

Proposal for a Java LDAP Application Programming Interface

The Lightweight Directory Access Protocol (LDAP, as defined in RFC 4510 et al.) provides a standard means of communicating with directory servers. It is commonly used in identity management, authentication/authorization, naming services, user/application configuration, and general data storage. There are a growing number of commercial and open source server implementations, and many enterprises and service providers are significantly increasing the use of LDAP in their web applications, desktop clients, and mission-critical infrastructure.

While LDAP innovation and development continues on the server side the same is not true of the client. Java SE and EE clients typically use Java Naming and Directory Interface (JNDI) to communicate with directory servers. This is primarily because JNDI and the accompanying LDAP SPI are embedded within Java SE and is rarely based on the merits of the API itself. JNDI is intended to be a generic interface for accessing very different kinds of data stores, and as such uses an abstract API. As a result, critical LDAP functionality is difficult to access and the developer experience is significantly impaired. For example:

- The abstract nature of JNDI terminology is confusing for the LDAP neophyte. That is, the Java LDAP developer must learn 2 different vocabularies, LDAP and JNDI. For example, the terms "bind" and "unbind" have standard meanings in LDAP which are very different from the way those terms are used in JNDI. In addition, a given JNDI method may result in different LDAP operations based on the arguments provided (e.g., while most variations of the JNDI "search" method will result in LDAP search operations, one usage will result in an LDAP compare operation, which may have different semantics).
- LDAP functionality can be enhanced through the use of elements such as controls, extended operations, and SASL mechanisms, and such extensions are frequently described in new specifications. JNDI does not provide an adequate mechanism for adding client-side support for these elements.
- Standard LDAP idioms are difficult or exposed in a bizarre manner; e.g., failed operation codes must be parsed out of exception messages, search enumerations must be closed.

We propose the creation of a new Java API specifically designed for communicating with LDAP directory servers. An IETF draft (<https://opends.dev.java.net/public/standards/draft-ietf-ldapext-ldap-java-api.txt>) addresses this subject and will likely serve as a starting point for future discussion as the the goal of this draft and our proposal are identical: a simple, developer-friendly interaction model for LDAP communication. In addition, the API should provide a convenient means of extending its functionality by adding support for new controls and extended operations as they are defined by various standards bodies. There have been successful implementations of the draft API in the past, but they were based on older versions of the Java language and as such did not not take advantage of recent improvements in Java performance and usability.

In many ways this API could provide for LDAP what JDBC has done for Java relational database clients. A Java-based LDAP API could easily be self-contained and free of external dependencies, making it more immediately accessible to developers. Standardization will relieve LDAP vendors of the mundane and repetitive task of maintaining vendor specific client libraries and provide developers with a pleasurable LDAP experience.