Administrator's Guide

iPlanet Directory Access Router

Release 2.1

806-5634-01
June 2000

Contents

List of Figures

List of Tables

Audience for This Guide

The iPlanet Directory Access Router (iDAR) Administrator's Guide is written for administrators who will configure and operate one or more iPlanet Directory Access Routers. This guide assumes that you have the following background:

- A general understanding of the Internet and LDAP.
- A general understanding of Netscape Directory Server 4.x and its administration. You should be able to read and modify directory data.

What's in This Guide

The guide is organized into 5 parts:

- Chapter 1 provides overview information regarding the different uses and deployment scenarios of iDAR.
- Chapters 2 through 4 cover the general configuration of iDAR including the use of the configuration generation tool and the configuration for SSL / TLS.
- Chapters 5 through 9 present the detailed description of each of the configuration items for iDAR.
- Chapters 10 through 12 cover iDAR operation.

Support

Thank you for choosing iPlanet Directory Access Router. Before requesting assistance, please check the FAQ's and technical notes at `http://www.iplanet.com/support/online`. If you have any questions, please visit Customer Service at `http://www.iplanet.com/support`.

Chapter 1

Overview of iDAR

The iPlanet Directory Access Router (iDAR) is an essential component of any mission-critical directory service for e-commerce solutions. iDAR is an LDAP application layer protocol gateway that offers significant real-world operational advantages for enhanced directory access control, schema compatibility, and high availability using application layer load balancing and failover.

Functionally, iDAR is an “LDAP access router” located between LDAP clients and LDAP directory servers. Requests from LDAP clients can be filtered and routed to LDAP directory servers based on rules defined in the iDAR configuration. Results from the directory server can be filtered and passed back to clients, again based on rules defined in the iDAR configuration. This process is totally transparent to the LDAP clients, which connect to iDAR just as they would to any LDAP directory server.

iDAR is a unique product that provides enhanced high availability, security, and client compatibility features for both extranet and intranet directory infrastructures, including:

- Automatic load balancing
- Transparent server failover & failback
- Automatic referral following
- Extranet / Intranet access control groups
- Secure client and server authentication
- Dynamic query and response filtering
- Dynamic schema mapping
- Directory-based or file-based configuration
- Configurable logging

iDAR coexists with and complements new and existing LDAP directory infrastructures and integrates seamlessly with directory-enabled applications already deployed in both enterprise extranets and intranets. It can be deployed to leverage the existing investment in a customer's directory infrastructure. iDAR will inter-operate with any LDAP compliant directory server. iDAR will work with any LDAP enabled and conformant directory, whether it's a native LDAP directory, an LDAP enabled X.500 directory, or an LDAP enabled relational database.

iDAR implements the LDAPv3 Internet specification and also supports the older and less functional LDAPv2 specification for compatibility with already deployed directory-enabled client applications that use LDAPv2. iDAR runs as a separate system server process on UNIX and Windows NT platforms. The server is multi-threaded and can handle thousands of LDAP client requests while applying access control rules and protocol filtering rules to each request. iDAR can help organizations protect their private directory information from unauthorized access, while making it safe for these organizations to publish their public information. iDAR can be used to configure a fine-grained, access control policy on LDAP directories, such as controlling who can perform different types of operations on different parts of the Directory Information Tree (DIT). iDAR can also be configured to disallow certain kinds of operations typically performed by web trawlers and robots to collect information.

Unlike a web proxy server, iDAR does not operate in a reverse proxy mode. It does not forward connections to arbitrary servers on the Internet from clients inside the firewall. Neither does it cache search results. The predominant reason for this is the problem of applying access controls to the data. This is currently only done in the LDAP directory server where the access controls are maintained. The iPlanet Directory Access Router has no knowledge of the directory server access controls.

iPlanet Directory Access Router Feature Set

The iDAR feature set provides three distinct functions: high availability, firewall-like security, and client-server compatibility.

High Availability

iDAR is designed to support high availability directory deployments by providing both automatic load balancing and automatic failover and failback among a set of replicated LDAP directory servers. For extranet and intranet environments it is often necessary to ensure that mission-critical directory-enabled clients and applications have 24x7 access to directory data. iDAR maintains connection state

information for all directory servers that it knows about, and is able to dynamically perform proportional load balancing of LDAP operations across a set of configured directory servers. Should one or more directory servers become unavailable, the load is proportionally redistributed among the remaining servers. When a directory server comes back on-line, the load is proportionally reallocated dynamically.

For example, suppose directory server A is configured to receive 40 percent of the LDAP client load, server B 20 percent, server C 20 percent and server D 20 percent. If directory server B fails, iDAR will recognize that server A is configured to carry twice the load of servers C and D, and will redistribute the 20 percent load from server B such that server A now receives 50 percent, server C 25 percent and server D 25 percent. When directory server B is recovered, iDAR will automatically detect this and revert back to the original load percentages configured across all four servers.

