UnboundID LDAP SDK for Java: LDAP Development Made Simple

Neil A. Wilson, Chief Architect, UnboundID Corp.

LDAPCon - September 21, 2009

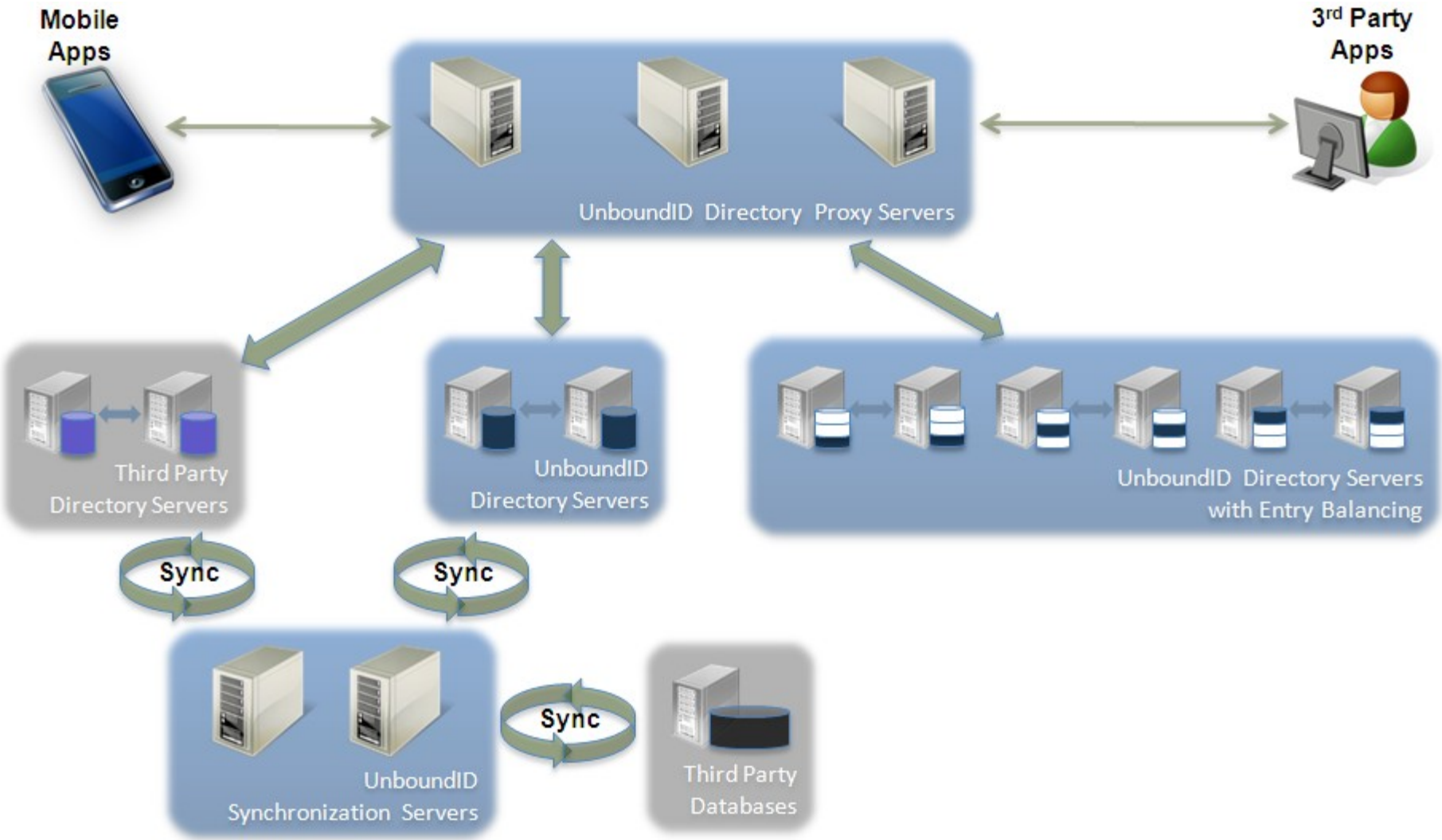
Agenda

- About UnboundID and the presenters
- About the UnboundID LDAP SDK for Java
- Why we wrote the LDAP SDK for Java
- Comparison with other APIs
- Developer-friendly features
- Android mobile platform demo
- LDAP SDK performance and scalability
- Tools provided with the LDAP SDK
- Codebase development and maturity
- Migrating from other APIs
- Getting the LDAP SDK
- Questions and feedback

About UnboundID

- Founded in December 2007 to focus on next-generation LDAP directory services
- Launched by former Sun employees who were previously responsible for OpenDS
  - Director of engineering, director of product marketing, engineering manager, architect, and community manager
  - Prior to that, heavily involved with directory services in other areas at Sun, Netscape, the Burton Group, TidePoint, and Caterpillar
- We were also the developers of SLAMD and continue to maintain it (http://www.slamd.com/)
- Partnered with Alcatel-Lucent in working with many of the largest telecommunications providers, both in the US and worldwide
  - Also working with other kinds of deployments, including financial and service provider industries, and enterprises

UnboundID Product Portfolio



Versions of the LDAP SDK

- Standard Edition
  - Designed to work with any LDAPv3 directory server
  - Completely open source under both GPLv2 and LGPLv2.1 (developer's choice)
- Commercial Edition
  - Includes everything included in the Standard Edition, and adds a number of additional features specific to UnboundID products
  - Doesn't provide any additional benefit unless you're using other UnboundID products
  - Adds support for an additional 15 controls, 8 extended operations, and 2 UnboundID-specific intermediate responses
  - APIs for interacting with administrative alerts, access and error log files, server monitor entries, and administrative tasks

About the LDAP SDK

- 100% pure Java (Java SE 5.0 or later)
- No third-party dependencies (single jar file)
- Provides support for synchronous and asynchronous operations
- Advanced connection pooling with failover, load balancing, and health checking
- Supports lots of standard controls and extensions
- We're adding a persistence API for easily storing and retrieving Java objects as LDAP entries in a usable format
- Also provides APIs for other directory-related processing, including ASN.1 BER, LDIF, SSL/TLS, and command-line tools
- Includes a framework to help migrate from other APIs

Standard Edition Controls and Extensions

- Supported Controls
  - Authorization identity
  - LDAP assertions
  - LDAP subentries
  - ManageDsaIT
  - Matched values
  - Password expired / expiring
  - Permissive Modify
  - Persistent search / entry change notification
  - Pre-read / post-read
  - Proxied authorization (v1 / v2)
  - Server-side sort
  - Simple paged results
  - Subtree delete
  - Virtual list view
- Supported Extended Operations
  - Cancel
  - Notice of disconnection
  - Password modify
  - StartTLS
  - Who Am I?

