. Therefore, you must repeat the **--listactiveservices** and **--stopservices** commands, perhaps several times:

```
issadmin.sh --listactiveservices
Fri Aug 24 00:29:58 GMT 2012
active index services:

a:Autosync:checkaccount:sync:mailhost.example.com:amam:INBOX:Fri_Aug_24_00_00_12_
GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:3000:3001:
Fri_Aug_24_00_30_07_GMT_2012:w
```

```
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:Fri_Aug_24_00_
28_45_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:javaone2009:Fri_Aug_24_00_
28_45_GMT_2012:w
a:28323:checkaccount:1:jennifer:mailhost.example.com:Fri_Aug_24_00_28_44_GMT_
2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:2000:2001:
Fri_Aug_24_00_30_07_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:0:1:Fri_
Aug_24_00_30_07_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:4000:4001:
Fri_Aug_24_00_30_07_GMT_2012:w
a:Autosync:checkaccount:sync:mailhost.example.com:amam:INBOX:edoc:0:189816754:Fri_
Aug_24_00_11_44_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:5000:5001:
Fri_Aug_24_00_30_07_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:javaone2009:edoc:0:1:Fri_
Aug_24_00_30_00_GMT_2012:w
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:1000:1001:
Fri_Aug_24_00_30_07_GMT_2012:w
Fri Aug 24 00:30:08 GMT 2012
```

issadmin.sh --stopservice a:28323:

```
Fri Aug 24 00:30:31 GMT 2012
stop of a:28323: :
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:3000:3001:
Fri_Aug_24_00_30_07_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:Fri_Aug_24_00_
28_45_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:javaone2009:Fri_Aug_24_00_
28_45_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:1:jennifer:mailhost.example.com:Fri_Aug_24_00_28_44_GMT_
2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:2000:2001:
Fri_Aug_24_00_30_07_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:0:1:Fri_
Aug_24_00_30_07_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:4000:4001:
Fri_Aug_24_00_30_07_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:5000:5001:
Fri_Aug_24_00_30_07_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:javaone2009:edoc:0:1:Fri_
Aug_24_00_30_00_GMT_2012:w writing service not canceled; marked to stop
a:28323:checkaccount:sync:mailhost.example.com:jennifer:ubuntu-art:edoc:1000:1001:
Fri_Aug_24_00_30_07_GMT_2012:w writing service not canceled; marked to stop
Fri Aug 24 00:30:45 GMT 2012
```

issadmin.sh --listactiveservices

```
Fri Aug 24 00:30:48 GMT 2012
active index services:
a:Autosync:checkaccount:sync:mailhost.example.com:amam:INBOX:Fri_Aug_24_00_00_12_
GMT_2012:w
a:Autosync:checkaccount:sync:mailhost.example.com:amam:INBOX:edoc:0:189816754:Fri_
Aug_24_00_11_44_GMT_2012:w
Fri Aug 24 00:30:54 GMT 2012
```

issadmin.sh --listactiveservices

```
Fri Aug 24 01:11:38 GMT 2012
active index services: none
Fri Aug 24 01:11:40 GMT 2012
```

Eventually all threads stop, as seen when **--listactiveservices** shows no more threads running. In this case, the **--sync** command did not complete, so the state of the account and its contents are not as expected. You can proceed by using commands like **--accountinfo** to see what the account looks like and clean it up. Console output from the original **--sync** command is not produced, because the output is lost when you interrupt the command.

You might also check the index directories of the account for **write.lock** files. Depending on when the interrupt occurred, these files might not be cleaned up properly.

Diagnosing Severe System Problems

If you find that simple **issadmin.sh** commands (such as **--liststats**) hang and do not return in a few minutes, the Index Service might have failed in a way which requires you to shut down the services and restart them by using the **svc_control.sh** command. Both indexing and search service are temporarily suspended.

If you detect a hanging condition, you can use the following commands to help diagnose the problem before shutting down services:

1. To show the process ID (pid) of the **IndexService** component, run the **jps** command.
2. Save all output to a file for later examination:

   ```
   jstack -F -l pid
   ```

3. To identify any index group directories that were open at the time of failure, run a command like the following:

   ```
   find /var/opt/sun/comms/jiss/store -name "write.lock"
   ```

   All **write.lock** files are found in the *IndexSearch_home*/**index/store/locks** directory, not the individual group index directories. The name of each file in this directory contains the number of the group to which it belongs.
4. Stop the services by using the **svc_control.sh stop** command.

The output from these commands identifies groups of accounts which might be causing problems. After shutting down the services, remove the **write.lock** files for each group and find accounts that might be affected by using **issadmin.sh** commands.

- If possible, leave the services offline while you make repairs, then start the servers after repairs are complete.
- To restore services quickly, use the **--setstate I** option to disable accounts that might be causing problems, then check these accounts after you restart the services to see if they need repair. Use the **--accountinfo** option on each account in such groups and look for any reason for the hanging.

After you restart services, if the **--accountinfo** header information does not look reasonable, or the hanging condition or other problems recur, the critical internal data structures might be damaged. For more information, see "Disaster Recovery".

Account Name Restrictions

The information in an Indexing and Search Service Store instance is organized into accounts. In the email environment, each account in the Indexing and Search Service

store uses the same user name as the corresponding account in the Messaging Server message store from which its content is derived.

The following characters are invalid in Indexing and Search Service account names:

```
<space> $ ~ = # * + % ! , { } ( ) / \ < > ; : " ` [ ] & ?
```

If any of these characters appear in an account name, Indexing and Search Service does not create or bootstrap the index for that account, and search queries for such an account fail. For more information about account name restrictions in Messaging Server, see the topic on message store valid UIDs and folder names in *Messaging Server System Administrator's Guide*.

Troubleshooting the Indexing and Search Service Web Services Proxy

This section describes how to troubleshoot the Web Services Proxy (**isshttpd)**.

isshttpd Service Start Fails Until LDAP Entries Are Created

When configuring the first Indexing and Search Service node to use LDAP to store the host mappings between Messaging Server and Indexing and Search Service hosts, the **isshttpd** service does not start until you run the **isshttpmgr.sh -u** *file* command. However, you cannot run this command before the **setup -t isshttpd** command because it requires configuration information provided in that command. Once the LDAP entries are created and replicated there should be no problem starting **isshttpd** on subsequent nodes.

Troubleshooting isshttpd Proxy Using wget

You can create **wget** scripts to simulate Oracle Communication Convergence behavior to Indexing and Search Service.

1. Edit the **/etc/wgetrc** file to point to **isshttpd** with the following entries:

```
http_proxy = localhost:5559
ftp_proxy = localhost:5559
```

2. Run the following shell script while updating the password for **indexeradmin**, the user name, and the mailhost:

```
#!/bin/sh
[ -f /tmp/isshttpd.out ] && rm -f /tmp/isshttpd.out
[ -f /tmp/cookie.out ] && rm -f /tmp/cookie.out
host=localhost
mailhost=ms1.example.com
username=username
password=password
/usr/sfw/bin/wget \--save-cookies=/tmp/cookie.out \
                          \--keep-session-cookies \
                          \--server-response \
                          \--no-check-certificate \
                          \-O /tmp/isshttpd.out \
                          \--max-redirect 0 \
                          \--post-data="j_username=indexeradmin%3B$username&j_
password=$password" \
                          \-t 1 \
                          "http://$host:8080/rest/j_security_check"

/usr/sfw/bin/wget \-O /tmp/isshttpd.out \
                          \--load-cookies=/tmp/cookie.out \
```

```
                                \--no-check-certificate \
                                \--server-response \
                                \-t 1 \

"http://$host:8080/rest/search?q=%2busername:$username%20%2bhostname:$mailhost%
20%2bfolder:INBOX&c=100"
```

This shell script searches the inbox of the user provided through **isshttpd**. You can validated different mappings by using a different user and mailhost combination.

Migrating from Java 6 to Java 7

This section describes how to migrate your Indexing and Search Service accounts from Java 6 to Java 7.

About Migrating from Java 6 to Java 7

If you used Java 7 to index the Indexing and Search Service store, then you do not need to migrate your Indexing and Search Service data. You can continue to update to future versions of Java 7, but you should not use any version of Java 6 with Indexing and Search Service.

The Java 7 release changed the Unicode representation of character strings compared to Java 6. If you created your Indexing and Search Services store using any Java 6 version, then updating to Java 7 may potentially cause some search queries to produce inaccurate results. For example, the search queries might return too few or too many matches, because of the changed data representation. To avoid this situation, you must reindex all or part of the Indexing and Search Service store data using Java 7. If you do not know what parts of the data are impacted, then you must reindex all accounts in the store.

Reindexing the Indexing and Search Service store can be very time consuming and disruptive to a production system. Depending on your data, the number of accounts impacted might be only a fraction of the total. Tools providing by Indexing and Search Service help you to manage indexing these accounts to minimize the Java 7 changes.

These Indexing and Search Service tools enable you to migrate to Java 7 by:

- Using the autosync process to identify which accounts must be reindexed under Java 7
- Halting all services, reconfiguring to use Java 7, and restarting services
- Submitting the accounts to be indexed using Java 7 to the autobootstrap process

The first and last steps can take a long time depending on how many accounts and how much data must be analyzed or indexed. The problem of inaccurate search results is limited to the time between restarting the service under Java 7 and the completion of the autobootstrap for each individual account. In this way all accounts are available for searching while Indexing and Search Service performs the corrections for Java 7.

Tools for Java 7 Migration

The **iss.indexsvc.periodic.autosync.deepcheck.enable** configuration parameter identifies which accounts to reindex for Java 7.

When you set the value of this parameter to **true**, the autosync processing checks for Java 7 character set problems in each folder in each account, and the folder state to be recorded in the index. When this parameter is set to **false** (the default), no checking for

Java 7 problems occurs, and no folder state updates occur. The folder state indicates whether any email in the folder must be reindexed under Java 7.

You must enable Indexing and Search Service autosync for the **iss.indexsvc.periodic.autosync.deepcheck.enable** parameter to have effect. When **iss.indexsvc.periodic.autosync.deepcheck.enable** is set to **true**, autosync runs normally but performs the extra checking for Java 7 character set problems. This checking causes autosync to take at least twice as long as usual, and possibly much longer, to complete.

The following command monitors the progress of the deep checking:

```
issadmin.sh --checkstore --detail deepcheck
```

This command collects and summarizes the information recorded in the folder state fields across all accounts. The output includes lines that you can extract to create a file suitable for use with the **--setautobootlist** *file* command. You can run this command whether deepcheck is enabled or not. Its output shows how much the autosync has completed and how much more indexing is needed before the Java 7 migration is complete. For more information, see "Java 7 Migration Example".

After you create the list of accounts that must be indexed under Java 7 (by using the **--checkstore--detail deepcheck** command), you submit the accounts to the autobootstrap queue for reindexing using the following additional option:

```
issadmin.sh --setautobootlist file --reboot
```

file is the file of accounts to be reindexed in the usual **--accountlist** format as extracted from the **--checkstore--detail deepcheck** output. The **--reboot** option informs the autobootstrap processing to delete the account if it already exists before reindexing it. In this manner Indexing and Search Service can still search the prior version of the account while it is waiting to be reindexed.

Java 7 Migration Example

The following example shows how to use Indexing and Search Service tools to migrate an Indexing and Search Service store from Java 6 to Java 7. This example assumes that you have created the Indexing and Search Service store and it is still running under Java 6.

1. Run the following command to determine how much data must be analyzed:

   ```
   issadmin.sh --checkstore --detail deepcheck
   ```

   The output of this command resembles the following:

   ```
   Wednesday, January 29, 2014 12:12:04 PM PST
   checkIndex succeeded for: /var/iss/index//store/dIndex will continue using this
   copy

   Header Summary found:
    Store ID: ISSID_9527c3c6-9fc0-47dd-af40-26e8aa4d4c25 created: 20140129164752
    default host name:              isshost.example.com
    default export/import path:     <none specified>
    total size of store instance:   0
    total number of accounts:       55
    total number of account groups: 55
    last known consecutive group:   155
    total search queries performed: 0
    total search query failures:    0
    total index events processed:   55
   ```

```
 last backup performed:          never
 last host number:               1
 last user number:               55
 attachment store enabled:       true
 dIndex memory locked:           false

 number of partitions (from acctMgr):0
 number of partitions (from dIndex):0

Checking contents of groups:
55 group documents found, as expected
no duplicate id records found in group documents
55 account documents found, as expected
no duplicate id records found in account documents
Checking folder status:
944 folders found, 0 have java 7 update issues,
   0 are OK, 0 are Java 7 safe, 0 are Java 7 indexed, 944 have no status
Time to check for   group duplicates: 0 seconds
Time to check for account duplicates: 0 seconds

Checking group assignments of accounts:
all account ids are assigned to groups, as expected
account numbers consistent
Time to check for group/account consistency: 0 seconds
# deepcheck folder analysis:
# number of accounts clear of problems: 0
# number of accounts safe from problems: 0
# number of accounts with java 7 update issues: 0
# number of accounts with java 7 indexed: 0
# number of accounts with no status: 55
# number of accounts with unknown status: 0

Wednesday, January 29, 2014 12:12:04 PM PST
```

2. Look at the section titled "Checking folder status" for the total folders in all accounts in the store, and those with status values marked. Because the deep checking has not yet begun, no folders have status yet. As the autosync checks each folder, these counts change over time.
3. Look at the last section titled "deepcheck folder analysis" for the status by accounts, which also shows that no status has been recorded. The information in this section grows as accounts are found that require reindexing.
4. Start the deep checking by setting the **iss.indexsvc.periodic.autosync.deepcheck.enable** configuration parameter to **true**.

   The autosync configuration parameters must also be enabled when you refresh the configuration. See "Selecting Appropriate Autosync Configuration Values" for information on sizing the autosync configuration to reflect the increased overhead that the deep checking incurs.
5. As the autosync checks the accounts, run the **--checkstore** command again to watch the progress. Expect at least twice the usual time for autosync to process all the accounts, and perhaps much longer if the accounts contain a great deal of data. For an Indexing and Search Service Store containing 100,000 accounts, it could take several days to deep check every account. After the autosync has checked every account, the process continues as new email and other changes to the accounts occur. Running the **--checkstore** command at this point produces output similar to the following:

```
Wednesday, January 29, 2014 03:25:17 PM PST
checkIndex succeeded for: /var/iss/index//store/dIndex will continue using this
copy

Header Summary found:
 Store ID: ISSID_9527c3c6-9fc0-47dd-af40-26e8aa4d4c25 created: 20140129164752
 default host name:              isshost.example.com
 default export/import path:     <none specified>
 total size of store instance:   0
 total number of accounts:       55
 total number of account groups: 55
 last known consecutive group:   155
 total search queries performed: 0
 total search query failures:    0
 total index events processed:   55
 last backup performed:          never
 last host number:               1
 last user number:               55
 attachment store enabled:       true
 dIndex memory locked:           false

 number of partitions (from acctMgr):0
 number of partitions (from dIndex):0

Checking contents of groups:
55 group documents found, as expected
no duplicate id records found in group documents
55 account documents found, as expected
no duplicate id records found in account documents
Checking folder status:
944 folders found, 115 have java 7 update issues,
   829 are OK, 0 are Java 7 safe, 0 are Java 7 indexed, 0 have no status
Time to check for   group duplicates: 0 seconds
Time to check for account duplicates: 0 seconds

Checking group assignments of accounts:
all account ids are assigned to groups, as expected
account numbers consistent
Time to check for group/account consistency: 0 seconds
# deepcheck folder analysis:
# number of accounts clear of problems: 34
# number of accounts safe from problems: 0
# number of accounts with java 7 update issues: 21
# number of accounts with java 7 indexed: 0
# number of accounts with no status: 0
# number of accounts with unknown status: 0
# 3 folders in account test have java 7 update issues
# folders:INBOX, Sent, info-ims
;;;;;isshost.example.com;test
# 1 folder in account durga has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;durga
# 1 folder in account test1 has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;test1
# 16 folders in account jennifer have java 7 update issues
# folders:???, INBOX, Sent, bar/foo bar, convergence, folderhierarchy/a/1
#    i18n, javaone2009, llnl.gov MIME tests, popAcct, problem attachment
#    stacy, ubuntu-art, |< |_| |\| -|-, time-exceeded-msgs,
time-exceeded-msgs-2
```

```
;;;;;isshost.example.com;jennifer
# 1 folder in account jeffb has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;jeffb
# 1 folder in account jeffa has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;jeffa
# 3 folders in account jake have java 7 update issues
# folders:Sent, INBOX, APOD
;;;;;isshost.example.com;jake
# 1 folder in account test8 has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;test8
# 2 folders in account test9 have java 7 update issues
# folders:INBOX, Sent
;;;;;isshost.example.com;test9
# 1 folder in account test5 has java 7 update issues
# folders:bug13521409
;;;;;isshost.example.com;test5
# 18 folders in account test2 have java 7 update issues
# folders:MultipleAttachments, ODFAttachments, INBOX, MULTIPLEWordAttachment
#    MultiplePDFAttachment, OtherAttachment, PDFAttachment, PDFFIVE, PDFFOUR
#    PDFNew, PDFTHREE, PDFTWO, PowerPointAttachments, RTFAttachment, Sent
#    demonov19_2007, nestedFolders/nested1, nestedFolders/nested1/c
;;;;;isshost.example.com;test2
# 2 folders in account paul have java 7 update issues
# folders:INBOX, Sent
;;;;;isshost.example.com;paul
# 1 folder in account david has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;david
# 1 folder in account anil has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;anil
# 1 folder in account dev has java 7 update issues
# folders:INBOX.old
;;;;;isshost.example.com;dev
# 27 folders in account test10 have java 7 update issues
# folders:DebuggingParseException, INBOX, support, forum, messaging
#    sac-interest, s10, sec list, ubuntu lists/bazaar
;;;;;isshost.example.com;test10
# 5 folders in account rick have java 7 update issues
# folders:INBOX, Sent, a16new, demonov19_2007, partnos
;;;;;isshost.example.com;rick
# 1 folder in account admin has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;admin
# 4 folders in account lea have java 7 update issues
# folders:Drafts, INBOX, Sent, leaemptyfolder/sub folder
;;;;;isshost.example.com;lea
# 22 folders in account i18n have java 7 update issues
# folders:INBOX, ??????/users-ru, 12May2011/?esky, ???/ubuntu-ko,
???/xen-japanese
#    12May2011/???, 12May2011/?????, 12May2011/???, 12May2011/????,
12May2011/Deutsch
#    12May2011/Espa?ol, 12May2011/Fran?ais, 12May2011/Polski,
12May2011/Portugu?s Brasileiro
#    12May2011/Pycc???, Drafts, ??/ubuntu-zh, Espa?ol/ubuntu-ar, Trash
#    Sent, Deutsch/ubuntu-de, Fran?ais/users-fr
;;;;;isshost.example.com;i18n
```

```
# 3 folders in account geetha have java 7 update issues
# folders:INBOX, Sent, messaging
;;;;;isshost.example.com;geetha

Wednesday, January 29, 2014 03:25:17 PM PST
```

6. Note how the "folder status" has changed. All folders have a status now, and some "have Java 7 update issues." At this point there are no folders marked as "Java 7 safe" or "Java 7 indexed" because the reindexing has not yet begun.
7. Note how the "deepcheck folder analysis" section has changed. You do not need to reindex the accounts marked "clear of problems". They do not contain any data that requires them to be rebootstrapped. You must reindex the accounts marked "with Java 7 update issues" under Java 7 to avoid search problems. The other totals should be zero after all accounts have been checked. However, you might see a small number of accounts or folders that still show "no status" after the autosync has completed processing all accounts. This situation could be due to accounts being added since the autosync cycle began, or accounts that are not in the Active state. (Autosync only processes accounts in the Active state, so if any accounts are in the Inactive, Bootstrap, or Unknown states, you should use the **--listbrief** command to identify them. You can then either change them to Active for autosync to find later, or delete them if unneeded, or keep them for manual processing after the Java 7 update.) The remaining part of the output shows the individual accounts that have Java 7 issues requiring reindexing. For each account, the number and names of the folders in which problems are detected are listed. This account information can give you a feel for how frequently problems were detected.
8. To generate the file of accounts to be submitted to autobootstrap, find the string ";;;" by using the **grep** command. Extract only the account records needed for the **--setautobootlist** *file* command into a file.
9. Once you have the file of accounts to reindex, you are ready to update to Java 7. Install Java 7 in addition to Java 6 on your host so that it is ready when you want to switch, and not delay the restart of the services.
10. Ensure that the **iss.indexsvc.periodic.autobootstrap.enabled** parameter is currently set to **false**.

    This setting prevents autobootstrapping from running at this point.
11. Run the following command to set up the accounts to be autobootstrapped before you shut down Indexing and Search Service services:

    ```
    issadmin.sh --setautobootlist file --reboot
    ```
12. Shut down Indexing and Search Service services.
13. After you shut down Indexing and Search Service services, change the following **jiss.conf** parameter to indicate the change to the Java 7 installation:

    ```
    java.home
    ```
14. After making this change, restart Indexing and Search Service services.
15. Enable autobootstrapping by setting the **iss.indexsvc.periodic.autobootstrap.enabled** parameter to **true**.

    If you preset the accounts in the autobootstrap queue, you should also enable the autobootstrap before restart. At this point all the accounts are using Java 7, but only the accounts that need reindexing should show any difference, and only if a

search query happens to occur that triggers a character representational problem. From this point on, do not configure the store to use Java 6.

16. Refresh the Indexing and Search Service services:

```
issadmin --refresh
```

17. The autosync and deepcheck are still enabled after the update to Java 7. The folder status continues to be updated. Use the **--checkstore** command to monitor the progress of the reindexing. The following output shows Java 7 reindexing:

```
Wednesday, January 29, 2014 03:41:56 PM PST
checkIndex succeeded for: /var/iss/index//store/dIndex will continue using this
copy

Header Summary found:
 Store ID: ISSID_9527c3c6-9fc0-47dd-af40-26e8aa4d4c25 created: 20140129164752
 default host name:              isshost.example.com
 default export/import path:     <none specified>
 total size of store instance:   0
 total number of accounts:       55
 total number of account groups: 55
 last known consecutive group:   155
 total search queries performed: 0
 total search query failures:    0
 total index events processed:   69
 last backup performed:          never
 last host number:               1
 last user number:               69
 attachment store enabled:       true
 dIndex memory locked:           false

 number of partitions (from acctMgr):0
 number of partitions (from dIndex):0

Checking contents of groups:
55 group documents found, as expected
no duplicate id records found in group documents
55 account documents found, as expected
no duplicate id records found in account documents
Checking folder status:
944 folders found, 14 have java 7 update issues,
   0 are OK, 192 are Java 7 safe, 738 are Java 7 indexed, 0 have no status
Time to check for   group duplicates: 0 seconds
Time to check for account duplicates: 0 seconds

Checking group assignments of accounts:
all account ids are assigned to groups, as expected
account numbers consistent
Time to check for group/account consistency: 0 seconds
# deepcheck folder analysis:
# number of accounts clear of problems: 0
# number of accounts safe from problems: 34
# number of accounts with java 7 update issues: 7
# number of accounts with java 7 indexed: 14
# number of accounts with no status: 0
# number of accounts with unknown status: 0
# 3 folders in account test have java 7 update issues
# folders:INBOX, Sent, info-ims
;;;;;isshost.example.com;test
# 1 folder in account durga has java 7 update issues
```

```
# folders:INBOX
;;;;;isshost.example.com;durga
# 1 folder in account test1 has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;test1
# 1 folder in account jeffa has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;jeffa
# 3 folders in account jake have java 7 update issues
# folders:Sent, INBOX, APOD
;;;;;isshost.example.com;jake
# 1 folder in account test8 has java 7 update issues
# folders:INBOX
;;;;;isshost.example.com;test8
# 4 folders in account lea have java 7 update issues
# folders:Drafts, INBOX, Sent, leaemptyfolder/sub folder
;;;;;isshost.example.com;lea

Wednesday, January 29, 2014 03:41:59 PM PST
```

18. The totals in the "folder status" are shifted to "Java 7 safe" and "Java 7 indexed" indicating folders had no previous problems or have been reindexed using Java 7 respectively.

19. The "deepcheck folder analysis" counts have changed, and the list of accounts needing reindexing has shrunk as the bootstrapping proceeds. Eventually all the accounts submitted for autobootstrap are indexed, and the **--checkstore** output resembles the following:

```
Wednesday, January 29, 2014 03:51:02 PM PST
checkIndex succeeded for: /var/iss/index//store/dIndex will continue using this
copy

Header Summary found:
 Store ID: ISSID_9527c3c6-9fc0-47dd-af40-26e8aa4d4c25 created: 20140129164752
 default host name:              isshost.example.com
 default export/import path:     <none specified>
 total size of store instance:   0
 total number of accounts:       55
 total number of account groups: 55
 last known consecutive group:   155
 total search queries performed: 0
 total search query failures:    0
 total index events processed:   76
 last backup performed:          never
 last host number:               1
 last user number:               76
 attachment store enabled:       true
 dIndex memory locked:           false

 number of partitions (from acctMgr):0
 number of partitions (from dIndex):0

Checking contents of groups:
55 group documents found, as expected
no duplicate id records found in group documents
55 account documents found, as expected
no duplicate id records found in account documents
Checking folder status:
944 folders found, 0 have java 7 update issues,
   0 are OK, 155 are Java 7 safe, 789 are Java 7 indexed, 0 have no status
```

```
Time to check for   group duplicates: 0 seconds
Time to check for account duplicates: 0 seconds

Checking group assignments of accounts:
all account ids are assigned to groups, as expected
account numbers consistent
Time to check for group/account consistency: 0 seconds
# deepcheck folder analysis:
# number of accounts clear of problems: 0
# number of accounts safe from problems: 34
# number of accounts with java 7 update issues: 0
# number of accounts with java 7 indexed: 21
# number of accounts with no status: 0
# number of accounts with unknown status: 0

Wednesday, January 29, 2014 03:51:08 PM PST
```

All folders and accounts show either "safe" or "indexed", indicating the update to Java 7 is complete.

20. Disable the deepcheck by setting the **iss.indexsvc.periodic.autosync.deepcheck.enable** parameter to **false**, to reduce the overhead of the autosync.
21. Refresh the configuration.

```
issadmin.sh --refresh
```

If the counts for other than "safe" and "indexed" are not zero, then there might be accounts that have not been corrected because they were not Active (as noted previously). Use the **--listbrief** command to identify any accounts that you might need to delete and reindex manually.

Configuration Considerations for Java 7 Migration

The previous example involved a small number of accounts (55). A production Indexing and Search Service store likely contains several thousand times more accounts and data, and so the **--checkstore** command might take a few minutes to complete instead of seconds as in this example. The time needed to deep check all accounts, and to reindex all accounts in which a Java 7 problem is detected, will also be long depending on the size of your data and how many problems are detected. Therefore, take care to configure the autosync and autobootstrap parameters to keep the overhead down and enable normal operation of Indexing and Search Service services. The following section provides guidelines for deciding how to balance the costs of the Java 7 upgrade with normal server operation.

Selecting Appropriate Autosync Configuration Values

Use the following guidelines to determine appropriate autosync configuration values for your deployment.

- **Check log files.** Before enabling the deep check feature, examine the **IndexSvc** log files for an indication of the current autosync overhead. If autosync is already enabled, and the log level allows for INFO messages, look for messages containing the phrase "findAllCandidates: time to create list." These messages occur when autosync has completed a cycle of all accounts and generated a new list to start the process again. The time stamp of such messages can help you determine approximately how long it takes the server to cycle through all the accounts for autosync using current configuration parameters. This time duration is less than

the lower bound for deep check processing, because deep checking is much more resource intensive that the typical autosync processing. Use this time duration to get an approximation for how long the deep check might take to complete.

- **Process accounts in parallel.** The deep checking overhead is significant on both Indexing and Search Service and Messaging Server. Each email in each folder of each account must be scanned for Java 7 migration issues. Thus the load on the Messaging Server is comparable to a full bootstrap of the account, and somewhat less on the Indexing and Search Service server. Spread this load out over time by adjusting the configuration parameters to process only a small number of accounts in parallel simultaneously, to avoid incurring delays in the normal Indexing and Search Service search services. As shown in the previous example, use the **--checkstore--detail deepcheck** command to monitor progress to achieve the right balance between normal Indexing and Search Service processing and the deep check in autosync.
- **Set parameters for the first time running autosync.** If you have not been running autosync, and do not have a way to estimate the full cycle duration, then set the **autosync iss.indexsvc.periodic.autosync.count** and **iss.indexsvc.periodic.autosync.thread.count** parameters to one third the default parameter values in the **jiss.comf.template** file to begin deep check processing. Observe the overhead and adjust these parameter values as appropriate to the load on the servers.
- **Start with small count and interval values.** You can modify the autosync parameters in the **jiss.conf** file by running the **issadmin.sh --refresh** command without having to restart Indexing and Search Service. However, the effects of these parameters do not take effect immediately. The current work period must finish before the new values are applied. Thus the "count" and "interval" values should be kept relatively small until you have determined the load on the system so they can be refreshed quickly without incurring a long wait for the current autosync work or interval to complete.
- **Run the deep check off hours.** Because the deep check processing might take a long time, run it during known times of low system load. The **iss.indexsvc.periodic.autosync.deepcheck.enable** parameter is refreshable, so you can turn it off or on as the load on the system varies. The work so far completed is recorded in the index, so no information is lost. However, completing the deep check takes much longer if not run all the time.
- **When to stop deep check.** If, after running the deep check for a while, the **--checkstore--detail deepcheck** command output indicates a very high rate of accounts that require reindexing (say 80 to 90 percent of all accounts checked), you might consider stopping the deep check. Then you can simply shut down the server, update to Java 7, and rebootstrap all accounts in the index. Taking these steps avoids the extra overhead of the deep check, and completes the Java 7 update in less time. The downside is that you do not know which accounts might return inaccurate search results during the period before being indexed using Java 7.
- **Create your own bootstrap commands.** The **--checkstore--detail deepcheck** command output includes comments showing those folders in an account which contain data that must be reindexed. Only the folders listed must actually be corrected. To reduce the amount of data to be reindexed, you can create your own list of **--bootstrap** commands that use the **--folder** *name* option on just the folders indicated. Creating your own list also reduces the time needed to complete the bootstrap. However, you must generate such a command list manually, which is harder to manage, and perhaps more error prone. Create your own list only in

special cases, such as if the **--checkstore--detail deepcheck** command output shows that most accounts have the same single folder (such as INBOX) to be reindexed. You could use a sequence of commands such as the following to reindex each account *XXXX*, and complete the Java 7 update more quickly than the general autobootstrap procedure outlined previously.

```
issadmin.sh --user XXXX --setstate I
issadmin.sh --user XXXX --deletefolder --folder INBOX
issadmin.sh --user XXXX --bootstrap --folder INBOX
```

Any accounts which do not follow this pattern would have to be indexed individually based on which folders are needed to complete the update.

Automatically Checking and Repairing dIndex Problems

The **--checkstore** command detects and repair inconsistencies in the dIndex, and corrects problems in the account group index directories (meta and content). Some checks complete with little impact on the system, while others are performance intensive. You can configure the **--checkstore** detections that cause little performance impact to run automatically when the IndexService starts. Doing so helps to catch dIndex problems sooner rather than later. In addition, you can configure Indexing and Search Service to automatically repair some of these problems, resulting in fewer failures overall.

You can also configure other **--checkstore** capabilities, such as checking the specific account group index for consistency, to run during the periodic autosync **--checkaccount** processing. The overhead for checking each account is minor, but over time, autosync checks every account. The additional checking delays the autosync processing slightly, but results in fewer problems that **--checkstore** needs to fix later.

To automatically check the dIndex when the IndexService starts:

1. Set the following configuration parameter to **true**:

   ```
   iss.indexsvc.periodic.autosync.checkstore.enabled
   ```

   By default, the parameter is set to **true**.

2. To view Warning level log messages in the log, set the following parameter to true:

   ```
   iss.indexsvc.periodic.autosync.enabled
   ```

During autosync checking for each account, an account's meta and content index directories are checked for existence and readability. This checking occurs every time autosync processes an account.

To automatically repair dIndex problems:

1. Ensure that the following configuration parameters are set to **true**:

   ```
   iss.indexsvc.periodic.autosync.enabled
   iss.indexsvc.periodic.autosync.checkstore.enabled
   ```

2. Set the following configuration parameter to **true**:

   ```
   iss.indexsvc.periodic.autosync.checkstore.sync.enabled
   ```

During autosync syncing for each account, the meta and content index directories are repaired for any account in need of correction.

Selecting Autobootstrap Configuration Values

Once the deep check processing has produced a list of accounts to be indexed under Java 7, estimate the reindexing time for those accounts to determine what configuration parameters to use for the autobootstrap processing. The account list that you use for the **--setautobootlist** *file* command can also be used in the **--accountinfo** command to find how many emails each account contains. However, if the number of accounts is large, this approach might not be practical.

As with the autosync parameters, start by using relatively small values for the autobootstrap "count," "interval," and "thread.count" parameters, so that you can quickly **--refresh** them as you monitor the autobootstrap process. The order of the accounts being bootstrapped is roughly the order of the names in your **--setautobootlist** *file*. Accounts later in the list have a larger likelihood of being searched before being rebootstrapped under Java 7. You can reorder the lines in the **--setautobootlist** *file* if you have any preference of which accounts you would rather have reindexed first. The order generated by the deep check is random.

After you have submitted the accounts for autobootstrap with the **--setautobootlist** *file***--reboot** command, you can adjust the autobootstrap parameters based on how quickly the various accounts finish and the load on the system. If the autobootstrap load causes service to degrade too much, you can reduce the autobootstrap "count," "interval," and "thread.count" parameter values by using the **--refresh** command. You can even disable autobootstrapping as you think best. The list of accounts is retained unless you use the **--unsetautobootlist** *file* command to remove accounts not yet finished bootstrapping. If you "unset" and then later "set" any accounts, remember to use the **--reboot** option with the **--setautobootlist** *file* command.

Disaster Recovery

Hardware failures, power loss, or software bugs can corrupt index store data. As a result, Indexing and Search Services might fail or stop responding.

When problems like these occur, normal operations might resume automatically, or you might need to perform significant intervention. The severity of and recovery from such failures depend on where the data corruption occurs.

The index store consists of two major parts:

- The master directory (dIndex) - Contains information about the organization of accounts
- The account group index directories - Contain data about individual emails and folders in each account

Recovery from failures in each of these parts requires different approaches. The following sections contain general information about disaster recovery approaches and specific scenarios.

Recovering from dIndex Directory Data Corruption

The first high-level step for disaster recovery is to ensure that the dIndex is usable.

1. Before attempting recovery operations on the dIndex data, ensure that the servers are stopped.
2. Look for any **write.lock** file in the dIndex directory and remove it. The presence of a **write.lock** file means that dIndex was likely being written when the failure occurred, so it may not be reliable.

3. Before starting the services or any **issadmin.sh** commands, run the following command to see if its structure is consistent:

   ```
   issadmin.sh --checkstore --detail dindex
   ```

   If no problems are reported, then dIndex might not be seriously effected. If you see any warnings of the form:

   ```
   WARNING: checkForCorruptdIndex: index path:
   /var/opt/sun/comms/jiss/index/store/dIndex
      dIndex is not clean, run checkIndex manually.
   ```

   then the **dIndex** must be fixed or replaced by a backup copy to restore server functions.

   > **Note:** This message is similar to the result of running the **checkIndex.sh** script on the dIndex.

   You can specify partitions to the **dIndex** (refer to "Improving Global Directory Index Performance"). If the value of configuration parameter **iss.store.partitions.count** is greater than zero, then in addition to the dIndex directory, there are **dIndex***NN* directories that contain parts of the dIndex information. The **--checkstore** command checks each of the partitions as well. Each **dIndex***NN* directory is backed up independently and therefore can be used to recover from corruption.

4. If **checkIndex.sh** problems are detected, make a backup copy of the entire dIndex directory to preserve the original data. If there are backup files in the store (named **dIndex.backupA**, **dIndex.backupB**, and **dIndex.backupC**), determine which is the most recently changed, and run **checkIndex.sh** on each to find if any are valid. If the **checkIndex.sh** output for one or more shows no sign of corruption, then decide whether the most recent backup is sufficiently current that you can use it to replace the corrupted dIndex. Your alternative is to run the **checkIndex.sh** command with the **-fix** option again on the dIndex. This action likely causes information loss in dIndex. In either case, using one of the backupA/B/C copies or using **-fix** probably means that some information has been lost. Depending on the nature of the corruption, using a recent backup might be a better approach, since the **dIndex** was in a known consistent state at the time the backup was created. However, there is no way to tell what data would be missing since the backup was created. (Any number of accounts might need to be synchronized again.)