Network layer IP load balancing devices don't have access to the LDAP protocol layer. However, iDAR integrates load balancing with access control, query filtering and query routing, and can make more intelligent application layer access control and LDAP routing decisions.

Firewall-like Security

iDAR provides flexible external directory access control facilities that enhance the basic access control provided by a directory server. The access control mechanisms allow different users and communities of users to be associated with specific access groups to which administrator defined security restrictions and query filters will be applied. The administrator can control access to entries based on LDAP authentication information, IP address, domain name and other criteria. The server also supports secure authentication using Internet standard security mechanisms including both challenge response "protected password" authentication and digital certificate based strong authentication and data encryption. For protected password, the CRAM-MD5 and DIGEST-MD5 SASL mechanisms are supported. For strong authentication and data confidentiality, support is available for both the Netscape alternate port 636 Secure Sockets Layer (SSL) mechanism and the Internet standard Transport Layer Security (TLS) mechanism. RSA, Digital Signature Standard, Triple-DES, and Diffie Hellman cryptographic algorithms are supported.

One of the significant security features that iDAR provides is the protection of the number of connections established between LDAP clients and the LDAP directory server. You can protect your LDAP directory server from connection attacks by configuring iDAR to monitor a number of specific metrics: the number of simultaneous client operations, the number of operations a client can request per connection, and the number of connections for a particular client group. It also has the ability to time-out inactive clients.

The customer can configure iDAR with specific threshold limits not to be exceeded for the given metrics. iDAR will monitor these metrics and ensure that the thresholds do not get exceeded.

These features can help prevent “denial of service” attacks and “flood attacks” that are so commonplace in the industry today. If iDAR detects a threshold has been reached it will start refusing connections to the directory server and prevent the directory server from being attacked and overwhelmed.

Client-server Compatibility

iDAR provides intelligent query routing by making query routing decisions based on LDAP distinguished names (DNs) and group access rights, including identifying mobile users based on authentication credentials. In addition, iDAR is able to automatically follow LDAP referrals that may be returned by a directory server, in support of highly distributed and scalable directory services. Automatic referral following is a significant advantage for large-scale directory deployments where it is necessary to physically distribute directory information among a set of directory servers, but have the distributed directory appear to users as one logical directory. iDAR supports this type of deployment scenario by providing the directory architect with the ability to logically unify otherwise distributed directory data in support of scalable distributed directory services.

iDAR supports any compliant LDAPv2 or LDAPv3 client application. Support is provided for schema rewriting to accommodate client applications with fixed schemas that do not always match the directory server's schema. For example, the Microsoft Outlook email client has a fixed schema that expects the directory server to implement Microsoft defined attributes, which may not match an enterprise's more general schema requirements. The schema rewriting capability allows the directory system administrator to implement a general purpose enterprise schema, and then dynamically map specific elements of that schema into the set of attribute types that are required by the less functional client application. iDAR is otherwise schema agnostic and accepts any attribute types and object classes defined by a large set of standard and ad hoc industry schema definitions, including RFC1274, X.520, X.521, LIPS, PKIX, inetOrgPerson, and DEN.

iPlanet Directory Access Router Deployment

Typical Customer Deployment of an Internal High Availability Configuration

In this configuration, (Figure 1-1) the customer has deployed an LDAP infrastructure for internal enterprise use only. There is no requirement for external network access to any of the enterprise LDAP services. The customer has deployed an enterprise firewall that will reject any access to internal LDAP services originating from outside the firewall. All client LDAP requests initiated internally must still go through iDAR via the Cisco LocalDirector (for high availability), which is shown here only as an example of IP packet switching which ensures clients have access to at least one iDAR. The customer prevents direct access to the directory servers by everyone except the hosts running iDARs by using TCP wrapper or equivalent program on UNIX platforms and TCP/IP port filtering on Windows NT.