Why We Wrote the LDAP SDK

- We needed a good SDK to use for our own products
  - Used as a core component of all of our products
  - Used by our administration frameworks for managing the products
  - Used in performance and scalability testing for the products
- Existing APIs just weren't good enough
- We wanted to help grow the directory market
  - Making it easier to write directory-enabled applications helps to increase the reliance on existing directory services and creates opportunities for new deployments
- We wanted to provide good APIs for our customers to use to interact with our products

Problems with Existing APIs

- They aren't as developer-friendly as they should be
  - There are gaps in functionality and missing convenience methods
  - Developers have to write too much code to make their applications production-ready (error handling, failover, load-balancing, debugging, etc.)
- They're showing their age
  - JNDI hasn't been significantly improved since Java 1.4 in 2002. The Netscape Directory SDK for Java also hasn't been updated since 2002. JLDAP hasn't been updated for over a year and a half
  - The Netscape SDK and JLDAP are based on an old spec and haven't been updated as Java has evolved
- They have performance and scalability limitations
  - The Netscape SDK has stability problems under heavy load

Code Comparisons: UnboundID LDAP SDK for Java versus JNDI and Netscape Directory SDK for Java

Code Comparison: Connect and Bind

- JNDI:

```
Properties env = new Properties();
env.setProperty(Context.INITIAL_CONTEXT_FACTORY,
     "com.sun.jndi.ldap.LdapCtxFactory");
env.setProperty(Context.PROVIDER_URL, "ldap://server.example.com:389/");
env.setProperty(Context.SECURITY_PRINCIPAL,
     "uid=test,ou=People,dc=example,dc=com");
env.setProperty(Context.SECURITY_CREDENTIALS, "password");
env.setProperty("java.naming.ldap.version", "3");
LdapContext conn = new InitialLdapContext(env, null);
```

Code Comparison: Connect and Bind

- Netscape Directory SDK for Java:

```
LDAPConnection conn = new LDAPConnection();
conn.connect(3, "server.example.com", 389,
     "uid=test,ou=People,dc=example,dc=com", "password");
```

Don't Forget This!

Code Comparison: Connect and Bind

- UnboundID LDAP SDK for Java:

```
LDAPConnection conn = new LDAPConnection("server.example.com",
     389, "uid=test,ou=People,dc=example,dc=com", "password");
```

Code Comparison: Search

- JNDI:

```
SearchControls searchControls = new SearchControls();
searchControls.setSearchScope(SearchControls.SUBTREE_SCOPE);
searchControls.setReturningAttributes(new String[] { "mail" });

String mail = null;
NamingEnumeration<SearchResult> results = conn.search(
     "dc=example,dc=com", "(uid=test)", searchControls);
try {
  SearchResult result = results.next();
  Attributes attributes = result.getAttributes();
  Attribute mailAttr = attributes.get("mail");
  if (mailAttr != null) {
    Object o = mailAttr.get();
    if (o != null) {
      if (o instanceof byte[]) {
        mail = new String((byte[]) o, "UTF-8");
      } else {
        mail = String.valueOf(o);
      }
    }
  }
} finally {
  results.close();
}
```

Confusing class name

Attribute values are objects and you need to figure out the type

Don't forget to close the NamingEnumeration!

Code Comparison: Search

No convenient way to get a single attribute value

- Netscape Directory SDK for Java

```
String mail = null;
LDAPSearchResults results = conn.search("dc=example,dc=com",
     LDAPConnection.SCOPE_SUB, "(uid=test)", new String[] { "mail" }, false);
while (results.hasMoreElements()) {
  try {
    LDAPEntry entry = results.next();
    LDAPAttribute mailAttr = entry.getAttribute("mail");
    if (mailAttr != null) {
      String[] values = mailAttr.getStringValueArray();
      if ((values != null) && (values.length > 0)) {
        mail = values[0];
      }
    }
  } catch (LDAPException le) {}
}
```

Watch out for exceptions in the SearchResults enumeration

Code Comparison: Search

- UnboundID LDAP SDK for Java

```
String mail = null;
SearchResult result = conn.search("dc=example,dc=com", SearchScope.SUB,
     "(uid=test)", "mail");
if (result.getEntryCount() > 0) {
  mail = result.getSearchEntries().get(0).getAttributeValue("mail");
}
```

Code Comparison: Add

No good way to create a multi-valued attribute

- JNDI:

```
Attributes attributes = new BasicAttributes();
Attribute objectClassAttribute = new BasicAttribute("objectClass", "top");
objectClassAttribute.add("domain");
attributes.put(objectClassAttribute);
attributes.put("dc", "example");
conn.bind("dc=example,dc=com", null, attributes);
```

Horrible method name!

Code Comparison: Add

- Netscape Directory SDK for Java:

```
LDAPAttributeSet attributeSet = new LDAPAttributeSet();
String[] ocValues = { "top", "domain" };
attributeSet.add(new LDAPAttribute("objectClass", ocValues));
attributeSet.add(new LDAPAttribute("dc", "example");
LDAPEntry entry = new LDAPEntry("dc=example,dc=com", attributeSet);
conn.add(entry);
```

Code Comparison: Add

- UnboundID LDAP SDK for Java:

```
conn.add("dc=example,dc=com",
     new Attribute("objectClass", "top", "domain"),
     new Attribute("dc", "example"));
```

Code Comparison: Add

- Also UnboundID LDAP SDK for Java:

```
conn.add(
     "dn: dc=example,dc=com",
     "objectClass: top",
     "objectClass: domain",
     "dc: example");
```

UnboundID LDAP SDK for Java Developer Friendliness

Developer Friendliness: Source Annotations

- The LDAP SDK source code is marked with annotations that help developers understand how it is intended to be used
  - @ThreadSafety
  - @Extensible / @NotExtensible
  - @Mutable / @NotMutable
  - @InternalUseOnly
- The annotations are visible in the javadoc documentation
- They are available at runtime via reflection
- Unit tests are in place to ensure that all classes are marked with an appropriate set of annotations

Developer Friendliness: Debugging

- The LDAP SDK offers a lot of debugging capabilities
  - ASN.1 encoding / decoding
  - Connect and disconnect
  - Exceptions caught
  - LDAP communication
  - LDIF reading and writing
  - Directory Server monitor data access
  - Improper use of the SDK
- Based on the Java logging framework
- Can be configured either using Java code or by providing properties to the JVM

Developer Friendliness: Connection Pools

- Three connection pool implementations:
  - Simple pool with initial and maximum connections
  - Thread-local pool, which eliminates the need to size the pool
  - Read-write pool, which can allow you to treat reads and writes differently
- Support server sets for high availability and load balancing
- Offer health checking to ensure connections are valid
  - Ability to specify a maximum age for connections in the pool
  - Ability to periodically perform validation on connections in the pool, as well as immediately before or after they are used and/or if a problem occurs
- All pools implement LDAPInterface, which is also implemented by LDAPConnection
  - In many cases, you can use a connection pool in the same way you would use a single connection

Developer Friendliness: Client-Side Processing

- Schema validation
  - Validating attribute values against the appropriate syntax
  - Ensure entries contain an appropriate set of attributes and object classes
  - Ensure entry DN complies with any defined name forms
- Filter evaluation
  - Supports all filter types except approximate match and extensible match
  - Can be configured to use schema-aware evaluation
- Sorting entries
  - EntrySorter class provides a Comparator<Entry> and can sort a provided Collection<Entry>
  - Can include hierarchy in the sorting if desired
  - Can be configured to use schema-aware ordering

Developer Friendliness: Java Framework Support

- The LDAP SDK is pure Java, so it should work with most Java frameworks
  - Works with both Java SE and Java EE
  - Isn't available for Java ME, but is supported on the Android platform
  - Includes a number of manifest properties to make it easy to use with OSGi
  - Easy to use with Apache Maven
  - Works with a number of alternate languages that run on the JVM and can invoke Java code, including JRuby, Jython, Scala, Clojure, and Groovy
- If there's another framework that doesn't have any external dependencies and you think we should support, then let us know