   If dIndex partitioning is being used, each individual **dIndex***XX* directory can be replaced by its backup in a similar manner. The effect of such replacement is much more limited than when dIndex is replaced, because each partition only contains information about a subset of the accounts. To enable the services to be run again, every partition of the dIndex must be able to pass the **checkIndex.sh** test.

   After these corrections to dIndex are complete, run the **--checkstore --detail dindex** command again. It should show no corruption, but may show other problems if any older backup copies have been incorporated.

5. After running the **checkIndex.sh** command to correct dIndex problems, if the **--checkstore --detail dindex** command still shows problems, run the following command to correct the store:

   ```
   issadmin.sh --checkstore --detail full --sync
   ```

If your store contains 100,000 accounts or more, this **--detail full** command might take at least 30 minutes to complete, because the command must check each individual account group index. Completion time varies and depends upon the number of accounts and the amount of data in the store. This command also generates a lot of diagnostic logging to standard output. When this command completes, account names that have been deleted by the **--sync** option (to resolve conflicts) are appended to the *iss.store.dir***/store/checkstore.accounts** file. This file has the same format as that used by the **--accountlist** *FILE* option. This file is automatically read when the **IndexService** restarts and the accounts in it are added to the autobootstrap queue to be recreated, as long as the value of the configuration parameter **iss.indexsvc.periodic.autobootstrap.enabled** is **true**. Once the accounts are added to the autobootstrap queue, they are deleted from this file. The system then manages the accounts as part of the autobootstrap queue. If you do not want to bootstrap any specific account in this file, remove the account from the file before restarting the services. You can also add other accounts to this file as needed.

6. After running the **--sync** command, run the following command to verify that the store is problem free:

```
issadmin.sh --checkstore --detail full
```

If errors still appear, retry the **--sync** command. If errors still persist, you should at least be able to restart the services. Accounts with errors might continue to have problems, but your system should perform better.

> **Note:** If running the first -**-checkstore --detail full --sync** command does not correct all errors, contact Oracle Support with the output.

The **--detail full** option is equivalent to the -**-detail dindex,store,group** option. While you can run the **--sync** command on the dIndex and store separately (the **group** option only matters when the store option is specified), use the **--detail full** option instead of separate commands. Limiting the checking to one or the other option reduces the information available for the checking. Thus, the system might not fully diagnose some problems, resulting in not enough information to clean up all the extraneous data in the store.

Correct any problems shown in the output before attempting to start services. You might also be able to use the backup copy of dIndex to diagnose what was lost by using the **lucli.sh** tool, but this process is complicated.

When the data in dIndex is corrupted, the effect might be global or local. In the worst situations, services cannot be started, or basic **issadmin.sh** commands like **--accountinfo**, **--listbrief**, **--listaccounts**, or **--liststats** fail to produce expected output or hang.

Corrupted global data can be detected in the header information of the output from many common **issadmin.sh** commands such as **--listbrief**. For example, if the number of accounts or groups is reported incorrectly, some critical global data might be corrupted. In this case, the main action you can take for recovery is to recreate (bootstrap) the entire index from Messaging Server store data. This process can be lengthy, so you might want to take the time to determine if parts of the store can be salvaged.

You can run the **issadmin.sh** commands whether the Index and Search Service servers are started or stopped. Failures in **issadmin.sh** commands when the services are not

running might indicate if the problem is global or local to only some accounts or groups.

If only some accounts or groups appear to be failing, use the following:

1. Disable those accounts using the **--setstate I** command. If the services start without failure, then the problem might be local and these accounts can be corrected while the rest of the services are active.
2. Try using the **--deleteaccount** or **--deletefolder** commands on suspicious accounts to see if the services can continue to operate. Sometimes services can be restored after you have removed a few accounts. You can then bootstrap these accounts again.
3. Ensure that the rest of the data is intact by using the **--checkaccount** or **--checkfolder** commands on every individual account (refer to the **--accountlist** option).
4. If the problems with dIndex appear to be corrected, you can use the **--sync** command to resolve any other problems in individual accounts.

If some **issadmin.sh** commands appear to work but dIndex is still corrupted, you can try the following approach to reduce the time that is required to recreate the store:

1. Use the **--export** command to export any accounts that you can.
2. Remove the corrupted store and start rebootstrapping.
3. Import into the new store any accounts that you can successfully export by using the **--import** command.
4. Use **--checkaccount** and **--sync** commands on every account to ensure that the data is current.

Recovering from Individual Account Groups Corruption

If the dIndex data is not corrupted but some accounts appear to be failing, then the data corruption might be localized in a set of account groups. Since each account group contains independent "meta" and "content" index directories, the damage might be limited to specific groups of accounts. You might be able to correct the damage without impacting the rest of the store, as described in the following steps:

1. With services stopped, start by searching for **write.lock** files in the rest of the store with a command such as the following:

   ```
   find /var/opt/sun/comms/jiss/index/store/ -name write.lock
   ```

   All **write.lock** files are found in the *IndexSearch_home***/index/store/locks** directory, not the individual group index directories. The name of each file in this directory contains the number of the group to which it belongs.
2. Remove all the **write.lock** files that you find. Any group directories identified by such files are candidates for further investigation, because they are likely to be corrupted.
3. Use the **checkIndex.sh** command on each of the pair of index directories in each suspect group, using the same procedure as described in the preceding dIndex corruption section. (The last four digits of the group number are used to find the directories in the store; for example, for group 12345, look for directories with the path **23/45/index12345_meta** and **23/45/index12345_content**.) The individual group directories do not have backup files like the **dIndex**, so if the **checkIndex.sh** command indicates corruption, that index directory must be fixed by using the **-fix**

option, which will likely cause data loss. (Sometimes, after running **checkIndex.sh -fix**, the group index might still not be clean. If the group index is not clean, try rerunning the **-fix**. More data is lost with each such command, but eventually the index should be cleaned. If an index cannot be cleaned by repeated **--fix** commands, then you must delete all files in that index directory and recreate the group. Refer to the section below about creating a new group for details.) If any of the **checkIndex.sh** commands indicate that data was lost, accounts in those groups are likely corrupted and need attention.

4. Run the **issadmin.sh --checkstore --detail store** command, and check its output for other accounts or groups which may show problems. (If your store contains more that 100000 accounts, the **--detail store** command might take a very long time (up to 40 minutes) to complete.)
5. Disable services to each suspect account by using the **issadmin.sh --setstate I** option. Then use the **--accountinfo** option, followed by **--checkaccount** or **--checkfolder** to diagnose account specific problems.
6. If necessary, try to delete any data that appears corrupted by using the **--deleteaccount** or **--deletefolder** options, and bootstrap or use the **--sync** option to try to correct the accounts.

If these attempts fail to correct the accounts, then a more global problem with dIndex might exist. The services might be able to run with these specific accounts disabled, but to restore full service to all accounts, you must correct the larger global problem. Running the **issadmin.sh --checkstore --detail dindex** command might help at this point. Some other approaches to consider include the following:

- If you suspect that the problem might be related to a specific account group directory, you might be able to move the accounts in that group into another group by using either the **--moveaccount** or **--export** and **--import** command options. If either the group directories or dIndex is very badly damaged, these commands might also fail, or only transfer part of the account. Always use **--checkaccount** after moving or importing an account to see if it succeeded. Afterward, the **--sync** option might be able to correct the account.
- If you are unable to create a group when trying to move an account, then dIndex is likely corrupted. Depending on how many accounts are configured as allowed per group, you might be able to move the accounts into existing groups, but not into a new group. Moving accounts into existing groups can allow the services to be resumed temporarily, but you can only correct this failure of dIndex by recreating the store again from the start.
- As a last resort, bootstrapping a folder or account into a different group should restore the data for specific corrupted accounts. Always remember to delete the folder or account from the damaged group directory before attempting to bootstrap again.

Finally, all data corruption might not be evident using these techniques, because the structure of the data is correct but the data values have been modified. Search results might prove inconsistent or wrong. This kind of corruption might be less serious but harder to detect because it only effects the correctness of individual searches, not the operation of the entire store. You can restore services to the other accounts while fixing the affected folder or account. In this case, you can delete the suspect folders or accounts and bootstrap them while other services are running.

Recreating an Individual Account Group Index Directory

If one or both of the index directories of an account group (the "meta" and "content" directories) cannot be corrected by using the **checkIndex.sh --fix** command, then the only way to correct this group is to replace the contents of the corrupted account group index completely, and all data for all accounts in that group is lost.

To recreate an individual account group directory:

1. Create an empty account group.
2. Delete all files in the corrupted index directory.
3. Copy all files from the empty account group into the corrupted directory.

   The accounts that were in that group can then be managed using the normal **issadmin.sh** commands, because the knowledge about those accounts still exists in the dIndex, although the account group index is empty.

A simple way to create an empty account group is to create a nonexistent account in a specific group, and then delete that account. This process creates the structure of an empty index directory that can then be copied to correct any number of group index directories that had to be deleted. When copying the empty account group directory, be sure to preserve the owner, group, and access rights of the empty account group directory (as with the "**cp -p**" command). To ensure that the normal default allocation of accounts to groups does not effect the empty group directories, use the **--group** option with the **--createaccount** command with a small group number, less than 100. (Group numbers less than 100 are not normally allocated by default, so that they can be used for administration purposes like this without interference from the rest of the services.)

Precautions to Reduce Recovery Time

Because recovery from a disaster can be so costly, you might want to perform some routine maintenance functions regularly to make recovery faster. For dIndex, backup copies of the entire dIndex are automatically made; these are named **dIndex.backupA**, **dIndex.backupB**, and **dIndex.backupC** and are located in the same directory as dIndex. These backup directories are rewritten periodically based on the value of the **iss.store.account.optimizeinterval** configuration parameter. (The path to the most recent backup appears in the header output for **--accountinfo** and other options of the **issadmin.sh** command.) These backups might allow you to recover some functionality if critical data is corrupted.

After the original bootstrap of accounts is complete, much of the global critical information in dIndex stays stable. Therefore, you might be able to temporarily replace the corrupted dIndex by a backup version while you check the rest of the system for failures. Depending how long ago the backup was made, you might be able to restore services in this manner for many users temporarily until you can complete a full rebuild of the store. However, doing so is not a fully reliable solution for the problem. You might need to perform a full rebootstrap of all accounts, although the system might be able to provide services to some accounts while the rest are being repaired.

You can back up individual accounts by using the **--export** option, and you can use the snapshot of the account to recover lost data by using the **--import** command. Then, after you have fixed any corruption, using the **--checkaccount** and **--sync** commands to correct the data might be quicker than a full rebootstrap of the account. However, the cost of exporting and importing each account in both time and disk space might be greater than is reasonable for large numbers of accounts.

Specific Disaster Scenarios

The topics in this section contain guidelines for how to respond to specific severe problems.

Recovering From Disk Space Full

Running out of disk space can create serious problems:

- If the disk storage system containing the Indexing and Search Service store becomes filled, indexing services begin to fail.
- If the disk containing the log files becomes filled, then further WARNING and SEVERE messages are also lost, so indications of what is wrong might also be missing. Some index data might be lost, and the index directories might be corrupted.

If you discover the disk has become full or is extremely close to full (98% capacity or more), immediately stop the services to prevent failing service requests from corrupting index data. Then recover space on the full disk until it is at least below 95% capacity. Follow the steps outlined in "Disaster Recovery" to determine if data corruption must be corrected before restarting the services.

If you detected the disk full problem and shut down services before any damage to the index occurred, you can restore services while you take steps to reduce the capacity of the disk further:

Some actions you can take to reduce the disk space for the store include:

1. Placing the log directory on a separate disk
2. Locating the export/import snapshot directory on a separate disk
3. Removing any defunct or inactive accounts from the Indexing and Search Service store

Using the disk space sizes from the output of the **--accountinfo** and **--listaccounts** commands, determine whether you should **--export** any accounts to other Indexing and Search Service store instances based on how much space would thus be recovered.

You might consider investing time to prevent the disk full condition, rather than spending more time and effort in recovering from a badly corrupted store. Monitoring the disk space left, for example with a **cron** job, would enable you to detect when a problem is imminent and to stop the services before serious damage can occur. If you allow enough margin, such as detecting when the disk is 85% to 90% full, then you can sometimes increase disk space as described previously without shutting down the services. How actively you check for problems depends on how much unused disk space your store usually contains, and how expensive a recovery from the disk full condition would be.

Out of Memory Exceptions

A warning message indicating "OutOfMemoryException" (OOME) might appear infrequently in the index log. This message is usually caused by an email that is unusually large or has a very complicated body structure, which causes an indexing service request to fail (typically the bootstrap or **--sync** commands). When this failure occurs, it might cause the index service process to hang, requiring you to stop and restart the service.

Whether or not the index service process hangs, the OOME message means that the account indicated in the message has suffered a failure, and the corresponding index data might be corrupted.

The general steps to begin recovery from an OOME message are:

1. Stop any running commands that affect the account. (See "Interrupting Commands" for details.)
2. Check the account group meta and content index directories of the damaged account for **write.lock** files.

All **write.lock** files are found in the *IndexSearch_home***/index/store/locks** directory, not the individual group index directories. The name of each file in this directory contains the number of the group to which it belongs.

The presence of any **write.lock** files means that an **IndexWriter** was opened to the account, and data might have been written into the file when the OOME occurred. This IndexWriter must be cleared and any corresponding **write.lock** files must be deleted before the account can be corrected.

Shutting down the index service clears the **IndexWriter**, but service to all accounts is interrupted for some length of time. To avoid this service interruption, you can try the following general approach:

1. To clear the specific **IndexWriter** that was used, delete any **write.lock** files from the index meta and content directories.
2. To avoid losing all the data in the account index, run the **--export** command.
3. Delete the account.
4. Recreate the account and use the **--import** command to restore the account data. The account is likely out of sync with the Messaging Store. It might be wise to create the account in a separate new group to avoid future problems.
5. To avoid repeating the failure, you must prevent the email that caused the original OOME problem from being processed. Use the **--ignorefolder** command and specify the folder containing the offending email to prevent all emails in that folder from being indexed again.
6. Use the **--sync** or bootstrap commands to recover the remainder of the account data. Unfortunately, this approach causes the loss of all data in the folder being ignored, including data unrelated to the failure. If you can move emails that are causing the OOME to another empty folder on the Messaging Store, then you can avoid the loss of indexing of other emails in the folder. However, this approach might not be acceptable to your users.

OOME problems might not be repeatable, since memory use varies significantly depending on what else is happening on the system at the time. After the account is indexed with a folder marked ignored, you might want to attempt to **--unignorefolder** the single folder and bootstrap it again when the system is less loaded. This approach might not succeed. If another OOME occurs, then you must repeat the same recovery process. Using such an approach depends upon what and how much data in the ignored folder will not be indexed.

Also, any **write.lock** found in an account index group directory indicates that there is an IndexWriter actively writing to an index, but only a single IndexWriter is used by all accounts in any group index. If there is only one account assigned to the group, then you know the account causing the OOME is the one affected.

However, if you have assigned multiple accounts to a group, this procedure of recovering from OOME failure might cause interaction with those accounts as well. To ensure that the accounts are valid, use one of the following approaches:

- Disable all accounts in the group (using **--setstate I**) and back them up with **--export** before starting to recover. After recovery is complete, you can restore these accounts and use the **--sync** command to bring them up to date.
- Alternatively, you can use the **--moveaccount** command to move these accounts into other groups before attempting the recovery.

Always do **--checkaccount** after moving or restoring accounts to ensure that the resulting accounts are valid.

Renaming an Indexing and Search Service Host

To rename an Indexing and Search Service host:

1. Change the host name in the operating system.

   See your operating system documentation for more information.

2. In Indexing and Search Service, perform the following on the renamed host:

   a. Edit the **jiss.conf** file to update the appropriate parameter (*hostname*, **mail.imq**, **imq.host**, or **ldap.host**) with the new host name.

   b. To uninstall settings, run the following command:

   ```
   IndexSearch_home/bin/setup -u
   ```

   c. To update settings, run the following command:

   ```
   IndexSearch_home/bin/setup -f IndexSearch_home/etc/jiss.conf
   ```

Troubleshooting FAQ

Topics in this section:

Why is my installation not picking up changes that I've made to the jiss.conf file?

1. Run the following command to see if changes have not been incorporated:

   ```
   IndexSearch_home/bin/issadmin.sh --checkconfig
   ```

2. Then try the following command:

   ```
   IndexSearch_home/bin/issadmin.sh --refresh
   ```

3. Run the following command again to see if the problem is corrected:

   ```
   IndexSearch_home/bin/issadmin.sh --checkconfig
   ```

   This command should update most but not all configuration changes. If changes are still not updated, then ensure that you stop and restart Indexing and Search Service after you make changes to the **jiss.conf** file. See "Stopping and Starting Indexing and Search Service".

How can I tell if a mailbox and index are synchronized?

1. Use the **issadmin.sh** command to check for synchronization:

   ```
   IndexSearch_home/bin/issadmin.sh --user user --checkaccount
   ```

2. If the mailbox and index are not synchronized, run the **issadmin.sh** command with the **--sync** parameter to fix:

   ```
   IndexSearch_home/bin/issadmin.sh --user user --checkaccount --sync
   ```

Why does the jmqconsumer log show large time differences or negative time for "time between generate and submit to index svc"?

If the Messaging Server host's time and the Indexing and Search Service host's time are not synchronized, the JMQConsumer log shows large timing discrepancies. The time when the **jmqconsumer** service logged a message and the time when the message was actually sent could be significantly off or even negative, for example:

```
Mar 31, 2009 11:59:32 AM
com.sun.comms.iss.jmqconsumer.JMQConsumer$EventQueueRunnable run
INFO: Account amam@mailhost.example.com is active on Tue Mar 31 11:59:32 GMT 2009,
processing event generated on Tue Mar 31 12:03:39 GMT 2009 with sequence number
8, time between generate and submit to index svc is -246813ms.
Mar 31, 2009 11:59:53 AM com.sun.comms.iss.jmqconsumer.MessageHeaders
displayJMSType
```

To fix this discrepancy, make sure both the Messaging Server system and the Indexing and Search Service system run an NTP daemon so that their times are synchronized.

If I run reconstruct on the mail store, are event notifications generated so that Indexing and Search Service remains synchronized?

Event notifications are not generated for actions generated from the Messaging Server **reconstruct** command. You must manually synchronize the accounts by using one of the following commands:

```
issadmin.sh --user user --checkaccount --sync
```

or

```
issadmin.sh --accountlist userlist --checkaccount --sync
```

How can I check the mail store IMQ broker?

Run the **imqcmd** command against the mail store IMQ broker, for example:

```
imqcmd list dst
Username: admin
Password:
Listing all the destinations on the broker specified by:

-------------------------
Host Primary Port
-------------------------
localhost 7676

-------------------------------------------------------------------------------
Name Type State Producers Consumers Msgs
Total Count UnAck Avg Size
-------------------------------------------------------------------------------
INDEXMS Queue RUNNING 32 1 0 0 0.0

Successfully listed destinations.
```

Verify that **INDEXMS** exists and that producers (from Messaging Server) and one consumer (from Indexing and Search Service) are present.

How can I check for problems with indexing emails or attachments, or generating thumbnail images?

Use the **issadmin.sh** command to view the status of an account:

```
issadmin.sh --user user --checkaccount --detail status
```

Problems might occur due to unsupported attachment types, errors in the attachment file (such as unreadable HTML or a faulty JPEG file that cannot be opened), or errors in the structure of the email. Look at the original email message to determine the specific problem.

Some errors could be due to text or plain attachments that are too large. You can increase the default setting of 2 MBytes by changing the **iss.indexsvc.attachment.sizelimit** parameter in the **jiss.conf** file. For example, the following entry increases the attachment size limit to 25 MBytes:

```
iss.indexsvc.attachment.sizelimit=25000000
```

If you increase the value of the **iss.indexsvc.attachment.sizelimit** parameter, then also increase the max heap size for **indexSvc** by using the **java.args** setting in the **jiss.conf** file. For example:

```
java.args=-Xmx4g
```

Is there a way to rotate the Indexing and Search Service logs?

At present, you cannot rotate logs on demand. Instead, use the following procedure:

1. Disable the **indexSvc** service:

   ```
   cd /var/opt/sun/comms/jiss/logs
   svcadm disable svc:/application/jiss-indexSvc:default
   ```

2. Move the log file to a new name:

   ```
   mv iss-indexsvc.log.0 iss-indexsvc.log.1
   ```

3. Enable the **indexSvc** service:

```
svcadm enable svc:/application/jiss-indexSvc:default
```

4. Run the **tail** command to ensure that Indexing and Search Service is running and writing to the new log, for example:

```
tail -f iss-indexsvc.log.0
```

What does a 403 error mean in the GlassFish Server server.log file?

The following is an example excerpt from the GlassFish Server **server.log** file.

```
[#|2009-03-25T14:42:56.299-0700|INFO|sun-appserver9.1|org.restlet.Component
(14487431)|_ThreadID=14;_ThreadName=httpSSLWorkerThread-8080-0;|2009-03-25
14:42:56 192.0.2.0 -
192.0.2.1 80 GET /rest/search
q=%2busername:testuser%20%2bhostname:bc011.example.com%20%2bfolder:Junk
%20%2bbody:"apple"&c=2147483647&contentformat=simpleuid&format=atom
403 337 - 3 http://host.example.com - -|#]
```

The last line in this example output shows that a 403 HTTP error (forbidden) was returned. This error occurs if the originating IP address (in this example, 192.0.2.0) is not stored in the **mail.server.ip** setting in the **jiss.conf** file. Indexing and Search Service currently grants access to the RESTful web service by IP address as a substitute for root user access.

If this error is occurring, ensure that you have correctly entered the IP address for the **mail.server.ip** setting. You can also enter a comma-delimited list of IP addresses when using multiple servers. If you change one or more IP addresses, make sure to stop and start the Indexing and Search Service services. For more information, see "Stopping and Starting Indexing and Search Service".

8

Improving Indexing and Search Service Performance

This chapter describes how to tune your Oracle Communications Indexing and Search Service deployment.

Improving Performance While Bootstrapping Users

During the bootstrap process, frequent access of the **dIndex** might cause some performance issues, slowing account creation. Use the techniques described in this section to improve system performance during bootstrapping.

Improving Overall Bootstrap Performance

To reduce overhead on the **dIndex**, increase the value of the **iss.store.account.optimizeinterval** parameter (the default is 50000).

Higher values cause less frequent optimization of the **dIndex**, reducing overhead while bootstrapping.

You can also increase the value of the **iss.store.account.optimizelevel** parameter to **5** (the default is 1).

Confirming Bootstrapping Optimization

To see how frequently the **dIndex** is being optimized, and how long it is taking, view the **iss-indexsvc.log**. You should see a log entry similar to the following one:

```
INFO: SharedWriter.close: optimizing index:/dIndex Time: 123ms
```

Improving Bootstrap Performance for Large Accounts

You can improve overall bootstrap performance, especially for large accounts, by enabling more segments to be generated before a **dIndex** merge operation occurs. This capability enables the bootstrap to use different merge factors for different accounts, and to dynamically adjust the merge factor based on account data.

To increase and dynamically adjust the segment merge factor:

1. Set the **iss.indexsvc.segmentmerge.factor** parameter to a value larger than **10** (the default).
2. Set the **iss.indexsvc.segmentmerge.factor.autoadjust.enabled** parameter to **true**.

   The segment merge factor used during bootstrap is dynamically adjusted, based on the size of the content being bootstrapped.

Note: When the **iss.indexsvc.segmentmerge.factor.autoadjust.enabled** parameter is set to **false**, the bootstrap process uses the value of the **iss.indexsvc.segmentmerge.factor** parameter.

Optimizing Event Backlog

Indexing and Search Service queues up message events in an account's backlog for later processing. However, certain event sequences in an account's backlog do not need to be processed. In fact, these events can be skipped without affecting the state of the user's index.

For example, email that has been "filtered" from the inbox to another folder for immediate deletion (think of spam filtering), and email drafts that are created and quickly destroyed, need not be indexed. These messages are typically created (with a **NewMsg** event) and deleted (with an **ExpungeMsg**) event) so quickly, it makes no sense for Indexing and Search Service to index a deleted message, which cannot be searched.

You can configure Indexing and Search Service so that the **jmqconsumer** process scans the event backlog and marks the useless events to be skipped before submitting them to the **indexSvc** process.

To disable event backlog for optimizing indexing, set the following configuration parameter to **true**:

```
iss.jmqconsumer.useless.event.skip.enabled
```

By default, the parameter is set to **true**.

You can review the log files for events that could be skipped out of event optimization. Some events, such as **ChangeFlag**, **ExpungeMsg**, and **Copy**, might contain multiple UIDs. An event is marked as skipped only if all the UIDs can be skipped. Events are counted under the **TotalEventsSkipped** metric. When only a few UIDs for an event can be marked as skipped, they are considered in the **TotalUIDsSkipped** metric.

When a **Copy** event is marked to be skipped, it might result in the creation of a new **NewMsg** event. This **NewMsg** event is again checked to see if it can be marked as skipped. If it is marked to be skipped, it is counted in the **DroppedNewMsg** metric. If it is not marked to be skipped, it is inserted in the backlog and is counted in the **InsertedNewMsg** metric.

Event optimization statistics are logged in the main JMQConsumer log and in the JMQConsumer statistics log. Account optimization is logged to the JMQConsumer main log at the **FINE** logging level.

A more comprehensive statistics of the event optimization is logged in JMQConsumer statistics log. Table 8–1 lists the metrics captured in log files.

Table 8–1 Metrics Captured in Log Files

Metric	Description
TotalEventsSkipped	Total number of skipped events.
NewMsg	Total number of skipped **NewMsg** events.
ChangeFlag	Total number of skipped **ChangeFlag** events.
Copy	Total number of skipped **Copy** events.

Table 8–1 (Cont.) Metrics Captured in Log Files

Metric	Description
Expunge	Total number of skipped **Expunge** events.
TotalUIDsSkipped	Total number of UIDs skipped.
ChangeFlagUIDs	Total number of skipped **ChangeFlag** event UIDs.
CopyUIDs	Total number of skipped **Copy** event UIDs.
ExpungeUIDs	Total number of skipped **Expunge** event UIDs.
Backlog	Total number of events in the scan during event optimization backlog. **Backlog** is counted only when the event optimization results in at least one event or one UID marked to be skipped.
TotalBacklog	Total number of events scanned during event optimization.
InsertedNewMsg	Number of **NewMsg** events created and inserted in the event backlog.
DroppedNewMsg	Number of **NewMsg** events created but later marked as skipped.
NumOfAccounts	Total number of accounts to which the scanned backlog belongs. An account is considered only when the corresponding event optimization results in at least one event or one UID is marked to be skipped.
TotalNumOfAccounts	Total number of accounts scanned during event optimization.
TotalRescan	Total number of times an account rescan is performed during event optimization.

Increasing RESTful Web Services Search Rate

Depending on the incoming search rate, you can modify the value of the **iss.rest.proxypool.size** parameter to accommodate a higher search rate.

The default is currently set to 512. Because this value is larger than 100, ensure that the **imq.autocreate.destination.maxNumProducers** parameter is set to **-1** in the IMQ broker **config.properties** file.

Otherwise, RESTful Web Services fail with the following error message:

```
Could not create connection and producer com.sun.messaging.jms.JMSException: [ADD_
PRODUCER_REPLY(19)] [C4036]: A broker error occurred. :[409] [B4183]: Producer can
not be added to destination SearchTopic [Topic], limit of 100 producers would be
exceeded
```

Also, when this value goes above 500, you must increase the value of the **imq.jms.max_threads** parameter to **2000**, to accommodate more than 2 * 500 threads in IMQ broker.

Setting Up Large Deployments

For large deployments, avoid bootstrapping users by using the default allocation policy or by randomly assigning accounts to groups. Instead, use the **--accountlist** *file* option, which enables you to provide a file containing the desired mapping of accounts to groups. Using this approach minimizes the contention between accounts in the dIndex. By using this method, you might reduce the bootstrap time of 20,000 accounts by over 15 percent.

Example:

```
issadmin.sh --bootstrap --accountlist /tmp/users.conf --host mailhost.example.com
--threads 10
```

Tuning RESTful Web Services

This section describes how to tune RESTful web services for Indexing and Search Service.

GlassFish Server domain.xml File

GlassFish Server stores configuration information in the **domain.xml** file, which is located in the **$INSTALL/domains/domain1/config** directory. Refer to the GlassFish Server documentation for information about updating **domain.xml**. You must restart GlassFish Server for configuration changes to take effect.

The GlassFish Server modifications in this section are based on the recommendations in the GlassFish Server documentation.

Tuning http-listener

For the **http-listener** entry, add the listener address to specify which network interface Indexing and Search Service uses to accept RESTful web requests. Also, set the acceptor threads equal to the number of virtual processors on the system.

For GlassFish Server 2, use an entry similar to the following:

```
<http-listener acceptor-threads="64" address="10.1.1.1" blocking-enabled="false"
default-virtual-server="server" enabled="true" family="inet" id="http-listener-1"
port="80" security-enabled="false" server-name="www.example.com"
xpowered-by="true">
</http-listener>
```

For GlassFish Server 3, use entries similar to the following:

```
<network-listener port="8070" protocol="http-listener-1" address="10.1.1.1"
transport="tcp" name="http-listener-1"
thread-pool="http-thread-pool"></network-listener>
<transport max-connections-count="64" acceptor-threads="64"
name="tcp"></transport>
```

Tuning request-processing

For GlassFish Server 2, the **request-processing** settings determine how HTTP requests are processed. GlassFish Server 3 uses the **thread-pool** and **http** settings for this purpose.

Increase the number of threads to enable the system to process more concurrent requests. The number of threads can be greater than the number of CPUs on the system, because all search requests are passed to the back-end search service, so threads are blocked waiting for responses. The request timeout value should be larger than the search timeout value in the **jiss.conf** file, which has a default value of 10 seconds.

For GlassFish Server 2, use an entry similar to the following:

```
<request-processing header-buffer-length-in-bytes="8192" initial-thread-count="32"
request-timeout-in-seconds="15" thread-count="128" thread-increment="8"/>
```

For GlassFish Server 3, use an entry similar to the following:

```
<thread-pool max-thread-pool-size="128" name="http-thread-pool"
```

```
min-thread-pool-size="32"></thread-pool>
<http request-timeout-seconds="15" timeout-seconds="30"
default-virtual-server="server" max-connections="4096">
```

Tuning keep-alive Connections

Adjusting the **keep-alive** connection settings minimizes the need to open and close sockets to service HTTP requests. GlassFish Server 2 uses separate **keep-alive** connections settings. For GlassFish 3, these settings are part of the **http** settings.

For GlassFish Server 2, use an entry similar to the following:

```
<keep-alive max-connections="4096" thread-count="64" timeout-in-seconds="30"/>
```

The maximum connection setting specifies the maximum number of requests that can use a connection. If no requests are initiated by the timeout duration, then the connection is closed.

For GlassFish Server 3, use an entry similar to the following:

```
<http request-timeout-seconds="30" timeout-seconds="30"
default-virtual-server="server" max-connections="4096">
```

Tuning JVM Options

The following JVM options should provide good performance for RESTful web services:

```
<jvm-options>-server</jvm-options>
<jvm-options>-d64</jvm-options>
<jvm-options>-XX:LargePageSizeInBytes=256m</jvm-options>
<jvm-options>-Xms2048m</jvm-options>
<jvm-options>-Xmx2048m</jvm-options>
<jvm-options>-Xmn1g</jvm-options>
<jvm-options>-XX:+UseParallelGC</jvm-option>
<jvm-options>-XX:+UseParallelOldGC</jvm-options>
<jvm-options>-XX:ParallelGCThreads=16</jvm-options>
<jvm-options>-XX:PermSize=192m</jvm-options>
<jvm-options>-XX:MaxPermSize=192m</jvm-options>
```

To monitor garbage collection activity, add the following options:

```
<jvm_options>-Xloggc:/opt/SUNWappserver/domains/domain1/logs/gclog</jvm-options>
<jvm-options>-XX:+PrintGCDetails</jvm-options>
<jvm-options>-XX:+PrintGCDateStamps</jvm-options>
```

Tuning Access Logging

You can disable access logging, which can be disk-intensive, by setting the following entry:

```
<property name="accessLoggingEnabled" value="false"/>
```

Message Queue Broker config.properties File Tunings

When using a standalone Message Queue, you can add nondefault configuration settings to the bottom of the **/var/imq/instances/imqbroker/props/config.properties** file. When using an embedded Message Queue, you add nondefault configuration settings to the *GlassFish_home*/**glassfish/domains/domain1/imq/instances/imqbroker/props/config.properties** file.

Tuning Java Message Service Threads

To enable more requests to be processed by the Indexing and Search Service **jmqbroker** in any given time interval, you can increase the value of the **jms.max_threads** parameter (say to **2000**). The default number of threads is 1000, but it is too small. The Indexing and Search Service **jmqbroker** handles both indexing and search requests, unlike the **jmqbroker** used by the Messaging Server's JMQ notification plugin, which handles only indexing requests. If you are using one **jmqbroker** for both Messaging Server event notifications and for Indexing and Search Service internal notifications, then increase the number of threads to at least **2000**.

Tuning Maximum Number of Producers

The default number of message producers that can be created by the **jmqbroker** is 100. This value is too low to handle the number of messages that are typically generated by an active Messaging Server deployment that supports more than 10,000 active users. You can disable the setting to enable an unlimited number of producers to be created to handle the load. Set the **imq.autocreate.destination.maxNumProducers** parameter to **-1**.

imqbroker.conf File Tunings

When using a standalone Message Queue, you specify broker properties by editing the **/etc/imq/imqbrokerd.conf** file.

When using an embedded Message Queue, you specify broker properties in the JMS service configuration or in the JMS host. See the topic on updating the JMS service configuration in *Oracle GlassFish Server 3.1 Administration Guide*.

Tuning jmqbroker JVM Options

Here are the JVM options for **jmqbroker**:

```
ARGS=-javahome /usr/jdk/latest -vmargs -d64 -vmargs -Xms6g -vmargs -Xmx6g
-vmargs -Xmn3g -vmargs -XX:PermSize=50m
-vmargs -XX:MaxPermSize=50m -vmargs -XX:+UseConcMarkSweepGC -vmargs
-XX:+UseParNewGC -vmargs -XX:+CMSParallelRemarkEnabled
-vmargs -XX:+CMSScavengeBeforeRemark -vmargs -XX:ParallelGCThreads=8 -vmargs
-XX:MaxTenuringThreshold=15 -vmargs
-XX:+DisableExplicitGC -vmargs
-Xloggc:/var/imq/instances/imqbroker/log/jmq-log.gclog -vmargs -XX:+PrintGCDetails
-vmargs -XX:+PrintGCDateStamps -vmargs -XX:+PrintTenuringDistribution
```

The JVM option allows use of the 64-bit JVM and reserves 4 GBytes of memory for the **jmqbroker**. Scalability tests show that the Messaging Server and the Indexing and Search Service **jmqbroker** use approximately 1.5 GBytes of the JVM heap.