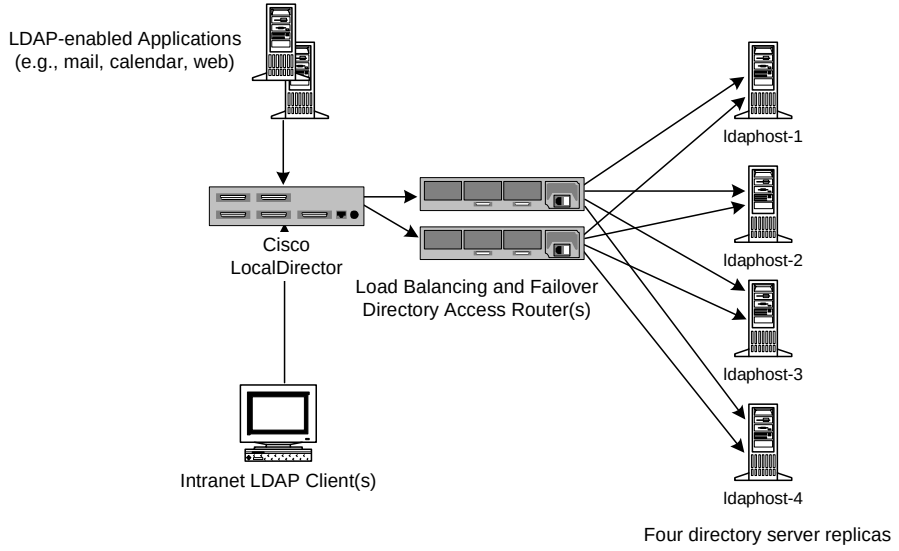

Figure 1-1 Typical Customer Deployment of an Internal High Availability Configuration

Typical Customer Deployment for a Distributed LDAP Directory Infrastructure

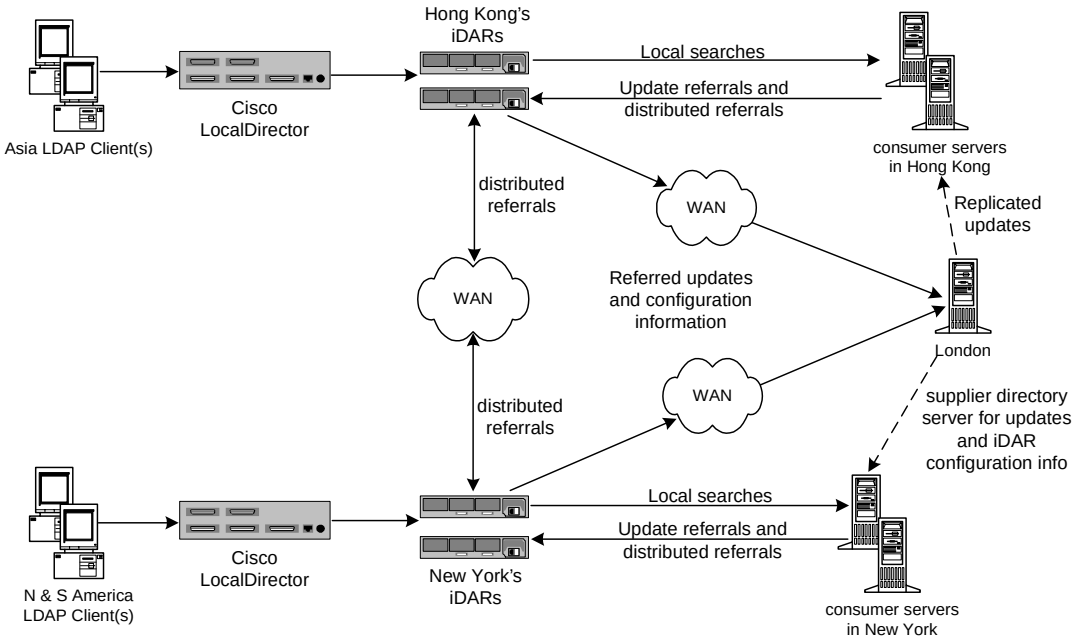

Figure 1-2 Typical Customer Deployment for a Distributed LDAP Directory Infrastructure

Customer Scenario

A large financial institution is headquartered in London with data centers in London, New York and Hong Kong. Currently, the vast majority of the data available to employees resides centrally in legacy RDBMS repositories in London. All access to this data from the financial institution's client community is via the WAN. The financial institution is experiencing scalability and performance problems with this centralized model and has decided to move to a distributed data model. The financial institution has also decided to deploy an LDAP directory infrastructure at the same time. The data in question is considered "mission critical" and should therefore be deployed in a highly available, fault tolerant infrastructure. An analysis of client application profiles has revealed that 95

percent of data accessed by a geographical client community is specific to that community since the data is customer based. It is rare for a client in Asia to access data for a customer in North America but it does happen infrequently. The client community also has a need to update customer information from time to time.

Customer Deployment

Given the profile of 95 percent local data access, the financial institution decided to geographically distribute its LDAP directory infrastructure. It deployed multiple directory consumer servers in each geographical location (i.e., Hong Kong, New York and London; London consumer servers are not shown in diagram). Each of these consumer servers is configured to hold the customer data specific to the geographical location. Data for European and Middle East customers is held in the London consumer servers, data for North and South American customers is held in the New York consumer servers, and data for Asian and Pacific Rim customers is held in the Hong Kong consumer servers. With this deployment, the overwhelming data requirement of the local client community is located geographically in the community. This provides significant performance improvements over the centralized model since client requests are processed locally thereby reducing the network overhead and the local directory servers are effectively partitioning the directory infrastructure thereby providing increased directory server performance and better scalability. Each set of consumer directory servers is configured to return referrals if a client submits an update request or if a client submits a search request for data located geographically elsewhere.