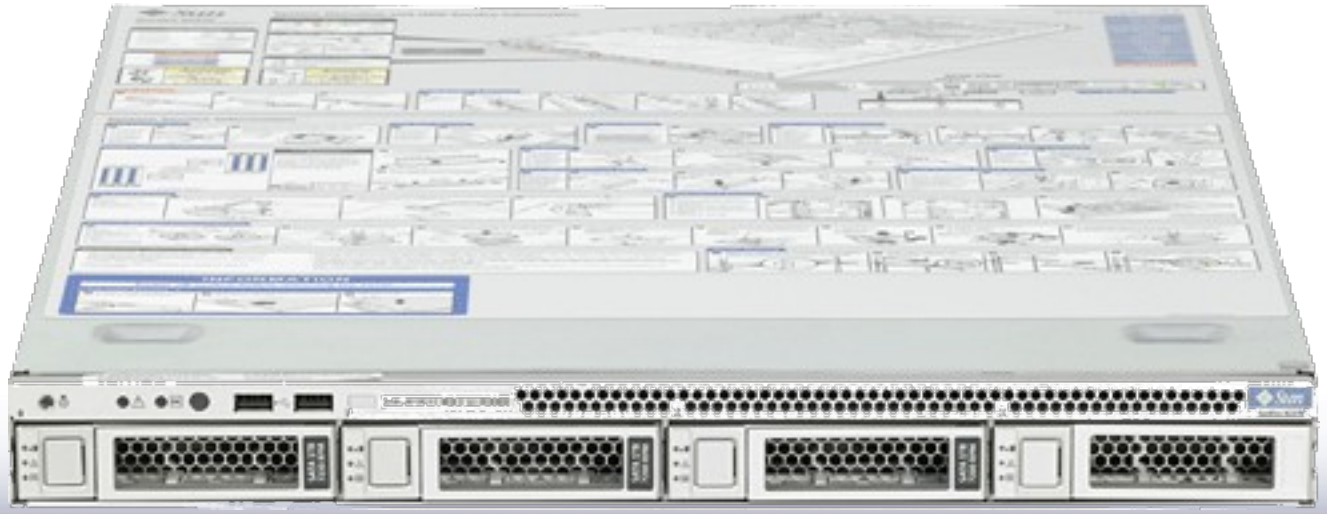
Android Application Demo

Tools Provided with the LDAP SDK

- Performance Tools
  - searchrate / modrate / authrate -- Provide the ability to perform repeated operations against one or more directories as quickly as possible to measure performance and response time
  - Include the ability to target a specific operation rate
  - Can generate output in CSV or human-readable text
- Other Tools
  - ldapsearch / ldapmodify / ldapcompare -- Provide the ability to perform individual operations against a directory server
  - validate-ldif -- Provides the ability to examine an LDIF file to determine whether it conforms to a specified directory schema
  - summarize-access-log -- Provides a tool to parse and summarize one or more UnboundID Directory Server access logs (only included in the Commercial Edition)

Performance Measurement Hardware

- Performance results given in this presentation were obtained on a SunFire x2270 with two Intel Xeon E5540 CPUs at 2.53GHz
  - Each CPU has 4 hyperthreaded cores (psrinfo reports 16 CPUs)
  - 24 GB DDR3 RAM
  - 2 internal 7200RPM hard drives
  - Solaris 10 Update 7 with Java SE 6.0 update 16
  - Client and server running on the same system



Sample modrate Output

```
$ bin/modrate --hostname localhost --port 389 \
     --bindDN "cn=Directory Manager" --bindPassword password \
     --entryDN "uid=user.[1-10000],ou=People,dc=example,dc=com" \
     --numThreads 10 --attribute description --valueLength 10 \
     --warmUpIntervals 3
      Recent       Recent       Recent      Overall      Overall
   Mods/Sec   Avg Dur ms   Errors/Sec     Mods/Sec   Avg Dur ms
----------- ------------ ------------ ------------ ------------
   17058.084        0.567        0.000   warming up   warming up
   21094.344        0.422        0.000   warming up   warming up
   22267.328        0.510        0.000   warming up   warming up
Warm-up completed.  Beginning overall statistics collection.
   23207.138        0.430        0.000    23207.138        0.430
   22150.252        0.451        0.000    22678.696        0.440
   22741.204        0.439        0.000    22699.532        0.440
   23750.592        0.420        0.000    22962.297        0.435
   23298.416        0.428        0.000    23029.521        0.433
   23104.952        0.432        0.000    23042.093        0.433
   23018.357        0.434        0.000    23038.702        0.433
   21197.050        0.471        0.000    22808.496        0.438
   20422.007        0.489        0.000    22543.331        0.443
```

Warm up intervals

Response times measured

Moving averages since starting data collection

Sample searchrate Output

```
$ bin/searchrate --hostname localhost --port 389 \
     --bindDN "cn=Directory Manager" --bindPassword password \
     --baseDN c=example,dc=com" --scope sub --attribute mail --numThreads 64 \
     --warmUpIntervals 3 --filter "(uid=user.[1-10000])" --ratePerSecond 100000
      Recent       Recent       Recent       Recent      Overall      Overall
Searches/Sec   Avg Dur ms Entries/Srch   Errors/Sec Searches/Sec   Avg Dur ms
------------ ------------ ------------ ------------ ------------ ------------
   93014.589        0.296        1.000        0.000   warming up   warming up
   99999.782        0.178        1.000        0.000   warming up   warming up
   99999.688        0.179        1.000        0.000   warming up   warming up
Warm-up completed.  Beginning overall statistics collection.
  100000.329        0.180        1.000        0.000  100000.329        0.180
   99998.846        0.177        1.000        0.000   99999.588        0.178
   99999.589        0.177        1.000        0.000   99999.588        0.178
  100000.389        0.184        1.000        0.000   99999.788        0.179
   99998.854        0.178        1.000        0.000   99999.602        0.179
   99999.105        0.178        1.000        0.000   99999.519        0.179
  100000.508        0.180        1.000        0.000   99999.660        0.179
  100000.253        0.183        1.000        0.000   99999.734        0.179
   99999.460        0.177        1.000        0.000   99999.704        0.179
   99999.930        0.178        1.000        0.000   99999.726        0.179
  100000.092        0.176        1.000        0.000   99999.760        0.179
```