Tuning jmqconsumer JVM Options

Here are the JVM options used for **jmqconsumer**:

```
java.args.jmqconsumer = -Xms12g -Xmx12g -Xmn6g -XX:PermSize=100m
-XX:MaxPermSize=100m -XX:+CMSClassUnloadingEnabled
-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled
-XX:+UseCMSInitiatingOccupancyOnly
-XX:CMSInitiatingOccupancyFraction=50 -XX:+CMSScavengeBeforeRemark
-XX:ParallelGCThreads=8 -XX:SurvivorRatio=4
-XX:MaxTenuringThreshold=15
-Xloggc:/var/opt/sun/comms/iss/logs/iss-jmqconsumer-$$.gclog -XX:+PrintGCDetails
-XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution
```

Tuning Search Service JVM Options

Here are the JVM options used for **searchSvc**:

```
java.args.searchsvc = -Xms12g -Xmx12g -Xmn6g -XX:PermSize=100m -XX:MaxPer
mSize=100m -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSClassUnloadingEnable
d -XX:+CMSParallelRemarkEnabled -XX:+UseCMSInitiatingOccupancyOnly
-XX:CMSInitiatingOccupancyFraction=50 -XX:+CMSScavengeBeforeRemark
-XX:ParallelGCThreads=8 -X
X:SurvivorRatio=4 -XX:MaxTenuringThreshold=15
-Xloggc:/var/opt/sun/comms/iss/logs/iss-searchsvc-$$.gclog -XX:+PrintGCDetails
-XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution
```

Tuning Index Service JVM Options

Set the size of heap size of Index Service as large as is practical. Garbage collection activity should be monitored to determine whether the size is sufficient. To adjust the heap size, set **java.args.indexsvc** as follows. Replace **YY** with a value between one-fourth and one-half the RAM on the system. Replace **ZZ** with one-half the size of **YY**.

```
java.args.indexsvc = -XmsYYg -XmxYYg -XmnZZg -XX:PermSize=100m
-XX:MaxPermSize=100m -XX:+CMSClassUnloadingEnabled
-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled
-XX:+UseCMSInitiatingOccupancyOnly
-XX:CMSInitiatingOccupancyFraction=50 -XX:+CMSScavengeBeforeRemark
-XX:ParallelGCThreads=8 -XX:SurvivorRatio=6
-XX:MaxTenuringThreshold=15 -Xloggc:/var/opt/sun/comms/iss/logs/iss-indexsvc-gclog
-XX:+PrintGCDetails
-XX:+PrintGCTimeStamps -XX:+PrintTenuringDistribution
```

For example, on a system with 128 Gbytes of memory, you can allocate memory for various heaps as follows:

- **searchSvc**, 12 Gbytes
- GlassFish Server, 2 Gbytes
- **jmqbroker**, 6 Gbytes
- **jmqconsumer**, 12 Gbytes
- **indexsvc**, 32 Gbytes

The following **java.args.indexsvc** settings would be appropriate:

```
-Xms32g
-Xmx32g
-Xmn16g
```

Tips:

- When debugging garbage collection issues, it might also be useful to add **-XX:+PrintHeapAtGC**.
- When debugging services, increase the log level on the services from **WARNING** to **INFO**.
- To ensure enough log data is retained, increase the size of the **iss.***service***.logfile.limit** parameter, or increase the **iss.***service***.logfile.count** parameter, or do both.

Part II

Administering a High-Availability System

Part II contains the following chapters:

9

Managing Indexing and Search Service High Availability

This chapter describes how to administer and troubleshoot a highly available Oracle Communications Indexing and Search Service system. For information on configuring Indexing and Search Service for high availability, including clusterv2, see *Indexing and Search Service Installation and Configuration Guide*.

Administering Indexing and Search Service High Availability

Administering Indexing and Search Service high availability involves:

- Adding Cluster Search Services
- Bootstrapping Users on Indexing Hosts
- Verifying Users on Web Hosts
- Adding an Additional Web Host
- Removing a Web Host

Adding Cluster Search Services

To add Cluster Search Services, run the **csearch.mgr.sh -A** command on the Indexing and Search Service web hosts. For more information, see "csearchmgr.sh".

Bootstrapping Users on Indexing Hosts

To bootstrap users, use the **issadmin.sh --bootstrap** command. For more information, see "issadmin.sh."

Verifying Users on Web Hosts

To verify that the Indexing and Search Service is correctly searching users:

1. On each Indexing and Search Service web host, log in as the bootstrapped user.

   In this example, log in to **bco01** and **bco22**.

2. Perform a search by using the RESTful interface.

   The search defaults to the default mail host. Thus, you might need to change the hostname parameter in the URL if the user resides on a different mail host. For example, on **bco01.example.com** the search URL might resemble the following for user **c1**:

   ```
   http://bco01.example.com:8070/rest/search?q=%2busername:c1%20%2bhostname:bco65
   ```

```
example.com%20%2battachment-type:at*&contentFormat=attachmentOnly&thumbnail=s&c
=100
```

You would then change the username and hostname fields in the URL to the following for user **u1**:

```
http://bco01.example.com:8070/rest/search?q=%2busername:u1%20%2bhostname:bco108
.example.com%20%2battachment-type:at*&contentFormat=attachmentOnly&thumbnail=s&
c=100
```

Adding an Additional Web Host

To add an Indexing and Search Service web host to your high availability deployment:

1. Repeat the installation for the new node.

   See *Indexing and Search Service Installation and Configuration Guide* for instructions.
2. Copy the index node configuration files to the new node.
3. Run the **csearchmgr.sh -A** command.
4. Add the new node to the load balancer.

Removing a Web Host

To remove an Indexing and Search Service web host from your high availability deployment:

1. Remove the node from the load balancer.
2. Run the **csearchmgr.sh -D** command.

Troubleshooting Indexing and Search Service High Availability

This section contains the following topics to troubleshoot your Indexing and Search Service high availability deployment:

- General Differences Between Indexing Hosts and Web Hosts
- Troubleshooting Web Hosts
- Troubleshooting Indexing Hosts

General Differences Between Indexing Hosts and Web Hosts

Table 9–1 shows the two main differences, other than host name, in the **jiss.conf** file between the indexing hosts and web hosts. The **setup** script should automatically configure these parameters based on the **iss.cluster.enable** parameter and type of node being configured.

Table 9–1 jiss.conf File Differences Between Indexing and Web Hosts

Parameter Name	Indexing Host	Web Host
java.naming.factory.initial	**com.sun.jndi.ldap.LdapCtxFactory**	**com.sun.jndi.fscontext.RefFSContextFactory**
iss.accountstate.dst.name	**AccountState.***instance.name*	**AccountState**

The naming factory change tells the web host to use file-based JNDI lookups that point to the local host's JMQ broker. The account state change is used to create a single point

where all account state updates are funneled through a single topic to the Oracle GlassFish Server WAR files (**rest**,**storeui**).

Troubleshooting Web Hosts

To troubleshoot the Indexing and Search Service web hosts:

1. Verify that the Cluster Search Services are running.

```
svcs -a | grep jiss
online         Jan_19   svc:/application/jiss-csearchSvc-bco04:default
online         Jan_19   svc:/application/jiss-csearchSvc-bco29:default
```

2. Check the Cluster Search Services log files for errors.

```
iss.log.dir/iss-csearchsvc-bco04.log.0
iss.log.dir/iss-csearchsvc-bco29.log.0
```

3. Check JMQ connections.

```
# imqcmd list dst
-------------------------------------------------------------------------------------------------
Name          Type     State       Producers      Consumers                Msgs
                                  Total Wildcard  Total Wildcard    Count Remote UnAck Avg Size
-------------------------------------------------------------------------------------------------
AccountState Topic    RUNNING   0     0          2     0           0     0       0    0.0
Indexbco04   Queue    RUNNING   0     -          1     -           0     0       0    0.0
Indexbco29   Queue    RUNNING   0     -          1     -           0     0       0    0.0
SearchTopic  Topic    RUNNING   512   0          2     0           0     0       0    0.0
mq.sys.dmq   Queue    RUNNING   0     -          0     -           0     0       0    0.0
```

For each running Cluster Search Service, you should see the following information.

- 1 consumer to **SearchTopic** topic
- 1 consumer to index host instance queue (**Indexbco04**, **Indexbco29**)
- 1 consumer per configuration in **iss.cluster.d** to **AccountState (**on indexing host; see **imqcmd list dst** output in "Troubleshooting Indexing Hosts")
- The Indexing and Search Service RESTful search servlet produces the following connections to JMQ running on the web host:
  - 1 consumer per configuration in **iss.cluster.d** to **AccountState**
  - 512 or **iss.rest.proxypool.size** producers to **SearchTopic** if at least one RESTful search has been issued (otherwise this value is 0)

Troubleshooting Indexing Hosts

To troubleshoot the Indexing and Search Service indexing hosts:

1. Check the log files.

```
iss.log.dir/iss-indexsvc.log.0
```

2. Run the **imqcmd** command to check JMQ connections on indexing node for the following information:

   1 consumer for **AccountState.***instance.name* topic for each cluster search service started plus one for **jmqconsumer**.

   For example:

```
# imqcmd list dst
-------------------------------------------------------------------------------------------------
Name          Type     State       Producers      Consumers                Msgs
                                 Total Wildcard  Total Wildcard    Count Remote UnAck Avg Size
-------------------------------------------------------------------------------------------------
AccountState.bco04 Topic RUNNING 0    0         3     0           0     0       0    0.0
```

Troubleshooting clusterv2

This section contains common problems and their solutions for clusterv2 high availability.

GlassFish Server (Web Node) Error

The following error message indicates that a problem occurred when setting up the local JNDI information:

```
GlassFish Server (Web Node) Log Contains "cannot findcn=CommsQueueFactory,
cn=CommsTopicFactory, or cn=SearchTopic"
```

Check the following settings:

- The **/etc/jiss/cluster.d/clusterv2.conf** file is present.
- Run *IndexSearch_home***/bin/csearchmgr.sh -A**.
- Run *IndexSearch_home***/bin/factorymgr.sh -l**, the output should be the following:

```
Listing all administered objects in the object store specified by:

 java.naming.factory.initial com.sun.jndi.fscontext.RefFSContextFactory
 java.naming.provider.url file:/etc/jiss/jms

 Lookup Name Object Class Name
 cn=CommsQueueFactory com.sun.messaging.QueueConnectionFactory
 cn=CommsTopicFactory com.sun.messaging.TopicConnectionFactory
 cn=IndexXXXX com.sun.messaging.Queue
 cn=SearchTopic com.sun.messaging.Topic

 Objects listed successfully.
```

Where *IndexXXXX* is the instance name from the **clusterv2.conf** file.

Searching Rest Interface Always Returns 404 Error

If you receive this error, check the following items:

- Ensure that the cluster search service is running on the cluster search node.

  ```
  Java_home/bin/jps -l | grep com.sun.comms.iss.csearch.CSearchService
  ```

- View the GlassFish Server log on the web node while updating the state on the indexing node by running the **issadmin.sh --setstate A --user** command.
- View the cluster search log on the cluster search node while updating the state on the indexing node by running the **issadmin.sh --setstate A --user** command.

10

Managing the Watcher Service

This chapter describes how to administer the Oracle Communications Indexing and Search Service watcher service.

Overview of the Indexing and Search Service Watcher Service

The watcher service provides local host monitoring of Indexing and Search Service services. When the watcher service detects a service outage, it generates log file warnings and email alerts to help you take recovery action. The watcher service does not automatically restart Indexing and Search Service services. You must manually restart Indexing and Search Service services when the watcher detects an outage. The watcher service's monitoring consists of connecting to services, authenticating if necessary, and checking for a configuration parameter or a response from the service beyond a simple TCP check.

How the Watcher Service Performs Monitoring

The watcher service monitoring consists of checking the availability of **jiss.conf** parameters at the TCP level. The watcher service connects to a configured host name and port to check if a TCP connect string is issued.

Table 10–1 describes the Indexing and Search Service configuration parameters that the watcher service monitors based on the Indexing and Search Service installation type.

Table 10–1 jiss.conf Parameters Monitored by the Watcher Service

iss.cluster.install Value	iss.cluster.type Value	jiss.conf Parameters Monitored
standalone	(**all**)	**mail.imq** **imq.host** **ldap.host** **appserv.web.port/ localhost**
multi-machine	**index**	**mail.imq**
multi-machine	**jmq**	**imq.host**
multi-machine	**ldap**	**ldap.host**
multi-machine	**web**	**appserv.web.port / localhost**
cluster	**web**	**imq.host (localhost)** **appserv.web.port/ localhost**

Table 10–1 (Cont.) jiss.conf Parameters Monitored by the Watcher Service

iss.cluster.install Value	iss.cluster.type Value	jiss.conf Parameters Monitored
cluster	**index**	**imq.host** (**localhost**) **ldap.host** **mail.imq**
clusterv2	**web**	**appserv.web.port** / **localhost**
clusterv2	**index**	**ldap.host** **imq.host** **mail.imq**

Table 10–2 describes which services the watcher service checks and how it performs those checks.

Table 10–2 Indexing and Search Service Services Monitored by the Watcher Service

iss.cluster.install	iss.cluster.type	Description of What Is Checked
standalone	(**all**)	JNDI lookup for **queue.connection.factory.name** JNDI lookup for **topic.connection.factory.name** JNDI lookup for **iss.searchsvc.dst.name** JNDI lookup for **iss.indexsvc.dst.name** Checks that **indexSvc** (**isAlive** method) is running Checks that **searchSvc** (**isSearch** method) is running Checks that **jmqconsumer** is running
multi-machine	**jmq**	JNDI lookup for **queue.connection.factory.name** JNDI lookup for **topic.connection.factory.name** JNDI lookup for **iss.searchsvc.dst.name** JNDI lookup for **iss.indexsvc.dst.name**
multi-machine	**index**	Checks that **indexSvc** (**isAlive** method) is running Checks that **searchSvc** (**isSearch** method) is running Checks that **jmqconsumer** is running
cluster	**web**	JNDI lookup for **queue.connection.factory.name** JNDI lookup for **topic.connection.factory.name** JNDI lookup for **iss.searchsvc.dst.name** Checks that **searchSvc** (**isSearch** method) is running per instance of cluster search service
cluster	**index**	JNDI lookup for **queue.connection.factory.name** JNDI lookup for **topic.connection.factory.name** JNDI lookup for **iss.indexsvc.dst.name** Checks that **indexSvc** (**isAlive** method) is running Checks that **jmqconsumer** is running

Table 10–2 (Cont.) Indexing and Search Service Services Monitored by the Watcher

iss.cluster.install	iss.cluster.type	Description of What Is Checked
clusterv2	**index**	JNDI lookup for **queue.connection.factory.name** JNDI lookup for **topic.connection.factory.name** JNDI lookup for **iss.searchsvc.dst.name** JNDI lookup for **iss.indexsvc.dst.name** Checks that **indexSvc** (**isAlive** method) is running Checks that **searchSvc** (**isSearch** method) is running per instance of cluster search service
clusterv2	**csearch**	JNDI lookup for **queue.connection.factory.name** JNDI lookup for **topic.connection.factory.name** JNDI lookup for **iss.searchsvc.dst.name** Checks that **searchSvc** (**isSearch** method) is running per instance of cluster search service

Configuring the Watcher Service

Configuring the watcher service involves:

- Enabling the Indexing and Search Service Watcher Service
- Configuring Email Notifications

Enabling the Indexing and Search Service Watcher Service

To enable the watcher service on an Indexing and Search Service node, configure the appropriate **iss.cluster** and **iss.watcher** parameters in the **jiss.conf** file as described in this procedure.

1. Edit the *IndexSearch_home*/**etc/jiss.conf** file.
2. Set **iss.watcher.enabled = true** to run the watcher service on this host.

   Alternately, you can run the **watchermrg.sh** command. For more information, see "watchermgr.sh".
3. (Optional) Set other watcher run-time parameters, such as notification and alert settings.

   For more information about optional **iss.watcher** parameters, see "Indexing and Search Service Configuration Parameters".
4. Save your changes to the **jiss.conf** file.
5. Run the **watchermgr.sh** command to enable the service.

   ```
   watchermgr.sh -a
   ```
6. Verify that the watcher has been configured correctly. For example, you can simulate failures on the following components and check the log file for entries or confirm that the appropriate email is sent.
   - Simulate a GlassFish Server failure:

     ```
     asadmin stop-appserv
     ```
   - Simulate a broker failure:

     ```
     /etc/init/imq stop
     ```

- Simulate an LDAP failure:

  ```
  /opt/SUNWdsee/dsee7/bin/dsadm stop path_to_instance
  ```

- Simulate an indexing service failure:

  Linux:

  ```
  /etc/init.d/indexSvc stop
  ```

  Solaris:

  ```
  svcadm disable jiss-indexSvc
  ```

The watcher service generates an alert each time that a single test fails. If more than one test fails in a configured interval, the alert level is increased by one for each test that fails. An alert level of zero (0) means that all basic and functionality tests have passed. Table 10–3 describes the watcher service maximum alert levels, depending on the **iss.cluster.install** and **iss.cluster.type** configurations.

Table 10–3 Watcher Service Maximum Alert Level

iss.cluster.install	iss.cluster.type	Maximum alert level
standalone	(**all**)	10
multi-machine	**index**	3
multi-machine	**jmq**	5
multi-machine	**ldap**	1
multi-machine	**web**	1
cluster	**web**	5 + number of indexing nodes
cluster	**index**	7
clusterv2	**web**	1
clusterv2	**index**	8 + number of indexing nodes - 1

Multiple host installations can combine various services into a single instance and thus **iss.cluster.type** has multiple values defined. Thus, you can increase the maximum alert level for that instance.

Configuring Email Notifications

To configure email notifications from the watcher service:

1. Edit the *IndexSearch_home*/**etc/jiss.conf** file.
2. Set or change the following configuration parameters:
   - **iss.watcher.notification.type**: **email**
   - **iss.watcher.notification.destination**: Email address to which to send watcher notifications
   - **iss.watcher.notification.source**: Source (from) email address for watcher notifications
   - **iss.watcher.notification.mail.host**: *host*:*port* of the messaging host sending the notifications

- **iss.watcher.notification.protocol**: Protocol to use when sending email notifications, either **ssl**, **tls**, or **plain**
- **iss.watcher.notification.email.interval**: Number of seconds between which the watcher service sends an email alert, once it first detects a failure
- **iss.watcher.email.user**: User name, if required, to send email
- **iss.watcher.email.user.password**: User name password, if required, to send email

For more information, see "Indexing and Search Service Configuration Parameters".

3. Save your changes to the **jiss.conf** file.
4. Refresh the Indexing and Search Service configuration for the change to take place.

```
issadmin.sh --refresh
```

How the Watcher Service Determines to Send Email Notifications

Table 10–4 describes how the watcher service sends email to the value of the **iss.watcher.notification.destination** parameter. If the value of the **iss.watcher.email.interval** parameter is less than or equal to the value of the **iss.watcher.interval** parameter, the watcher service sends an alert email every **iss.watcher.interval** seconds.

Table 10–4 Error Action Table

Current Level	New Level	iss.watcher.email. interval Reached?	Email Sent?
0	Greater than or equal to 1	Not checked	Yes
Greater than or equal to 1	0	Not checked	Yes
Greater than or equal to 1	Greater than or equal to 1	No	No
Greater than or equal to 1	Greater than or equal to 1	Yes	Yes

If an alert level changes but does not reach zero, no new email is sent if the **iss.watcher.email.interval** has not been reached.

Part III

Indexing and Search Service API

Part III contains the following chapters:

11

Overview of the Web Service API

This chapter describes the Oracle Communications Indexing and Search Service web service API and the HTTP GET parameters currently supported.

Overview of Indexing and Search Service Web Service API

The Indexing and Search Service web service API is a RESTful web service. It takes a Uniform Resource Identifier (URI) through the HTTP GET method, and returns search results by following the OpenSearch 1.1 specification in either Rich Site Summary (RSS) 2.0, Atom 1.0, or JavaScript Object Notation (JSON) format.

The search URI consists of **http://***isshost***:***port***/rest/search** and mandatory **q=** and optional parameters. The default port is 8080.

> **Note:** The examples in this chapter assume that you are already logged in to Indexing and Search Service. To log in, in a browser, navigate to **http://***isshost***:***port***/rest**.

The following search returns all emails with **test** in the subject line for **user1** on the email server **demo.example.com**:

```
http://demo.example.com:8080/rest/search?q=%2busername:user1%20%2bhostname:isshost
.example.com%20%2bsubject:test
```

More sample URIs are provided at the welcome page, **http://***isshost***:***port***/rest**.

HTTP GET Parameters

The following sections describe the HTTP GET parameters that Indexing and Search Service currently supports.

Search Query Parameter

q=*search_query_in_lucene_query_parser_syntax*

The search query needs to follow the Lucene query parser syntax using a specific list of Lucene field names that were used for indexing the content. For more information, see "Using Search Query and Sort Criteria."

Optional Data Type Parameter

t=*supported_data_type*

The optional data type parameter enables you to search across all data types or a specific data type within Indexing and Search Service indexed data. Currently, Indexing and Search Service supports only the **email** data type.

Optional Format Parameter

format=*RSS_or_ATOM_or_JSON*

The optional format parameter enables a client to receive search results in either RSS 2.0, Atom 1.0, or JSON format per OpenSearch 1.1 specification. The default is RSS.

Optional Sort Parameter

sort=*sort_criteria*

The optional sort criteria enables you to sort search results based on particular criteria. Currently, Indexing and Search Service enables you to sort on only a subset of indexed field names. See "Using Search Query and Sort Criteria" for a complete list of sort criteria for the email data type.

Optional Start Index Parameter

s=*start_from_this_item_number*

The optional start index enables a client to choose from which number to return the search result. The default start index is 0.

Optional Count per Page Parameter

c=*num_of_results_per_page*

The optional number of search results per page enables a client to request a specific number of returned items. The default number of results per page is 10. This number can be used with or without the optional start index parameter.

Optional Content Format Parameter

contentFormat=*format_selector*

The optional **contentFormat** selector enables a client to request additional formatting for results returned from a search call. The default format is standard format (for email, currently). When the standard content format is specified, the results contain IMAP URIs providing a compact representation of the account, folder, and UID of the message with the match.

> **Note:** The IMAP URI should be consumed by an IMAP client to provide access to the data. The URI is not directly usable by a web browser.

Other formats that can be specified are **attachmentOnly**, **simpleUID**, and **MJS**. When **attachmentOnly** is specified, each result contains URLs to the original attachment and any thumbnails that correspond to it (refer to the **thumbnail** parameter). Also, the **totalResults** and **count-per-page** values returned refer to the number of attachments located, not the number of emails. When **simpleUID** is specified, the result for each entry contains only the UID number of the matched email. This option applies only when you specify **format=ATOM**. It is intended to speed up queries to a single folder, such as those queries through IMAP. When **MJS** is specified (only valid when

format=JSON is specified), the output is formatted in compact WMAP (webmail) format.

Optional Thumbnail Parameter

thumbnail=*size*

The optional **thumbnail** parameter enables a client to request the size of the thumbnail referenced when the **contentFormat=attachmentOnly** option is also specified. The **DEFAULT** size causes no thumbnail URL to be generated. Other sizes that can be specified are **S**, **M**, **L**, and **XL** for small, medium, large, and extra large. If one of these sizes is specified, a URL reference for that size thumbnail appears in the description field of each result.

Optional JavaScript Callback Parameter (JSON Format Only)

callback=function_name

The optional JavaScript callback parameter enables a client to request that the JSON results be wrapped inside a JavaScript callback function specified by the client. For example, setting **callback=function_name** generates the following output:

```
function_name({
               "title": "iss-host:iss-port search: search-query",
               "link":
"http://iss-host:iss-port/rest/search?q=search-query&amp;amp;c=100&amp;amp;format=json",
        "modified": "Mon Feb 08 22:02:48 GMT 2010",
        "opensearch:totalResults": "0",
        "opensearch:startIndex": "0",
        "opensearch:itemsPerPage": "100",
        "items": [
        ]
})
```

Optional Timeout Parameter

timeoutmsec=*number_of_milliseconds*

The optional timeout parameter enables a client to request how many milliseconds to wait for a search query response before returning from a search request. If the search response was not obtained before the timeout, a 500 error (Waiting for response timed out on request) is returned. The default timeout is specified in the **jiss.conf** file with the **iss.searchsvc.response.timeout** parameter.

12

Using Search Query and Sort Criteria

This chapter provides guidance for generating Oracle Communications Indexing and Search Service search queries. It specifies the field names that you can use in query terms, and assists in translating well-known IMAP SEARCH queries into Indexing and Search Service search queries. It also describes how search results can be sorted by certain field names.

About Search Results and Pattern Matching

The search web service enables pagination of results through the start and count parameters, but when the queries come in from Oracle Communications Messaging Server, the count is always set to the max (count=2147483647).

You might not see all the search results that you expect because Indexing and Search Service does not search the same way that IMAP does. A partial match does not result in a match unless you provide a wildcard character to the search web service.

Table 12–1 compares searching between Indexing and Search Service and IMAP.

Table 12–1 Indexing and Search Service and IMAP Searches

Search Option	Search Query Syntax	Matching Strings
Indexing and Search Service with no wildcard matching	+search "apple*"	apple* (the string, not a wild card match)
IMAP search with no wildcard matching	subject:apple*	apple* (the string, not a wild card match)
Indexing and Search Service wildcard matching with * after	+subject:apple*	apple, apples, applet
IMAP **service.imap.indexer.suffix_search** (In the search query syntax example, subject is included in the field list.)	subject apple	apple, apples, applet
Indexing and Search Service wildcard matching with * before	+subject:*mine	determine, mine
IMAP **service.imap.indexer.prefix_search** (In the search query syntax example, subject is included in the field list.)	subject mine	determine, mine
Indexing and Search Service wildcard matching with * both before and after	+subject:*ear*	search, ear, earth
IMAP **service.imap.indexer.substring_search** (In the search query syntax example, subject is included in the field list.)	subject ear	search, ear, earth

IMAP wildcard searches depend on the use of the Messaging Server **service.imap.indexer.suffix_search**, **service.imap.indexer.prefix_search**, and **service.imap.indexer.substring_search** options. You need to enable these options on Messaging Server to be able to perform IMAP wildcard searches. For more information, see the chapter on indexer options in *Messaging Server Reference*.

For example, searching for **apple** does not match **apples** but searching for **apple*** does match both **apple** and **apples**. Messaging Server puts the terms in double quotes, so if you put **"apple*"** in your IMAP client search query, this string is interpreted by Indexing and Search Service as **apple*** and the asterisk (*) is not interpreted as a wildcard.

To experiment with the richer search experience of using the web services directly, in a browser, navigate to the following URLs:

- **http://***iss-host***:8080/rest**
- **http://***iss-host***:8080/searchui/**

Indexing and Search Service provides sample searches at these URLs. These searches access the web service directly and use the thumbnail images.

You can configure the default web services port, **8080**, to a port of your choosing, if necessary.

Search Query Field Names

Table 12–2 describes the search query field names for the email data type.

Table 12–2 Search Query Field Names

Field Name	Comments
answered	No comments.
attachgroup-author	No comments.
attachgroup-contents	No comments
attachgroup-imagecompress	Only defined for picture type like **atjpeg**.
attachgroup-imageheight	Only defined for picture type like **atjpeg**.
attachgroup-imageprecision	Only defined for picture type like **atjpeg**.
attachgroup-imagesource	Only defined for picture type like **atjpeg**.
attachgroup-imagewidth	Only defined for picture type like **atjpeg**.
attachgroup-name	No comments.
attachgroup-pubdate	No comments.
attachgroup-size	No comments.
attachgroup-title	No comments.
attachment-type	Contains an entry for every attachment detected in document. Valid values are lowercase and prefixed by "at", such as "atpdf" defined in the AttachmentType enumeration for attachments. Currently, *type* in the at*type* values can be any of the following: applefile, audio, compres, doc (Microsoft Word), html, image, iwork (Apple iWork), jpeg, odf (any OpenOffice file), other (uncategorized), pdf, pgpsign, plain (plain text), ppt (Microsoft Powerpoint), rtf, ssign, sencr, vcf, video, vsd (Microsoft Visio), xls (Microsoft Excel), xml.

Table 12–2 (Cont.) Search Query Field Names

Field Name	Comments
bcc	No comments.
body	Applies to **contents** or **attachgroup-contents**.
cc	No comments.
content-description	No comments.
content-id	Search not yet fully supported.
content-md5	Search not yet fully supported.
content-transfer-encoding	Search not yet fully supported.
content-type	No comments.
contents	No comments.
deleted	No comments.
draft	No comments.
email-headers	Currently not implemented.
flagged	No comments.
folder	No comments.
from	No comments.
hostname	No comments.
message-id	Search not yet fully supported.
messagekey	Currently not implemented.
received	No comments.
reply-to	No comments.
seen	No comments.
sent	No comments.
size	No comments.
subject	No comments.
text	Applies to **email-headers**, **contents**, or **attachgroup-contents**.
to	No comments.
uid	No comments.
uidvalidity	No comments.
username	No comments.

Sort Criteria

Table 12–3 describes the search field names you use to specify sort criteria.

Table 12–3 Sort Criteria Field Names

Field Name	Comments
cc	No comments.
folder	No comments.

Table 12–3 (Cont.) Sort Criteria Field Names

Field Name	Comments
from	No comments.
received	IMAP SORT arrival (not yet fully implemented).
sent	IMAP SORT date (not yet fully implemented).
size	No comments.
subject	No comments.
to	No comments.
uid	Implicit in IMAP SORT.

Specify reverse sort criteria by inserting a hyphen (-) as a prefix to the individual field name. Specify multiple sort criteria as a simple ordered list of field names, such as:

```
+size -subject +to -from
```

The default sort uses the **uid** field name. You can also explicitly specify this field name in a multiple sort criteria list.

Note: Because sort by folder is a post meta search process, folder sort is always the last sort performed. That is, sort = **+subject +folder** is the same as **+folder +subject**. The results are always ordered first by folder, then by other sorting criteria.

You can use the following search field names for flags as sort criteria: **answered**, **deleted**, **draft**, **flagged**, and **seen**.

Sorting on these terms occurs after any folder sort is performed. The sort for each of these specifiers returns results with the corresponding flag value of **true** before those that are **false**. The reverse sort (using the minus) returns results with **false** before **true**.

How IMAP SEARCH Uses Search Query Field Names

Most IMAP SEARCH criteria use the same (or similar) name as the field name in a search query to Indexing and Search Service. However, some criteria must be mapped in non-obvious ways. Table 12–4 describes how these features in IMAP SEARCH map into corresponding Indexing and Search Service search query terms.

Table 12–4 IMAP SEARCH Mapping to Indexing and Search Service

IMAP SEARCH Flag	Indexing and Search Service Query Term
ANSWERED	**+answered:true**
UNANSWERED	**+answered:false** or **-answered:true**
DELETED	**+deleted:true**
UNDELETED	**+deleted:false** or **-deleted:true**
DRAFT	**+draft:true**
UNDRAFT	**+draft:false** or **-draft:true**
FLAGGED	**+flagged:true**
UNFLAGGED	**+flagged:false** or **-flagged:true**

Table 12–4 (Cont.) IMAP SEARCH Mapping to Indexing and Search Service

IMAP SEARCH Flag	Indexing and Search Service Query Term
SEEN	**+seen:true**
UNSEEN	**+seen:false** or **-seen:true**
LARGER *n*	**-size:{0 TO** *n***}** **-size:[0 TO** *n***]** (corresponds to **LARGER** than or equal to *n*)
SMALLER *n*	**+size:{0 TO** *n***}** (technically excludes zero)
UID *sequence_set* - single or range of UIDs only (arbitrary list not currently supported)	**+uid:44** (single UID) **+uid:[44 TO 444]** (range of UIDs, inclusive of 44 and 444)
ON *date*	**+received:***YYYYMMDD*
SENT ON *date*	**+sent:***YYYYMMDD*
BEFORE *date*	**+received:{19700101 TO** *YYYYMMDD***}**
SENTBEFORE *date*	**+sent:{19700101 TO** *YYYYMMDD***}**
SENTSINCE *date*	**-sent:[19700101 TO** *YYYYMMDD***]**
SINCE *date*	**-received:[19700101 TO** *YYYYMMDD***]**

The Indexing and Search Service field names support for the following IMAP SEARCH features has not yet been implemented:

- **KEYWORD** *flag* (also **UNKEYWORD**)
- **HEADER** *field_name string*

Search Query Syntax

A search query consists of a list of terms, each separated by blanks. A term that you prefix with a plus sign (+) means "AND MATCHES", a term that you prefix with a minus sign (-) means "AND NOT MATCHES", and a term that you prefix with nothing (that is, a blank space) means "OR MATCHES".

Every query must contain one term that defines the host name and one that defines the user name of the account that you want to search. Thus, the simplest valid search query looks like the following:

```
+hostname:hhh +username:uuu
```

This search returns all emails in the account **uuu@hhh**. These two field names must appear as the first two terms in any search query. (Otherwise, terms may appear in any order in the query.) You can use additional terms in the query to select fewer results from this set. Only terms of the form **-term** or **+term** are allowed (OR cannot restrict the result).

A term may contain a keyword from the list of field names separated by a colon (:) from the value to be searched. A term value without a field name uses the default field name, which is **contents**. A term can also consist of a parenthesized list of terms (with an optional prefix character) to form more complex queries, including the OR MATCHES feature. Parenthesized terms can be nested.

The value of a term can be a single string, a phrase, or a range specifier. A single string term contains only the characters to be matched with no spaces, such as:

```
+contents:java
```

A single string term matches the specific string as a word in the field.

A phrase is a quoted string containing a list of single string terms separated by spaces, such as:

```
+contents:"java code"
```

A phrase term matches the sequence of string values as words in the field.

A range specifier is a pair of single string terms separated by the string TO and enclosed in either square or curly brackets, such as:

```
+size:{0 TO 1000}
    +size:[100 TO 2000]
    +sent:{19700101 TO 20010101}
```

A range term matches all values from the first to the second in the field. The square brackets signify to include the end points in the match; curly brackets signify to exclude the end points. Notice that date matches like the preceding example require the values to be fully specified in the form of *YYYYMMDD* for year, month, day.

Range terms containing the **uid** field name have changed so that the semantics of the search more closely match those of the IMAP SEARCH UID term. You can use the bound of "*" in a uid range term to indicate the upper bound matches through the largest uid available in the folder. For more information, see IETF RFC 3501 (INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1) at:

https://tools.ietf.org/html/rfc3501

Within a parenthesized list term, you can use the reserved words **AND**, **NOT**, and **OR** in addition to the corresponding + or - prefix characters to combine terms. These reserved words must be uppercase, for readability.

The **hostname** and **username** field names can only appear once at the beginning of the search query, and cannot appear in a parenthesized list. Range term values (that is, using "{ }" and "[ ]" with **TO** to specify a range of values) are not permitted within parenthesized expressions.

Range terms containing only **uid** or only **received** field names can appear in parenthesized expressions. No term with any other field name can appear in such a parenthesized expression.

Within any single parenthesized list term, only field names of the same "kind" may appear. Table 12–5 describes these kinds by specific function.

Table 12–5 Field Names for Functions

Function	Permitted Field Names
Folder terms	"folder" only
Flag terms	"answered," "deleted," "draft," "flagged," and "seen"
Meta terms	"received," "sent," "uid," and "uidvalidity"
Generic content terms	"attachgroup-contents"
Content terms	All other field names, including "body," and "text,"

The value of a term with a field name can also be a parenthesized list of values, for example:

```
+folder:(INBOX Trash Sent)
```

The preceding query is a shortened form of the following equivalent form:

```
+(folder:INBOX folder:Trash folder:Sent)
```

This query matches results in any one of the specified folders. In this form, the list of values must not have field names specified, because all values share the same field name.

Term Modifiers

The single string and phrase terms can contain modifiers that can be used for more complicated matching.

A single string term value can contain the special characters "?" and "*" to perform single character and multiple character wildcard matching respectively. For example, the following term matches the words "text" and "test" and any other single character in the third position of a word of this form:

```
+contents:te?t
```

The following term matches the words test, tests or tester, and so forth:

```
+contents:test*
```

Multiple wildcard characters can be used in combination to match more complicated strings. Wildcard characters as the first character of the string can be very expensive to match, so you can enabled and disable this feature by using the iss.searchsvc.leadingwildcard.enabled configuration parameter (enabled by default).

You can also append the single string term with a trailing "fuzzy search" modifier. This approach allows for matching words that are similar in spelling. For example, the following term also matches the words rest, nest, and tests:

```
+contents:test~
```

The phrase term can be modified with a trailing "proximity search" modifier, finding words that are a within a specific distance away. For example, the following term matches the words "java" and "code" within 10 words of each other in the field:

```
+contents:"java code"~10
```

Other Search Features

Various email clients have search criteria for relative dates such as "yesterday," "past month," "age in days," and so on. Each of these must be mapped into a **sent:** or **received:** term based on the specific date (such as "yesterday") or on a range of dates (like "past month"). It is the client's responsibility to generate the correct *yyyymmdd* form for each term.

For example, the following searches for all emails received this month, compute the date and use the year and month with the day in a wildcard match:

```
+received:201103??
```

This query matches all emails received in March 2011.