LDAP Request Flow

Client LDAP requests are sent to the iPlanet Directory Access Routers via the Cisco LocalDirector. The LocalDirector product is shown here only as an example of IP packet switching that ensures clients always have access to at least one iDAR. The locally deployed iDARs route all requests initially to the array of local directory servers holding the local customer data. The iDARs are configured to load balance across the array of directory servers thereby also providing automatic failover and failback. Client search requests for local customer information are satisfied by a local directory and appropriate responses returned to the client via the iDAR. Client search requests for geographically ‘foreign’ customer information are initially satisfied by the local directory server by returning a referral back to the iDAR. This referral contains an LDAP URL that points to the appropriate geographically distributed iDARs. The local iDAR processes the referral on behalf of the local client and sends the search request to the appropriate distributed iDAR. The distributed iDAR forwards the search request onto the distributed directory server and receives the appropriate response. This response is then returned to the local client via the distributed and the local iDARs.

Update requests received by the local iDAR are also satisfied initially by a referral returned by the local directory server. Again, iDAR follows the referral on behalf of the local client but this time forwards the update request onto the supplier directory server located in London. The supplier directory server applies the update to the supplier database and sends a response back to the local client via the local iDAR. Subsequently, the supplier directory server will propagate the update down to the appropriate consumer directory servers.

All the iPlanet Directory Access Routers are configured to start up and look for their configuration in the supplier directory server. This allows you to distribute the iDARs geographically but manage their configurations centrally.

Typical Customer Deployment for a Centralized LDAP Directory Infrastructure

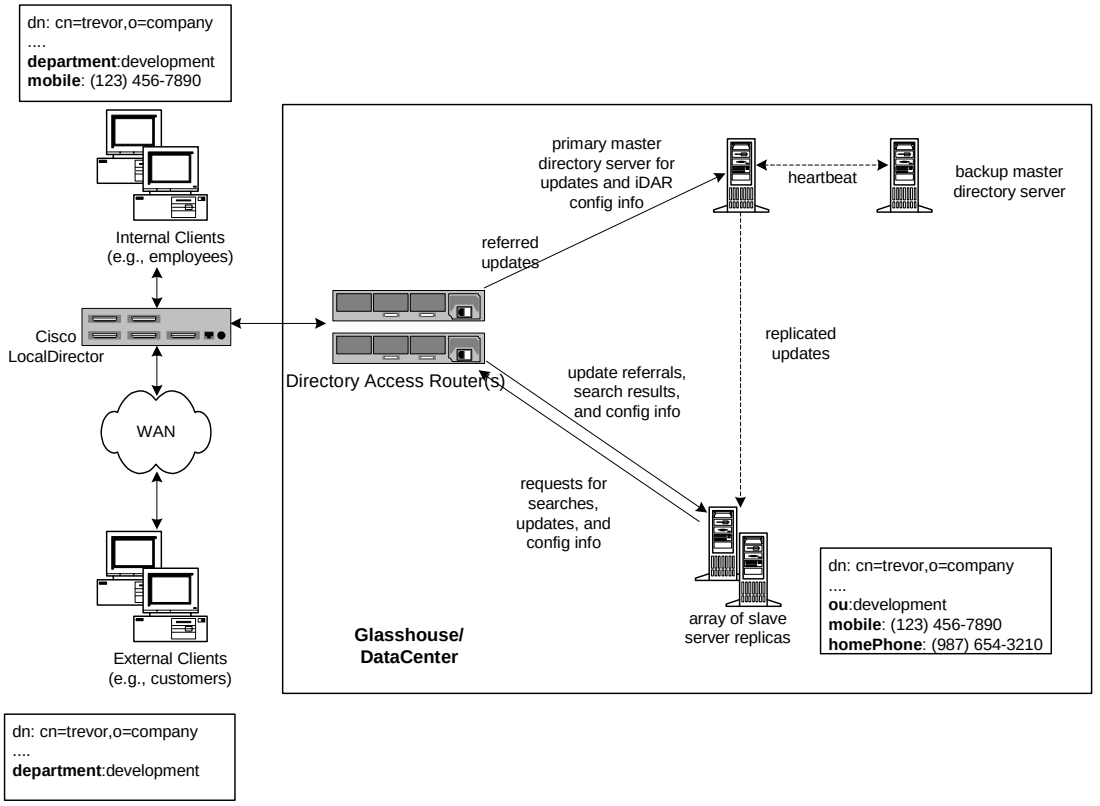

Figure 1-3 Typical Customer Deployment for a Centralized LDAP Directory Infrastructure

Customer Scenario

A large global enterprise, with customers and employees distributed throughout the world, needed to deploy a corporate white and yellow pages (electronic phone book) to reduce the cost of printing a paper phone book, to increase the accuracy of the corporate information and to reduce the use of environmental resources. The white and yellow pages information had to be available to both customers and employees with appropriate access controls. They also had to be available 24x7 and were classified as mission critical due to customers and employees being distributed throughout the world across all time zones.