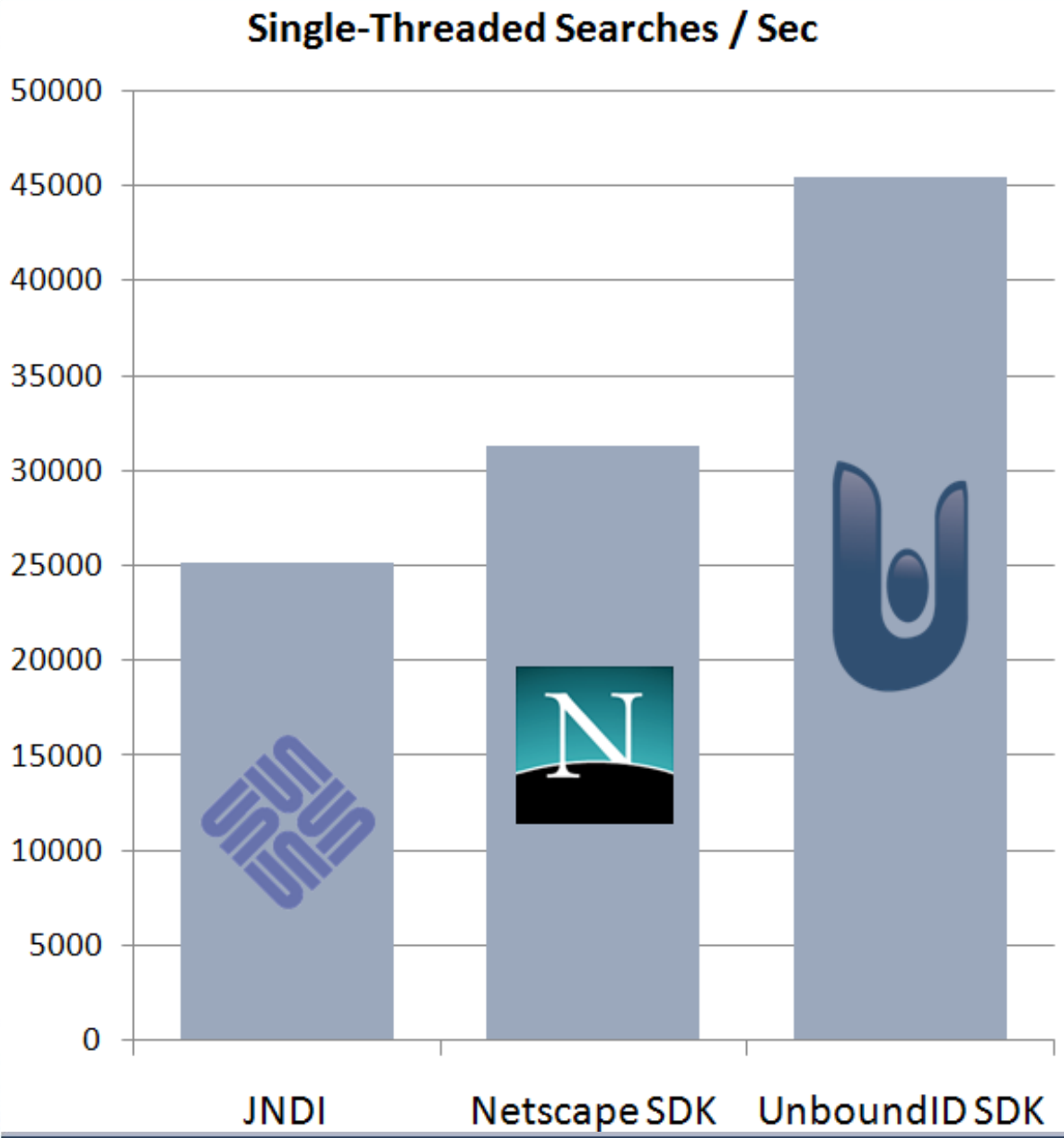
Within 0.001% of target rate

Confirm that entries are being returned

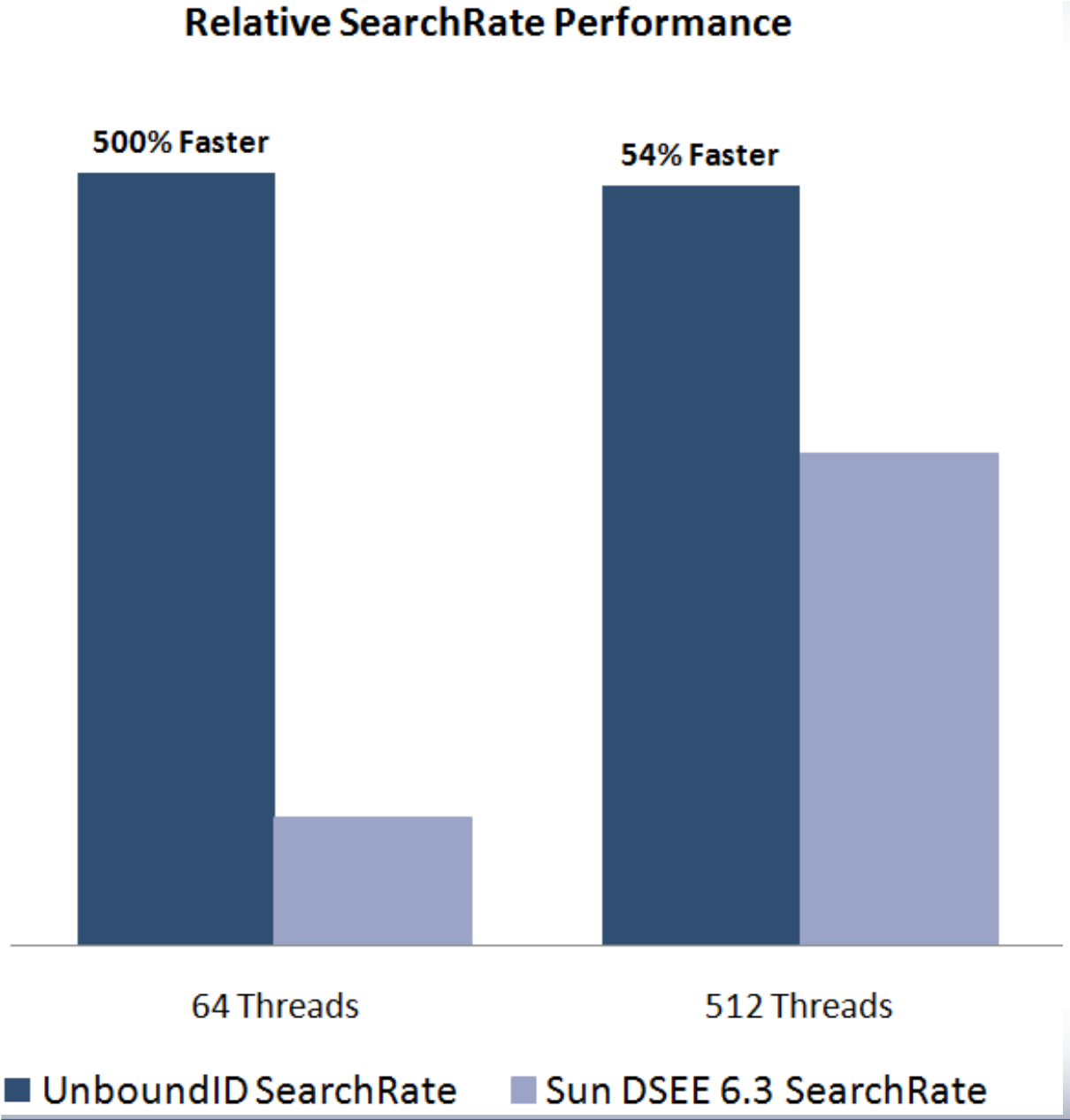
Measuring LDAP SDK Performance

A One-Slide Directory Server Built Using the LDAP SDK

```
package com.unboundid.crds;

import java.io.*;
import java.net.*;
import java.util.*;
import com.unboundid.asn1.*;
import com.unboundid.ldap.protocol.*;
import com.unboundid.ldap.sdk.*;

public final class CannedResponseDirectoryServer extends Thread {
  public static void main(final String[] args) throws Exception {
    final ServerSocket serverSocket = new ServerSocket(1389, 512);
    serverSocket.setReuseAddress(true);
    while (true) { new CannedResponseDirectoryServer(serverSocket.accept()).start(); }
  }

  private static final AddResponseProtocolOp ADD_RESPONSE =
       new AddResponseProtocolOp(0, null, null, null);
  private static final BindResponseProtocolOp BIND_RESPONSE =
       new BindResponseProtocolOp(0, null, null, null, null);
  private static final CompareResponseProtocolOp COMPARE_RESPONSE =
       new CompareResponseProtocolOp(6, null, null, null);
  private static final DeleteResponseProtocolOp DELETE_RESPONSE =
       new DeleteResponseProtocolOp(6, null, null, null);
  private static final ExtendedResponseProtocolOp EXTENDED_RESPONSE =
       new ExtendedResponseProtocolOp(53, null, null, null, null, null);
  private static final ModifyResponseProtocolOp MODIFY_RESPONSE =
       new ModifyResponseProtocolOp(0, null, null, null);
  private static final ModifyDNResponseProtocolOp MODIFY_DN_RESPONSE =
       new ModifyDNResponseProtocolOp(0, null, null, null);
  private static final SearchResultEntryProtocolOp SEARCH_ENTRY_RESPONSE =
       new SearchResultEntryProtocolOp("dc=example,dc=com",
       Collections.<Attribute>emptyList());
  private static final SearchResultDoneProtocolOp SEARCH_DONE_RESPONSE =
       new SearchResultDoneProtocolOp(0, null, null, null);

  private final Socket socket;

  private CannedResponseDirectoryServer(final Socket socket) {
    this.socket = socket;
  }

  @Override() public void run() {
    try
    {
      socket.setKeepAlive(true);
      socket.setReceiveBufferSize(8192);
      socket.setSendBufferSize(8192);
      socket.setReuseAddress(true);
      socket.setSoLinger(true, 1);
      socket.setTcpNoDelay(true);

      final BufferedInputStream inputStream = new BufferedInputStream(socket.getInputStream());
      final OutputStream outputStream = socket.getOutputStream();
      final ASN1StreamReader asn1Reader = new ASN1StreamReader(inputStream);
      final ASN1Buffer asn1Buffer = new ASN1Buffer();