The following searches for all emails received in the last 90 days, computes the date as of 90 days ago, and uses a range term to select all emails since then:

```
+received:[20101223 TO 20110322]
```

This query matches all emails received from December 23, 2010 through March 22, 2011 inclusively.

13

Search Query Output Format

This chapter provides guidance for understanding Oracle Communications Indexing and Search Service search query results. It also describes how to specify different result formats in the search query. Several examples of search queries and search query responses are included.

Supported Output Formats

You can request a specific output format from a search query by specifying an optional format parameter and content format parameter. The supported format values include Rich Site Summary (RSS) 2.0, Atom 1.0, or JavaScript Object Notation (JSON) format as described in the OpenSearch 1.1 specification. The supported content format parameters include the standard format (for email, currently), or **attachmentOnly**, **simpleUID**, and **MJS**.

- When you specify the standard format, the results from the search query contain IMAP URIs that provide a compact representation of the account, folder, and UID of the message with the match.

  Note: The IMAP URI should be consumed by an IMAP client to provide access to the data. The URI is not directly usable by a web browser.

- When you specify **attachmentOnly**, each result contains an IMAP URI to the original attachment. Also, the **totalResults** and **itemsPerPage** values that are returned refer to the number of attachments that are located, not the number of emails. When you use **format=JSON**, each result includes a URI to the thumbnail that corresponds to the attachment.
- When you specify **simpleUID**, the result for each entry contains the folder name, UID Validity, and UID number of each matched email. This option applies only when you use **format=ATOM**. It is intended to speed up queries generated through IMAP by including only the minimum information necessary to identify each result email.
- When you specify **MJS** (which is only valid when you use **format=JSON**), the output is formatted in compact webmail (WMAP) format.

Sample JSON Output: Standard Format

The following example demonstrates a standard email search that requests three search results out of all the messages in a user's account that match the word "java" in the message body and attachments.

Search query:

```
http://iss.example.com:8070/rest/search?q=%2busername:jennifer%20%2bhostname:mail.
example.com%20%2bbody:java&c=3&format=json
```

Response:

```
{
                "title": "iss.example.com:8070 search: +username:jennifer
+hostname:mail.example.com +body:java",
                "link":
"http://iss.example.com:8070/rest/search?q=+username:jennifer%20+hostname:mail.
example.com%20+body:java&amp;c=3&amp;format=json",
        "modified": "Fri May 21 04:40:29 GMT 2010",
        "opensearch:totalResults": "160",
        "opensearch:startIndex": "0",
        "opensearch:itemsPerPage": "3",
        "items": [
        {
                "title": "JSP quick reference guide",
                "link":
"imap://jennifer@example.com/convergence;UIDVALIDITY=1246921910/;UID=2",
                "id":
"imap://jennifer@example.com/convergence;UIDVALIDITY=1246921910/;UID=2",
                "date": "2008-04-19T03:16:41Z",
                "folder": "convergence",
                "uid": "2",
                "from": "test <test@example.com>",
                "description": "From:test <test@example.com>
To:jennifer@example.com Folder:convergence"
        },
        {
                "title": "VTA Bike Map",
                "link":
"imap://jennifer@example.com/INBOX;UIDVALIDITY=1195248456/;UID=28",
                "id":
"imap://jennifer@example.com/INBOX;UIDVALIDITY=1195248456/;UID=28",
                "date": "2008-04-19T03:12:12Z",
                "folder": "INBOX",
                "uid": "28",
                "from": "test <test@example.com>",
                "description": "From:test <test@example.com>
To:jennifer@example.com Folder:INBOX"
        },
        {
                "title": "Installing Sun Java Enterprise System 6",
                "link":
"imap://jennifer@example.com/javaone2009;UIDVALIDITY=1243541192/;UID=30",
                "id":
"imap://jennifer@example.com/javaone2009;UIDVALIDITY=1243541192/;UID=30",
                "date": "2009-05-28T20:47:16Z",
                "folder": "javaone2009",
                "uid": "30",
                "from": "Jennifer <jennifer@example.com>",
                "description": "From:Jennifer <jennifer@example.com>
```

```
To:jennifer@example.com Folder:javaone2009"
        }
        ]
}
```

Sample JSON Output: attachmentOnly Format

The following example demonstrates a sample **attachmentOnly** search that requests three search results from all attachments in a user's account.

Search query:

```
http://iss.example.com:8070/rest/search?q=%2busername:jennifer%20%2bhostname:mail.
example.com%20%2battachment-type:%28atdoc%20atppt%20atxls%20atvcf%20atvsd%20ataudi
o%20atcompress%20athtml%20atimage%20atiwork%20atjpeg%20atodf%20atpdf%20atplain%20
atrtf%20atsencr%20atssign%20atvideo%20atxml%29&contentFormat=attachmentOnly&thumb
nail=s&c=3&format=json
```

Response:

```
{
                "title": "iss.example.com:8070 search: +username:jennifer
+hostname:mail.example.com +attachment-type:(atdoc atppt atxls atvcf atvsd ataudio
atcompress athtml atimage atiwork atjpeg atodf atpdf atplain atrtf atsencr atssign
atvideo atxml)",
                "link":
"http://iss.example.com:8070/rest/search?q=+username:jennifer%20+hostname:mail.
example.com%20+attachment-type:(atdoc%20atppt%20atxls%20atvcf%20atvsd%20ataudio%2
0atcompress%20athtml%20atimage%20atiwork%20atjpeg%20atodf%20atpdf%20atplain%20atr
tf%20atsencr%20atssign%20atvideo%20atxml)&amp;c=3&amp;format=json",
        "modified": "Fri May 21 04:26:50 GMT 2010",
        "opensearch:totalResults": "1511",
        "opensearch:startIndex": "0",
        "opensearch:itemsPerPage": "3",
        "items": [
        {
                "title": "100_0059.JPG",
                "link":
"imap://jennifer@mail.example.com/Sent;UIDVALIDITY=1272042833/;UID=1/;section=2",
                "id":
"imap://jennifer@mail.example.com/Sent;UIDVALIDITY=1272042833/;UID=1/;section=2",
                "type": "atjpeg",
                "content-type": "image/jpeg",
                "size": "3563598",
                "date": "2010-01-11T14:36:13Z",
                "folder": "Sent",
                "uid": "1",
                "part": "2",
                "from": "user71 71 <user71@example.com>",
                "subject": "email with attachment",
                "media": {"m":
"http://iss.example.com:8070/store/data/01/26/h1/u26/f3/1272042833/00/1/tbn
_medium_1_.jpg?fqdn=iss.example.com&mailhost=mail.example.com"},
                "description": "Subject: email with attachment Folder: Sent Size:
3563598 bytes<p><a
href=\"http://iss.example.com:8070/store/data/01/26/h1/u26/f3/1272042833/00/1/pic
_1_.jpg?fqdn=iss.example.com&mailhost=mail.example.com&filename=100
_0059.JPG\"><img
src=\"http://iss.example.com:8070/store/data/01/26/h1/u26/f3/1272042833/00/1/tbn
_small_1_.jpg?fqdn=iss.example.com&mailhost=mail.example.com\" height=\"75\"
width=\"75\" alt=\"100_0059.JPG\"></a>"
```

```
        },
        {
                "title": "72285838.mov",
                "link": "imap://jennifer@mail.example.com/MIME
tests;UIDVALIDITY=1266451357/;UID=1/;section=2",
                "id": "imap://jennifer@mail.example.com/MIME
tests;UIDVALIDITY=1266451357/;UID=1/;section=2",
                "type": "atvideo",
                "content-type": "video/quicktime",
                "size": "13892792",
                "date": "2010-02-17T23:52:51Z",
                "folder": "MIME tests",
                "uid": "1",
                "part": "2",
                "from": "Jennifer <jennifer@example.com>",
                "subject": "72285838.mov",
                "media": {"m":
"http://iss.example.com:8070/store/data/icons/video.png?fqdn=iss.example.com&mail
host=mail.example.com"},
                "description": "Subject: 72285838.mov Folder: MIME tests Size:
13892792 bytes<p><a
href=\"http://iss.example.com:8070/store/data/01/26/h1/u26/f7/1266451357/00/1/vid
eo_1_?fqdn=iss.example.com&mailhost=mail.example.com&filename=72285838.mov\"><img
src=\"http://iss.example.com:8070/store/data/icons/video.png?fqdn=iss.example.com
&mailhost=mail.example.com\"  alt=\"72285838.mov\"></a>"
        },
        {
                "title": "osol_poster.odg",
                "link":
"imap://jennifer@mail.example.com/INBOX;UIDVALIDITY=1195248456/;UID=33/;section=2
",
                "id":
"imap://jennifer@mail.example.com/INBOX;UIDVALIDITY=1195248456/;UID=33/;section=2
",
                "type": "atodf",
                "content-type": "application/vnd.oasis.opendocument.graphics",
                "size": "665190",
                "date": "2008-04-19T03:27:13Z",
                "folder": "INBOX",
                "uid": "33",
                "part": "2",
                "from": "test <test@example.com>",
                "subject": "OpenSolaris poster",
                "media": {"m":
"http://iss.example.com:8070/store/data/01/26/h1/u26/f1/1195248456/00/33/tbn
_medium_1_.png?fqdn=iss.example.com&mailhost=mail.example.com"},
                "description": "Subject: OpenSolaris poster Folder: INBOX Size:
665190 bytes<p><a
href=\"http://iss.example.com:8070/store/data/01/26/h1/u26/f1/1195248456/00/33/od
f_1_?fqdn=iss.example.com&mailhost=mail.example.com&filename=osol
_poster.odg\"><img
src=\"http://iss.example.com:8070/store/data/01/26/h1/u26/f1/1195248456/00/33/tbn
_small_1_.png?fqdn=iss.example.com&mailhost=mail.example.com\"  alt=\"osol
_poster.odg\"></a>"
        }
        ]
}
```

Part IV

Indexing and Search Service Log Messages

Part IV contains the following chapters:

14

Utility Service Log Messages

This chapter provides sample Utility Service (**utilSvc**) log messages and descriptions in alphabetical order.

FINE Level

FINE:
None.

INFO Level

INFO: read socket: found totalRead: 12 msg:isalive;5152
This message indicates the start of a utility request (is a process ID alive). You might see several such messages when the system runs normally.

INFO: isAlive: dataArg:5152 result:T
This message indicates the result of a utility request (is a process ID alive). You might see several such messages when the system runs normally.

INFO: read socket: found totalRead: 38 msg:diskusage;/var/iss/index//store/01/04/
This message indicates the start of a utility request (disk space for a specific account group). You might see several such messages when the system runs normally.

INFO: processed Request: 182M /var/iss/index//store/01/0/index104_content 140M /var/iss/index//store/01/04/index104_meta/
This message indicates the result of a utility request (disk space for a specific account group). You might see several such messages when the system runs normally.

WARNING Level

WARNING: Exception while checking pid 8347 java.io.IOException: Cannot run program "/bin/ps": error=12, Not enough space
While attempting to fork and exec a program (**/bin/ps** in this case), the system ran out of memory and thus failed. Check that the max heap size for **utilSvc** is not too large (the **java.args.utilsvc** option), and check that enough swap space is configured on the system.

WARNING: Util service shutdown in progress, after 7 requests processed, no further requests will be processed...

This message is logged when the utility service shuts down, indicating how many utility requests have been processed since the service was started. This message is normally the last message in the log file before the service shuts down, and records the time of service shutdown in the log file.

SEVERE Level

SEVERE: Util Exception: *various_generic_exception_details*

This message is logged when an unexpected exception occurs, causing the service to stop. It might indicate a configuration or system-level problem. A stack trace back is usually generated. Report details of such messages to Oracle Support.

SEVERE Util Exception: *various_generic_exception_details* continuing...

Similar to the previous message, the exception reported is local to a specific service request that failed. The service continues to respond to requests but might cause errors in other parts of the system. Restarting the service might correct the problem, depending on the specific exception details. Report details of such messages to Oracle Support.

15

Search Service Log Messages

This chapter provides a list of sample Search Service (**searchSvc**) log messages and descriptions in alphabetical order.

INFO Level

INFO: (debug) New account manager created: id=ACCTMGRID_20090404044319

This is a startup message.

INFO: 1 total matching documents to query: +username:user1 +hostname:mailhost.example.com +folder:"INBOX" +deleted:false +body:"testing", search time was 69ms

This message indicates that the search query for undeleted messages in the INBOX for **user1** with body containing "testing" matched one document and the search took 69 milliseconds.

INFO: 42 total matching documents to query: +username:user1 +hostname:mailhost.example.com +folder:"INBOX" +subject:"intern", search time was 65ms

This message indicates that the search query for messages in the INBOX for **user1** with subject containing "intern" matched 42 documents and the search took 65 milliseconds.

INFO: 52 total matching documents to query: +username:user1 +hostname:mailhost.example.com +folder:"INBOX" +from:"John.Smith@example.com", search time was 501ms

This message indicates that the search query for messages in the INBOX for **user1** from "John.Smith@example.com" matched 52 documents and the search took 501 milliseconds.

INFO: 8 total matching documents to query: +username:user1 +hostname:mailhost.example.com +folder:"INBOX" +subject:"AI's from call", search time was 46ms

This message indicates that the search query for messages in the INBOX for **user1** with subject "AI's from call" matched 8 documents and the search took 46 milliseconds.

INFO: AccountManager configuration: # of accounts per group: 1, max group allowed in store: 150000, default optimize level: 1, default # of writes between dIndex optimize: 50

This is a startup message.

INFO: Clean up before shutdown...

This message indicates that a service shutdown is in progress.

INFO: query=+username:user1 +hostname:mailhost.example.com +body:java, type=EMAIL, sort=, format=JSON, count=100, start=-1, serverName=isshost.example.com:8080, contentFormat=DEFAULT, thumbnail=DEFAULT

This message indicates that a search request has been received. The search is for mail for **user1**, any folder, with body containing "java", no sorting, JSON output format, start at match 0 and return up to 100 results, and use the default content format (email format rather than attachment format).

INFO: query=+username:user1 +hostname:mailhost.example.com +folder:"INBOX" +from:"John.Smith@example.com", type=EMAIL, sort=, format=SIMPLEUID, count=2147483647, start=-1, serverName=isshost.example.com:-1, contentFormat=SIMPLEUID, thumbnail=DEFAULT

This message indicates that a search request has been received from Messaging Server (which uses the **SIMPLEUID contentformat**). The search is for mail for **user1**, folder INBOX, from **John.Smith@example.com**, requesting all results to be returned.

INFO: query=+username:user1 +hostname:mailhost.example.com +attachment-type:at*, type=EMAIL, sort=, format=JSON, count=100, start=-1, serverName=isshost.example.com:8080, contentFormat=ATTACHMENTONLY, thumbnail=M

This message indicates that a search request has been received for the attachment format content type (thumbnails and attachment links). The search is for mail for **user1**, all folders, match any attachment (only PDF, Open Office documents and JPEGs are available in the attachmment store), requesting up to 100 results to be returned. Medium size thumbnails are requested and the JSON output format.

INFO: query=+username:user1 +hostname:mailhost.example.com +attachment-type:at*, type=EMAIL, sort=, format=RSS, count=5, start=10, serverName=isshost.example.com:8080, contentFormat=ATTACHMENTONLY, thumbnail=S

This message indicates that a search request has been received for the attachment format content type (thumbnails and attachment links). The search is for mail for **user1**, all folders, match any attachment (only PDF, Open Office documents and JPEGs are available in the attachment store), requesting up to five results to be returned, starting at result 10. Small size thumbnails are requested and the RSS output format.

WARNING Level

WARNING: AccountManager.getAccount: no such account (account could not be assigned to group) host=mailhost.example.com user=user1 searching: true acctId: xmailhostzexamplezcomxuser1x

A search was performed against an unindexed user.

WARNING: Search service shutdown in progress, no requests will be processed...

This message indicates that a service shutdown is in progress.

WARNING: Starting search service.

This is a startup message.

WARNING: StoreSearcher.search: query missing required EMAIL_HOSTNAME, EMAIL_USERNAME

This message indicates a malformed search query. Both username and hostname are required.

SEVERE Level

SEVERE: Will not respond search request, caught: com.sun.comms.iss.common.IssException: AccountManager.getAccount: no such account (account could not be assigned to group) host=mailhost.example.com user=user1 searching: true acctId: xmailhostzexamplezcomxuser1x

A search was performed against an unindexed user.

16

Message Queue Consumer and Broker Log Messages

This chapter provides a list of sample Message Queue Consumer (**jmqconsumer**) and Message Queue broker log messages and descriptions in alphabetical order.

Message Queue Consumer Log Messages

This section describes the Message Queue Consumer log messages and descriptions.

INFO Level

INFO: Account user1@mailhost.example.com is active on Mon Apr 06 00:00:14 PDT 2009, processing event generated on Mon Apr 06 00:00:14 PDT 2009 with sequence number 1061, time between generate and submit to index svc is 108ms.

This message indicates that the **user1** account is in Active state (as opposed to Bootstrapping or Inactive), so the real-time event 1061 has been passed to Index Services for indexing. The time between the event being generated on Messaging Server and it being passed to Index Services was 108 milliseconds (this depends on the Messaging Server and the Indexing and Search Service server having clocks which are synchronized).

INFO: Event queue: user1@mailhost.example.com:1 user2@mailhost.example.com:5

The accounts **user1** and **user2** have queued up events (one for **user1** and five for **user2**). This can happen if the accounts are not in Active state or if a flood of events have been received for the user (events are processed for each user in order).

INFO: Event queue: none

This regularly displayed log message indicates that there are no events queued up for processing. This is the normal state, if all accounts are Active.

INFO: Finished indexing event for account user1@mailhost.example.com with sequence number 10030, time between submit to and return from index svc is 243ms.

This message indicates that Index Services has returned successfully from a submitted real-time event. The total time elapsed between submitting the event from JMQ Consumer to Index Services and the successful return was 243 milliseconds.

INFO: JMSType: ChangeFlag (UID 102142,102146:102149,102153) Op 1-> Sys 0 PerUser 0

This message indicates that a **ChangeFlag** event was received for UIDs 102142, 102146-102149 and 102153.

INFO: JMSType: ChangeFlag (UID 110616:110617,110619,110622:110623,110627) Op 1-> Sys 0 PerUser 3

This message indicates that a **ChangeFlag** event was received for UIDs 110616-110617, 110619, 110622-110623, and 110627.

INFO: JMSType: ChangeFlag (UID 110621) Op 1-> Sys 0 PerUser 1

This message indicates that a **ChangeFlag** event was received for UIDs 110621.

INFO: JMSType: ChangeFlag (UID 110635) Op 2-> Sys 0 PerUser 0

This message indicates that a **ChangeFlag** event was received for UIDs 110635.

INFO: JMSType: Copy

This message indicates that a **Copy** event was received.

INFO: JMSType: Create

This message indicates that a **Create** event (new folder) was received.

INFO: JMSType: ExpungeMsg

This message indicates that an **ExpungeMsg** event was received.

INFO: JMSType: NewMsg

This message indicates that a **NewMsg** event (new message through SMTP/LMTP) was received.

INFO: JMSType: UpdateMsg

This message indicates that a **UpdateMsg** event (new message through IMAP) was received. This is frequently a new message to the **Sent** or **Drafts** folder.

INFO: Queuing event for user1@mailhost.example.com generated on Fri Apr 03 23:33:08 PDT 2009, with sequence number 79 on Fri Apr 03 23:33:08 PDT 2009

This message indicates that an event was received for **user1** and was generated at 23:33:08 on Messaging Server.

INFO: Received event ChangeFlag generated on Fri Apr 03 21:27:01 PDT 2009 with sequence number 13 uidValidity is 1198003994 operation is 1 uidlist is 435190 peruser_flags is 0 system_flags is 0 user_flags1 is 32 user_flags2 is null user_flags3 is null URL is imap://user1@mailhost.example.com/INBOX;UIDVALIDITY=1198003994/;UID=null

This message indicates that a **ChangeFlag** event was received for UID 435190, folder **INBOX**, user **user1**.

INFO: Received event ChangeFlag generated on Mon Apr 06 06:36:35 PDT 2009 with sequence number 2045 uidValidity is 1198002898 operation is 1 uidlist is 620857:620862,620865,620874,620876,620883:620884 peruser_flags is 3 system_flags is 0 user_flags1 is null user_flags2 is null user_flags3 is null URL is

imap://user1@mailhost.example.com/INBOX;UIDVALIDITY=1198002898/;UID=null

This message indicates that a **ChangeFlag** event was received for UIDs 620857-620862, 620865, 620874, 620876, 620883-620884, folder **INBOX**, user **user1**.

INFO: Received event Copy generated on Mon Apr 06 06:36:34 PDT 2009 with sequence number 2038 from-username is user1 from-foldername is INBOX to-username is user1 to-foldername is mac-users from-uidValidity is 1198002898 to-uidValidity is 1128698094 from-uidlist is 20804,620807,620830:620837,620843,620850:620851,620863,620867,620870:620873,620879:620880,620882,620885:620887 to-uidlist is 58001:58025 URL is imap://user1@mailhost.example.com/mac-users;UIDVALIDITY=1128698094/;UID=null

This message indicates that a **Copy** event was received for **user1**, from the INBOX to the **mac-users** folder.

INFO: Received event Create generated on Mon Apr 06 11:05:20 PDT 2009 with sequence number 3444 URL is imap://user1@mailhost.example.com/new-folder;UIDVALIDITY=1239041120/;UID=null

This message indicates that a **Create** event was received for **user1**, creating the **new-folder** folder.

INFO: Received event ExpungeMsg generated on Mon Apr 06 06:24:19 PDT 2009 with sequence number 2015 uidValidity is 962866757 uidlist is 612235:612238,612240,612242:612243,612245:612256 URL is imap://user1@mailhost.example.com/INBOX;UIDVALIDITY=962866757/;UID=null

This message indicates that the following UIDs were expunged the **INBOX** for **user1**: 12235-612238,612240,612242-612243,612245-612256.

INFO: Received event NewMsg generated on Fri Apr 03 21:24:35 PDT 2009 with sequence number 4 (size h:6526 e:10131 t:10131 - Using text) URL is imap://user1@mailhost.example.com/INBOX;UIDVALIDITY=961021076/;UID=51990

This message indicates that a **NewMsg** event was received for **user1**, folder **INBOX**, UID 51990. Header size was 6526 bytes, email size was 10131, matching the text size of the event, so the email was completely contained within the text of the event.

INFO: Received event NewMsg generated on Fri Apr 03 23:13:24 PDT 2009 with sequence number 77 (size h:8348 e:435411 t:270492 - Using imap) URL is imap://user1@mailhost.example.com/INBOX;UIDVALIDITY=1170916439/;UID=39376

This message indicates that a **NewMsg** event was received for **user1**, folder **INBOX**, UID 39376. Header size was 8348 bytes, email size was 435411 bytes, so larger than the text size of the event, which was 270492, so the email was not completely contained within the text of the event and the email is fetched instead by using IMAP.

INFO: Received event UpdateMsg generated on Fri Apr 03 21:38:32 PDT 2009 with sequence number 28 (size h:350 e:469 t:469 - Using text) URL is imap://user1@mailhost.example.com/Sent;UIDVALIDITY=1002212041/;UID=14555

This message indicates that an **UpdateMsg** event was received for **user1**, folder **Sent**, UID 14555. Header size was 350 bytes, email size was 469, matching the text size of the event, so the email was completely contained within the text of the event.

INFO: Received event UpdateMsg generated on Mon Apr 06 21:23:45 PDT 2009 with sequence number 7132 (size h:654 e:962626 t:262798 - Using imap) URL is imap://user1@mailhost.example.com/Sent;UIDVALIDITY=956794653/;UID=14480

This message indicates that an **UpdateMsg** event was received for **user1**, folder **Sent**, UID 14480. Header size was 654 bytes, email size was 962626 bytes, so larger than the text size of the event, which was 262798, so the email was not completely contained within the text of the event and the email is fetched instead by using IMAP.

INFO: Set account user1@mailhost.example.com state to A

This indicates that **user1** has been set to Active state. This message is generated when the account finishes bootstrapping.

INFO: Set account user1@mailhost.example.com state to B

This indicates that **user1** has been set to Bootstrapping state. This message is generated when the account starts bootstrapping.

INFO: Set account user1@mailhost.example.com state to I

This indicates that **user1** has been set to Inactive state. This message is generated when the account is synced by using the **issadmin.sh --checkaccount --sync** command.

INFO: Total: 102 Users: 32 Total active: 2 Active Users: 1 Total inactive: 3 Inactive Users: 2 Total bootstrap: 97 Bootstrap Users: 29

This is the last line of the "printQueues" messages in the JMQ log for checking the account queues. It summarizes the number of events in the backlog for all accounts in each of the account states. Thus:

```
Total of all events in backlog:    102   over 32  users

Total events from Active users:      2   over  1  user
Total events from Inactive users:    3   over  2  users
Total events from Bootstrap users:  97   over 29  users

So all the "Total" fields add up to the first value (2+3+97=102)
and all the "User" fields add up to the second value (1+2+29=32).
```

WARNING Level

WARNING: Account user1@mailhost.example.com doesn't exist, will not process event with sequence number 11

This message indicates that an event was received for an account that does not exist on the Indexing and Search Service host.

WARNING: [I500]: Caught JVM Exception: java.net.ConnectException: Connection refused

This message indicates that a connection could not be made to the IMQ broker. Check that the **mail.imq** setting is correct and that the IMQ broker is running.

WARNING: Connection problem: com.sun.messaging.jms.JMSSecurityException: [C4060]: Login failed: user=jmquser, broker=mailhost.example.com:7676(46078)

This message indicates that the setting for **mail.imq.user** or **mail.imq.password** is incorrect. If **mail.imq.password** is incorrect and must be changed, you might also need to edit the value stored in the **mail.imq.passfile** file.

WARNING: Exception while indexing, sequence number: 19006 com.sun.comms.iss.common.IssException: javax.mail.internet.ParseException : Expected parameter name, got "filenameMatrix.odt" Exception while processing: Message UID : 522565 On Host : mailhost.example.com For User : user1 In Folder : INBOX

This message indicates that an error occurred during indexing of a message.

WARNING: Shutdown in progress, no requests will be processed...

This message indicates that a service shutdown is underway.

WARNING: Starting up. Listening to Queue INDEXMS on mailhost.example.com:7676

This is a startup message indicating that JMQ consumer has connected to the broker at **mailhost.example.com**, port **7676**, and is listening to the Queue destination named **INDEXMS**.

WARNING: Unable to obtain JMS objects to send message response: [SEND_REPLY(9)] [C4036]: A broker error occurred. :[500] com.sun.messaging.jmq.jmsserver.util.BrokerException: Destination Q:temporary_destination:__queue_10.5.185.151_46656_1 could not be found in the store user=misoadmin, broker=isshost.example.com:7676(62279)

This message indicates that account state listener could not send a response back to the producer through the temporary queue. You can ignore this error as it has no impact on functionality.

Broker Log Messages

This section describes the Message Queue broker log messages and descriptions.

WARNING Level

WARNING [B3004]: No threads are available to process a new connection on service jms. 1000 threads out of a maximum of 1000 threads are already in use by other connections. A minimum of 2 threads must be available to process the connection. Please either limit the # of connections or increase the imq.<service>.max_threads property. Closing the new connection.

This indicates that the value of the **imq.jms.max_threads** property needs to increased (the default is 1000). It needs to accommodate 2 * number of connections to the Indexing and Search Service IMQ broker. The default location of the configuration file on Solaris is **/var/imq/instances/imqbroker/props/config.properties** and on Linux, the default location is **/var/opt/sun/mq/instances/imqbroker/props/config.properties**.

17

Index Service Log Messages

This chapter provides a list of sample Index Service log messages and descriptions in alphabetical order.

FINE Level

FINE: AccountManager.deleteDocuments: (debug) using ACCTMGRID_ 20090428002551

Seen when a user account is deleted using the **issadmin.sh --deleteaccount** command.

FINE: AccountManager.expungeAccount: ACCTMGRID= ACCTMGRID_ 20090408174626

Seen when a user account is deleted using the **issadmin.sh --deleteaccount** command.

FINE: Attachment type athtml is not stored. Host : <mailhost.example.com> User : <user114> Folder : <INBOX> Uid : <110222> NameOfAttachment : Attachment.html

HTML attachment files are not saved in attachment store; currently only JPEG, Open Office and PDF files are saved.

FINE: Attachment type atother is not stored. Host : <mailhost.example.com> User : <user4> Folder : <To Me> Uid : <7256> NameOfAttachment : smime.p7s

Unrecognized attachment types are classified under the category "other" and are not saved to the attachment store. This message indicates that an attachment with name "smime.p7s" in UID 7256 of Folder "To Me", user **user4** was found, but not saved.

FINE: Attachment type atplain is not stored. Host : <mailhost.example.com> User : <user24> Folder : <CY2001> Uid : <4424> NameOfAttachment : db_load_ msg.c

Plain text attachment files are not saved in attachment store; currently only JPEG, Open Office and PDF files are saved.

FINE: Attachment type atrtf is not stored. Host : <mailhost.example.com> User : <user148> Folder : <2008> Uid : <5314> NameOfAttachment : mail.rtf

RTF attachment files are not saved in attachment store; currently only JPEG, Open Office and PDF files are saved.

FINE: Attachment type atvcf is not stored. Host : <mailhost.example.com> User : <user24> Folder : <CY2002> Uid : <13453> NameOfAttachment : user24.vcf

vCard attachment files are not saved in attachment store; currently only JPEG, Open Office and PDF files are saved.

FINE: Attachment type atxml is not stored. Host : <mailhost.example.com> User : <user24> Folder : <INBOX> Uid : <461034> NameOfAttachment : Requirements document.xml

XML attachment files are not saved in attachment store; currently only JPEG, Open Office and PDF files are saved.

FINE: Skipping conversion of MIME type: APPLICATION/GZIP filename: 02.tar.gz Host : <mailhost.example.com> User : <user111> Folder : <Sent> Uid : <5135>

Gzipped files are skipped and are not indexed.

FINE: Skipping conversion of MIME type: APPLICATION/MS-TNEF filename: WINMAIL.DAT Host : <mailhost.example.com> User : <user114> Folder : <INBOX> Uid : <74682>

Files with MS-TNEF MIME type are skipped and are not indexed.

FINE: Skipping conversion of MIME type: APPLICATION/OCTET-STREAM filename: f810.exe Host : <mailhost.example.com> User : <user24> Folder : <INBOX> Uid : <35355>

.exe files are skipped and are not indexed.

FINE: Skipping conversion of MIME type: APPLICATION/PKCS7-SIGNATURE filename: smime.p7s Host : <mailhost.example.com> User : <user24> Folder : <INBOX> Uid : <400410>

Files with PKCS7-SIGNATURE MIME type are skipped and are not indexed.

FINE: Skipping conversion of MIME type: APPLICATION/POSTSCRIPT filename: rules1.2.ps Host : <mailhost.example.com> User : <user99> Folder : <RELEASE> Uid : <125>

Postscript files are skipped and are not indexed.

FINE: Skipping conversion of MIME type: APPLICATION/X-TAR filename: xtract.tar Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE02> Uid : <4311>

Tar files are skipped and are not indexed.

FINE: Skipping conversion of MIME type: IMAGE/GIF filename: top.gif Host : <mailhost.example.com> User : <user116> Folder : <INBOX> Uid : <11855>

GIF images are skipped and are not indexed.

FINE: Skipping conversion of MIME type: IMAGE/PNG filename: UI.png Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE05> Uid : <5020>

PNG images are skipped and are not indexed.

FINE: Skipping conversion of MIME type: MESSAGE/DELIVERY-STATUS filename: attachment Host : <mailhost.example.com> User : <user107> Folder : <testfolder> Uid : <75893>

Files with DELIVERY-STATUS MIME type are skipped and are not indexed.

FINE: Skipping conversion of MIME type: MESSAGE/EXTERNAL-BODY filename: draft-ietf-12.txt Host : <mailhost.example.com> User : <user114> Folder : <RFC> Uid : <3>

Files with EXTERNAL-BODY MIME type are skipped and are not indexed.

FINE: Found multipart MIME message, stop converting rest of input stream. Host : <mailhost.example.com> User : <user5> Folder : <INBOX> Uid : <612569> NameOfAttachment : [Fwd: performance tuning].eml

This message indicates that the message is a multipart MIME type. To avoid duplicated indexes, only the top part is indexed, assuming further attachment processing handles rest parts separately.

FINE: TextPlainConverter: large attachment found totalRead:10022047 Host : <mailhost.example.com> User : <user148> Folder : <ME> Uid : <5339> NameOfAttachment : vpn

Message in uid 5339, folder ME for user user148 contains a large plain text attachment file. This log message is printed out on every 1 MB that was read from the file. The maximum size that will be indexed defaults to 25 MB and is configurable.

FINE: The document is really a RTF file: Attendance_Sheet.rtf Host : <mailhost.example.com> User : <user113> Folder : <2006> Uid : <1246> NameOfAttachment : Attendance_Sheet.rtf

The file **Attendance_Sheet.rtf** from uid 1246, folder 2006, user **user113**, has been determined to be an RTF file instead of an MS Word file. **RTFConverter** is used to process this file.

FINE: EmailFlagManager: folder flag document empty, folderid: xmailhostzexamplezcomxuser24x_cy2002_fid_s_b_d_c_u_35_36 starting new one

Seen during the bootstrap of an account, this indicates a new record containing flag information for folder **cy2002** for user **user24@mailhost.example.com** needed to be created.

FINE: Waiting 300000 ms for ThreadPool termination:call#6103

Seen when multiple threads are used for bootstrapping, or during **issadmin.sh --checkacount** or other processing. This message is generated when the thread pool has completed submitting all its tasks, and is waiting for them all to complete.

FINE: missing thumbnail file, original name: Customers.sxc, using Default Icon: OOWriter.png Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE04> Uid : <8579>

This message indicates that there was no thumbnail image created for file **Customers.sxc**, found in uid 8579, folder **ARCHIVE04**, user **user99**. The default icon **OOWriter.png** is used as the thumbnail image for this file. This message is not displayed if the attachment file is of Microsoft document type, when default icon is always used.

INFO Level

INFO: Begin checking write.lock files.

Seen when starting the indexing service. A background thread is running to check leftover **write.lock** files in the index directory, which might cause index corruption.

INFO: End checking write.lock files.

Seen when starting the indexing service. A background thread is finished to check leftover **write.lock** files in the index directory, which might cause index corruption.

INFO: Initializing account state records.

Seen when starting services, this indicates the services are initializing account state records from dIndex.

INFO: Initializing account transition date records.

Seen when starting services, this indicates the services are initializing account transition date records from dIndex.

INFO: (debug) New account manager created: id=ACCTMGRID_20090405174109 (dIndex writable)

Seen when starting services, this indicates the time of creating the account manager. Only one such message indicating "writable" should appear (for the index services).

INFO: AccountManager configuration: # of accounts per group: 1, max group allowed in store: 150000, default optimize level: 1, default # of writes between dIndex optimize: 50

Seen when starting services, this indicates configured settings in effect for index services.

INFO: AccountManager.copyMessages: host: mailhost.example.com user: user61 fromUser: user61 toFolder: bloggers fromFolder: INBOX toUidV: 1192063983 fromUidV: 1198002898 toUidList: 4583:4584 fromFolderUidList: 621294:621295

Seen when messages are copied within an account, this indicates which messages were affected by the copy in the account.

INFO: AccountManager.finalCountSynch: time to finish host: mailhost.example.com user: user2 is 14ms

Seen after bootstrapping an account, this indicates the total time it took to complete writing all of the count information of emails in each folder.

INFO: AccountManager.getStatusMessages: time to find 10 status messages (acctid:xmailhostzexamplezcomxuser61x folderId:null index:00/61/index61_meta) is 81ms

Seen when using the **issadmin.sh --checkaccount** or **--checkfolder --detail status** commands, this indicates the time it took to find the indicated number of the status messages for the account or folder. The "null" folderId indicates the search was done for the entire account, not just one folder.

INFO: AccountManager.setAccountState: group index directories 00/04/index4_meta and 00/04/index4_content have been optimized

Seen when using the **issadmin.sh --setstate O** command, this indicates which directories are being optimized for the account.

INFO: All account state notification failure, server not running. This failure is only expected when using svc_control.sh to stop/start services, otherwise check if JMQConsumer is working.