Customer Deployment

The global enterprise decided to deploy a centralized LDAP directory infrastructure to support the deployment of the white and yellow pages. A centralized deployment was chosen in this instance because the white and yellow pages are for corporate employee information only. This was not to be a customer database although the intent was for customers to have access to some of the information. It was decided that the projected size of the directory database (~200,000 entries) was not sufficient to require a more complex distributed deployment model since neither scalability nor performance were anticipated to be a problem.

Since there was a high availability requirement, the enterprise decided to deploy multiple consumer directory server replicas supplied by a single supplier directory server. To remove the single point of failure introduced by a single supplier directory server the enterprise deployed a backup supplier directory server.

The iPlanet Directory Access Routers were deployed for three different reasons. First, to provide the load balancing and automatic failover and failback between all LDAP clients and the array of directory server replicas. Second, to be able to differentiate between external and internal clients and to set appropriate access controls accordingly. Third, to provide compatibility between the LDAP clients using the white and yellow pages and the directory servers themselves. In addition to utilizing a purpose built white and yellow pages application, the LDAP clients also used a number of off-the-shelf LDAP enabled applications that came with fixed schema requirements. These schema requirements did not always match the directory schema designed by the enterprise therefore requiring some basic schema attribute mapping. In addition, not all the LDAP enabled applications used by the clients were capable of processing referrals received from the directory servers correctly. The iDARs were configured to follow these referrals on behalf of the clients.

LDAP Request Flow

All client requests, whether from internal or external clients and whether search requests or update requests, are sent to the iDARs via the Cisco LocalDirector. The LocalDirector product is shown here only as an example of IP packet switching that ensures clients always have access to at least one iDAR. Multiple iDARs are deployed to ensure there is no single point of failure. The iDARs load balance all the requests received from the clients across all the consumer directory servers in the array. The iDARs will also detect the failure of any of the consumer servers and failover to the available consumer servers in the array.

Since consumer servers are read only replicas they are configured to return an LDAP referral when an update request is received from a client. This referral contains an LDAP URL pointing to the supplier directory server. When the directory server returns the referral, the iDAR recognizes it and follows the referral on behalf of the client. It binds to the supplier directory server and sends the update request to it. The supplier directory server applies the update to the supplier database and sends a response back to the client via iDAR. Subsequently, the supplier directory server will propagate the update down to the appropriate consumer directory servers.

Search requests sent by clients get routed via the iDARs to the array of consumer directory server replicas. iDARs can be configured to ‘inspect’ these search requests before sending them onto the directory servers and filter out any requests that don't meet the access control and security rules configured for a particular client group and perform any necessary mappings. iDAR can also be configured to 'inspect' the search result returned by the directory server and again perform appropriate filtering and mapping. In the example shown above, both internal and external clients have requested a search for the entry belonging to ‘Trevor’. These inbound requests are treated identically by iDAR irrespective of the client type. The directory server executes the request successfully and returns the entry for ‘Trevor’ back to the iDAR. iDAR has been configured to manipulate the search result differently depending on whether the original request came from an internal or external client. In the case of the external client, both the mobile phone number and the home phone number fields in the entry are filtered out since they are deemed to be data inappropriate for customers. Note also that the ou: development attribute / value pair has been mapped to department: development. This is necessary since one of the applications the client is using to access the directory (e.g., Outlook, Outlook Express) has fixed schema elements that do not match the schema elements deployed in the enterprise directory servers. In the case of the internal client it was determined that the mobile phone number was an important data element to share among employees whereas the home phone number was not. So for internal clients, iDAR is configured to filter out only the home phone number and to permit the client to see the mobile phone number. Note the same mapping of the ou attribute to the department attribute is also performed.

All iPlanet Directory Access Routers are configured to start up and look for their configuration in the supplier directory server. This allows for the management of multiple iDAR configurations centrally from a directory.