      while (true) {
        final LDAPMessage requestMessage = LDAPMessage.readFrom(asn1Reader, true);
        if (requestMessage == null) {
          return;
        }

        switch (requestMessage.getProtocolOpType()) {
          case LDAPMessage.PROTOCOL_OP_TYPE_ABANDON_REQUEST:
            break;
          case LDAPMessage.PROTOCOL_OP_TYPE_ADD_REQUEST:
            writeResponse(requestMessage, ADD_RESPONSE, asn1Buffer, outputStream);
            break;
          case LDAPMessage.PROTOCOL_OP_TYPE_BIND_REQUEST:
            writeResponse(requestMessage, BIND_RESPONSE, asn1Buffer, outputStream);
            break;
          case LDAPMessage.PROTOCOL_OP_TYPE_COMPARE_REQUEST:
            writeResponse(requestMessage, COMPARE_RESPONSE, asn1Buffer, outputStream);
            break;
          case LDAPMessage.PROTOCOL_OP_TYPE_DELETE_REQUEST:
            writeResponse(requestMessage, DELETE_RESPONSE, asn1Buffer, outputStream);
            break;
          case LDAPMessage.PROTOCOL_OP_TYPE_EXTENDED_REQUEST:
            writeResponse(requestMessage, EXTENDED_RESPONSE, asn1Buffer, outputStream);
            break;
          case LDAPMessage.PROTOCOL_OP_TYPE_MODIFY_REQUEST:
            writeResponse(requestMessage, MODIFY_RESPONSE, asn1Buffer, outputStream);
            break;
          case LDAPMessage.PROTOCOL_OP_TYPE_MODIFY_DN_REQUEST:
            writeResponse(requestMessage, MODIFY_DN_RESPONSE, asn1Buffer, outputStream);
            break;
          case LDAPMessage.PROTOCOL_OP_TYPE_SEARCH_REQUEST:
            writeResponse(requestMessage, SEARCH_ENTRY_RESPONSE, asn1Buffer, outputStream);
            writeResponse(requestMessage, SEARCH_DONE_RESPONSE, asn1Buffer, outputStream);
            break;
          default:
            return;
        }
      }
    } catch (Exception e) {} finally {
      try {
        socket.close();
      } catch (Exception e) {}
    }
  }

  private static void writeResponse(final LDAPMessage requestMessage,
       final ProtocolOp protocolOp, final ASN1Buffer buffer, final OutputStream outputStream)
       throws IOException {
    final LDAPMessage responseMessage =
         new LDAPMessage(requestMessage.getMessageID(), protocolOp);
    buffer.clear();
    responseMessage.writeTo(buffer);
    buffer.writeTo(outputStream);
  }
}
```



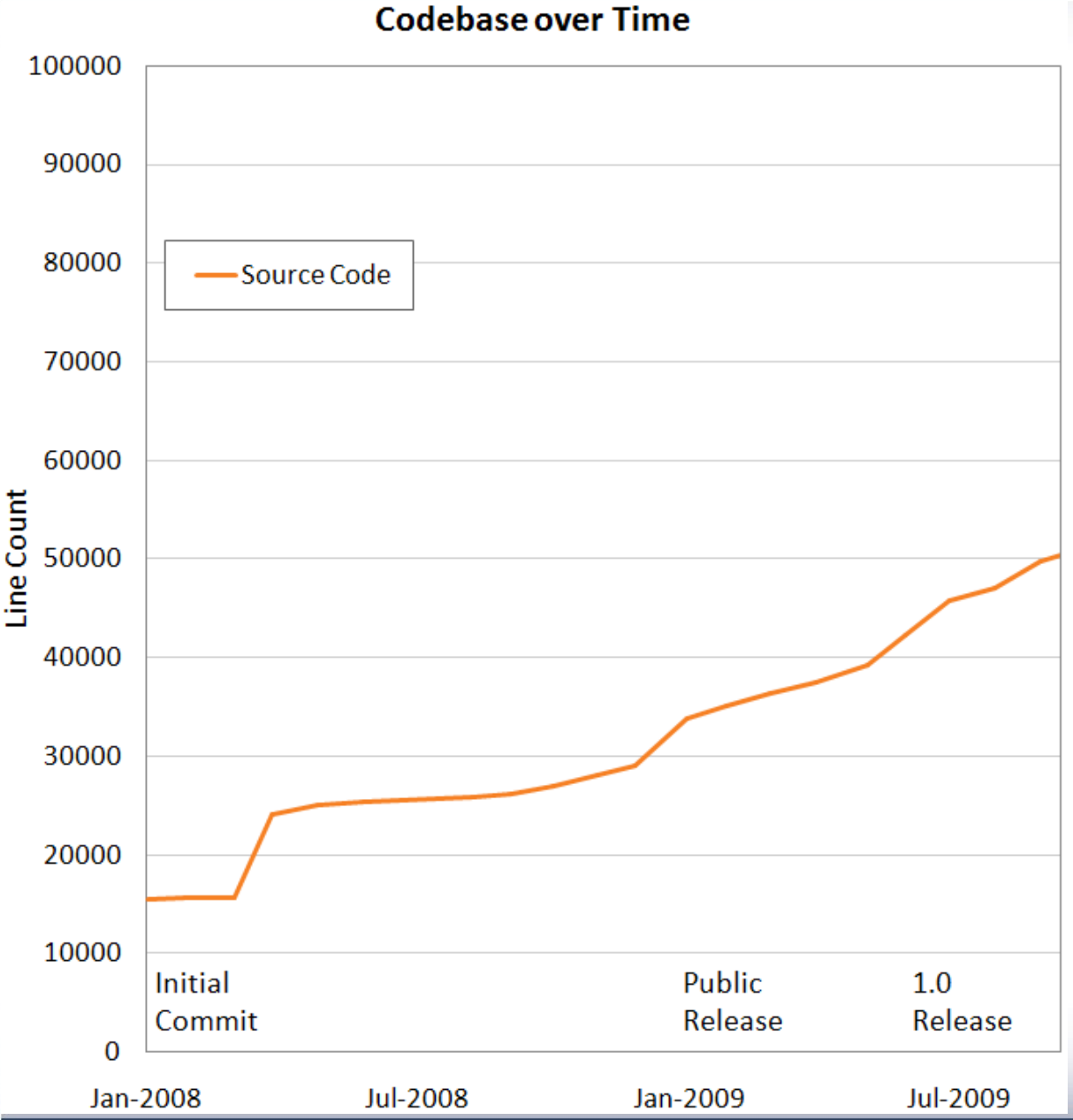
- A simple single-threaded program to perform exact searches against the "Canned Response" directory server
  - Performance ignored for 1 million searches as a warm-up
  - Performance measured for the next 1 million searches
- 80.52% faster than JNDI
- 45.12% faster than Netscape Directory SDK for Java