Index service attempts to send out all account state notification at start up, to sync up with JMQ Consumer, but JMQ Consumer is not running. This is usually expected during installation or services restart using the **svc_control.sh** command, because JMQ Consumer is started after the index service.

INFO: Clean up before shutdown...

This is a normal shutdown message indicating cleanup up of resources before completely shutting down the index service.

INFO: EmailFlagManager.updateFlagStringsList: entering for folderid: xmailhostzexamplezcomxuser1x_bugs_fid_s_b_d_c_u_38 521 times, # of append calls: 299 longest string written: 245764

Seen during the update of email flags, this message indicates how often flag data was written to the index, and the size of the largest string written thus far. If many such messages appear, especially in a short period, this indicates frequent changes to the flag data, and may show a potential performance problem that should be reported.

INFO: EmailFlagManager.updateFlags: time to update host: mailhost.example.com user: user5 folder: INBOX 10 Email flags: 9ms

Seen after processing a change to email flags, this indicates the time to complete the update of the flag information for the folder of the account specified.

INFO: EmailFlagManager:copyFlags: time to update host: mailhost.example.com user: user34 folder: Trash 136 Email flags: 29ms

Seen after processing a copy of emails, this indicates the time needed to update the flags for the number of emails in the folder.

INFO: Exception: net.sf.jmimemagic.MagicMatchNotFoundException using default mime type converter: APPLICATION/OCTET-STREAM Host : <mailhost.example.com> User : <user24> Folder : <CY2002> Uid : <15592>

This message indicates that the MIME type of the attachment could not be determined through analysis of the attachment bitstream, so the default MIME type of APPLICATION/OCTET-STREAM is used.

INFO: Exception: net.sf.jmimemagic.MagicMatchNotFoundException using default mime type converter: TEXT/PLAIN Host : <mailhost.example.com> User : <user120> Folder : <INBOX> Uid : <49865>

This message indicates that the MIME type of the attachment could not be determined through analysis of the attachment bitstream, so the default MIME type of TEXT/PLAIN is used.

INFO: FolderCount.updateCounts: host: mailhost.example.com user: user1 updating folder: INBOX to 39674/39674/4971

Seen when messages are added or deleted from an account, this message shows the total number of the meta/content/attachment records that the indicated folder of the account now has in the index. (The last number, the number of attachments, is not always accurate, so ignore it.)

INFO: Found same converter class JPEGConverter in retry, ignored, Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE01> Uid : <3194>

In the attempt to find a suitable converter to process an attachment, the same **JPEGConverter** was determined in retry. The first try using **JPEGConverter** most likely had an error that was not caused by choosing the wrong converter. Index service reports the previous error and continues.

INFO: Found same converter class MSExcelConverter in retry, ignored, Host : <mailhost.example.com> User : <user99> Folder : <INBOX> Uid : <136110>

In the attempt to find a suitable converter to process an attachment, the same **MSExcelConverter** was determined in retry. The first try using **MSExcelConverter** most likely had an error that was not caused by choosing the wrong converter. Index service reports the previous error and continues.

INFO: Found same converter class MSWordConverter in retry, ignored, Host : <mailhost.example.com> User : <user120> Folder : <INBOX> Uid : <75007>

In the attempt to find a suitable converter to process an attachment, the same **MSWordConverter** was determined in retry. The first try using **MSWordConverter** most likely had an error that was not caused by choosing the wrong converter. Index service reports the previous error and continues.

INFO: Found same converter class PDFConverter in retry, ignored, Host : <mailhost.example.com> User : <user24> Folder : <CY2008> Uid : <18892>

In the attempt to find a suitable converter to process an attachment, the same **PDFConverter** was determined in retry. The first try using **PDFConverter** most likely had an error that was not caused by choosing the wrong converter. Index service reports the previous error and continues.

INFO: Free memory(MB):1002, Total memory(MB):4133, Max memory(MB):4133

Tracks the current memory usage: Free memory is the amount of free memory (in MB) in the JVM. Total memory is the amount of memory (in MB) currently in the JVM. Max memory is the max amount of memory (in MB) the JVM will attempt to use.

INFO: Processing user user24: 11884 Messages in Folder: INBOX

This message is displayed during bootstrapping. It indicates that bootstrapping of INBOX has commenced and there are 11884 messages in the folder.

INFO: Received event on account: user1@mailhost.example.com with sequence number 10136

Index service received a real time event request from JMQ Consumer, the sequence number was assigned by the JMQ Consumer.

INFO: Retrying different converter for filename: 0100000005100986.pdf Host : <mailhost.example.com> User : <user24> Folder : <INBOX> Uid : <434768>

In the attempt to find a suitable converter to process attachment file 0100000005100986.pdf, found in uid 434768, folder **INBOX** for user **user24**, the first try found an error which triggers a retry using different converter.

INFO: Sent notification of all account state.

Account state information for all accounts was posted to JMQ Consumer. This might happen if JMQ Consumer restarts.

INFO: SharedWriter.close: optimizing index:/dIndex Time: 101ms

This message indicates the time to complete optimizing the dIndex. Various commands can cause one or more such message, including bootstrapping. The time is proportional to the size, complexity, and amount of change that has occurred to the dIndex. Unusually long times, over 10 minutes, might indicate a performance bottleneck, and should be investigated. Also, if such messages for the same index appear very frequently, consider taking steps to improve the performance of the index store.

INFO: SharedWriter.close: optimizing index:/index61_content Time: 8859ms

This message indicates the time to complete optimizing the 61st group's content index. Various commands can cause one or more such message, including bootstrapping and **issadmin.sh --setstate O** commands. The time is proportional to the size, complexity, and amount of change that has occurred to the accounts in the index directories. Unusually long times, over 10 minutes, might indicate a performance bottleneck, and should be investigated. Also, if such messages for the same index appear very frequently, consider taking steps to improve the performance of the index store.

INFO: SharedWriter.close: optimizing index:/index61_meta Time: 3278ms

This message indicates the time to complete optimizing the 61st group's meta index. Various commands can cause one or more such message, including bootstrapping and **issadmin.sh --setstate O** commands. The time is proportional to the size, complexity, and amount of change that has occurred to the accounts in the index directories. Unusually long times, over 10 minutes, may indicate a performance bottleneck, and should be investigated. Also, if such messages for the same index appear very frequently, consider taking steps to improve the performance of the index store.

INFO: Total number of messages in folder is 33984; folderName is INBOX

Seen during bootstrapping of an account, this message shows the current number of messages found (33984) in the indicated folder (**INBOX**) at the current time.

INFO: Unable to find a converter for MIME type: APPLICATION/APPLEFILE filename: %imap.jar Host : <mailhost.example.com> User : <user24> Folder : <CY2003> Uid : <15314>

There is no registered converter for JAR files, and one such file was found as an attachment in an email with UID **15314**, folder **CY2003**, and user **user24**.

INFO: Unable to find a converter for MIME type: APPLICATION/OCTET-STREAM filename: CreateUser.class Host : <mailhost.example.com> User : <user24> Folder : <INBOX> Uid : <239260>

There is no registered converter for class files, and one such file was found in UID **239260**, folder **INBOX**, and user **user24**.

INFO: Unable to find a converter for MIME type: APPLICATION/X-MAKER filename: start-here.book Host : <mailhost.example.com> User : <user107> Folder : <testfolder> Uid : <73880>

There is no registered converter for Framemaker files, and one such file was found in UID **73880**, folder **testfolder**, and user **user107**.

INFO: Unable to find a converter for MIME type: APPLICATION/X-MSDOWNLOAD filename: list.htm.pif Host : <mailhost.example.com> User : <user114> Folder : <INBOX> Uid : <64234>

There is no registered converter for pif files, and one such file was found as an attachment in an email with UID **64234**, folder **INBOX**, and user **user114**.

INFO: Unable to find a converter for MIME type: APPLICATION/X-PKCS12 filename: cert.p12 Host : <mailhost.example.com> User : <user120> Folder : <INBOX> Uid : <100485>

There is no registered converter for p12 files, and one such file was found as an attachment in an email with UID **100485**, folder **INBOX**, and user **user120**.

INFO: Unable to find a converter for MIME type: APPLICATION/X-SUN-COREFILE filename: core Host : <mailhost.example.com> User : <user24> Folder : <Archive/1997> Uid : <4040>

There is no registered converter for core files, and one such file was found in UID **4040**, folder **Archive/1997**, and user **user24**.

INFO: Unable to read file mime type: java.io.IOException: Stream closed continuing with default converter(APPLICATION/OCTET-STREAM) Host : <mailhost.example.com> User : <user24> Folder : <CY2003> Uid : <14459>

Index service was not able to determine the file mime type while trying to find a suitable converter, use default converter for mime type APPLICATION/OCTET-STREAM for file found in UID **14459**, folder **CY2003**, and user **user24**.

INFO: Unable to read file mime type: java.io.IOException: Stream closed continuing with default converter(TEXT/PLAIN) Host : <mailhost.example.com> User : <user111> Folder : <dev> Uid : <7088>

Index service was not able to determine the file mime type while trying to find a suitable converter, use default converter for mime type TEXT/PLAIN for file found in UID **7088**, folder **dev**, and user **user111**.

INFO: [Begin: Sat Apr 04 19:36:02 PDT 2009] Processing User - user108 On Host - mailhost.example.com

This message indicates that bootstrapping of **user108** has begun.

INFO: [End: Sat Apr 04 19:40:55 PDT 2009] Processing User - user109 On Host - mailhost.example.com Elapsed time: 288 seconds

This message indicates that bootstrapping of **user109** has finished and took 288 seconds in total.

INFO: changeFlags: time to change flags in host: mailhost.example.com user: user5 folder: Drafts 60ms

Seen after processing a change to email flags, this message shows the total time it took (60 milliseconds) to modify the flags of the list of emails in the indicated folder.

INFO: deleteUidList: MetaDocs deleted: 113 content: 113

Seen after processing commands which delete messages from the index, this message shows the number of records in the meta data and content index directories deleted for an account. This gives some idea of the amount of change to the index; the two

numbers cannot be the same because of sharing of the content records when copying of emails between folders has occurred in the account.

INFO: endElement(): not adding invalid matcher '320 kBits'

Messages of this type can be ignored. They are displayed when attachment processing and the MIME detection library is started.

WARNING Level

WARNING: BootStrap of account, User: user147 Host: mailhost.example.com Empty account Optimize not needed, User: user147 Host: mailhost.example.com

This indicates that **user147** was bootstrapped, but no messages were found in the account.

WARNING: failed to index attachment, content type: APPLICATION/BINARY attachment type: atdoc filename: file.doc Exception Message: java.lang.ArrayIndexOutOfBoundsException Host : <mailhost.example.com> User : <user111> Folder : <INBOX> Uid : <144026>

Index service encountered an error while indexing file **file.doc**, found in uid **144026**, folder **INBOX**, and user **user111**. The file is not indexed.

WARNING: failed to index attachment, content type: APPLICATION/DOC attachment type: atdoc filename: file.sxi Exception Message: org.apache.poi.poifs.filesystem.OfficeXmlFileException: The supplied data appears to be in the Office 2007+ XML. POI only supports OLE2 Office documents Host : <mailhost.example.com> User : <user120> Folder : <INBOX> Uid : <48897>

Index service encountered an error while indexing file **file.sxi**, found in uid **48897**, folder **INBOX**, and user **user120**. The file is not indexed.

WARNING: failed to index attachment, content type: APPLICATION/MSWORD attachment type: atdoc filename: Final.doc Exception Message: org.apache.poi.hpsf.IllegalPropertySetDataException: The property set claims to have a size of 16 bytes. However, it exceeds 16 bytes. Host : <mailhost.example.com> User : <user24> Folder : <INBOX> Uid : <471123>

Index service encountered an error while indexing file **Final.doc**, found in uid **471123**, folder **INBOX**, and user **user24**. The file is not indexed.

WARNING: failed to index attachment, content type: APPLICATION/MSWORD attachment type: atdoc filename: benchmark.doc Exception Message: java.lang.IllegalStateException: Table Stream '0Table' wasn't found - Either the document is corrupt, or is Word95 (or earlier) Host : <mailhost.example.com> User : <user24> Folder : <CY2000> Uid : <1801>

Index service encountered an error while indexing file **benchmark.doc**, found in uid **1801**, folder **CY2000**, and user **user24**. The file is not indexed.

WARNING: failed to index attachment, content type: APPLICATION/MSWORD attachment type: atdoc filename: file.doc Exception Message: org.apache.poi.hpsf.IllegalPropertySetDataException: The property set claims to

have a size of 16 bytes. However, it exceeds 16 bytes. Host : <mailhost.example.com> User : <user120> Folder : <INBOX> Uid : <67738>

Index service encountered an error while indexing file **file.doc**, found in uid **67738**, folder **INBOX**, and user **user120**. The file is not indexed.

WARNING: failed to index attachment, content type: APPLICATION/MSWORD attachment type: atdoc filename: Plan.doc Exception Message: java.lang.StringIndexOutOfBoundsException: String index out of range: 18878 Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE01> Uid : <4089>

Index service encountered an error while indexing file **Plan.doc**, found in uid **4089**, folder **ARCHIVE01**, user **user99**. The file is not indexed.

WARNING: failed to index attachment, content type: APPLICATION/MSWORD attachment type: atdoc filename: Requirements.doc Exception Message: java.lang.ArrayIndexOutOfBoundsException Host : <mailhost.example.com> User : <user24> Folder : <CY2002> Uid : <11745>

Index service encountered an error while indexing file **Requirements.doc**, found in uid **11745**, folder **CY2002**, and user **user24**. The file is not indexed.

WARNING: failed to index attachment, content type: APPLICATION/OCTET-STREAM attachment type: atdoc filename: file.doc Exception Message: java.lang.ClassCastException: org.apache.poi.poifs.property.DocumentProperty cannot be cast to org.apache.poi.poifs.property.DirectoryProperty Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE01> Uid : <127>

Index service encountered an error while indexing file **file.doc**, found in uid **127**, folder **ARCHIVE01**, and user **user99**. The file is not indexed.

WARNING: AccountManager.checkCounts: mismatch in (mailhost.example.com user1) Bugs counts, count contains: 282 but counted 283 contents with attachments

Seen during the **issadmin.sh --checkaccount** or **--checkfolder** command, this message indicates the count of the attachments in folder named **Bugs** does not match what was expected in the index. Because the method used to count attachments is not fully functional, this kind of message is not correct and should be ignored.

WARNING: AccountManager.getAccount: no such account (account could not be assigned to group) host=mailhost.example.com user=user0 searching: true acctId: xmailhostzexamplezcomxuser0x

Seen when a failure has been detected, this message indicates that the requested account could not be found in the group it should have. Either the **issadmin.sh --group** value has an error, or the account does not exist in the store.

WARNING: [I500]: Caught JVM Exception: java.io.EOFException

This might be logged if the credentials to the Indexing and Search Service IMQ broker are not correct. Check the settings of **iss.imq.user** and **iss.imq.password**. If **iss.imq.password** must be changed, you might also need to change the value stored in the **iss.imq.passfile** file.

WARNING: General exception failure indexing XML Attachments: An invalid XML character (Unicode: 0x2) was found in the element content of the document.

Host : <mailhost.example.com> User : <user144> Folder : <in-archive> Uid : <1115> NameOfAttachment : sample.xml

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: Attribute name "noresize" associated with an element type "frame" must be followed by the ' = ' character. Host : <mailhost.example.com> User : <user111> Folder : <dev> Uid : <4351> NameOfAttachment : config_WINNT.xmlo

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: Content is not allowed in prolog. Host : <mailhost.example.com> User : <user144> Folder : <Trash> Uid : <24170> NameOfAttachment : mod_req.xml

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: Content is not allowed in trailing section. Host : <mailhost.example.com> User : <user120> Folder : <INBOX> Uid : <105565> NameOfAttachment : deleteFromIcal302HTTPResponse

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: Open quote is expected for attribute "{1}" associated with an element type "version". Host : <mailhost.example.com> User : <user144> Folder : <INBOX> Uid : <5914> NameOfAttachment : app.xml

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: The element type "br" must be terminated by the matching end-tag "</br>". Host : <mailhost.example.com> User : <user24> Folder : <CY2002> Uid : <14207> NameOfAttachment : Untitled Document.xml

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: The markup in the document following the root element must be well-formed. Host : <mailhost.example.com> User : <user144> Folder : <INBOX> Uid : <2774> NameOfAttachment : baseComponents.xml

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: The prefix "jabberd" for element "jabberd:cmdline" is not bound. Host : <mailhost.example.com> User : <user111> Folder : <Sent> Uid : <5621> NameOfAttachment : tel.xml

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: The processing instruction target matching "[xX][mM][lL]" is not allowed. Host : <mailhost.example.com> User : <user24> Folder : <INBOX> Uid : <469411> NameOfAttachment : mls-mta.xml

This message indicates a parse error processing an XML attachment.

WARNING: General exception failure indexing XML Attachments: The value of attribute "DN" associated with an element type "null" must not contain the '<' character. Host : <mailhost.example.com> User : <user24> Folder : <CY2005> Uid : <24155> NameOfAttachment : createservices.xml

This message indicates a parse error processing an XML attachment.

WARNING: Index service shutdown in progress, no requests will be processed...

This message indicates that a shutdown signal has been received to stop the index service, no further index requests are processed.

WARNING: IO failure indexing ODF Attachments: I/O error reading PNG header! Host : <mailhost.example.com> User : <user120> Folder : <INBOX> Uid : <77785> NameOfAttachment : StaffDec-07v1.2.odp

Index service encountered an I/O failure while indexing the Open Office attachment **StaffDec-07v1.2.odp**, found in uid **77785**, folder **INBOX**, and user **user120**. The file is not indexed.

WARNING: IO failure indexing ODF Attachments: Unexpected end of ZLIB input stream Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE04> Uid : <4589> NameOfAttachment : voc-11_0.sxc

Index service encountered an I/O failure while the indexing Open Office attachment **voc-11_0.sxc**, found in uid **4589**, folder **ARCHIVE04**, user **user99**. The file is not indexed.

WARNING: IO failure indexing ODF Attachments: null Host : <mailhost.example.com> User : <user24> Folder : <CY2002> Uid : <11252> NameOfAttachment : store-1.sxi

Index service encountered an I/O failure while indexing the Open Office attachment **store-1.sxi**, found in uid **11252**, folder **CY2002**, and user **user24**. The file is not indexed.

WARNING: IO failure indexing ODF Attachments: only DEFLATED entries can have EXT descriptor Host : <mailhost.example.com> User : <user24> Folder : <CY2006> Uid : <384> NameOfAttachment : offer.sxw

Index service encountered an I/O failure while indexing the Open Office attachment **offer.sxw**, found in uid **384**, folder **CY2006**, and user **user24**. The file is not indexed.

WARNING: IO failure indexing ODF Attachments: unexpected EOF Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE04> Uid : <7737> NameOfAttachment : Init_template05.sxw

Index service encountered an I/O failure while indexing the Open Office attachment **Init_template05.sxw**, found in uid **7737**, folder **ARCHIVE04**, and user **user99**. The file is not indexed.

WARNING: IO failure indexing XML Attachments: Invalid byte 2 of 3-byte UTF-8 sequence. Host : <mailhost.example.com> User : <user24> Folder : <CY2005> Uid : <27056> NameOfAttachment : fa90780.xml

Index service encountered an I/O failure while indexing the XML attachment **fa90780.xml**, found in uid **27056**, folder **CY2005**, and user **user24**. The file is not indexed.

WARNING: Starting index service.

Seen when starting the indexing service.

WARNING: Unable to convert PDF to image: java.awt.geom.IllegalPathStateException: missing initial moveto in path definition Host : <mailhost.example.com> User : <user24> Folder : <INBOX> Uid : <480841> NameOfAttachment : Download_Anaylsis.pdf

This message indicates that an error has occurred while converting PDF file **Download_Anaylsis.pdf**, found in uid **480841**, folder **INBOX**, and user **user24**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.io.IOException: Error: Unknown colorspace 'CS1' Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE04> Uid : <373> NameOfAttachment : 2P.pdf

This message is indicating an error has occurred while converting PDF file **2P.pdf**, found in uid **373**, folder **ARCHIVE04**, and user **user99**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.io.IOException: Not implemented Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE00> Uid : <5533> NameOfAttachment : WAP.pdf

This message is indicating the third party library is not able to convert PDF file **WAP.pdf**, found in uid **5533**, folder **ARCHIVE00**, and user **user99**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.lang.ArrayIndexOutOfBoundsException Host : <mailhost.example.com> User : <user24> Folder : <CY2006> Uid : <8745> NameOfAttachment : 0090093471.pdf

This message is indicating an error has occurred while converting PDF file **0090093471.pdf**, found in uid **8745**, folder **CY2006**, and user **user24**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.lang.ClassCastException Host : <mailhost.example.com> User : <user24> Folder : <Archive/1998> Uid : <17505> NameOfAttachment : Platinum.pdf

This message indicates taht an error has occurred while converting PDF file **Platinum.pdf**, found in uid **17505**, folder **Archive/1998**, and user **user24**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.lang.IllegalArgumentException: Invalid ICC Profile Data Host : <mailhost.example.com> User : <user99> Folder : <ARCHIVE05> Uid : <3835> NameOfAttachment : OMR-1.pdf

This message indicates that an error has occurred while converting PDF file **OMR-1.pdf**, found in uid **3835**, folder **ARCHIVE05**, and user **user99**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.lang.IllegalArgumentException: Width (-1218) and height (-1580) must be > 0 Host : <mailhost.example.com> User : <user111> Folder : <dev> Uid : <6445> NameOfAttachment : RESUME.pdf

This message indicates that an error has occurred while converting PDF file **RESUME.pdf**, found in uid **6445**, folder **dev**, and user **user111**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.lang.NegativeArraySizeException Host : <mailhost.example.com> User : <user112> Folder : <INBOX> Uid : <7391> NameOfAttachment : SignUp.pdf

This message indicates that an error has occurred while converting PDF file **SignUp.pdf**, found in uid **7391**, folder **INBOX**, and user **user112**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.lang.NullPointerException Host : <mailhost.example.com> User : <user1> Folder : <INBOX> Uid : <39760> NameOfAttachment : Special-2009D.pdf

This message indicates that an error has occurred while converting PDF file **Special-2009D.pdf**, found in uid **39760**, folder **INBOX**, and user **user1**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to convert PDF to image: java.lang.RuntimeException: Not yet implemented Host : <mailhost.example.com> User : <user8> Folder : <Sent> Uid : <14462> NameOfAttachment : feature-list.pdf

This message indicates that the third-party library is not able to convert PDF file **feature-list.pdf**, found in uid **14462**, folder **Sent**, and user **user8**, into thumbnail image. A default PDF icon is used as thumbnail image for this file.

WARNING: Unable to create thumbnail: Bg.jpg Host : <mailhost.example.com> User : <user56> Folder : <INBOX> Uid : <43513> java.lang.IllegalArgumentException: Width (0) and height (240) cannot be <= 0

This message indicates that an error has occurred while converting JPEG file **Bg.jpg**, found in uid **43513**, folder **INBOX**, and user **user56**, into thumbnail image. A default JPEG icon is used as thumbnail image for this file.

WARNING: Unable to extract metadata: atjpeg156142497245031460 6image022.jpg com.drew.metadata.MetadataException: Tag '306' cannot be cast to a java.util.Date. It is of type 'class java.lang.String'. Host : <mailhost.example.com> User : <user111> Folder : <Trash> Uid : <120223>

This message indicates that the index service is not able to extract metadata information for JPEG file **atjpeg156142497245031460 6image022.jpg**, found in uid **120223**, folder **Trash**, and user **user111**.

WARNING: Unable to process image: atjpeg104959767636257046117_01.gif com.drew.imaging.jpeg.JpegProcessingException: not a jpeg file Host : <mailhost.example.com> User : <user144> Folder : <Trash> Uid : <23532>

This message indicates that index service is not able process the image to extract metadata information for file **atjpeg104959767636257046117_01.gif**, found in uid **23532**, folder **Trash**, and user **user144**.

WARNING: Unable to set the Creation Date java.io.IOException: Error converting date: Thu Sep 29 17:24:24 2005

This message indicates that the index service is not able to set the creation date for this attachment file.

WARNING: Unsupported encoding printable-ascii specified in document, using preliminary encoding ISO-8859-1 instead

The encoding "printable-ascii" in an HTML attachment is not recognized. ISO-8859-1 encoding is used instead.

WARNING: computed thumbHeight 0 for C:DOCUME~1saraLOCALS~1TempnsmailP4.jpeg thumbWidth:240 imageRatio:612.0 thumbRatio:240.0 imageWidth:612 imageHeight:1 using thumbHeight=1 Host : <mailhost.example.com> User : <user24> Folder : <CY2000> Uid : <4051> NameOfAttachment : C:DOCUME~1saraLOCALS~1TempnsmailP4.jpeg

This message indicates an error during thumbnail creation for file **C:DOCUME~1saraLOCALS~1TempnsmailP4.jpeg**, found in uid **4051**, folder **CY2000**, and user **user24**. A default icon is used as thumbnail image for this file.

WARNING: java.io.IOException: Too many close-groups in RTF text : RTF Attachment : Unable to extract contents Host : <mailhost.example.com> User : <user111> Folder : <dev> Uid : <7088> NameOfAttachment : CV.doc

This message indicates a parse error processing an RTF attachment.

WARNING: javax.mail.internet.ParseException : Expected parameter name, got "filenameTeam Matrix.odt" Exception while processing: Message UID : 612836 On Host : mailhost.example.com For User : user5 In Folder : INBOX

This message indicates a parse error processing message with uid **612836**, in folder **INBOX**, for user **user5**.

SEVERE Level

SEVERE: JMS Exception: com.sun.messaging.jms.JMSSecurityException: [C4060]: Login failed: user=jmquser, broker=isshost.example.com:7676(50596)

This error might be logged if the credentials to the Indexing and Search Service IMQ broker are not correct. Check the settings of **iss.imq.user** and **iss.imq.password**. If **iss.imq.password** must be changed, you might also need to change the value stored in the **iss.imq.passfile** file.

18

GlassFish Server Log Messages

This chapter provides a list of sample Oracle GlassFish Server log messages and descriptions in alphabetical order.

INFO Level

INFO: Found LDAP session cookie.

This indicates the user session has already been authenticated.

INFO: Received connection from xxx.xxx.xxx.xxx

This indicates there was a request from the email server.

INFO: Set account user1@mailhost.example.com state to A

This indicates that **user1** has been set to Active state and is searchable. This message is generated when the account finishes bootstrapping.

INFO: Set account user1@mailhost.example.com state to B

This indicates that **user1** has been set to Bootstrapping state. This message is generated when the account starts bootstrapping.

INFO: Set account user1@mailhost.example.com state to I

This indicates that **user1** has been set to Inactive state. This message is generated when the account is synchronized by running the **issadmin.sh --checkaccount --sync** command.

INFO: Starting account state listener.

This indicates a listener is registered to listen to account state topic for account state change event.

INFO: Starting remove user path listener.

This indicates a listener is registered to listen to account state topic for user path removal event.

INFO: endElement(): not adding invalid matcher 'XXXX'

These types of messages are from the MIME type parser library, and can be ignored.

INFO: ip:xxx.xxx.xxx.xxx fqdn:null auth:FORM user:user1 pathinfo:/01/45/h1/u45/f8/1208230097/00/72/tbn_small_6_.jpg

This message logs a request to access **tbn_small_6_.jpg** from **user1**.

WARNING Level

WARNING: Unable to obtain JMS objects to send message response: [SEND_ REPLY(9)] [C4036]: A broker error occurred. :[500] com.sun.messaging.jmq.jmsserver.util.BrokerException: Destination Q:temporary_destination:__queue_10.5.185.151_46656_1 could not be found in the store user=misoadmin, broker=isshost.example.com:7676(62279)

This message indicates that account state listener could not send a response back to the producer through the temporary queue. The error has no impact on functionality, and this message can be ignored.

WARNING: Exception during auth com.sun.comms.iss.shared.domainmap.DomainResolutionException: Failed. could not find:example.com under o=internet

This message indicates a user authentication error against the LDAP server. Ensure that you have correctly set the **mail.basedn**, **mail.schemalevel**, and **mail.dcroot** parameters in the configuration file.

SEVERE Level

SEVERE: Could not create connection and producer com.sun.messaging.jms.JMSException: [ADD_PRODUCER_REPLY(19)] [C4036]: A broker error occurred. :[409] [B4183]: Producer cannot be added to destination SearchTopic [Topic], limit of 100 producers would be exceeded user=misoadmin, broker=brokerhost.example.com:7676(56692)|#]

This message indicates that the following parameter is missing in the Message Queue 4.3 **config.properties** file, located in the **/var/imq/instances/imqbroker/props/ directory** on Solaris and the **/var/opt/sun/mq/instances/imqbroker/props/** directory on Linux):

imq.autocreate.destination.maxNumProducers=-1

Part V

Indexing and Search Service Reference

Part V contains the following chapters:

19

Indexing and Search Service Command-Line Utilities

This chapter describes the Oracle Communications Indexing and Search Service command-line utilities.

Common Information

The Indexing and Search Service command-line utilities are located in the *IndexSearch_home*/**bin** directory, unless otherwise stated. To run these utilities, you must log in as or become the superuser (**root**), or the Indexing and Search Service user.

List of Indexing and Search Service Utilities

Table 19–1 describes the command-line utilities that you use to administer Indexing and Search Service.

Table 19–1 Indexing and Search Service Command-line Utilities

Utility	Description
checkIndex.sh	Checks individual index directories for consistency, and corrects some errors.
checkIss	Checks and verifies that the Indexing and Search Service services are running and that the log files have been written to recently.
checkStack	Checks the Indexing and Search Service installation by bootstrapping a user then performing a search through the RESTful interface.
csearchmgr.sh	Adds and removes Cluster Search Services for indexing nodes.
factorymgr.sh	Views and manages Java Naming and Directory Interface (JNDI) lookup objects.
issadmin.sh	Administers the Indexing and Search Service store.
isshttpdmgr	Starts and stops the **isshttpd** service, and configures Directory Server entries for the lookup table.
issrehostuser.sh	Automates the movement of accounts from one Indexing and Search Service store to another.
issversion	Reports the Indexing and Search Service version.
mergeIndex.sh	Merges multiple index files.

Table 19–1 (Cont.) Indexing and Search Service Command-line Utilities

Utility	Description
search_query_number.sh	Reports the number of search queries that have been run since the last time that **searchSvc** was restarted.
searchRun.sh	Searches the index interactively, from the command line. Prompts for input and displays the results of the query to standard output.
svc_control.sh	Starts and stops Indexing and Search Service services.
watchermgr.sh	Enables or disables the watcher service directly, instead of doing so by running the Indexing and Search Service **setup** script.

checkIndex.sh

The **checkIndex.sh** utility is a Lucene tool that you use to debug problems with the Indexing and Search Service store. You use it to check individual index directories in the Indexing and Search Service store for consistency, and to correct some kinds of errors. If indexing services are running, you might see errors from a running index, because the index can change while the utility is running. In any case, you are warned that the index services are running. **checkIndex.sh** is sometimes helpful in determining if an index directory has been corrupted, possibly during an "optimize" step. Use the **-fix** option with discretion, because if anything is wrong with the index, it usually causes data to be lost (perhaps a large amount of data). Before using the **checkIndex.sh** utility, ensure that you understand Lucene concepts and how data is indexed into documents.

Syntax

```
checkIndex.sh pathToIndex [-fix] [-segment X] ...
```

Options

Table 19–2 describes the options for the **checkindex.sh** utility.

Table 19–2 checkIndex.sh Options

Option	Description
pathToIndex	Required. The full directory path to an individual Indexing and Search Service index directory under **/var/iss**, for example, **/var/iss/index/store/dIndex**.
-fix	Writes a new **segments_N** file, removing any problematic segments.
-segment *X*...	Checks only the specified segments. You can specify this option multiple times to check more than one segment, for example: **-segment_2 -segment _a**. You cannot use this option with the **-fix** option.

Caution: Use the **-fix** option only on an emergency basis, as it causes documents (perhaps many documents) to be permanently removed from the index. Always make a backup copy of your index first before running the **checkindex.sh** utility. Do not run this utility on an index with active write activity.

When you run **checkIndex.sh** without the **-fix** option, it opens the index, reports version information, and reports any exceptions it encounters and what action it would take if the **-fix** option were specified. When you run **checkIndex.sh** with the **-fix** option, it removes any segments that have issues and writes a new **segments_N** file. All documents contained in the affected segments are removed.

Exit code 1 indicates either that the index cannot be opened or has been corrupted.

checkIss

The **checkIss** utility checks the Indexing and Search Service status. **checkIss** verifies that the Indexing and Search Service services are running and that the log files have been written to recently.

Syntax

```
checkIss [-t] [-B] [-h]
```

Options

Table 19–3 describes the options for the **checkIss** utility.

Table 19–3 checkIss Options

Option	Description
-t	Specifies the timeout for send and receive of JMQ messages.
-h	Prints out the usage summary.
-B	Checks dIndex backups for account state corruption.

Exit code 1 indicates a failure status along with information on the failure. Exit code 0 indicates no issues.

Example crontab Entry

You can use a crontab entry to schedule **checkIss** to run periodically, for example:

```
0,15,30,45 * * * * /opt/sun/comms/jiss/bin/checkIss 2>&1 > /tmp/checkIss || cat
/tmp/checkIss | mailx -s "checkISS warning" admin@example.com
```

Example Output from checkIss

In the following example, **checkIss** verifies that all services but the JMQ Consumer service are running.

```
checkIss
WARNING: JMQConsumer is not running
```

checkIss Functions Based on Indexing and Search Service Installation Type

Table 19–4 describes the services, processes, and parameters that the **checkIss** command verifies based on the Indexing and Search Service installation type. For more information about Indexing and Search Service processes and services, see "Indexing and Search Service Processes".

Table 19–4 checkIss Functions

Value of iss.cluster.install	Value of iss.cluster.type	What checkIss Performs
standalone	None	■ Checks Service Install for **utilSvc**, **indexSvc**, **searchSvc**, and **jmqconsumer**. ■ Checks pid files for **utilSvc**, **indexSvc**, **searchSvc**, and **jmqconsumer**. ■ Checks **jps** for signature for **utilSvc**, **indexSvc**, **searchSvc**, **jmqconsumer**, Message Queue, and Oracle GlassFish Server. ■ Checks **indexSvc** and **jmqcomsumer** log for activity. If log has been modified in over a hour, **checkIss** returns 1. ■ Checks JNDI lookup for factories and destinations. ■ Sends **isAlive** JMQ message to **indexSvc**. ■ Sends **isAlive** JMQ message to **searchSvc**. ■ Checks credentials for **mail.server:mail.imap.port** by using **mail.imap.admin.username** and **mail.imap.admin.password**. ■ Checks for the presence of producers on **mail.imq** for the destination of **mail.imq.name**. ■ Checks credentials for **mail.imq** by using **mail.imq.user** and **mail.imq.password**.
multi-machine	**index**	■ Checks Service Install for **utilSvc**, **indexSvc**, **searchSvc**, and **jmqconsumer**. ■ Checks pid files for **utilSvc**, **indexSvc**, **searchSvc**, and **jmqconsumer**. ■ Checks **jps** for signature for **utilSvc**, **indexSvc**, **searchSvc**, **jmqconsumer**, and Message Queue. ■ Checks **indexSvc** and **jmqcomsumer** log for activity, if log has been modified in over a hour, **checkIss** returns 1. ■ Checks JNDI lookup for factories and destinations. ■ Sends **isAlive** JMQ message to **indexSvc**. ■ Sends **isAlive** JMQ message to **searchSvc**. ■ Checks credentials for **mail.server:mail.imap.port** by using **mail.imap.admin.username** and **mail.imap.admin.password**. ■ Checks for the presence of producers on **mail.imq** for the destination of **mail.imq.name**. ■ Checks credentials for **mail.imq** by using **mail.imq.user** and **mail.imq.password**.
multi-machine	**web**	■ Checks **jps** for signature for GlassFish Server.
multi-machine	**ldap**	■ None.
multi-machine	**jmq**	■ None.
cluster	**web**	■ Checks Service Install for **utilSvc** and **csearchSvc**. ■ Checks pid files for **utilSvc** and **csearchSvc**. ■ Checks **jps** for signature for **utilSvc**, **csearchSvc**, and GlassFish Server. ■ Checks JNDI lookup for factories and destinations.