Deploying iDAR with a Single Firewall

Your organization's firewall must be configured to only allow LDAP clients to access the machine and port on which the iDAR is running. Typically, LDAP clients will connect on TCP port 389. This will protect the host running iDAR from clients who may potentially try to gain unauthorized access to it. Also, placing the host running the proxy server on its own LAN, by using the router switch, will protect your internal network from denial of service attacks such as flooding your network with unnecessary traffic. The firewall should also disallow LDAP access to the machine(s) and port(s) on which the LDAP Directory Server(s) are "hiding" thereby protecting the LDAP directory database(s).

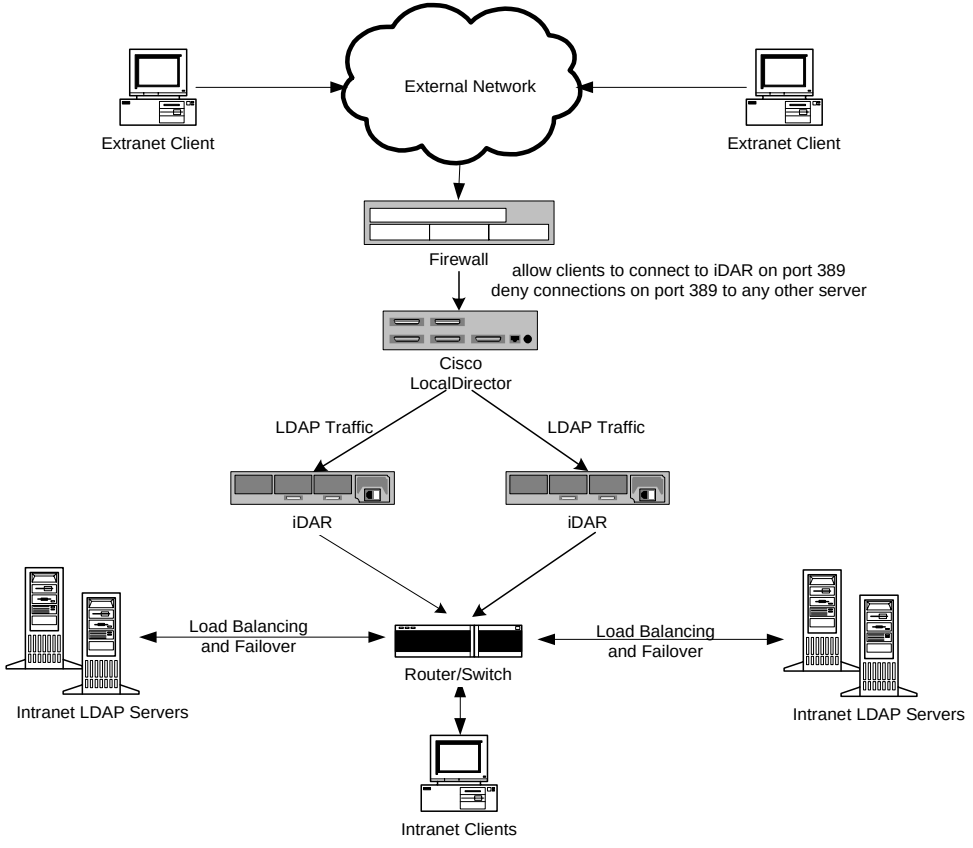

Figure 1-4 iDAR Setup with One Firewall

Deploying iDAR with Two Firewalls

This configuration shown in Figure 1-5 has all the benefits of the configuration described above with some additional security. Installing two firewalls creates a zone of control around the ‘proxies’, allowing the site administrator to rate limit traffic from the external networks. It also ensures that a compromise of one of the ‘proxy’ servers cannot be used directly to attack other machines in the interior network. Firewall A would be configured to only allow incoming packets if the destination IP address is that of the proxy handling that TCP or UDP protocol. Firewall B would be configured to only allow packets from the proxy machines that are appropriate to the servers which the proxy needs to access.