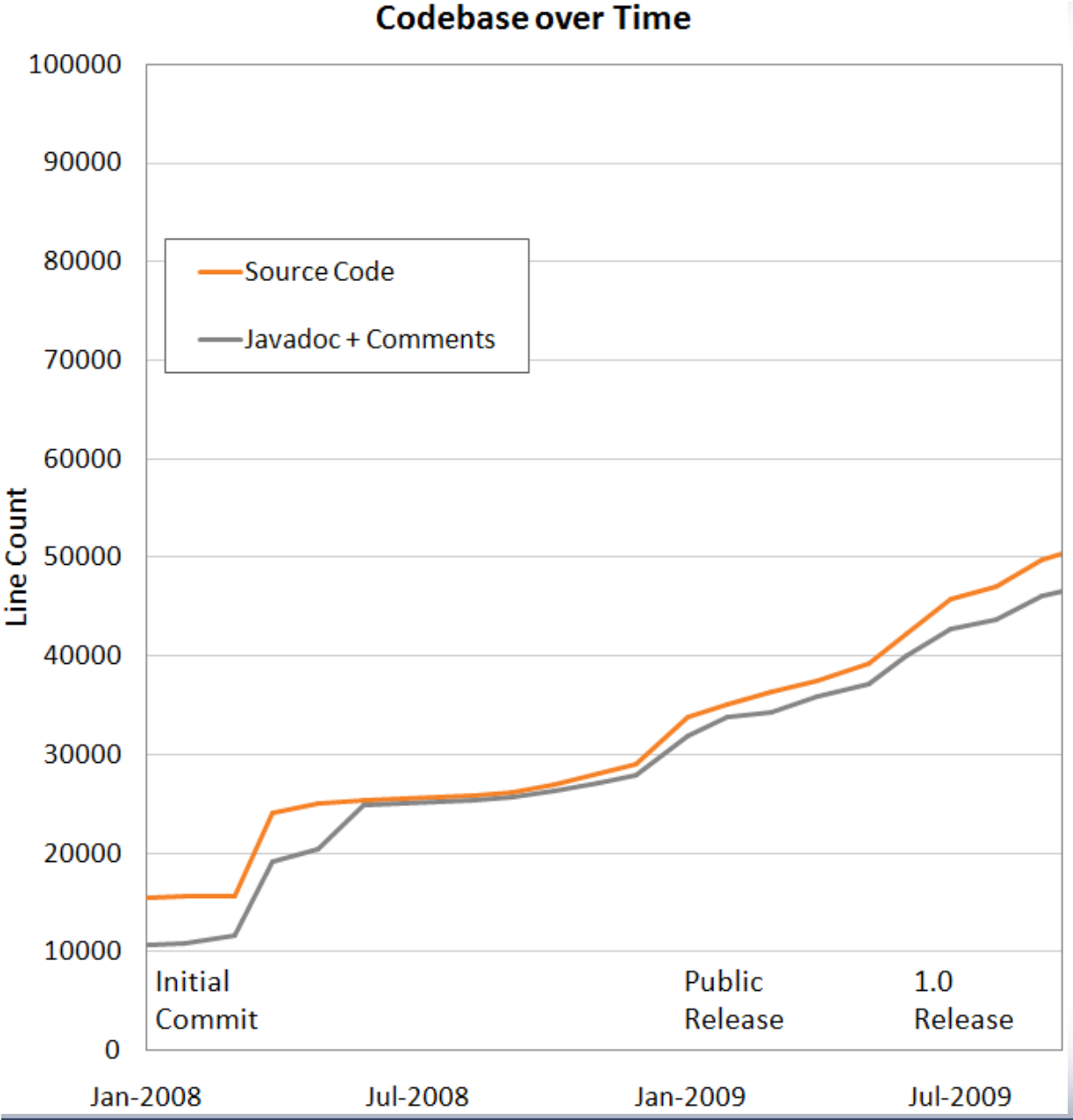
- Java-based UnboundID searchrate vs C-based Sun DSEE 6.3 searchrate
  - Run against the "Canned Response" directory server
  - Results show optimal number of threads for each version of searchrate
  - Actual performance numbers cannot be disclosed due to restrictions in the DSEE license agreement, but the graphs show relative performance

Migrating from/Coexisting with Other LDAP APIs

- Netscape Directory SDK for Java
  - We provide a compatibility layer that exposes much of the Netscape API as a wrapper around our own SDK
  - In many cases, you can simply change your imports from "netscape.ldap" to "com.unboundid.ldap.sdk.migrate.ldapjdk"
  - Supports synchronous operations, custom socket factories, controls and extended operations, and many common data structures
  - Should also be helpful for JLDAP, since it is based on the same IETF draft as the Netscape API
- JNDI
  - The "com.unboundid.ldap.sdk.migrate.jndi" package provides support for converting between several JNDI data structures, including attributes, modifications, search result entries, controls, and extended requests and responses



- "Source Code" only includes meaningful Java code directly included in the LDAP SDK
  - Excludes comments, whitespace, copyright and license headers, unit tests, and lines with just braces
- Despite the relatively recent 1.0 release, it's undergone over a year and a half of development and is still being actively improved



- "Javadoc + Comments" only includes meaningful comments:
  - Excludes copyright and license headers, as well as lines with only comment headers (e.g., just "`/**`", "`/*`", "`*`", "`*/`" or "`//`")
- The majority of the comments are in javadoc documentation

All Classes

Packages

com.unboundid.asn1
com.unboundid.ldap.matchingr
com.unboundid.ldap.protocol
com.unboundid.ldap.sdk
com.unboundid.ldap.sdk.contro
com.unboundid.ldap.sdk.exam
com.unboundid.ldap.sdk.exten
com.unboundid.ldap.sdk.migra
com.unboundid.ldap.sdk.migra
com.unboundid.ldap.sdk.schem
com.unboundid.ldif
com.unboundid.util
com.unboundid.util.args
com.unboundid.util.ssl

com.unboundid.ldap.sdk.

Classes

AssertionRequestControl
AuthorizationIdentityRequestCon
AuthorizationIdentityResponseCo
EntryChangeNotificationControl
ManageDsaITRequestControl
MatchedValuesFilter
MatchedValuesRequestControl
PasswordExpiredControl
PasswordExpiringControl
PermissiveModifyRequestControl
PersistentSearchRequestControl
PostReadRequestControl
PostReadResponseControl
PreReadRequestControl
PreReadResponseControl
ProxiedAuthorizationV1RequestC
ProxiedAuthorizationV2RequestC
ServerSideSortRequestControl
ServerSideSortResponseControl
SimplePagedResultsControl
SortKey
SubentriesRequestControl
SubtreeDeleteRequestControl
VirtualListViewRequestControl
VirtualListViewResponseControl

com.unboundid.ldap.sdk.controls

Class AssertionRequestControl

```
java.lang.Object
  └ com.unboundid.ldap.sdk.Control
      └ com.unboundid.ldap.sdk.controls.AssertionRequestControl
```

All Implemented Interfaces:

java.io.Serializable

```
@NotMutable
@ThreadSafety(level=COMPLETELY_THREADSAFE)
public final class AssertionRequestControl
extends Control
```

Class annotations

This class provides an implementation of the LDAP assertion request control as defined in RFC 4528. It may be used in conjunction with an add, compare, delete, modify, modify DN, or search operation. The assertion control includes a search filter, and the associated operation should only be allowed to continue if the target entry matches the provided filter. If the filter does not match the target entry, then the operation should fail with an ResultCode.ASSERTION_FAILED result.