Table 19–4 (Cont.) checkIss Functions

Value of iss.cluster.install	Value of iss.cluster.type	What checkIss Performs
cluster	**index**	■ Checks Service Install for **utilSvc**, **indexSvc**, and **jmqconsumer** ■ Checks pid files for **utilSvc**, **indexSvc**, and **jmqconsumer** ■ Checks **jps** for signature for **utilSvc**, **indexSvc**, **jmqconsumer**, and Message Queue ■ Checks **indexSvc** and **jmqcomsumer** log for activity. If log has been modified in over a hour, **checkIss** returns 1 ■ Checks JNDI lookup for factories and destinations ■ Sends **isAlive** JMQ message to **indexSvc** ■ Checks credentials for **mail.server:mail.imap.port** by using **mail.imap.admin.username** and **mail.imap.admin.password** ■ Checks for the presence of producers on **mail.imq** for the destination of **mail.imq.name** ■ Checks credentials for **mail.imq** by using **mail.imq.user** and **mail.imq.password**
clusterv2	**web**	■ Checks **jps** for signature for GlassFish Server. ■ Checks JNDI lookup for factories and destinations.
clusterv2	**csearch**	■ Checks Service Install for **utilSvc** and **csearchSvc**. ■ Checks pid files for **utilSvc** and **csearchSvc**. ■ Checks JNDI lookup for factories and destinations. ■ Checks **jps** for signature for **utilSvc**, **csearchSvc**, and Message Queue.
clusterv2	**index**	■ Checks Service Install for **utilSvc**, **indexSvc**, and **jmqconsumer**. ■ Checks pid files for **utilSvc**, **indexSvc**, and **jmqconsumer**. ■ Checks **jps** for signature for **utilSvc**, **indexSvc**, **jmqconsumer**, and Message Queue. ■ Checks **indexSvc** and **jmqcomsumer** log for activity If log has been modified in over a hour, **checkIss** returns 1. ■ Checks JNDI lookup for factories and destinations. ■ Sends **isAlive** JMQ message to **indexSvc**. ■ Checks credentials for **mail.server:mail.imap.port** by using **mail.imap.admin.username** and **mail.imap.admin.password**. ■ Checks for the presence of producers on **mail.imq** for the destination of **mail.imq.name**. ■ Checks credentials for **mail.imq** by using **mail.imq.user** and **mail.imq.password**.

The check service install verification performs different checks depending on the operating system:

- Solaris: Verifies the Service Management Facility (SMF) entries and that the service is in an enabled state.
- Linux: Verifies the **chkconfig** entries and that the service is in an on state.

checkStack

The **checkStack** utility checks the Indexing and Search Service installation by bootstrapping a user then performing a search of that user through the RESTful interface.

Syntax

```
checkStack [-h|--help] [--appserver-host host] [--host host] [--folder folder]
           [--password] [--passwordfile file] [--port port] [--user user] [-h]
```

Options

Table 19–5 describes the options for the **checkStack** command.

Table 19–5 checkStack Options

Option	Description
--appserver-host *host*	Specifies the GlassFish Server host name, defaults to **localhost**. Used for **cluster**, **clusterv2** and **multi-machine** installations.
--port *port*	Specifies the GlassFish Server port, defaults to 8080.
--host *host*	Specifies the mail host for the bootstrapped user.
--password	Prompts for the user password to authenticate to GlassFish Server.
--passwordfile *file*	Reads the user password from the file name provided.
--user *user*	Bootstraps and searches this user name.
--folder *folder*	Only bootstraps and searches this folder, default for search is **INBOX**.
-h | **--help**	Prints out the usage summary.

Example

```
checkStack.sh --user c273 --host mail.example.com --passwordfile /var/tmp/userpass
--appserver-host web.example.com --port 8070

Starting bootstrap of user c273
Running command [/opt/sun/comms/jiss/bin/issadmin.sh, --bootstrap, --user, c273,
--host, mail.example.com]
Wed Aug  8 15:59:33 GMT 2012
Starting indexing c273                at Wed Aug 08 15:59:36 GMT 2012
Ending task       c273                at Wed Aug 08 15:59:46 GMT 2012 after 10
seconds
Wed Aug  8 15:59:46 GMT 2012
Successfully bootstrapped user c273
Successfully searched user c273 returned 592 results
```

csearchmgr.sh

The **csearchmgr.sh** utility adds and removes Cluster Search Services for indexing nodes. Use this utility only in a clustered environment where the **iss.cluster.enabled** parameter is set to **true**.

Syntax

```
csearchmgr.sh [-b basedir] [-l] [-A|-D] [-a|-d] [-n instance_name] [-h]
```

Options

Table 19–6 describes the options for the **csearchmgr.sh** utility.

Table 19–6 csearchmgr.sh Options

Option	Description
-b *basedir*	Specifies the Indexing and Search Service base installation directory. Optional.
-l	Lists all Cluster Search Service entries. Optional.
-A	Adds all Cluster Search Service configuration found in the **iss.cluster.dir** parameter. Optional.
-D	Removes all Cluster Search Service configurations. Optional.
-a	Adds a Cluster Search Service entry for name. Optional.
-d	Removes a Cluster Search Service entry for name. Optional.
-n *instance_name*	Specifies the name of the Cluster Search Service entry on which to perform action. Required if **-a** or **-d** is provided.
-h	Prints out the usage summary.

Exit code 0 indicates a success. Exit code 1 indicates a failure.

Options can appear in any order.

Examples

List Cluster Search Services:

```
csearchmgr.sh -l
```

Add all Cluster Search Services found in the **iss.cluster.dir** parameter:

```
csearchmgr.sh -A
```

Remove all Cluster Search Services:

```
csearchmgr.sh -D
```

Add a single Cluster Search Service:

```
csearchmgr.sh -a -n instance_name
```

Remove a single Cluster Search Service:

```
csearchmgr.sh -d -n instance_name
```

factorymgr.sh

The **factorymgr.sh** utility is used to view and manage JNDI lookup objects.

Syntax

```
factorymgr.sh [-h] [-l] |[-m] [-b basedir] [-p parameter] [-n lookupname] [-v
value]
```

Options

Table 19–7 describes the options for the **factorymgr.sh** utility.

Table 19–7 factorymgr.sh Options

Option	Description
-l	Lists all objects or object provided by **-n** *lookupname*. Required if **-m** is not provided.
-m	Modifies object properties. Requires **-n**, **-p**, and **-v**. Required if **-l** is not provided.
-n *lookupname*	Lists or modifies the lookup name of the object. Required if **-m** is provided.
-b *basedir*	Optional. Provides installation base directory.
-p *parameter*	Specifies the parameter to be modified. Required if **-m** is provided. Optional if **-l** is provided.
-v *value*	Sets value of parameter. Required if **-m** is provided.
-h	Optional. Prints out the usage summary.

Exit code 0 indicates a success. Exit code 1 indicates a failure.

Options can appear in any order.

Examples

List all objects:

```
factorymgr.sh -l
```

List parameters for an object:

```
factorymgr.sh -l -n cn=CommsQueueFactory
```

Modify a parameter for an object:

```
factorymgr.sh -m cn=CommsQueueFactory -p imqBrokerHostName -v imq.example.com
```

issadmin.sh

The **issadmin.sh** utility displays information about the Indexing and Search Service store instance, and provides options to administer the store, enabling you to modify the accounts and folders of accounts. You can use this utility whether the Indexing and Search Service services are running or not.

The **issadmin.sh** utility takes list, action, and modifier options. List options print information contained in the Indexing and Search Service store. List options do not modify any Indexing and Search Service store data. Action options make changes to the Indexing and Search Service store data. Modifier options provide specific details required for each action or list option. Options can appear in any order.

Syntax

```
[-h|--help] [--listaccountlistfile] [--listaccounts]
[--listactiveservices] [--listbacklog] [--listbrief] [--listfolders]
[--listgroups] [--liststats] [--host host] [--user user]
[--folder folder] [--accountinfo] [--accountlist accountfile]
[--altoutput file] [--autobootaccounts] [--backup] [--bootstrap]
[--checkaccount] [--checkconfig] [--checkfolder] [--checkstore]
[--continueonerror] [--converttoformat fmt] [--createaccount]
[--createfolder] [--datapath path] [--deleteaccount]
[--deleteautobootlist] [--deletefolder] [--detail detaillist]
[--eximpath path] [--export] [--group groupnum] [--groupinfo]
```

```
[--ignorefolder] [--import] [--importversion version] [indexmode full|IMAPonly]
[--lockmemoryindex] [--loglevel all|config|fine|finer|finest|info|servere|warning]
[--moveaccount] [--password] [--passwordfile file] [--port port]
[--prompt] [--protocol ssl|tls] [--reboot] [--refresh] [--rehost host]
[--renamefolder newfoldername] [--runoptimizer true|false]
[--selectstate statelist] [--selecttime h[:m]]
[--setautobootlist file] [--setdefaulteximpath path]
[--setdefaulthostname host] [--setstate newstate] [--singleton]
[--skipfolder folder] [--stopservice usid] [--storepath path] [--sync]
[--threads n] [--timeout seconds] [--uid uidlist] [--unignorefolder]
[--unlockmemoryindex] [--unsetautobootlist file] [--useramdir]
```

The **--accountinfo**, **--bootstrap**, **--checkaccount**, **--checkfolder, --createaccount**, **--createfolder**, **--deleteaccount**, **--deletefolder**, **--export**, **--ignorefolder**, **--import**, **--moveaccount**, **--renamefolder**, **--setstate**, and **--unignorefolder** options require the **--user** and **--host** options.

If you do not specify a value for **--host**, **issadmin.sh** uses the default host for the Indexing and Search Service store.

The **--deleteautobootlist** option can optionally specify the **--host** and **--user** options.

The **--checkfolder**, **--createfolder**, **--deletefolder**, **--ignorefolder**, **--renamefolder**, and **--unignorefolder** options require the **--folder** option.

The **--moveaccount** option requires the **--group** option.

The **--groupinfo** option requires either the **--group** option or the **--host** and **--user** options.

The **--password**, **--passwordfile** *file*, **--port**, and **--protocol** options only apply to the **--bootstrap**, **--checkaccount**, and **--checkfolder** options.

The **--sync** and **--detail** options apply only to the **--checkaccount**, **--checkfolder**, and **--checkstore** options.

The **--detail** *detaillist* option can contain only one or more of the values **dindex**, **store**, **group**, **full**, **ignoreok**, **flags**, **status**, **deepcheck**, or **verbose** in a list separated by commas.

The **--uid** option applies only to **--deletefolder**.

The **--threads** and **--continueonerror** options apply only to **--accountlist**.

The **--password** and **--passwordfile** options are for non-production use only.

The **--password** option prompts you for the password.

You can specify the **--setautobootlist** *file1* and **--unsetautobootlist** *file2* options in the same command, but only once each. *file1* cannot reference the same file as *file2*.

You can specify **--indexmode** with **--bootstrap**, **--createaccount**, or **--setautobootlist** options, to override the value of the **iss.indexsvc.attachment.indexmode** configuration parameter. The **iss.indexsvc.attachment.indexmode** configuration parameter specifies to either generate all attachment records along with the "fullcontents" field, or only the "fullcontents" field.

Options

Table 19–8 describes **issadmin.sh** list options. The **--list*** options are mutually exclusive. Only the last option appearing on the command line has any effect. These options print after any action specified in the command has been performed, so they show Indexing and Search Service store information after any change has occurred,

not before. The **--info*** options act like actions, so the **--list*** options can also be used when these options are used. Output goes to **stdout** unless otherwise noted.

Table 19–8 List Options

Option	Description
--accountinfo	Prints out all information for a single account. Requires the **--host** and **--user** options.
--checkaccount	Compares the specified account in the Indexing and Search Service store against the information in the corresponding account in the Messaging Server message store for consistency. Requires the **--host** and **--user** options. Prints out to **stderr** any discrepancies detected, including problems detected during the bootstrap process. (Fine-grain details such as message flags or problems detected during the bootstrap process are checked only when you specify the **--detail** option.) By default, the system values for connecting to the Messaging Server message store by using IMAP are used. You can use the **--password**, **--passwordfile**, **--port**, and **--protocol** options to override these defaults.
--checkfolder	Conducts consistency checks of the specified folder in the specified account in the Indexing and Search Service store against the information for the corresponding account and folder in the original Messaging Server message store. Requires the **--folder**, **--host**, and **--user** options. Prints out to **stderr** any discrepancies detected, including problems detected during the bootstrap process. (Fine-grained details such as message flags or problems detected during the bootstrap process are checked only when you specify the **--detail** option.) By default, the system values for connecting to the Messaging Server message store by using IMAP are used. You can use the **--password**, **--passwordfile**, **--port**, and **--protocol** options to override these defaults.
--detail *detaillist*	Specifies that the data in the store or the specific folder or account should be checked in detail. Used only with the **--checkfolder**, **--checkaccount**, and **--checkstore** options. Output reflects the differences found at a more detailed level. **detaillist** is a list of attributes, delimited by commas, from the following set: **dindex store group full flags ignoreok status deepcheck verbose** If this option is not specified with **--checkaccount** or **--checkfolder** options, an account or folder is checked for the correct number of emails in the folder only. If this option is not specified with the **--checkstore** option, the store is checked for **dindex** consistency only. Values of **flags**, **ignoreok**, and **status** can only be used with the **--checkaccount** and **--checkfolder** options. The **dindex, store, group, deepcheck,** and **verbose** values can only be used with the **--checkstore** option. If **status** is specified, the status messages from the bootstrap process in the index are also displayed in the command output. (These indicate when the index was incomplete, for example, if an attachment failed to convert.) If **flags** is specified, the flag values of each message are checked to match the content server values (Message Server message store). If **ignoreok** is specified, then if the only command output shows folders marked as ignored, the command omits these warnings, and returns success. If **store** is specified, then all the data in every account group in the entire store is checked for consistency. If **group** is specified with the **--checkstore** option, each group meta and content index is opened when **store** is also specified to verify its integrity. (This is a time consuming operation when the store consists of thousands of group index directories. You should do this only one time while the **--sync** option is used.) If **deepcheck** is specified with the **--checkstore** option, the output includes information about which accounts must be reindexed, such as after upgrading to Java 7. This output is formatted in the form of the **--accountlist** file to be used with the **--setautobootlist** *file* command. If **verbose** is specified with the **--checkstore** option, more details of the checking are generated. If **full** is specified, every level of detailed checking is performed. Detailed checking can impact system performance. Running detailed checking can be up to four times slower for **--checkaccount** when using **--detail full** on larger accounts.

Table 19–8 (Cont.) List Options

Option	Description
--groupinfo	Prints out all information for a single group. To identify which group, this option requires either the **--group** option or the **--host** and **--user** options for any account in the group.
-h or **--help**	Prints out the usage summary.
--listaccounts	Prints out summary information for all accounts in all groups in the store instance.
--listaccountlistfile	Prints summary of account information that is formatted suitably for use as a file with the **--accountlist** option. This command provides a simple way to track new accounts generated during autoprovisioning.
--listactiveservices	Prints out the list of services currently active in the Index Server related to **issadmin.sh** commands. The unique service identifier (USID) of each service is printed. For more information, see the **--stopservice** *usid* option.
--listbrief	Prints out simplified summary information for all accounts in all groups in the store instance. The output contains only basic account information, one account per line, and is faster than the **--listaccounts** option for large numbers of accounts.
--listfolders	Prints out all information for all folders of all accounts in all groups in the store instance.
--listgroups	Prints out summary information for all groups in the store instance.
--liststats	Prints out internal statistics for debugging purposes.

Table 19–9 describes action options.

Table 19–9 Action Options

Option	Description
--autobootaccounts	For use only with the **--listaccountlistfile** command. Modifies output to contain only the list of accounts waiting for autobootstrap in the standardized **--accountlist** format. The output includes comments indicating the event error count for each account. You can use the list to select accounts manually based on these counts.
--backup	Creates an immediate backup of the dIndex, triggering the notification of the disk full status to service requestors. After the disk space reaches the limit allowed (by default 95 percent), all index services stop being processed until the disk space available shows this threshold is no longer exceeded. After enough disk space has been freed up, use this command to trigger normal processing of indexing services to continue.
--bootstrap	Creates the initial index for an account, known as bootstrapping the account. Bootstraps the account or folder specified. If the **--folder** option is used with this action, then only that folder (and any descendants) are bootstrapped. Bootstrapping involves indexing all the content from an account by using several parallel threads when possible. If you do not create the account beforehand, **--bootstrap** creates the account based on the **--group** or **--singleton** options specified. If no such information is specified, then the account is assigned to a group by using the default allocation algorithm. (To provide maximum control over the placement of accounts in groups, create accounts by using the **--createaccount** command before you bootstrap them.) If you create the account before using **--bootstrap**, then the account must be in the Inactive (I) or Unknown (X) state for this command to operate. During the bootstrap process, the state of the account is Bootstrap (B), and is changed to Active (A) upon completion. Depending on the amount of information in the account, this command can take several minutes to complete, or even hours for very large accounts. See also the **--runoptimizer** and **--skipfolder** options, which can be used with **--bootstrap**. For more information about bootstrapping many accounts, see "Setting Up Large Deployments".
--checkconfig	Compares the current values in the configuration files against the values currently active in the running services. Services must be running to use this option. Any changes that have occurred in the configuration files that have not been applied to the services either through the **--refresh** command or restarting the services are directed to standard error.
--checkstore	Compares the index store for consistency. When used alone or with the **--detail** option, this command reports any problems found in the internal consistency of the dIndex and account index directories. This is useful during recovery from a major failure, such as loss of power which might have corrupted data in the store. When problems with the store are detected, use with the **--sync** option. The **--sync** option resolves the problem by using different parts of the store data. If the **issadmin.sh --checkstore --sync** command detects problems with a user account and cannot fix the errors, it deletes the account and schedules it for autobootstrap. You can configure some **--checkstore** detections that cause little performance impact to run automatically when the IndexService starts. In addition, you can configure Indexing and Search Service to automatically repair some of these problems, resulting in fewer failures overall. For more information, see "Automatically Checking and Repairing dIndex Problems".
--converttoformat *fmt*	Converts the format of the dIndex implementation. The *fmt* value determines the format to convert to. Currently this value must be either **1** or **2**. No other options may be used with this command. When the format of the dIndex is the same as the *fmt* value specified, the command causes no change to the dIndex. When the *fmt* value is different than the format of the dIndex, the current dIndex file is renamed and replaced by the reformatted dIndex. The configuration parameter iss**.store.partitions.count** is used to determine the number of the resulting dIndex partitions. This command is only allowed when Indexing and Search Service services are not running.

Table 19–9 (Cont.) Action Options

Option	Description
--createaccount	Creates an account in the Indexing and Search Service store. Requires the **--host** and **--user** options. You can also specify the **--group** and **--singleton** options. If the **--group** option is not also specified, the default allocation policy selects a group into which to place the account. For more information about default allocation policy, see "Overview of the Indexing and Search Service Store Instance".
--createfolder	Creates a folder in an account in the Indexing and Search Service store. Requires the **--folder**, **--host**, and **--user** options.
--deleteaccount	Deletes an account in the Indexing and Search Service store. Requires the **--host** and **--user** options.
--deleteautobootlist	Removes all accounts from the current autobootstrap list. When the **--user** and **--host** options are also specified, only the specific account is removed from the autobootstrap list.
--deletefolder	Deletes a folder tree in an account in the Indexing and Search Service store. Requires the **--folder**, **--host**, and **--user** options. Any folders underneath the specified folder are also deleted. If **--deletefolder** detects nested folders, you are prompted to continue, enabling you to quit the command. See also the **--uid** option.
--export	Copies data for an account, creating a snapshot directory under the current snapshot repository. Requires the **--user** and **--host** options. Use this command to back up accounts, and to create snapshots that can be used with the **--import** option to move accounts between store instances. The account must not be in the Active state, that is, the account must be Inactive. This command creates the snapshot repository if it does not exist. See the **--eximpath** modifier and **--setdefaulteximpath** option for how to alter the snapshot repository. Also, see "Exporting and Importing Accounts Example".
--ignorefolder, **--unignorefolder**	Marks a folder to be ignored. Requires the **--folder** option. The **--ignorefolder** action causes the folder specified by the **--folder** option to be ignored for all indexing operations, including bootstrap, real-time update events, **--checkaccount**, and **--sync** operations. The **--unignorefolder** action reverses the process, returning the named folder to normal operation. If the account is deleted from the index, any ignored folders are also deleted, and all ignore markings for folders of that account are also removed. The **--deletefolder** action does not affect any folder marked as ignored. The **--unignorefolder** action must be performed first. The ignored property applies to all nested folders. Only the highest-level folder must be marked and unmarked. Folders marked to be ignored are also ignored by any search queries on the account.
--import	Copies data for an account into the index from the current snapshot repository. Requires the **--user** and **--host** options. The account must already exist but be empty for this to work: use the **--createaccount** command to create the account in the desired group. The account must not be in the Active state, that is, it must be in the Inactive state. The snapshot may have been created from any store instance. Use this command to move an account from one store instance to another, and to recover from a backup snapshot. The **--importversion** *version* option enables you to select between multiple snapshot directories for the account which may be present in the snapshot repository. To be able to import the account while changing the host, you must use the **--rehost** *host* option. See the **--eximpath** option and the **--setdefaulteximpath** option for how to alter the snapshot repository. Also, see "Exporting and Importing Accounts Example".
--listbacklog	Lists to standard output the current event backlog for all accounts currently with one or more events queued. The output can be filtered using the **--user** and **--host** options, the **--accountlist** option, the **--selectstate** *statelist* option, and the **--selecttime** *h*[:*m*] option.

Table 19–9 (Cont.) Action Options

Option	Description
--lockmemoryindex, **--unlockmemoryindex**	Forces the directory index (dIndex) to be retained in memory to improve performance across multiple indexing commands (such as when the **--accountlist** option is used with the **--bootstrap** command). This property remains in effect until the index is released with the **--unlockmemoryindex** command, which returns the index to disk. (If the server is shutdown, the current memory index is written back to disk, but it is locked back into memory when the server is restarted.) While locked in memory, information is updated, and can be inspected (using **--list*** commands for example), but any changes are not available to other parts of the system, such as the search or IMQ services. To enable the system to be fully operational, the **--unlockmemoryindex** action must be performed. (These commands are intended to be useful during the bootstrap process when many threads may update the dIndex repeatedly in parallel with large amounts of data; use avoids large amounts of disk I/O in the rapidly changing data structures, which can become a bottleneck. It is not recommended to use this feature for general system operation, as updated data is not visible to other processes until the unlock.)
--moveaccount	Moves an account from one group to another. Requires the **--group**, **--host**, and **--user** options. The account must not be in the Active state. If **--singleton** is specified, the target group must not already contain another account, and the target group is marked as **--singleton**.
--refresh	Forces values for all refreshable parameters changed in the configuration file since the last **--refresh** command (or since the services were last restarted) to be applied to all running services. Changes affecting log file locations, count, and size occur after all previous log messages have been written for the modified parameters.
--renamefolder *newfoldername*	Renames a folder in an account to *newfoldername*. Requires the **--folder**, **--host**, and **--user** options. **--folder** specifies the old folder name. Any folders underneath the specified folder are also renamed.
--setdefaulteximpath *path*	Assigns the value for *path* to be the default snapshot repository for this store instance. Once set, this directory path name is used as the default value for **--import** and **--export** commands whenever **--eximpath** is not specified. The **--eximpath** option can be used on the command line to override this default. Like **--setdefaulthostname**, **--setdefaulteximpath** establishes the default snapshot repository directory for **--export** and **--import** to use. If never specified, the **iss.store.dir/snapshots** directory is used as the default. Unlike **--setdefaulthostname**, you can set this option more than once, and the last such setting becomes the default setting. You can use this option to remove the default so that it behaves as if no **--setdefaulteximpath** *path* was ever specified. To return the behavior to the original unspecified state, use this option specifying the empty string (**""**) as the *path* value.

Table 19–9 (Cont.) Action Options

Option	Description
--setdefaulthostname *host*	Assigns the value *host* to be the default host name for this store instance. This value can be set only once for any single store instance. After it is set, the value permits all commands requiring the **--host** option to be used without it. The value set as the default host name is automatically used whenever **--host** is not specified. You can use the **--host** option at the command line to override this default. You can use this option more than once. Each subsequent use of this option replaces the default value. To remove the default so that none is applied, use this option specifying the empty string (**""**) as the *host* value.
--setstate *newstate*	Sets the state of the account to *newstate*. The values for *newstate* are **A**, **B**, **C**, **I**, **0**, and **X**, which represent Active, Bootstrapping, Cleared, Inactive, Optimized and Unknown, respectively. Requires the **--host** and **--user** options. You must set the state of an Active account to Inactive to bootstrap any folder in the account. Real-time notification events and search queries are only processed for accounts in the Active state. Specifying **O** causes the group of the account to be Optimized. This setting also affects any other accounts in the group. (Optimization of the index improves search performance and does not usually need to be performed manually in this fashion.) An Optimized (**O**) state is a transient state. As soon as an account is modified by an event, then the account is no longer in the Optimized state. Specifying **C** causes the command lock for the account to be cleared. You usually only use **C** to clean up after catastrophic failures. Each of these states (except **C** and **O**) can appear in the output of the list options.
--stopservice *usid*	Specifies the unique service identifier (*usid*) for services to stop in the Index Server. Using the output from **--listactiveservice**, this command option stops threads in the Index Server which might have been left running when various **issadmin.sh** commands were interrupted. Any single *usid* from the **--listactiveservice** output can be stopped. Also, any prefix ending with a colon (**:**) can be used to stop all threads whose *usid* begins with such a prefix. When stopping threads, it might be necessary to repeat the **--listactiveservice** and **--stopservice** commands because new threads can continue to be created while these commands are executed. (After you finish using **--stopservice**, check any accounts being modified by the services you stopped, as the interrupted commands have not terminated normally. Interrupted commands that modify accounts might leave index locks in the meta or content index directories. Check for problems using a command such as **find** *basedir***/index/store/ -name "write.lock"** where *basedir* is the value of the **basedir** parameter in the configuration file. Any such **write.lock** files found should be deleted, and any accounts in such groups should be checked for inconsistencies, and, if corrupted, should be deleted and bootstrapped again.) All **write.lock** files are found in the *basedir***/index/store/locks** directory, not the individual group index directories. The name of each file in this directory contains the number of the group to which it belongs.

Table 19–10 describes modifier options.

Table 19–10 Modifier Options

Option	Description
--accountlist *file*	Specifies a file containing account information for one or more accounts used to repeat the command action over multiple accounts. You can execute commands such as **--createaccount** or **--deleteaccount** over a list of accounts, all within one executable command. (This is much more efficient than invoking **issadmin.sh** once for each account at the command line.) Each line of *file* specifies information for one account, minimally the user name. Any command-line option values found in the file override the corresponding value specified on the command line. The command is executed once for each line in the file, with the appropriate values in the line used instead of any from the command line. *file* can also contain empty, blank, and comment lines. Any line whose first non-white-space character is "#" or "@", or consists of only white space or carriage return, is ignored. Each line may contain several command option values. The format of a line in the file can be one of the following: *username* or *groupnum;state;singleton;detail;hostname;username* In the first format, only the user name appears on a line, and all the other attributes are defaulted. In the second format, each of the individual terms can appear. If a term is not specified, the separators are still required, and any missing terms are defaulted. The values for the various terms must be valid values that can appear on the command line. (For *singleton*, an empty entry means not a singleton and any string starting with the letter 's' means **--singleton**.) Consider the following example: `;101;I;;;;user30` It specifies the group, state, and user name to use for any commands for which those fields are meaningful. Likewise, the following example specifies the group, state, singleton, host name, and user name: `;104;A;single;;host4;user25` Thus if these lines appeared in file **/tmp/userslist**, then the following first command would be equivalent to the last two commands: `issadmin.sh --createaccount --accountlist /tmp/userslist` `issadmin.sh --createaccount --group 101 --user user30` `issadmin.sh --createaccount --group 104 --singleton --host host4 --user user25` Additionally, the following first command would be equivalent to the last two commands: `issadmin.sh --setstate X --accountlist /tmp/userslist` `issadmin.sh --setstate I --user user30` `issadmin.sh --setstate A --host host4 --user user25` Similarly, the "detail" field can be specified to override the value in the **--detail** option used in commands such as the following: `issadmin.sh --checkaccount --detail flags --accountlist /tmp/userslist` The following line is equivalent to a line containing just *username*: `;;;;;;username` For more information, see "Bootstrapping Examples."

Table 19–10 (Cont.) Modifier Options

Option	Description
--altoutput *file*	Specifies a file into which to write command output. The value of *file* must be the path to a file that either does not exist or is writable. Standard error and output from all commands is appended to *file*. This appending of standard error and output enables the results of long commands, such as those using **--accountlist** and **--bootstrap**, to be viewed before the command completes. If this option is not used, output is generated but does not appear on standard error or output until the command completes.
--continueonerror	Specifies that the **--accountlist** commands should continue running if one gets an error. By default, without this option, processing of the **--accountlist** stops if one of the commands produces an error.
--datapath *path*	Specifies the path of an alternative Indexing and Search Service data (that is, attachment) store to use. The default data store (from the configuration parameter **iss.data.dir**) is used unless this option is specified. The path specified includes the final directory, in the same fashion as is specified in the **iss.data.dir** parameter in the **jiss.conf** file. See also the **--storepath** option. Be careful when specifying **--datapath** because you might also need to use **--storepath** to obtain the results you want.
--detail *detaillevel*	Specifies that the data in the specific folder or account should be checked in detail. Used only with the **--checkstore**, **--checkfolder**, and **--checkaccount** options. If not specified, the account or folder is checked for the right number of emails in the folder only. Values for this option are **full**, **flags**, or **status**. The **--checkfolder** and **--checkaccount** options also take the **ignoreok** value. When used with these commands, **ignoreok** causes folders marked as "ignored" to be treated as normal, log messages do not appear, and the return code result is as if such folders did not exist. When used with the **--checkstore** option, additional values are **dindex**, **store**, **group deepcheck**, and **verbose**. The "full" option, when used with the **--checkstore** option, means to include **dindex**, **store**, **group deepcheck**, but not **verbose**. If **status** is specified, the status messages from the bootstrap process in the index are also displayed. These messages indicate when the index was incomplete such as if an attachment failed to convert. If **flags** is specified, the flag values of each message are checked to match the values in the Messaging Server store. If **full** is specified, every level of detailed checking is performed. Output reflects the differences found at a more detailed level. Detailed checking can be expensive to perform. Preliminary tests indicate performance of up to four times slower for **--checkaccount** when using **--detail full** on larger accounts.
--eximpath *path*	Specifies the path of an alternate snapshot repository for the **--export** and **--import** commands. Used only with **--export** or **--import** commands. If not specified, the default snapshot repository is used.
--folder *foldername*	Specifies the folder name of an account as needed by various actions.
--group *groupnum*	Specifies the integer group number of an account as needed by various actions.
--host *host*	Specifies the host (or domain) name of an account as needed by various actions.
--importversion *version*	Specifies which version of an account to use for the **--import** command from the snapshot repository. The value of *version* must match one of the valid "Created:" time stamps found in the summary file in the snapshot repository. Used only with **--import** to specify which version of the account to import from.
--loglevel [all | config | fine | finer | finest | info | severe | warning]	Specifies the log level to use for commands. Only applies when the Index Service is not running. Behavior in this case is similar to and overrides the **iss.indexsvc.log.level** parameter.
--password	Causes the command to prompt for a password. It is read from the command line as plain text.
--passwordfile *file*	Specifies the file containing the password of an account needed by **--checkfolder** and **--checkaccount** options. The password is read from *file*. It is treated as plain text.
--port *port*	Specifies the port of the Messaging Server host needed by **--checkfolder** and **--checkaccount** options. The value is a simple integer.

Table 19–10 (Cont.) Modifier Options

Option	Description
--prompt	Specifies that the **--checkstore --sync** command prompts interactively at the command line. The individual steps in the **--sync** command prompt for confirmation, enabling you to skip steps. By default, no prompting is performed.
--protocol *protocol*	Specifies the protocol to use when accessing the Messaging Server host as needed by **--checkfolder** and **--checkaccount** options. Values for this option are **ssl** and **tls**.
--reboot	Permits accounts in **--setautobootlist** *file* to be bootstrapped even when they already exist. This option causes the account to be deleted and completely bootstrapped again. (Refer to **--detail deepcheck** for example of use.)
--rehost *host*	Specifies the host name of a snapshot being imported when it is being changed. Using this option permits the **--import** command to change the name of the host from what is in the snapshot. When an account is exported using **--export**, the current host name is used to identify the snapshot. To import such a snapshot into an account in another store with a different host value, specify the host of the account as specified in the snapshot in this option. For example, if the snapshot was created for **--user** *U* **--host** *H*, and the new account is created for **--user** *U* **--host** *NEWH*, then the **--import** command needs to use **--user** *U* **--host** *NEWH* **--rehost** *H* options.
--runoptimizer [**true** | **false**]	Specifies if the account in the **--bootstrap** command should be optimized after indexing. If **true** is specified, the account is optimized. If **false** is specified, the account is not optimized. If this option is not specified with the **--bootstrap** command, the default is not to optimize the account. It takes somewhat longer to optimize an account after the bootstrap, but subsequent searches of the account are generally faster. See also the **--setstate O** option.
--setautobootlist *file*	Specifies a file containing information for accounts in **--accountlist** format. This command adds each entry in *file* to the autobootstrap list so it is bootstrapped by the periodic bootstrap process. Refer to the **--unsetautobootlist** *file* command: this option can be specified in the same command as the **--unsetautobootlist** *file* option, but only once and the two *file* specifiers must not then reference the same file. In this case, the **--setautobootlist** *file* changes are applied before the **--unsetautobootlist** *file* changes.
--selectstate *statelist*	Specifies a comma-delimited list of one or more account states from the set **A**, **B**, **I**, and **X**. When specified with the **--listbacklog** command with or without any **--user/- host** or **--accountlist** options, only those accounts found with more than zero events in the event backlog which match one of the specified states are included in the output. When specified with the **--listbrief** option, only users in the selected state are displayed.
--selecttime *h*[:*m*]	Specifies one or two integer values indicating the number of hours *h* (or hours *h* and minutes *m*) to use to select only those accounts which have had events backlogged for longer than that amount of time (and so would be technically out of sync during this interval). This can be used with the **--selectstate** option as well to further restrict the results. The hours and minutes values can each be zero, but not negative; the minutes value is optional. If the total time interval is zero, then no selection occurs, and all results from the **--listbacklog** command are returned.
--singleton	Specifies that the target group of the **--createaccount** or **--moveaccount** options must contain only one account.
--skipfolder *folder*	Specifies the folder name of an account which is skipped during the **--bootstrap** command. No data from *folder* and any of its descendants appears in the index for the account. The **--skipfolder** and **--folder** options cannot specify the same folder in the **--bootstrap** command.
--storepath *path*	Specifies the path of an alternative Indexing and Search Service store to use. The default store (from the configuration parameter **iss.store.dir**) is used unless this option is specified. The path specified does not include the final "/store", just the full path name up to it (such as **/var/iss/index**). See also **--datapath** option. Be careful when specifying **--storepath** because you might also need to use **--datapath** to obtain the results you want.