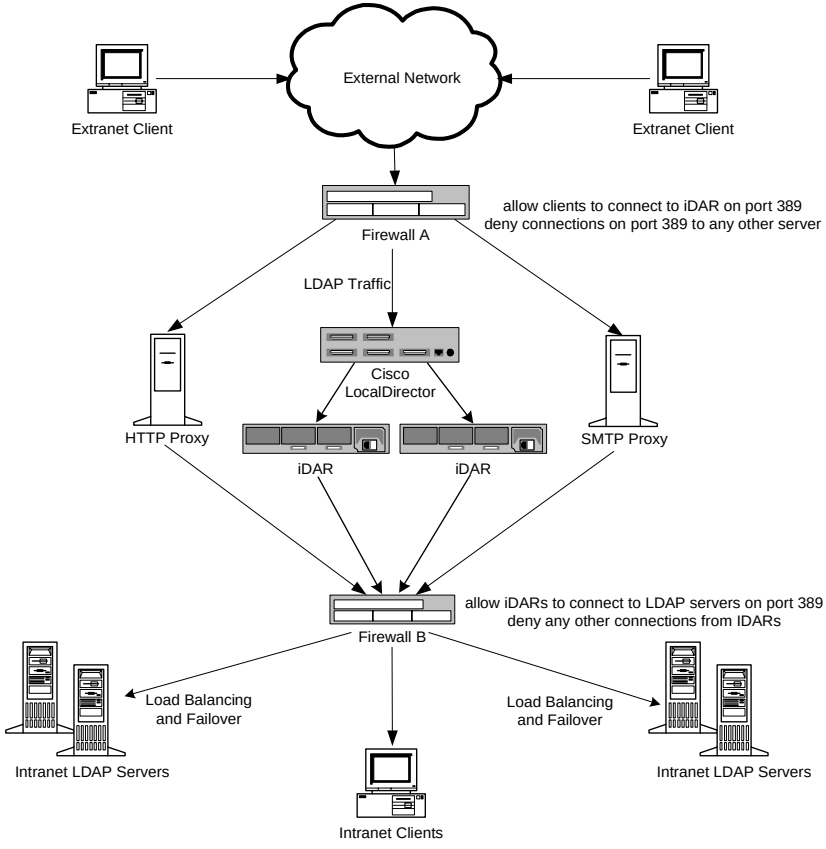

Figure 1-5 iDAR Setup with Two Firewalls

Chapter 2

Configuration Overview

Configuration of iPlanet Directory Access Router is a two step process. First, a startup configuration file must be created with information that will enable the iDAR to find its configuration (see “Startup Configuration File,” on page 48.) Then, the iDAR configuration must be created, which must reside in another file on disk or in a LDAP directory server. In the latter case, the startup configuration must also contain information about what mechanism to use to bind to the directory server. The rest of this chapter is an overview of the iDAR configuration.

Format

The iDAR configuration file (`tailor.ldif`) is in LDAP Data Interchange Format (LDIF) format. LDIF is used to represent LDAP entries in a simple text format. This format is used for iDAR configuration files for two reasons.

Firstly a simple text file is easily modified by most commonly used editors and, secondly, the use of an LDIF file allows for the storage of the iDAR configuration in an LDAP directory server. This may be important for customers that decide to distribute their LDAP infrastructure geographically but want to manage the administration of that infrastructure centrally. By choosing to keep all iDAR configurations in a central LDAP repository, central administration of the iDARs can be achieved. The configuration file is divided into a number of distinct sections, each potentially comprising multiple directory entries:

- *Server* - The server entry identifies the iDAR server instance. It is constructed by including the `ids-proxy-sch-LDAPProxy` object class. The startup configuration file points to this entry. The server entry has some basic attributes that identify where the Global Configuration object is located, the name of the server instance, and the DN's under which the Group, Property, Rule and Action object entries reside. Figure 2-1 shows these relationships graphically.

- *Global* - The global configuration entry is constructed of attributes that identify those iDAR properties that are common to all clients. Global entry is constructed by including the `ids-proxy-sch-GlobalConfiguration` object class. See “Global Configuration,” on page 59.
- *Groups* - Each group entry identifies a specific client community, and specifies the restrictions to enforce on clients that match that group. Group entries are constructed by including the `ids-proxy-sch-Group` and the `ids-proxy-sch-NetworkGroup` object classes. See “Groups Configuration,” on page 67.
- *Properties* - Each group entry may include a set of properties defined by property entries. These property entries describe more specialized restrictions on the clients. Property entries are constructed by including the `ids-proxy-sch-Property` object class together with its associated object classes. Properties are included within group entries by defining the property entry name as the value of the `ids-proxy-con-include-property` attribute. See “Properties Configuration,” on page 81.
- *Rules* - A group or property entry may include one or more rule entries. These rules are evaluated at predetermined points of execution (determined by the rule type), and one or more actions (defined separately) can be taken based on the result of the evaluation. Rule entries are constructed by including the `ids-proxy-sch-Rule` object class together with its associated object classes. Rules are included within group entries by defining the rule entry name as the value of the `ids-proxy-con-include-rule` attribute. See “Rules Configuration,” on page 91.
- *Actions* - Depending on the evaluation of a rule, one or more actions may get executed. Action entries specify what the iDAR needs to do in the case where the rule that includes them evaluates to `TRUE`. Action entries are constructed by including the `ids-proxy-sch-Action` object class together with associated object classes. Actions are included as part of rule entries by defining the action entry name as part of the value of the `ids-proxy-con-execute` attribute. See “Actions Configuration,” on page 95.