The behavior of the assertion request control makes it ideal for atomic "check and set" types of operations, particularly when modifying an entry. For example, it can be used to ensure that when changing the value of an attribute, the current value has not been modified since it was last retrieved.

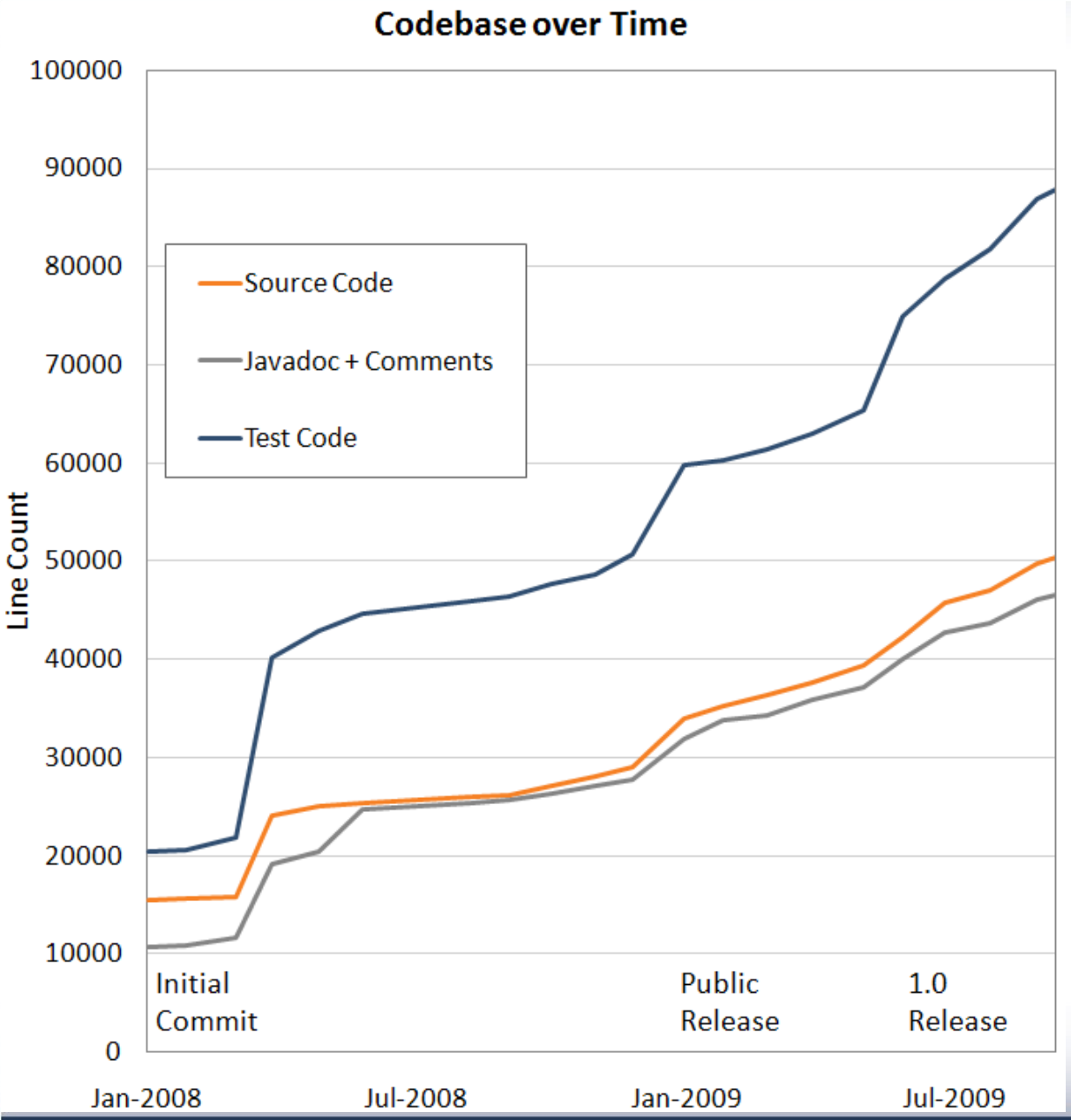
Link to specification

Example

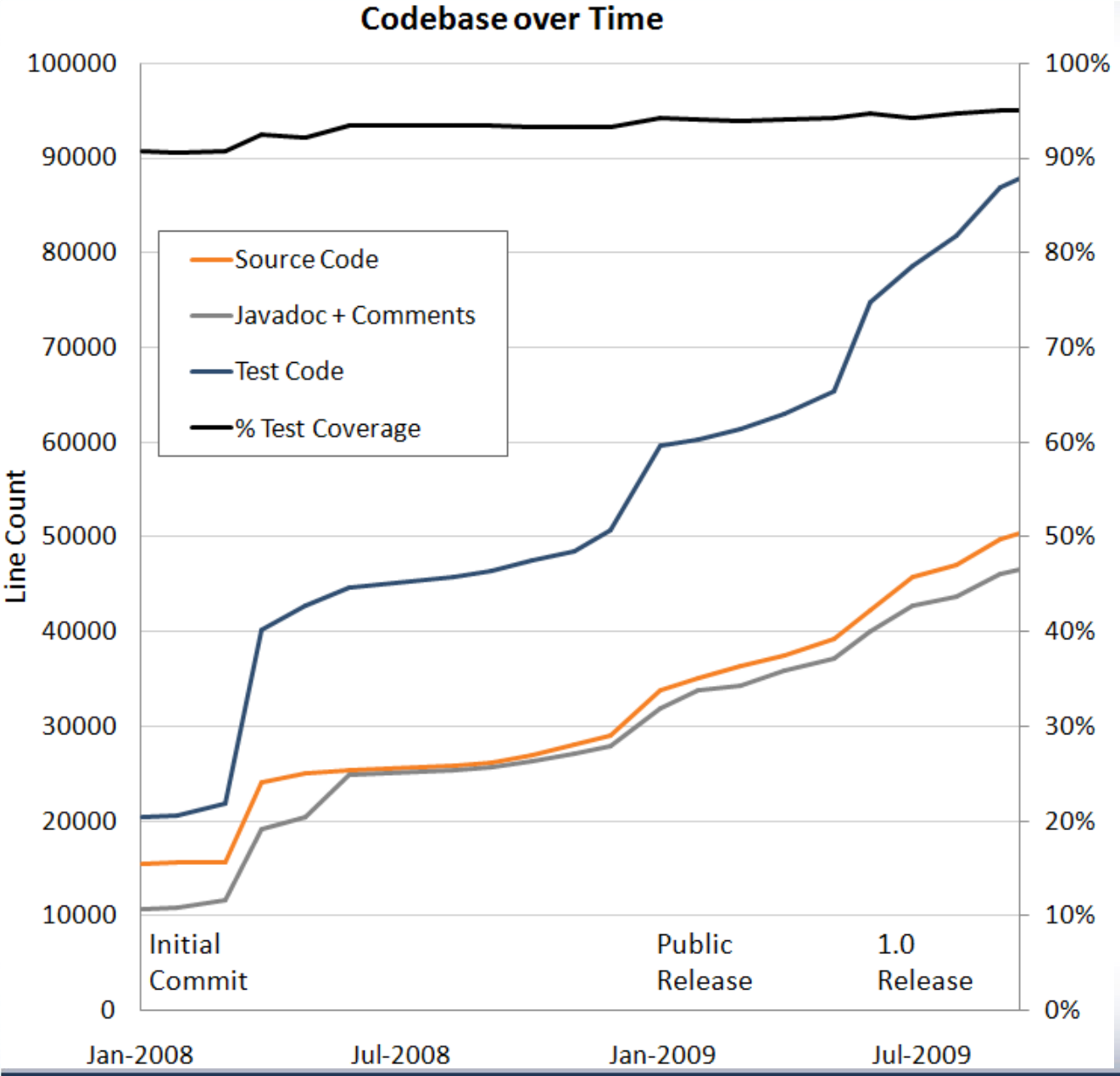
The following example demonstrates the use of the assertion request control. It shows an attempt to modify an entry's "accountBalance" attribute to set the value to "543.21" only if the current value is "1234.56":

```
Modification mod = new Modification(ModificationType.REPLACE,
                                    "accountBalance", "543.21");
ModifyRequest modifyRequest =
     new ModifyRequest("uid=john.doe,ou=People,dc=example,dc=com", mod);
modifyRequest.addControl(
     new AssertionRequestControl("(accountBalance=1234.56)"));

try
{
  LDAPResult modifyResult = connection.modify(modifyRequest);
  // If we've gotten here, then the modification was successful.
}
catch (LDAPException le)
{
```