Table 19–10 (Cont.) Modifier Options

Option	Description
--sync	Specifies that the data in the specific folder or account should be synchronized with the Messaging Server store. Used only with the **--checkstore**, **--checkfolder**, and **--checkaccount** options. See **--checkstore**, **--checkfolder**, and **--checkaccount** descriptions in Table 19–8, " List Options" for more information. If the **--detail flags** modifier is used, then message flags are also checked, which causes additional synchronization to occur. (Reindexing to correct "incomplete" indexing caused by, for example, attachment conversion failures is not attempted, so such warnings remain a reason for the folder or account to be out of sync.)
--threads *n*	Specifies the number of threads to use when running with the **--accountlist** option. If not specified, the commands run sequentially in the order found in the **--accountlist** file. If specified, then commands are submitted to run concurrently in groups of *n* threads in the order found in the **--accountlist** file. However, the order that the commands complete is not fixed, so this feature should only be used when the commands in each line are independent of each other. (If two lines of the file specify actions on the same account, the order in which the actions are run is not deterministic.) The maximum permissible value for *n* is 100. Any error in any of the commands causes processing to halt unless the **--continueonerror** option is also specified.
--timeout *seconds*	This option is not currently implemented. The system recognizes it but the value is ignored.
--uid	Specifies a list of Unique Identifiers (as defined by IMAP) relative to the folder in the **--folder** modifier. Each value is an integer. Separate multiple values with commas. White space within the UID list is not allowed. This option can only be specified when using the **--deletefolder** option. This option causes only the records associated with each UID to be deleted.
--unsetautobootlist *file*	Specifies a file containing information for accounts in **--accountlist** format. This command removes each entry in *file* from the autobootstrap list so it is not bootstrapped by the periodic bootstrap process. (Unless other action, such as manual bootstrap, is taken, subsequent error events cause an account to be placed on the list again.) Refer to the **--setautobootlist** *file* command: this option can be specified in the same command as the **--setautobootlist** *file* option, but only once and the two *file* specifiers must not then reference the same file. In this case, the **--setautobootlist** *file* changes are applied before the **--unsetautobootlist** *file* changes.
--user *user*	Specifies the user name of an account as needed by various actions.
--useramdir	Specifies that the **--accountlist** option should use in-memory processing of dIndex to improve speed. This modifier causes all changes to the dIndex to be delayed from showing up in the rest of the system until the action using **--accountlist** completes. Only some actions that allow **--accountlist** are affected by this option: these are currently **--createaccount**, **--createfolder**, **--deleteaccount**, **--deletefolder**, **--setstate**, **--ignorefolder**, and **--unignorefolder**. Use caution when using this option. Using this option can cause loss of index data if processing is interrupted before the command completes. Use should be limited to when the **--accountlist** file is extremely large (tens of thousands of entries in the file) and when load is low on the system. You can also use this option when services other than indexing are not running, such as when performing maintenance on many accounts in the instance (for example, large scale initial account creation, modification of account state, account or folder addition or deletion, and so on).

Examples

- To list all groups in a store instance:

```
issadmin.sh --listgroups
Thu Aug 27 11:32:44 PDT 2009
 Store ID: ISSID_20090824223352 created: 20090824223352
 default host name:              <none specified>
 default export/import path:     <none specified>
 total size of store instance:   0
```

```
 total number of accounts:       54
 total number of account groups: 6
 last known consecutive group:   0
 total search queries performed: 0
 total search query failures:    0
 total index events processed:   434
 last backup performed:          never
 last host number:               0
 last user number:               0
 attachment store enabled:       true
 dIndex memory locked:           false
 Group #   # accounts
    101     10    meta index:  6.4M       content index:   18M
    102     10    meta index:   42M       content index:   94M
    103     10    meta index:   91K       content index:  7.5M
    104     10    meta index:   91K       content index:  299K
    105     10    meta index:   12M       content index:   28M
    106     4    meta index:  382M        content index:  697M
        total space in group indices: meta: 442.5777MB   content: 844.7919MB

 Time spent in index du:        107ms
 Time spent searching meta index:    0ms
 Time spent searching content index: 0ms
Thu Aug 27 11:32:46 PDT 2009
```

- To show all information about a given account:

```
issadmin.sh --accountinfo --user user1 --host mailhost.example.com
Thu Aug 27 11:35:41 PDT 2009
 Store ID: ISSID_20090824223352 created: 20090824223352
 default host name:              <none specified>
 default export/import path:     <none specified>
 total size of store instance:   0
 total number of accounts:       54
 total number of account groups: 6
 last known consecutive group:   0
 total search queries performed: 0
 total search query failures:    0
 total index events processed:   438
 last backup performed:          never
 last host number:               0
 last user number:               0
 attachment store enabled:       true
 dIndex memory locked:           false
  Group #   # accounts  status username     hostname   foldername
      104     10    meta index:   91K       content index:  299K
                         A      user1        mailhost.example.com
                          last state transition: 20090107125507
                                Account created: 20090107125451
                                  0                threeleadingspaces
                                  0              thirdemptyfolder
                                  0              test_cr6735149
                                  0              a16new
                                  0              secondemptyfolder
                                  16 at:7        INBOX
                                       5 emails have 36 attachment files, 8.3M
                                  1              simplename
                                  1              new_folder/a_new_folder
                                  3              simpleName
                                  3              user1_shared_folder
                                  2              abc(def) test folder
```

```
                                    3              new_folder/subfolder
                                    2              new_folder
                                    38 at:6        demonov19_2007
                                         1 email has 4 attachment files, 35K
                                    24 at:8        Sent
                                         7 emails have 52 attachment files, 10M
                                    1              Junk
                                    2              Trash
                                    1              Drafts
                                    97 at:21     total in account
                                         13 emails have 92 attachment files,
18.33417MB
          97 at:21       total in group 104
                      13 emails have 92 attachment files, 18.33417MB
          total space in group indices: meta: 0.088867MB   content: 0.291992MB

 Time spent in index du:        96ms
 Time spent searching meta index:    0ms
 Time spent searching content index: 4ms
Thu Aug 27 11:35:43 PDT 2009
```

The account status is specified in the line:

```
A user1 mailhost.example.com
```

In this case, the A specifies "active."

- To check that an account matches the content store, run the following two commands:

```
issadmin.sh --checkaccount --user user1 --host mailhost.example.com
Thu Jan  8 02:09:22 GMT 2015
Account user1 mailhost.example.com is in sync
Thu Jan  8 02:09:24 GMT 2015

issadmin.sh --checkaccount --user user2 --host mailhost.example.com
Thu Jan  8 02:10:11 GMT 2015
NOT SYNCHED folder: MSVisioAttachments number of emails match but found 1
indexing problem
    uid: 0000000001 incomplete: failed to index attachment content type:
APPLICATION/OCTET-STREAM attachment type: atdoc filename: assignment_1.vsd
Exception Message MSWord Attachment : Error Creating WordExtractor Object:
java.io.FileNotFoundException: no such entry: "WordDocument" NameOfAttachment :
assignment_1.vsd
        folder: MSVisioAttachments MS shows nMsgs: 1 uidVal: 1205273656
nextUID: 2 new: 0

Thu Jan  8 02:10:28 GMT 2015
```

- To bootstrap an account by using the **--accountlist** option:

```
issadmin --bootstrap --accountlist /tmp/users.conf --host mailhost.example.com
--threads 10
```

- To bootstrap an account by using the default allocation policy:

```
issadmin --bootstrap --user user2 --host mailhost.example.com --runoptimizer
true
```

- To bootstrap an account by allocating to a specific singleton group:

```
issadmin --bootstrap --user user2 --host mailhost.example.com --group group2
```

```
--singleton --runoptimizer true
```

isshttpdmgr

The **isshttpdmgr** utility starts and stops the **isshttpd** service, and configures Directory Server entries for the lookup table.

Location: *IndexSearch_home*/**isshttpd/scripts/isshttpdmgr**

Syntax

```
isshttpdmgr [-b basedir] [-a|-d] [-v] [-h] [-l] [-u file] [-s] [-S]
```

Options

Table 19–11 describes the options for the **isshttpdmgr** utility.

Table 19–11 isshttpdmgr Options

Option	Description
-b *basedir*	Specifies the Indexing and Search Service installation directory. The default is **/opt/sun/comms/jiss**.
-a	Enables **isshttpd**.
-d	Disables **isshttpd**.
-l	Lists current file or LDAP entries.
-u *file*	Sets the current LDAP to value in file.
-s	Configures but does not start **isshttpd** services.
-S	Does not update the **iss.isshttpd.enabled** parameter in the **jiss.conf** file.
-v	Enables verbose debugging.
-h	Prints out the usage summary.

Examples

- To list current entries:

```
isshttpdmgr -l

ms1.example.com=iss1.example.com,8080
ms2.example.com=iss2.example.com,8080
```

- To set LDAP entries:

```
cat inputfile
sunkeyvalue: ms1.example.com=iss1.example.com,8080
sunkeyvalue: ms2.example.com=iss2.example.com,8080

isshttpdmgr -u inputfile
```

- To enable the **isshttpd** service:

```
isshttpdmgr -a
```

- To disable the **isshttpd** service:

```
isshttpdmgr -d
```

issrehostuser.sh

The **issreshostuser.sh** utility automates the movement of accounts from one Indexing and Search Service instance to another. This command is intended to be invoked by the Messaging Server **rehostuser** command, not directly from the command line.

Location: *IndexSearch_home*/**store/scripts/issrehostuser.sh**

Syntax

```
issrehostuser.sh -a action -s srchost -d desthost -u username
                 -t destiss -o thisiss -c scpuser -p sshcmd
                 [-r true|false] [-x true|false] [-g groupnum]
                 [-n snapowner] [-e eximpath] [-w destpath]
```

Options

Table 19–12 describes the options for the **issrehostuser.sh** utility.

Table 19–12 issrehostuser.sh Options

Option	Description
-a *action*	Required. The action to perform. *action* can be one of **prep**, **fini**, or **isscleanup**.
-s *srchost*	Required. The Messaging Server host name from which the account is to be moved. Must match the host name of the account in the source Indexing and Search Service store instance.
-d *desthost*	Required. The Messaging Server host name to which the account is to be moved. This is the host name of the account in the destination Indexing and Search Service store instance.
-u *username*	Required. The user name of the account to be moved. Must match the user name of the account in the source Indexing and Search Service store instance.
-t *destiss*	Required. The Indexing and Search Service instance host name to which the account is to be moved.
-o *thisiss*	Required. The Indexing and Search Service instance host name from which the account is to be moved. This is the host on which the **issrehostuser.sh** command runs.
-c *scpuser*	Required. User name used by the secure copy **scp** command used to transfer data from the *thisiss* host to the *destiss* host.
-p *sshcmd*	Required. The secure shell command used on *thisiss* host for running remote commands on *destiss* host.
-r *copy*	Optional. *copy* can be either **true** or **false**. If **true**, the command uses secure copy (**scp**) to transfer data from the *thisiss* host to the *destiss* host. If **false**, the command assumes the two hosts share an NFS disk instead. Default is **true**. You should not need to ever specify **false**; this setting is intended primarily for testing purposes.
-x *delete*	Optional. *delete* can be either **true** or **false**. If **true**, the account is removed from the *thisiss* store after the rehost is complete. If **false**, the account is left on the *thisiss* machine after the rehost is complete. Default is **true**. You should not need to ever specify **false**; this setting is intended primarily for testing purposes.
-g *groupnum*	Optional. The number of the group into which the rehosted account is moved on *destiss*. Default is empty (the system determines what group to use).

Table 19–12 (Cont.) issrehostuser.sh Options

Option	Description
-n *snapowner*	Optional. Used to change owner of data copied from the *thisiss* host to the *destiss* host. Default is **/bin/chown -R ${iss_group}:${iss_user}**. You should not specify an alternate to this command.
-e *eximpath*	Optional. The export/import directory path used when creating the copy of the account being rehosted. Default is **iss.exim.dir** or, if that is not defined, **iss.tmp.dir**. You should not specify an alternate path.
-w *destpath*	Optional. Path to commands on *destpath*. The default is the same as the *thisiss* path (*base_dir*). If the instance on the *destpath* host is not installed in the same place as on the *thisiss* host, use this option to specify the correct path.
-h	Prints out the usage summary.
-v	Generates verbose debugging output.

Rehosting an Account from One Indexing and Search Service Instance to Another

Normally, you do not use the **issreshostuser.sh** utility directly from the command line. However, you can use it to rehost an account from one Indexing and Search Service instance to another.

To rehost an account from one Indexing and Search Service instance to another:

1. Run the **issreshostuser.sh** command on the Indexing and Search Service host containing the account to be rehosted, specifying the **-a prep** action.
2. Use the Messaging Server **rehostuser** command to move the account on the Messaging Server.
3. To complete the rehost, run the **issrehostuser.sh** command using the **-a fini** action.

If the rehost succeeds, **issreshostuser.sh** returns an exit code of 0 (zero). Otherwise, the return code varies depending on the failure. If either the **prep** or **fini** actions fail, run the **isscleanup** action to restore the account to its original state.

issversion

The **issversion** utility reports the version of Indexing and Search Service.

Syntax

```
issversion
```

Example

```
issversion
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
Indexing and Search Service for Oracle Communications Unified Communications Suite
1u5-26.13901(1.0.5.26.0) (built 20151221)
SunOS sc11152314 5.11 11.0 sun4v sparc sun4v
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
common.jar reports version: 1u5-26.13901
indexapi.jar reports version: 1u5-26.13901
jmqconsumer.jar reports version: 1u5-26.13901
search.jar reports version: 1u5-26.13901
store.jar reports version: 1u5-26.13901
auth.jar reports version: 1u5-26.13901
```

```
shared.jar reports version: 1u5-26.13901
watcher.jar reports version: 1u5-26.13901
isshttpd.jar reports version: 1u5-26.13901
utilsvc.jar reports version: 1u5-26.13901
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
Checking deployed wars:
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
rest reports version: 1u5-26.13901
    common.jar in rest reports version: 1u5-26.13901
    indexapi.jar in rest reports version: 1u5-26.13901
    search.jar in rest reports version: 1u5-26.13901
    store.jar in rest reports version: 1u5-26.13901
    auth.jar in rest reports version: 1u5-26.13901
    shared.jar in rest reports version: 1u5-26.13901
searchui reports version: 1u5-26.13901
storeui reports version: 1u5-26.13901
    common.jar in storeui reports version: 1u5-26.13901
    indexapi.jar in storeui reports version: 1u5-26.13901
    store.jar in storeui reports version: 1u5-26.13901
    auth.jar in storeui reports version: 1u5-26.13901
    shared.jar in storeui reports version: 1u5-26.13901
```

mergeIndex.sh

The **mergeIndex.sh** utility merges multiple index files. You must supply at least three directories to **mergeIndex.sh**. The first directory is the result index, and the rest are input index directories to be merged. This utility implements the **--moveaccount** option. It is provided as a standalone utility that might be useful for recovering corrupted accounts under special situations. This utility is not generally needed, but is used during the export, import, back up, and recovery of index data. Do not attempt using the **mergeIndex.sh** utility without a thorough understanding of Lucene concepts and how data is indexed into documents. You can use this utility whether the Indexing and Search Service services are running or not.

Syntax

```
mergeIndex.sh mergedIndex index1 index2 [index3] ...
```

Options

Table 19–13 describes the options for the **mergeIndex.sh** utility.

Table 19–13 mergeIndex.sh Options

Option	Description
mergedIndex	Required. The resulting index directory from the merge.
index1 index2 index3...	At least two index directories are required. The index directories are merged in the order presented into the *mergedIndex* directory.

Exit code 1 indicates a a problem parsing the options. Exit code 0 indicates no issues.

search_query_number.sh

The **search_query_number.sh** utility reports the number of search queries that have been run since the last time that the **searchSvc** service was restarted.

Syntax

```
search_query_number.sh [--timeout int] [-h|--help]
```

Options

Table 19–14 describes the options for the **search_query_number.sh** utility.

Table 19–14 search_query_number.sh Options

Option	Description
--timeout *int*	Specifies the timeout for send and receive of JMQ messages.
-h | **--help**	Prints out the usage summary.

Example

```
search_query_number.sh
Current search query number is: 560
```

searchRun.sh

The **searchRun.sh** utility performs searches on the index from the command line interactively. The utility prompts for input and displays the results of the query to standard output.

Requirements: The search services must be running to use this utility.

Syntax

```
searchRun.sh [--promptcallback] [--promptcompresssize] [--promptcontentformat]
             [--promptcount] [--promptformat] [--promptsize]
             [--promptsort] [--promptstart]
             [--loglevel all|config|fine|finer|finest|info|severe|warning]
             [--nojmssvc] [--queryfile file] [--storepath path]
```

Options

Table 19–15 describes the options for the **searchRun.sh** utility. Each of the following options can appear once in a command, in any order:

Table 19–15 searchRun.sh Options

Option	Description
--promptsort	Causes the script to prompt for sort criteria. Form of response to prompt is the same as the **SORT=** parameter of RESTful search services commands.
--promptcount	Causes the script to prompt for the number of results to return. Form of response to prompt is an integer.
--promptstart	Causes the script to prompt for the start number of the results. Form of response to prompt is an integer.
--promptformat	Causes the script to prompt for the format of the results. Form of response to prompt is one of the choices the prompt presents.
--promptcompresssize	Causes the script to prompt for the compression size of the results. Form of response to prompt is an integer indicating how large the search results should be to cause compression.
--promptcontentformat	Causes the script to prompt for the content format of the results. Form of response to prompt is one of the choices the prompt presents.

Table 19–15 (Cont.) searchRun.sh Options

Option	Description
--promptsize	Causes the script to prompt for the size of the thumbnail to use when formatting the results. Form of response to prompt is one of the choices the prompt presents.
--promptcallback	Causes the script to prompt for the JavaScript callback to use in the JSON format response. If none is provided, the response is not wrapped in a function.
--loglevel all | **config** | **fine** | **finer** | **finest** | **info** | **severe** | **warning**	Specifies the level of logging information to be generated by the search process.
--nojmssvc	Required if the command is run on a cluster node or when the services are not running.
--queryfile *file*	Alternate input for search query requests. Each line in the file *file* is a single query, terminated with a newline. Enables search queries to exceed the limit of 255 characters on the command line.
--storepath *path*	Specifies the path of an alternative Indexing and Search Service store to use. The default store (from the configuration parameter **iss.store.dir**) is used unless this option is specified. The path specified does not include the final "**/store**", just the full path name up to it (such as **/var/iss/index**).

svc_control.sh

The **svc_control.sh** utility starts and stops Indexing and Search Service services.

Syntax

```
svc_control.sh start|stop
```

watchermgr.sh

The **watchermgr.sh** utility enables or disables the watcher service outside of the standard Indexing and Search Service **setup** script.

Syntax

```
watchermgr.sh [-a] [-d] [-b basedir] [-S] [-s] [-v] [-h]
```

Table 19–16 describes the options for the **watermgr.sh** utility.

Table 19–16 watchermgr.sh Options

Option	Description
-a	Configures the watcher service.
-d	Unconfigures the watcher service.
-b	Sets the base directory of the Indexing and Search Service installation (default is **/opt/sun/comms/jiss**).
-S	Does not update the **iss.watcher.enabled** parameter in the **jiss.conf** file.
-s	Configures but does not start the watcher service.
-v	Enables verbose debugging for the watcher service.
-h	Prints out the usage summary.

Examples

- To configure the watcher service:

```
watchermgr.sh -a
No basedir defined defaulting to /opt/sun/comms/jiss
Running configure watcher
Starting watcher
```

- To unconfigure the watcher service:

```
watchermgr.sh -d
No basedir defined defaulting to /opt/sun/comms/jiss
Running unconfigure watcher
Stopping watcher
```

Additional Indexing and Search Service Scripts

Indexing and Search Service provides the following scripts, based on Lucene, for manipulating individual index directories within the store instance. To use these scripts, you need an understanding of Lucene and how the data is indexed into documents.

Table 19–17 describes the additional Indexing and Search Service scripts.

Table 19–17 Additional Indexing and Search Service Scripts

Tool	Description
lucli.sh	Command-line Lucene index inspection script. Invoke this script and use the help command to inspect the individual index directories under the store.
luke.sh	GUI version of the Lucene Index Toolbox (Luke). Invoke this script and use the Help menu for commands and features to inspect the individual index directories in a store instance. Requires separate download of various **.jar** files. The script indicates where to find any missing files.

Deprecated Commands

This section lists the Indexing and Search Service commands that have been deprecated.

indexSvcBootstrap.sh

Beginning with **Indexing and Search Service 1 Update 2**, the **indexSvcBootstrap.sh** command is deprecated. Use **issadmin.sh --bootstrap** instead.

The **indexSvcBootstrap.sh** utility is used to create the initial index for an account, known as bootstrapping the account. For accounts which exist on Messaging Server prior to installing Indexing and Search Services, the data in each account must be initially indexed before the search can be used. This command is used to extract data from the content server (by using the standard IMAP interface) and create the index for each account. (Depending on how many accounts and how much data (email) each has, this can take considerable time. This command is multithreaded to speed up this process; however, it also might produce significant load on the Messaging Server IMAP service, because it must access every email in each account.)

Requirements: Must be **root** or the Indexing and Search Service user. The content server must be configured to enable access to each account by using the IMAP

interface. An account may or may not exist in the index when you use this command. If the account already exists, then it must not be already in the Active state. The index must not contain any records for any folders to be bootstrapped by this command, or duplicate index records may be created.

Location: *IndexSearch_home*/**indexapi/scripts/indexSvcBootstrap.sh**

Indexing and Search Service 1 Update 1: Also in *IndexSearch_home*/**bin/indexSvcBootstrap.sh**

Syntax

```
indexSvcBootstrap.sh --host host --user user [--datapath pathname]
 [--folder folder] [--group groupnum]
 [--loglevel all|config|fine|finer|finest|info|server|warning] [--password]
 [--passwordfile file] [--port port] [--protocol ssl|tls]
 [--runoptimizer true|false] [--singleton] [--skipfolder skipfolder]
 [--storepath pathname]
```

Options

Table 19–18 describes the options for the **indexSvcBootstrap.sh** command.

Table 19–18 indexSvcBootstrap.sh Options

Option	Description
--host *host*	Required. Specifies the host or domain of the account to be indexed.
--user *user*	Required. Specifies the user name of the account to be indexed.
--datapath *pathname*	Optional. Specifies the path of an alternative Indexing and Search Service data (that is, attachment) store to use. The default data store (from the configuration parameter **iss.data.dir**) is used unless this option is specified. The path specified includes the final directory, in the same fashion as is specified in the **iss.data.dir** parameter. See also **--storepath** option. Be careful when specifying **--datapath** because you might also need to use **--storepath** to obtain the results you want.
--folder *folder*	Optional. Specifies to index only this one folder of the account. This includes any nested folders.
--group *groupnum*	Optional. Specifies the integer group number into which the account index will be placed. If not specified, the default allocation policy will select a group into which to place the account. For more information on default allocation policy, see "Overview of the Indexing and Search Service Store Instance".
--loglevel all | config | fine | finer | finest | info | severe | warning	Optional. Specifies the level of logging information to be generated by the bootstrapping process.
--password	Optional. For development only. It is not meant for production use because it puts the password in the command-line option, which is not secure.
--passwordfile *file*	Plaintext. If **--password** option is provided, you are prompted for the password.
--port *port*	Optional. Used to specify the port for the IMAP connection to the Messaging Server host. Default is 143.
--protocol sls | tls	Optional. Used to specify the security protocol used by the IMAP connection to the Messaging Server host. By default, no security protocol is assumed to be needed.

Table 19–18 (Cont.) indexSvcBootstrap.sh Options

Option	Description
--runoptimizer true | **false**	Optional. Used to indicate whether the index being created should be optimized at the conclusion of the bootstrapping. Optimization makes the process take a little longer, but the additional time is usually not significant relative to that of the rest of the processing. However, search performance of an optimized index may be significantly better than the unoptimized index.
--singleton	Optional. Specifies that the target group of the indexing must contain only the one account being bootstrapped.
--skipfolder *skipfolder*	Optional. Specifies that the folder named *skipfolder* in the account is not to be indexed during this bootstrap process. This is sometimes useful if there is some problem with a certain folder or if it contains a huge number of emails such that processing it would take an inordinate amount of time.
--storepath *pathname*	Optional. Specifies the path of an alternative Indexing and Search Service store to use. The default store (from the configuration parameter **iss.store.dir**) is used unless this option is specified. The path specified does not include the final "/store", just the full path name up to it (such as **/var/iss/index**). See also **--datapath** option. Be careful when specifying **--storepath** because you might also need to use **--datapath** to obtain the results you want.

Options can appear in any order.

Example

```
indexSvcBootstrap.sh --user user1 --host mailhost.example.com --runoptimizer true
```

indexSvcFork.pl

In **Indexing and Search Service 1 Update 1**, **indexSvcFork.sh** replaces **indexSvcFork.pl**.

The **indexSvcFork.pl** utility provides a means to bootstrap a list of users in parallel.

Requirements: Must be run as **root** or the Indexing and Search Service user.

Location: *IndexSearch_home*/**indexapi/scripts/indexSvcFork.pl**

Syntax

```
indexSvcFork.pl [-f filename ] [-m mailhost] [-n max_number_of_spawns]
                [ -p polling_interval] [-t timeout]
```

Options

Table 19–19 describes the options for the **indexSvcFork.pl** command.

Table 19–19 indexSvcFork.pl Options

Option	Description
-f *filename*	Required. File name containing list of users to index, single user name per line.
-m *mailhost*	Optional. Mail host of the users to be indexed. Defaults to **mail.server** in the **jiss.conf** file.

Table 19–19 (Cont.) indexSvcFork.pl Options

Option	Description
-n *max_number_of_spawns*	Optional. Number of **indexSvcBootstrap.sh** calls to create simultaneously. Defaults to 5.
-p *polling_interval*	Optional. How often processes should be polled for completion (seconds). Defaults to 1 second.
-t *timeout*	Optional. How long an **indexSvcBootstrap.sh** should run before timing out (seconds). Defaults to 3600 seconds.

Example

```
indexSvcFork.pl -f users.conf -m mailhost.example.com -n 10
```

indexSvcFork.sh

Beginning with **Indexing and Search Service 1 Update 2**, the **indexSvcFork.sh** command is deprecated. Use **issadmin.sh --bootstrap** with the **-accountlist** option instead.

This command was introduced with **Indexing and Search Service 1 Update 1**.

The **indexSvcFork.sh** utility provides a means to bootstrap a list of users in parallel.

Requirements: Must be run as **root** or the Indexing and Search Service user.

Location: *IndexSearch_home***/indexapi/scripts/indexSvcFork.sh** (or *IndexSearch_home***/bin/indexSvcFork.sh**)

Syntax

```
indexSvcFork.sh  --file file  [--host hostname] [--threads num_threads]
[--runoptimizer true|false] [--timeout num_seconds]
[--port port] [--protocol ssl|tls]
```

Options

Table 19–20 describes the options for the **indexSvcFork.sh** command.

Table 19–20 indexSvcFork.sh Options

Option	Description
--file *filename*	Required. File name containing list of users to index. The file format is the same as for the **--accountlist** option of the **issadmin.sh** utility.
--host *mailhost*	Optional. Mail host of the users to be indexed, if not specified in the file.
--threads *num_threads*	Optional. Number of bootstraps to run in parallel. Defaults to 8.
--runoptimizer true | **false**	Optional. Whether or not to optimize the account. Defaults to **false**.
--timeout *num_seconds*	Optional. How long in seconds a bootstrap should run before timing out. Defaults to 50000 seconds.
--port *port*	Optional. Used to specify the port for the IMAP connection to the Messaging Server host. Default is **mail.imap.port** in the **jiss.conf** file.
--protocol ssl | **tls**	Optional. Use specified type of encryption for IMAP instead of the default in the **jiss.conf** file.

Example

```
indexSvcFork.sh --file /tmp/users.conf --host mailhost.example.com --threads 10
```

20

Indexing and Search Service Configuration Parameters

This chapter describes the Oracle Communications Indexing and Search Service configuration parameters found in the **jiss.conf** file and the Java keystore.

Local Installation Configuration

Table 20–1 describes local installation configuration parameters.

Table 20–1 Local Installation Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
hostname	NA	Fully qualified domain name of this Indexing and Search Service system, for example, **isshost.example.com**.
instance.name	NA	Instance name of the installation for an indexing node, for example, **iss1**.
basedir	**/opt/sun/comms/jiss**	The base installation directory for Indexing and Search Service.
iss.user	**jiss**	User under which all Indexing and Search Service services run.
iss.group	**jiss**	Group under which all Indexing and Search Service services run.
iss.store.dir	**/var/opt/sun/comms/jiss/index**	Location to store the Lucene indexes.
iss.data.dir	**/var/opt/sun/comms/jiss/attach**	Location of attachment data.
iss.tmp.dir	**/var/tmp/iss**	Temporary directory used for processing attachments.
iss.boottmp.dir	**/var/tmp/bootstrap**	Temporary directory used for processing account indexing during bootstrap.

Table 20–1 (Cont.) Local Installation Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.boottmp.enabled	The default depends on how the **iss.store.dir** directory is mounted.	Enables the use of the directory specified by the **iss.boottmp.dir** parameter to improve indexing performance. When the value is **false**, the **iss.boottmp.dir** parameter is ignored, and indexing uses the index store directly. If not explicitly specified in the **jiss.conf** file, the **iss.boottmp.dir** parameter is used if the **iss.store.dir** directory is mounted through NFS. Otherwise, it is not.
iss.log.dir	**/var/opt/sun/comms/jiss/logs**	Location of Indexing and Search Service log files.
iss.exim.dir	**/var/opt/sun/comms/jiss/snapshots**	Location of account snapshots for export or import.
java.home	**/usr/jdk/latest**	Base directory for Java.
java.args.indexsvc	**-XX:PermSize=100m -XX:MaxPermSize=100m -XX:+CMSClassUnloadingEnabled -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled -XX:ParallelGCThreads=8 -XX:+UseCMSInitiatingOccupancyOnly -XX:CMSInitiatingOccupancyFraction=50 -XX:+CMSScavengeBeforeRemark -XX:SurvivorRatio=6 -XX:MaxTenuringThreshold=15 -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution -Xloggc:/tmp/iss-indexsvc.gclog**	Java Virtual Machine options for the Index Service process.
java.args.searchsvc	**-XX:PermSize=100m -XX:MaxPermSize=100m -XX:+CMSClassUnloadingEnabled -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled -XX:+UseCMSInitiatingOccupancyOnly -XX:CMSInitiatingOccupancyFraction=50 -XX:+CMSScavengeBeforeRemark -XX:ParallelGCThreads=**8 **-XX:SurvivorRatio=4 -XX:MaxTenuringThreshold=15 -Xloggc:/tmp/iss-searchsvc.gclog -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution**	Java Virtual Machine options for the Search Service process.

Table 20–1 (Cont.) Local Installation Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
java.args.jmqconsumer	**-XX:PermSize=100m -XX:MaxPermSize=100m -XX:+CMSClassUnloadingEnabled -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled -XX:+UseCMSInitiatingOccupancyOnly -XX:CMSInitiatingOccupancyFraction=50 -XX:+CMSScavengeBeforeRemark -XX:ParallelGCThreads=8 -XX:SurvivorRatio=4 -XX:MaxTenuringThreshold=15 -Xloggc:/tmp/iss-jmqconsumer.gclog -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution**	Java Virtual Machine options for the JMQ Consumer process.
java.args.utilsvc	**-Xmx128m**	Java Virtual Machine options for the **utilSvc** process.
iss.indexsvc.maxfds	**65536**	Maximum number of open file descriptors for the **indexSvc** process.
iss.searchsvc.maxfds	**65536**	Maximum number of open file descriptors for the **searchSvc** process.
appserv.issuser.enabled	**false**	Set to **true** to run non-root Oracle Communications GlassFish Server user.

Message Store Configuration

Table 20–2 describes message store configuration parameters.

Table 20–2 Message Store Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
mail.ldap	None	User/Group Directory URL, in format *hostname*:**389**. For example: **host.example.com:389,host2.example.com:389**
mail.basedn	None	User/Group Directory Base DN. For example: **dc=example,dc=com**
mail.schemalevel	**2**	Schema level (1 or 2).
mail.dcroot	**o=internet**	DC Tree Base DN (only required for LDAP schema 1).
mail.loginseparator	**@**	Login separator.
mail.proxyseparator	**;**	Proxy separator.
mail.ugfilter	**(uid=%U)**	User/Group filter.
mail.defaultdomain	None	User/Group default domain.
mail.searchbind	**cn=Directory manager**	User/Group Directory Manager DN.
mail.searchbind.password	None	User/Group Directory Manager DN password.

Table 20–2 (Cont.) Message Store Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
mail.ldap.minpool	**1**	Minimum number of connections in User/Group LDAP pool.
mail.ldap.maxpool	**20**	Maximum number of connections in User/Group LDAP pool.
mail.ldap.timeout	**60**	LDAP operation timeout in seconds.
mail.ldap.monitoringinterval	**60**	Monitoring interval (in seconds) for LDAP pool when the LDAP server is down.
mail.ldap.refreshinterval	**60**	Time interval (in seconds) after which connection in LDAP pool is recreated. **0** means that no refresh is required.
mail.ldap.enablessl	**false**	Whether LDAP SSL is enabled.
mail.ldap.port	**389**	Default LDAP port, overwritten if LDAP host is specified in *host*:*port* format.
mail.server	None	Messaging Server store fully qualified host name.
mail.server.ip	None	Comma-delimited list of mail server IPs corresponding to **mail.server**. For example: **10.0.0.1,10.0.0.2**
mail.imap.admin.username	**indexeradmin**	Messaging Server read-only store administrator user name (**store.indexeradminsconfigutil** parameter). This user also has read-only administrative access to Indexing and Search Service web services.
mail.imap.admin.password	None	Messaging Server read-only store administrator password.
mail.imap.port	**143**	Messaging Server IMAP port number.
mail.imap.protocol	**ssl**	Specifies which IMAP protocol to use in URLs of search result output. Value can be either **ssl** or **tls**. A value of **ssl** means use IMAPS protocol. A value of **tls** means use IMAP protocol. The default is IMAP.
mail.ldap.timeout	**60**	LDAP operation timeout in seconds.
mail.imq	None	IMQ broker *hostname*:*port* for Messaging Server JMQ notifications. For example: **mailhost.example.com:7676**.
mail.imq.type	**queue**	Messaging Server JMQ notifications destination type (**queue** or **topic**) (must match **local.store.notifyplugin.index.destinationtype** in **configutil** command).

Table 20–2 (Cont.) Message Store Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
mail.imq.name	**INDEXMS**	Messaging Server JMQ notifications destination name (must match **local.store.notifyplugin.index.jmqtopic** in **configutil** command.)
mail.imq.user	**jesuser**	User name for Messaging Server JMQ notifications (must match **local.store.notifyplugin.index.jmquser** in **configutil**).
mail.imq.password	None	Password for Messaging Server JMQ notifications (must match **local.store.notifyplugin.index.jmqpwd** in **configutil** command.)

Message Queue Configuration

Table 20–3 describes Message Queue configuration parameters.

Table 20–3 Message Queue Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
imq.enabled	**false**	Enables or disables the JMQ broker.
imq.host	None	Comma-delimited list of Indexing and Search Service IMQ broker fully qualified host names. For example: **host1.example.com:7676,host2.example.com:7676**
iss.imq.user	**jmquser**	User name for Indexing and Search Service IMQ broker.
iss.imq.password	None	Password for Indexing and Search Service IMQ broker.
iss.imq.admin.password	None	Password for Indexing and Search Service IMQ broker administrator.
iss.imq.delivery.mode	**NON_PERSISTENT**	Type of IMQ messages Indexing and Search Service should send, **PERSISTENT** or **NON_PERSISTENT**.
iss.imq.bin	None	Specifies the full path to the **imqcmd** and **imqusermgr** binaries on GlassFish Server.
iss.imq.var	None	Specifies the full path to the IMQ broker **var** directory on GlassFish Server (**IMQ_DEFAULT_VARHOME**).

Directory Server Configuration for Java Naming and Directory Interface

Table 20–4 describes parameters for Directory Server configuration for Java Naming and Directory Interface (JNDI).

Table 20–4 Directory Server Configuration Parameters for JNDI

Parameter	Initial Setting in jiss.conf.template File	Description
ldap.enabled	**false**	Indexing and Search Service Directory Server host enabled. In a multiple host Indexing and Search Service deployment, defines the system on which the LDAP configuration is applied. With all Indexing and Search Service components deployed on a single host, this parameter should be **true**. Because Indexing and Search Service has LDAP configuration data that must be applied to the LDAP host, this parameter controls which Indexing and Search Service host applies that data. This parameter does not control whether a local LDAP instance is running on the Indexing and Search Service host.
jndi.type	**ldap**	Specifies either file-based or Directory Server-based JNDI lookups (**file** or **ldap**).
ldap.host	None	Indexing and Search Service Directory Server fully qualified host name. Supports a comma-delimited list. For example: **dshost1.example.com:389,dshost2.example.com:389**
ldap.password	None	Indexing and Search Service-installed Directory Server password.
java.naming.factory.initial	**com.sun.jndi.ldap.LdapCtxFactory**	Indexing and Search Service Java naming initial context factory.
java.naming.security.authentication	**simple**	Indexing and Search Service Directory Server authentication type.