Code example



- Significantly more test code than code in the LDAP SDK itself
- "Test Code" means the same as "Source Code", but for unit tests
  - Excludes comments, whitespace, license headers, etc.



- Over 95% of SDK code exercised by unit tests
- Over 35,000 tests invoked
- Includes unit tests that ensure API compatibility, proper source annotations, OSGi imports and exports, etc.
- The LDAP SDK also benefits from testing performed against other products that use it

Plans for Future Improvement

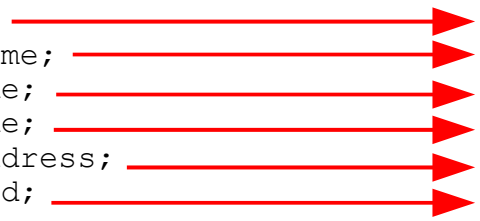
- Java object persistence framework
- Improved LDIF manipulation (LDIFConnection class)
- Enhanced support for groups
- Further debugging support (e.g., access and change logging)
- Content synchronization client support
- Additional matching rules
- More example and utility programs
- Directory-related applications in the Android market
- DSML support?
- Many improvements in the Commercial Edition

Java Object Persistence

- Uses Java annotations to mark classes and fields that should be persisted in a directory
  - @LDAPObject used to mark classes for objects to be persisted and specifies the object classes to use for the entries
  - @LDAPField used to mark fields in objects to be persisted and specifies information about the attributes for the entries
- Objects to be persisted are stored in a meaningful format that is usable by other applications
  - It's also possible to create objects from existing directory data without needing to alter the content or structure of that data
- Includes the ability to generate LDAP schema from annotated classes, and to generate Java source from LDAP schema

Java Object Persistence: Example Object

```
package com.unboundid.example;

public class User {
  private String userID;
  private String firstName;
  private String lastName;
  private String fullName;
  private String eMailAddress;
  private String password;

  // Constructors, getters, setters,
etc.
}
```



```
dn: uid=john.doe,ou=People,dc=example,dc=com
objectClass: top
objectClass: myUser
uid: john.doe
givenName: John
sn: Doe
cn: John Doe
mail: john.doe@example.com
userPassword: password
```

Java Object Persistence: RFC 2713 Approach

```
package com.unboundid.example;

import java.io.Serializable;

public class User
       implements Serializable {
  private String userID;
  private String firstName;
  private String lastName;
  private String fullName;
  private String eMailAddress;
  private String password;

  // Constructors, getters, setters,
etc.
}
```

```
dn: uid=john.doe,ou=People,dc=example,dc=com
objectClass: top
objectClass: javaObject
objectClass: javaSerializedObject
objectClass: extensibleObject
uid: john.doe
javaClassName: com.unboundid.example.User
javaSerializedData:: rO0ABXNyABpjb20udW5ib3V
 uZGlkLmV4YW1wbGUuVXNlcgAAAAAAADA5AgAGTAAMZU1
 haWxBZGRyZXNzdAASTGphdmEvbGFuZy9TdHJpbmc7TAA
 JZmlyc3ROYW1lcQBAAFMAAhmdWxsTmFtZXEAfgABTAAI
 bGFzdE5hbWVxAH4AAUwACHBhc3N3b3JkcQBAAFMAAZ1c
 2VySURxAH4AAXhwdAAUSm9obi5Eb2VAZXhhbXBsZS5jb
 210AARKb2hudAAISm9obiBEb2V0AANEb2V0AAhwYXNzd
 29yZHQACEpvaG4uRG91
```

Java Object Persistence: Annotated Object

```
package com.unboundid.example;

import com.unboundid.ldap.sdk.persist.*;

@LDAPObject(structuralClass="myUser")
public class User {
  @LDAPField(attibute="uid", inRDN=true,
             required=true)
  private String userID;

  @LDAPField(attibute="givenName")
  private String firstName;

  @LDAPField(attibute="sn", required=true)
  private String lastName;

  @LDAPField(attibute="cn", required=true)
  private String fullName;

  @LDAPField(attibute="mail")
  private String eMailAddress;

  @LDAPField(attibute="userPassword")
  private String password;

  // Constructors, getters, setters, etc.
}
```

Java Object Persistence: Annotated Object

```
package com.unboundid.example;

import com.unboundid.ldap.sdk.persist.*;

@LDAPObject(structuralClass="myUser")
public class User {
  @LDAPField(attibute="uid", inRDN=true,
             required=true)
  private String userID;

  @LDAPField(attibute="givenName")
  private String firstName;

  @LDAPField(attibute="sn", required=true)
  private String lastName;

  @LDAPField(attibute="cn", required=true)
  private String fullName;

  @LDAPField(attibute="mail")
  private String eMailAddress;

  @LDAPField(attibute="userPassword")
  private String password;

  // Constructors, getters, setters, etc.
}
```

```
dn: uid=john.doe,ou=People,dc=example,dc=com
objectClass: top
objectClass: myUser
uid: john.doe
givenName: John
sn: Doe
cn: John Doe
mail: john.doe@example.com
userPassword: password
```

Getting the LDAP SDK

- UnboundID Website
  - http://www.unboundid.com/products/ldapsdk/
- SourceForge
  - http://sourceforge.net/projects/ldap-sdk/
  - `svn checkout https://ldap-sdk.svn.sourceforge.net/svnroot/ldap-sdk/trunk ldap-sdk`
- Maven
  - Available in the Central Repository
  - groupId = "com.unboundid"
  - artifactId = "unboundid-ldapsdk"
  - version = "1.1.0"
  - Also works seamlessly with the Ivy dependency manager

Getting Help with the LDAP SDK

- Documentation
  - Detailed javadoc documentation, including lots of examples
  - Additional documents include an FAQ, a comparison with other APIs, and a getting started guide
  - Online at http://www.unboundid.com/products/ldapsdk/docs/ and included as part of the LDAP SDK download
- Mailing Lists
  - ldap-sdk-announce, ldap-sdk-commits, ldap-sdk-discuss @lists.sourceforge.net
  - http://sourceforge.net/mail/?group_id=275998
- Discussion Forum
  - http://sourceforge.net/forum/?group_id=275998
- Paid support available from UnboundID

UnboundID LDAP SDK for Java: LDAP Development Made Simple

Any questions?