GlassFish Server Configuration

Table 20–5 describes the Oracle GlassFish Server configuration parameters.

Table 20–5 GlassFish Server Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
appserv.enabled	**false**	Enables or disables GlassFish Server this host.
appserv.dir	**/opt/glassfish3/glassfish**	Base installation directory for GlassFish Server.
appserv.web.port	**8080**	GlassFish Server port number.
appserv.admin.port	**4848**	GlassFish Server administrative port number.
appserv.domain	**domain1**	GlassFish Server domain name.
appserv.domain.dir	NA	GlassFish Server domain directory if different from default.
appserv.admin.user	**admin**	GlassFish Server administrative user name.

Table 20–5 (Cont.) GlassFish Server Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
appserv.admin.password	None	GlassFish Server administrative password.
appserv.admin.passfile	**/var/opt/software/.domain1.as adminpass**	GlassFish Server password file for operation.
appserver.issuser.enabled	**false**	Enables or disables using **iss.user** and **iss.group** to manage services by using the **svcadm** command.
apache.port	**81**	Apache port (multiple host deployment).
appserv.searchui.enabled	**false**	Deploy the SearchUI web application used for testing and debugging.

Indexing and Search Service Services Run Time Configuration

Table 20–6 describes Indexing and Search Service services run time configuration parameters.

Table 20–6 Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.services.enabled	**false**	Enables or disables Indexing and Search Service services enabled on this host.
iss.searchsvc.log.level	**WARNING**	Search Service logging level (one of **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**).
iss.searchsvc.logfile.name	**iss-searchsvc.log**	Search Service log file name.
iss.searchsvc.logfile.limit	**20000000**	Maximum Search Service log file size in bytes.
iss.searchsvc.logfile.count	**20**	Maximum number of Search Service log file rotation files.
iss.searchsvc.logfile.formatter	**com.sun.comms.iss.common. SingleLineFormatter**	Search Service logging formatter.
iss.searchsvc.stats.log.enabled	**true**	Alternate search service log for statistics enabled.
iss.searchsvc.stats.log.level	**INFO**	Search Service statistics logging level (one of **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**).
iss.searchsvc.stats.logfile.name	**iss-searchsvc-stats.log**	Search Service statistics log file name.
iss.searchsvc.stats.logfile.limit	**20000000**	Maximum Search Service statistics log file size in bytes.
iss.searchsvc.stats.logfile.count	**20**	Maximum number of Search Service statistics log file rotation files.
iss.searchsvc.stats.logfile.formatter	**com.sun.comms.iss.common. SingleLineFormatter**	Search Service statistics logging formatter.

Table 20–6 (Cont.) Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.indexsvc.log.level	**WARNING**	Index Service logging level (one of **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**).
iss.indexsvc.logfile.name	**iss-indexsvc.log**	Index Service log file name.
iss.indexsvc.logfile.limit	**20000000**	Maximum Index Service log file size in bytes.
iss.indexsvc.logfile.count	**100**	Maximum number of Index Service log file rotation files.
iss.indexsvc.logfile.formatter	**com.sun.comms.iss.common.SingleLineFormatter**	Index Service logging formatter.
iss.indexsvc.periodic.autosync.checkstore.enabled	**true**	When the value of this parameter is **true**, the **--checkstore** checking for duplicate account, folder, group, and partition documents is performed as the initial document cache is created during startup of IndexService. Also, Warning messages are logged if the value of **iss.indexsvc.periodic.autosync.enabled** is **true**. Depending on the size of the index, the writing of log messages might delay the start of **indexSvc**
iss.indexsvc.periodic.autosync.checkstore.sync.enabled	**true**	When the value of this parameter is **true**, then problems detected during the **--checkstore** checking during initial document cache creation are be corrected, if the values of **iss.indexsvc.periodic.autosync.enabled** and **iss.indexsvc.periodic.autosync.checkstore.enabled** are both **true**. Also, during autosync syncing for each account, the meta and content index directories are fixed for any account found to need correction.
iss.indexsvc.segmentmerge.factor	**10**	Enables more segments to be generated before a merge occurs, which should reduce overhead when the bootstrap process is generating a lot of new index data using several simultaneous threads.
iss.indexsvc.segmentmerge.factor.autoadjust.enabled	**false**	When the value of this parameter is **false**, the value from **iss.indexsvc.segmentmerge.factor** is used. When the value of this parameter is **true**, the value of the segment merge factor used during bootstrap is adjusted to differ from the value in **iss.indexsvc.segmentmerge.factor**, based on the size of the content being bootstrapped. Use this feature when you want to dynamically adjusting the value based on the data found in the account.

Table 20–6 (Cont.) Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.indexsvc.stats.log.enabled	**true**	Alternate index service log for statistics enabled.
iss.indexsvc.stats.log.level	**INFO**	Index Service statistics logging level (one of **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**).
iss.indexsvc.stats.logfile.name	**iss-indexsvc-stats.log**	Index Service statistics log file name
iss.indexsvc.stats.logfile.limit	**20000000**	Maximum Index Service statistics log file size in bytes.
iss.indexsvc.stats.logfile.count	**20**	Maximum number of Index Service statistics log file rotation files.
iss.indexsvc.stats.logfile.formatter	**com.sun.comms.iss.common.SingleLineFormatter**	Index Service statistics logging formatter.
iss.jmqconsumer.log.level	**WARNING**	JMQConsumer logging level (one of **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**).
iss.jmqconsumer.logfile.name	**iss-jmqconsumer.log**	JMQConsumer log file name.
iss.jmqconsumer.logfile.limit	**20000000**	Maximum JMQConsumer log file size in bytes.
iss.jmqconsumer.logfile.count	**100**	Maximum number of JMQConsumer log file rotation files.
iss.jmqconsumer.logfile.formatter	**com.sun.comms.iss.common.SingleLineFormatter**	JMQConsumer logging formatter.
iss.jmqconsumer.stats.log.enabled	**true**	Alternate JMQConsumer service log for statistics enabled.
iss.jmqconsumer.stats.log.level	**INFO**	JMQConsumer statistics logging level (one of **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**).
iss.jmqconsumer.stats.logfile.name	**iss-jmqconsumer-stats.log**	JMQConsumer statistics log file name.
iss.jmqconsumer.stats.logfile.limit	**20000000**	Maximum JMQConsumer statistics log file size in bytes.
iss.jmqconsumer.stats.logfile.count	**20**	Maximum number of JMQConsumer statistics log file rotation files.
iss.jmqconsumer.stats.logfile.formatter	**com.sun.comms.iss.common.SingleLineFormatter**	JMQConsumer statistics logging formatter.
iss.jmqconsumer.useless.event.skip.enabled	**true**	Checks event queue for useless event sequences before sending the first event of the account to IndexSvc.
iss.utilsvc.log.level	**WARNING**	Util Service logging level (one of **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**).
iss.utilsvc.logfile.name	**iss-utilsvc.log**	Util Service log file name.
iss.utilsvc.logfile.limit	**20000000**	Maximum Util Service log file size in bytes.

Table 20–6 (Cont.) Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.utilsvc.logfile.count	**20**	Maximum number of Util Service log file rotation files.
iss.utilsvc.logfile.formatter	**com.sun.comms.iss.common.SingleLineFormatter**	Util Service logging formatter.
iss.utilsvc.port	**5558**	Util Service communication port.
iss.utilsvc.thread.count	**25**	Util Service request thread count.
iss.svc.response.timeout	**60000**	Default timeout for all services in milliseconds.
iss.uid.cache.time	**3600**	Time (in seconds) that a login ID or UID is kept in cache.
iss.searchsvc.response.timeout	**10000**	Search Service response timeout in milliseconds.
iss.searchsvc.host.inactive.list	**null**	Specifies a comma-delimited list of regular expression patterns used to match host names of accounts to be treated as Inactive for purposes of search only. Treating host names as Inactive for search only causes all search requests for any account whose host name matches to return an error regardless of the state of the account in the index. Example: **.+**
iss.searchsvc.user.inactive.list	**null**	Specifies a comma-delimited list of regular expression patterns used to match user names of accounts to be treated as Inactive for purposes of search only. Treating accounts as Inactive for search only causes all search requests for any account whose user name matches to return an error regardless of the state of the account in the index. Example: **.+**
iss.searchsvc.statisticsinterval	**1800**	Time interval (in seconds) after which statistics are periodically logged to search service log file.
iss.indexsvc.indexthread.count	**768**	Index Service request thread count.
iss.indexsvc.checksyncthread.count	**25**	Check/Sync command thread count.
iss.indexsvc.folderthread.count	**5**	Number of folders to bootstrap in parallel per user.
iss.indexsvc.sync.folderthread.count	**10**	Number of folders allowed to synchronize in parallel.
iss.indexsvc.singlefolderthread.count	**1**	Number of threads to run in parallel per folder.
iss.indexsvc.admincorethread.count	**25**	For Index Service threads configuration and attachment file size limit.

Table 20–6 (Cont.) Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.indexsvc.message.perthread.sizeinmb	**16**	Average size (in MB) of folder data to load at a time over IMAP.
iss.indexsvc.attachmentstore.enabled	**true**	Attachment store enabled, storing thumbnails of attachments.
iss.indexsvc.attachment.indexmode	**full**	When the value of this configuration parameter is **full**, all attachment records are generated along with the "fullcontents" field. When the value is **IMAPonly**, only the "fullcontents" field is generated. See "issadmin.sh" for more information on overriding this configuration parameter with **--bootstrap**, **--createaccount**, or **--setautobootlist** actions.
iss.indexsvc.attachment.timeout	**300**	Attachment processing timeout in seconds.
iss.indexsvc.attachment.sizelimit	**25000000**	Maximum size text attachment that will be indexed (in bytes).
iss.indexsvc.attachment.thumbnail.xlarge	**true**	Whether to generate extra-large thumbnails.
iss.indexsvc.attachment.maxbreadth	**100**	Maximum number of MIME body parts that will be processed at a single level.
iss.indexsvc.attachment.maxdepth	**50**	Maximum number of MIME body part levels, default matches Messaging Server's hard coded limit of 50.
iss.indexsvc.attachment.pdf.thumbnail.enabled	**false**	Generate thumbnails for PDF attachments (might dramatically increase indexing time when enabled).
iss.indexsvc.statisticsinterval	**600**	Time interval (in seconds) after which statistics are periodically logged to index service log file.
iss.store.account.maxgroup	**500000**	Maximum number of groups allowed in the index.
iss.store.account.pergroup	**1**	Maximum number of accounts allowed in a group.
iss.store.account.optimizeinterval	**50000**	Number of **dIndex** writes before optimize is done.
iss.store.account.optimizelevel	**1**	Default level of optimize (number of segments left).

Table 20–6 (Cont.) Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.store.diskfullpercent.limit	**95**	Defines the disk capacity limit for the index store. When used disk space exceeds this limit, **IndexService** shuts down services to avoid running out of disk space and causing a catastrophic failure. Value is a percentage (0 to 100). When the **IndexService** detects that used disk space exceeds this limit, severe level messages are written to the log file. If the disk space remains too full to perform normal index operations, Indexing and Search Service does not add new data to the store until sufficient space is made available again. (For more detail, refer to the recovery procedures concerning how to administer this situation.)
iss.store.ignorefolder.list	**null**	Specifies a comma-delimited list of folder names which are created and marked as ignored automatically during the **--bootstrap** and **--createaccount**commands. Folder names are case sensitive. If comma, space, or quote characters are present in a folder name, the name must be enclosed in quotation marks ("). An embedded quote is represented by backslash quote. Example: **Trash,"hidden,ignoredfolder"**
iss.store.partitions.count	**0**	Specifies an integer value of the number of partitions across which to divide the dIndex. Value can be 0 to 20. If 0, the single dIndex is created using the format 1 implementation. If the value is 1 or greater, the format 2 implementation is used.
iss.store.partitions.file	**null**	Not currently supported.
iss.store.reboot.age	**365**	If a user's bootstrap time stamp is greater than the value of **iss.store.reboot.age**, then the user is scheduled for rebootstrap. The minimum value is **31**.
iss.storeui.access.method	**disk**	**storeui** access method (**disk** for single system install or **http** for multi-system install).
iss.searchsvc.leadingwildcard.enabled	**true**	**searchsvc** allows leading wildcard in query.
index.runoptimizer	**true**	Indicates whether to optimize index after bootstrap.
queue.connection.factory.name	**cn=CommsQueueFactory**	Name of connection factories and destinations.

Table 20–6 (Cont.) Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
topic.connection.factory.name	**cn=CommsTopicFactory**	Name of connection factories and destinations.
iss.searchsvc.dst.name	**cn=SearchTopic**	Name of connection factories and destinations.
iss.indexsvc.dst.name	**cn=isshostIndex**	Name of connection factories and destinations.
iss.accountstate.dst.name	**cn=isshostAccountState**	Name of connection factories and destinations.
iss.jmqsvc.response.timeout	**10000**	JMQConsumer Service response timeout in milliseconds.
iss.jmqconsumer.queue.timeout	**86400000**	JMQConsumer event queue timeout.
iss.jmqconsumer.queue.limit	**5000**	JMQConsumer event queue limit.
iss.jmqconsumer.expire.limit	**5**	JMQConsumer event expire size limit.
iss.jmqconsumer.thread.count	**128**	JMQConsumer index request thread count.
iss.jmqconsumer.statecheck.interval	**300**	Number of seconds between account state check by JMQConsumer.
iss.rest.proxypool.size	**512**	Size of proxy connection pool in restful web services.
iss.rest.searchsvc.timeout	**1000**	Timeout for rest web services checking if search service is running in milliseconds.
iss.autoprovision.enabled	**false**	Enables auto provisioning of new accounts.
iss.autoprovision.host.include.list	**.+**	Specifies a comma-delimited list of regular expression patterns used to match host names of accounts to be included in automatic provisioning. Example: *host.*, **mshost.**
iss.autoprovision.user.include.list	**.+**	Specifies a comma-delimited list of regular expression patterns used to match user names of accounts to be included in automatic provisioning. Example: **a.,b.,c.+**
iss.autoprovision.host.exclude.list	**None**	Specifies a comma-delimited list of regular expression patterns used to match host names of accounts to be excluded from automatic provisioning. Example: **nonmail.***
iss.autoprovision.user.exclude.list	**None**	Specifies a comma-delimited list of regular expression patterns used to match user names of accounts to be excluded from automatic provisioning. Example: **d.,e.,f.+**
iss.indexsvc.periodic.autosync.enabled	**false**	Enables automatic periodic synchronization of Indexing and Search Service accounts.

Table 20–6 (Cont.) Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.indexsvc.periodic.autosync.flags.enabled	**true**	Enables use of **--detail** *flags* during automatic periodic synchronization of Indexing and Search Service accounts.
iss.indexsvc.periodic.autosync.interval	**300**	Specifies an integer value indicating the number of seconds to wait until the start of the next period of the autosync.
iss.indexsvc.periodic.autosync.count	**1000**	Specifies an integer value indicating the number of accounts to process from the list of accounts yet to be synced during each period.
iss.indexsvc.periodic.autosync.thread.count	**10**	Specifies an integer value indicating the number of threads to use to process accounts to be synced during each period.
iss.indexsvc.periodic.autosync.deepcheck.enabled	**false**	Enables deep checking during automatic periodic synchronization of Indexing and Search Service accounts. For more information, see "Migrating from Java 6 to Java 7".
iss.indexsvc.periodic.autobootstrap.enabled	**false**	Enables automatic periodic bootstrapping of Indexing and Search Service accounts.
iss.indexsvc.periodic.autobootstrap.interval	**300**	Specifies an integer value indicating the number of seconds to wait until the start of the next period of autobootstrap accounts.
iss.indexsvc.periodic.autobootstrap.count	**500**	Specifies an integer value indicating the number of accounts to process from the list of accounts yet to be bootstrapped during each period.
iss.indexsvc.periodic.autobootstrap.thread.count	**10**	Specifies an integer value indicating the number of threads to use to process accounts to be bootstrapped during each period.
iss.indexsvc.periodic.autobootstrap.triggerlist	**All**	Specifies a comma-delimited list of event names that are used to count event errors toward autobootstrapping. Entries in the list may be any of the following values: **None**, **All**, **NewMsg**, **UpdateMsg**, **ChangeFlags**, **CopyMsg**, **Create**, **Delete**, **Expunge**,**Rename**
iss.indexsvc.periodic.autobootstrap.triggercount	**1**	Specifies an integer value indicating the number of event errors that must occur on an account before it is scheduled for autobootstrapping.

Table 20–6 (Cont.) Indexing and Search Service Services Run Time Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.indexsvc.periodic.autodelete.enabled	**true**	Enables automatic periodic deletion of orphaned Indexing and Search Service accounts.
iss.indexsvc.periodic.autodelete.interval	**900**	Specifies the time in seconds between the autodelete periods.
iss.indexsvc.periodic.autodelete.purge.interval	**3600**	Specifies the time in seconds that must have passed since the last transaction on an account to allow it to be deleted.

Watcher Service Configuration

Table 20–7 describes watcher service configuration parameters.

Table 20–7 Watcher Service Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.watcher.enabled	**false**	Specifies that the watcher service is enabled on this host.
java.args.watcher	**-Xmx128m**	Sets Java Virtual Machine (JVM) options for the watcher process.
iss.watcher.interval	**60**	Specifies interval in seconds of how often the watcher service performs its checks.
iss.watcher.timeout	**30**	Specifies number of seconds watcher should wait for a response from a service before generating an alert.
iss.watcher.notification.type	**log-only**	Specifies type of watcher notification to use: **log-only** or **email**
iss.watcher.notification.destination	null	Specifies destination email address for watcher notifications.
iss.watcher.notification.source	null	Specifies source (from) email address for watcher notifications.
iss.watcher.notification.mail.host	null	Specifies mail host and port to send messages: *host*:*port*
iss.watcher.notification.protocol	**plain**	Specifies protocol to use when sending email notifications: **ssl**, **tls**, or **plain**
iss.watcher.email.user	null	Specifies user name if required to send email.
iss.watcher.email.user.password	null	Specifies user name password if required to send email.
iss.watcher.log.level	**WARNING**	Sets watcher service log level. Log levels from lowest to highest are **OFF**, **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**, and **ALL**.
iss.watcher.logfile.name	**iss-watcher.log**	Specifies watcher service log file name.

Table 20–7 (Cont.) Watcher Service Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.watcher.logfile.limit	**20000000**	Specifies watcher service log file size limit.
iss.watcher.logfile.count	**20**	Specifies watcher service number of log files to preserve.
iss.watcher.logfile.formatter	**com.sun.comms.iss.common.SingleLineFormatter**	Specifies watcher service log file formatter.

Web Services Proxy (isshttpd)

Table 20–8 describes web services proxy (**isshttpd**) parameters.

Table 20–8 Web Services Proxy (isshttpd) Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.isshttpd.enabled	**false**	Enables or disables **isshttpd** service on this host.
java.args.isshttpd	**-Xmx1g**	Specifies JVM options for the **isshttpd** process.
iss.isshttpd.thread.count	**50**	Specifies **isshttpd** service request thread count.
iss.isshttpd.port	**5559**	Specifies the port on which **isshttpd** listens to process requests.
iss.isshttpd.ssl.port	**5558**	Specifies the SSL port on which **isshttpd** listens to process requests.
iss.isshttpd.lookup.source	**ldap**	Type of mapping for the user requesting the Indexing and Search Service search to its mail host, either **ldap** or **file**.
iss.isshttpd.lookup.file	**/etc/jiss/isshttpd.map**	Specifies location of the file that contains the mail server to Indexing and Search Service server mapping.
iss.isshttpd.bind.localhost	**true**	Enables **isshttpd** to bind only the localhost.
iss.isshttpd.log.level	**WARNING**	Specifies **isshttpd** service log level. Log levels from lowest to highest are **OFF**, **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**, and **ALL**.
iss.isshttpd.logfile.name	**iss-isshttpd.log**	Specifies **isshttpd** service log file name.
iss.isshttpd.logfile.limit	**20000000**	Specifies **isshttpd** service log file size limit.
iss.isshttpd.logfile.count	**20**	Specifies **isshttpd** service number of log files to preserve.
iss.isshttpd.logfile.formatter	**com.sun.comms.iss.common.SingleLineFormatter**	Specifies **isshttpd** service log file formatter.
iss.isshttpd.outgoing.ssl.enabled	**false**	Enables SSL for communicating to Indexing and Search Service web services
iss.isshttpd.incoming.ssl.enabled	**false**	Starts SSL enabled socket listener.
iss.isshttpd.incoming.plain.enabled	**true**	Starts plain socket listener.

Table 20–8 (Cont.) Web Services Proxy (isshttpd) Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.ssl.keystore	null	SSL keystore file path.
iss.ssl.keystorepassword	null	SSL keystore password.
iss.isshttpd.ssl.selfsigned.enabled	**true**	Accepts self-signed certificates when accessing Indexing and Search Service web services.

Cluster Configuration for Indexing and Search Service

Table 20–9 describes parameters for cluster configuration parameters.

Table 20–9 Cluster Configuration Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.cluster.install	**standalone**	Cluster configuration: **multi-machine**, **cluster**, **clusterv2**, or **standalone**
iss.cluster.jndi.namespace	NA	JNDI lookup name space for multi-machine installations, for example: **indexing01**
iss.cluster.dir	**cluster.d**	The directory containing the cluster configuration files relative path to *IndexSearch_home*/**etc/**.

Deprecated Parameters

Table 20–10 describes deprecated parameters.

Table 20–10 Deprecated Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
imq.port Removed in **Indexing and Search Service 1 Update 2.**	**7676**	Indexing and Search Service IMQ broker port number.
ldap Renamed to **ldap.host** in **Indexing and Search Service 1 Update 2**.	**isshost.example.com**	Indexing and Search Service Directory Server fully qualified host name.
ldap.port Removed in **Indexing and Search Service 1 Update 2**.	**389**	Indexing and Search Service Directory Server port number.
ldap.sslport Removed in **Indexing and Search Service 1 Update 2**.	**636**	Indexing and Search Service Directory Server SSL port.
java.naming.security.principal	**cn=Directory Manager**	Indexing and Search Service Directory Manager DN format.

Table 20–10 (Cont.) Deprecated Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
java.util.logging.FileHandler.limit Removed as of **Indexing and Service Search 1 Update 1**, where it is replaced by the new parameters **iss.*.logfile.limit**, where * is **indexsvc**, **jmqconsumer**, **searchsvc**, or **utilsvc**.	**20000000**	Maximum log file size in bytes.
java.util.logging.FileHandler.count Removed as of **Indexing and Service Search 1 Update 1**, where it is replaced by the new parameters **iss.*.logfile.count**, where * is **indexsvc**, **jmqconsumer**, **searchsvc**, or **utilsvc**.	**20**	Maximum number of log file rotation files.
java.util.logging.FileHandler.formatter Removed as of **Indexing and Service Search 1 Update 1**, where it is replaced by the new parameters **iss.*.logfile.formatter**, where * is **indexsvc**, **jmqconsumer**, **searchsvc**, or **utilsvc**.	**com.sun.comms.iss.common.SingleLineFormatter**	Logging formatter.
iss.indexsvc.corethread.count Removed as of **Indexing and Service Search 1 Update 1**.	**25**	Index Service threads configuration and attachment file size limit.
iss.indexsvc.message.perthread Removed as of **Indexing and Service Search 1 Update 1**.	**1000**	Maximum number of messages indexed per thread.
iss.indexsvc.niofsdirectory.enabled Removed as of **Indexing and Service Search 1 Update 1**, where **NIOFSDirectory** is always used.	**true**	Use **NIOFSDirectory** for index.

Table 20–10 (Cont.) Deprecated Parameters

Parameter	Initial Setting in jiss.conf.template File	Description
iss.cluster.enabled Renamed to **iss.cluster.install** in **Indexing and Search Service 1 Update 4**.	**false**	If the Indexing and Search Service instance is running in clustered mode, defaults to **false**.
iss.cluster.type Renamed to **iss.cluster.install** in **Indexing and Search Service 1 Update 4**.	**none**	The type of node, either **web** or **index** that the Indexing and Search Service instance is running.
iss.isshttpd.ssl.enabled Converted to to **iss.isshttpd.outgoing.ssl.enabled** in **Indexing and Search Service 1.0.5.30**.	**false**	Enables SSL for communicating to Indexing and Search Service web services.

21

IMAP Search Behavior in Indexing and Search Service

This chapter describes the differences between the Indexing and Search Service search implementation and Messaging Server IMAP SEARCH implementation.

Overview of IMAP Search Behavior Differences

The Indexing and Search Service implementation and Messaging Server IMAP SEARCH implementation differ in their results in several ways. These differences come from fundamentally distinct approaches to the problems of search. RFC 3501 is based on a simple substring comparison, while more recently developed search systems such as Indexing and Search Service are based on indexing and lookup. The former is simple but expensive when searching large amounts of data, while the latter is specifically designed to speed up searches over massive amounts of data, at the expense of lost matching capability.

Substring Matches

The IMAP standard (RFC 3501) states:

In all search keys that use strings, a message matches the key if the string is a substring of the field. The matching is case-insensitive.

The Indexing and Search Service implementation of search is not based on comparing substrings of the field. Rather, it is based on indexing technology, which intentionally ignores parts of the field string and uses index lookup for string comparisons to dramatically speed up the search process.

The following table contains examples of search strings that match when using Messaging Server IMAP SEARCH but do not match when using Indexing and Search Service.

Table 21–1 IMAP SEARCH Strings That Do Not Match to Indexing and Search Service Search

Query	Reason for ISS Nonmatch
+subject:,	Punctuation marks are ignored.
+body:the	Many small stopwords (such as "the") are ignored.
+to:geo	Full tokens, not substrings, are matched (for example George, Georg, Georgie, Ingeo).

These examples are all standard conforming queries. These are fundamental differences. Some queries could be modified by Indexing and Search Service to support the last example (by using stems, or wildcards, or both). In general, though, a string match does not occur because too much information is thrown away.

Quoted Phrases

The Messaging Server IMAP SEARCH "collapses" white spaces whereas Indexing and Search Service does not. For example, searching for "rent or buy" in Messaging Server IMAP SEARCH returns results containing "rentorbuy" but Indexing and Search Service does not return such results.

Searches in Foreign Alphabets

A search including an accented character (foreign alphabet) returns only an exact match on that character in Indexing and Search Service. Messaging Server IMAP SEARCH returns a match on the character with and without the accent.

Indexing and Search Service Versus IMAP SEARCH

Search queries from the IMAP SEARCH, ESEARCH, and SORT commands may be performed by Indexing and Search Service, however, queries from the IMAP THREAD command are not.

Indexing and Search Service, rather than Messaging Server, performs the IMAP search unless the IMAP SEARCH Indexing and Search Service gateway finds one of the following conditions:

- An ESEARCH return option (except for RETURN (ALL)) on a SEARCH command is specified in the search.
- None of the search keys of SUBJECT, FROM, TO, CC, BCC, TEXT, or BODY is specified in the search.
- Any one of the search keys KEYWORD, UNKEYWORD, HEADER, OLDER, YOUNGER, MODSEQ, ANNOTATION, OLD, or NEW is specified in the search.

Indexing and Search Service handles the IMAP search request if the preceding conditions are false, regardless of how complex the search might be. Additionally, Indexing and Search Service can handle AND, OR, and NOT operators in the search request.

> **Note:** Currently, you either enable or do not enable Indexing and Search Service on IMAP for all search queries. You cannot select a finer granularity of search method, at the client, query, or account level, to use both Indexing and Search Service and Messaging Server IMAP SEARCH features in the same IMAP server.

For more information about constructing Indexing and Search Service search queries, see "Using Search Query and Sort Criteria".

Handling Search Errors and Timeouts

The search is performed by IMAP SEARCH under the following conditions, even if the search query would normally be an Indexing and Search Service search:

- If Indexing and Search Service encounters an error, for example, a user has not yet been indexed
- If the Indexing and Search Service search query times out

Glossary

account

The Indexing and Search Service store instance organizes information for individual users into accounts. The Indexing and Search Service account is derived from the corresponding Messaging Server email user account. The Indexing and Search Service store uses the same user name, host name, and password as the corresponding account in the Messaging Server message store. Also, the Indexing and Search Service indexed data mirrors the folder directory structure of the email account.

An Indexing and Search Service account is uniquely identified by two strings: the user name and host (or domain) name. The account is the unit of security for content. Each account must have a password to control access to its data.

Each Indexing and Search Service account is assigned to one and only one group in the Indexing and Search Service store. Groups organize the underlying data files in the index. The data for all accounts assigned to a single group is indexed and stored together. This way of indexing and storing data reduces the overhead of managing multiple index directories for many accounts.

account snapshot

All information for an Indexing and Search Service account that is captured at a given time by running the **issadmin.sh --export** command. The export creates a snapshot of the account information in a file structure separate from the store instance. This structure can be used as an archive and as a backup.

account state

The state of an Indexing and Search Service account, set by the index service (process), which can be one of the following:

- Active: The account is searchable. Account state must be active to be searchable.
- Bootstrapping: The account is being bootstrapped.
- Inactive: The account is in maintenance mode and is not active.
- Unknown: The account state is unknown. A new account is in this state after running the **issadmin.sh --createaccount** command.

You can also manually change an account's state by running the **issadmin.sh** command, for example, to change an account from Active to Inactive state.

An account being bootstrapped remains in the B (Bootstrapping) state. When bootstrapping has completed, the status changes to A (Active). After it is active, an account is available for searching, and real-time indexing events are enabled.

attachment store

Data store that Indexing and Search Service creates and maintains for only thumbnail images of attachments. Convergence displays these thumbnail images in its **Attachments** folder.

autobootstrap

See **autobootstrapping**.

autobootstrapping

An Indexing and Search Service feature that bootstraps accounts that have not been bootstrapped or autoprovisioned, due to errors with event notifications of changes. To correct these accounts, you can use the autobootstrapping feature, which is useful if you want your deployment to bootstrap uninitialized accounts under such conditions.

autoprovision

See **autoprovisioning**.

autoprovisioning

An Indexing and Search Service feature that automatically adds new accounts to the Indexing and Search Service store. However, unlike running the **issadmin.sh** command, which gives you control over the group or singleton status for an account, autoprovisioning simply creates the account in the next available index group in the Indexing and Search Service store by using the default allocation policy.

bootstrap

See **bootstrapping**.

bootstrapping

The act of creating each account in the Indexing and Search Service store.You can bootstrap accounts by either manually running the **issadmin.sh --bootstrap** command or by enabling account autoprovisioning.

Cluster Search Service

An Indexing and Search Service feature that makes its search component highly available, enabling access to the index and attachment thumbnail stores through NFS. When Indexing and Search Service search is unavailable from an Indexing and Search Service web node, the Cluster Search Service redirects the clients' search requests to another Indexing and Search Service web node, which accesses the highly available NFS, and locates the appropriate index. Thus, the Indexing and Search Service search component can fail without an effective loss of the overall search functionality.

clusterv2

An Indexing and Search Service feature that makes its search component highly available. See **Cluster Search Service**. clusterv2 provides enhanced security by not exposing the NFS tier, on which the indexes are stored, through a firewall.

default allocation policy

When you create Indexing and Search Service accounts by using the default allocation policy, the default allocation policy places accounts in the lowest numbered, previously defined group that does not already contain the maximum allowed number of accounts. The first group by default is 101. The **iss.store.account.pergroup** configuration parameter defines the number of accounts that each group can contain (the default value is 1). If all previously defined groups are full, the number of the next

group created is the successor to the lowest numbered group after the first group that has no successor group, thus filling in gaps in the sequence of group numbers. (Groups with numbers less than the first group are not affected by the default allocation policy.)

dindex

See **global directory index**.

exporting (an account)

Each Indexing and Search Service account is uniquely assigned to a single Indexing and Search Service store instance, which contains all its index and data information. To manage the size of the store and locality of accounts, you must be able to move an account from one store instance to another. This process consists of two steps:

1. Exporting the information from its current store instance
2. Importing the exported information into another store instance

global directory index

The global "directory index" (dIndex) controls the Indexing and Search Service store. It is a single, unique Lucene index directory that contains the following information:

- Each account's location in the group index directories
- Account state information
- Various details about accounts such as folders and counts
- All information needed to manage the system. In contrast, the account group index directories contain all the indexing information for searching emails organized by account, with the various fields that can be used to search, such as subject, to, from, body, attachments, and so on.

See also **partition count**.

group

Each Indexing and Search Service account is assigned to one and only one group in the Indexing and Search Service store. Groups organize the underlying data files in the index. The data for all accounts assigned to a single group is indexed and stored together. This way of indexing and storing data reduces the overhead of managing multiple index directories for many accounts.

IMAP SEARCH Indexing and Search Service gateway

Because Indexing and Search Service can communicate directly with Messaging Server, any IMAP client that can connect to Messaging Server can interoperate with Indexing and Search Service. The Indexing and Search Service architecture permits two ways of conducting a search. Either the mail client communicates directly to Indexing and Search Service by using the RESTful web service (deployed in a GlassFish Server web container), or Messaging Server communicates to the Indexing and Search Service interface. In the latter case, IMAP clients perform searches of the index store by first connecting to the Messaging Server IMAP daemon. The IMAP SEARCH Indexing and Search Service gateway component of the Messaging Server then diverts appropriate searches to Indexing and Search Service. From the client perspective, the client continues to communicate over the IMAP protocol with the Messaging Server with no knowledge of there being a separate Indexing and Search Service server.

importing (an account)

Each account is uniquely assigned to a single Indexing and Search Service store instance, which contains all its index and data information. To manage the size of the store and locality of accounts, you must be able to move an account from one store instance to another. This process consists of two steps:

1. Exporting the information from its current store instance
2. Importing the exported information into another store instance

index store

The Indexing and Search Service data store that contains the account information.

See also **bootstrapping**.

Indexing and Search Service Web Services Proxy

An Indexing and Search Service feature that provides a Web Services Proxy (**isshttpd**). The Web Services Proxy is a standalone Java daemon that Convergence utilizes to route search and thumbnail requests to the correct Indexing and Search Service web services host.

indexing service

The component of Indexing and Search Service that bootstraps new users and indexes email data in real time.

isshttpd

See **Indexing and Search Service Web Services Proxy**.

JMQ broker

Message-oriented middleware server that hosts messaging destinations (that is, queues and topics) for the purposes of asynchronous communication. In Indexing and Search Service, the JMQ broker hosts the search queue and email update notifications for the indexing queue.

Lucene

A free and open-source software library, suitable for any application that requires full text indexing and searching capability. Indexing and Search Service uses Lucene for all its indexing functionality, including the global directory index, groups, and account snapshots.

message store

Contains the user mailboxes for a particular Messaging Server instance.

orphaned account

An Indexing and Search Service account that no longer appears to have a corresponding Messaging Server account is "orphaned." This situation can occur, for example, when you move Messaging Server user accounts between message stores, for load balancing purposes, without updating Indexing and Search Service.

partition count

An integer value of the number of partitions across which to divide the global directory index. Value can be 0 to 20. If 0, the single global directory index is created using the format 1 implementation. If the value is 1 or greater, the format 2 implementation is used.

periodic autobootstrap

See **autobootstrapping**.

periodic automatic synchronization

An Indexing and Search Service feature that checks that the Messaging Server account contains the same information as the Indexing and Search Service index. When the accounts are out-of-sync, periodic automatic synchronization updates the Indexing and Search Service index. Periodic automatic synchronization also repairs accounts that have lost events or show inconsistencies due to server outages. Orphaned accounts are handled by the automatic delete feature.

See also **orphaned account**.

rehosting an account

The process of moving accounts between Messaging Server stores, and the corresponding process of rehosting the Indexing and Search Service index between Indexing and Search Service stores.

search service

The component of Indexing and Search Service that enables clients to search the index store. In an Indexing and Search Service deployment, the GlassFish Server system hosts the Search Service programs, along with the RESTful web API.

singleton group

An Indexing and Search Service group that can contain only one account. After you create a singleton group with an account, you cannot assign another account to that group. If the account in a singleton group is deleted from the Indexing and Search Service store, that group is no longer a singleton, until explicitly assigned again. See also group.

snapshot repository

Directory containing zero or more account snapshots.

See also **account snapshot**.

watcher

See **watcher service**.

watcher service

Provides local host monitoring of Indexing and Search Service services. When the watcher service detects a service outage, it generates log file warnings and email alerts to help you take recovery action.