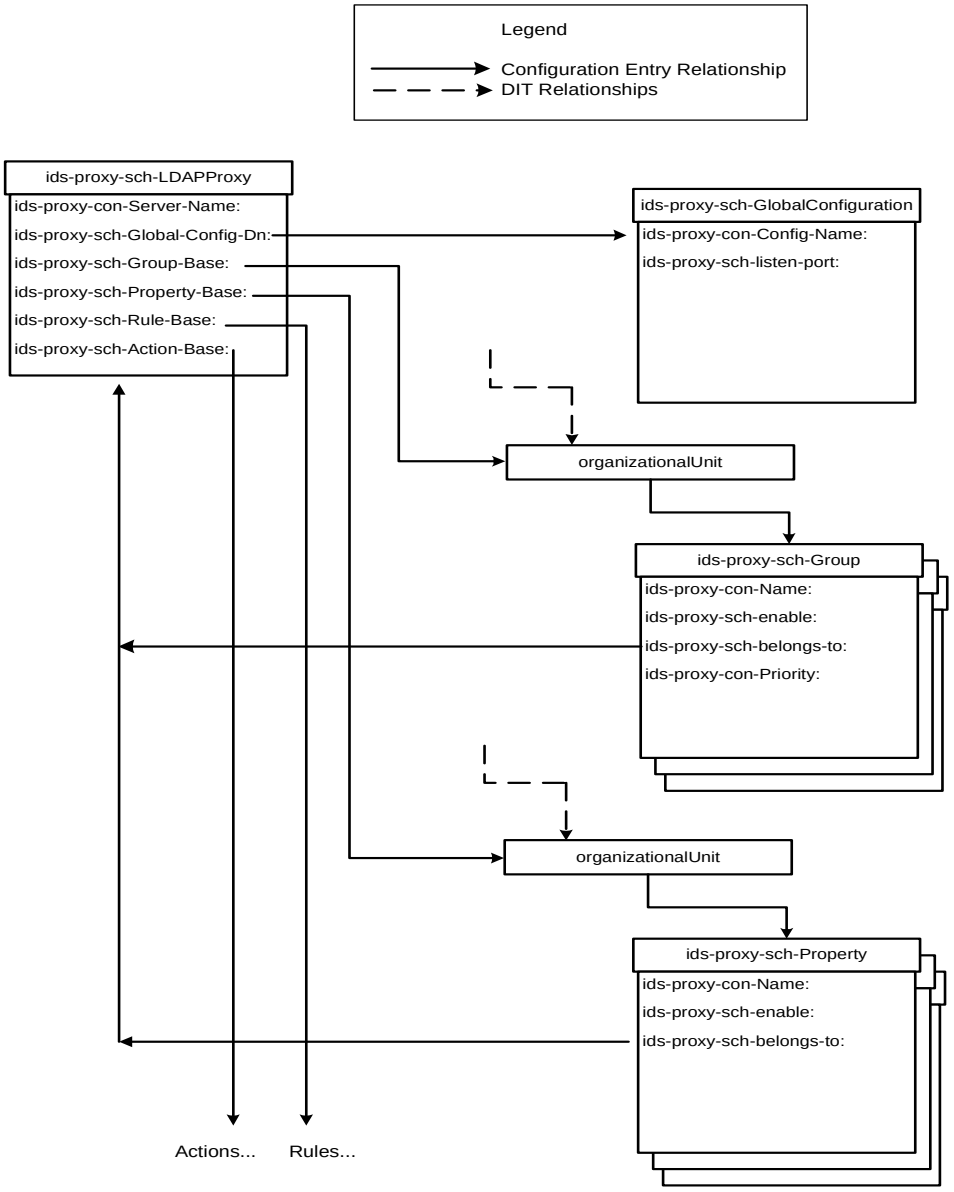

Figure 2-1 Information Model and Configuration Entry Relationship

It is worth noting at this point that the iDAR configuration file adheres to the following naming convention:

Those elements of the configuration file that specifically relate to the schema structure (as opposed to the iDAR configuration elements) start with the prefix `ids-proxy-sch` where the `sch` component indicates the schema. Examples of such elements are names of object classes (e.g., `ids-proxy-sch-Group`) and attributes with a syntax of DN (e.g., `ids-proxy-sch-Property-Base`) Those elements of the configuration file that are specifically related to the iDAR configuration start with the prefix `ids-proxy-con` where the `con` component indicates configuration. Almost all attributes defined as part of the iDAR schema (other than those with DN syntax) start with `ids-proxy-con` (e.g., `ids-proxy-con-listen-port`). This is because the vast majority of attributes relate to iDAR configuration options. The two notable exceptions are `ids-proxy-sch-Enable` and `ids-proxy-sch-belongs-to`. The first attribute determines if the entry is part of the configuration or not and the second attribute which server configuration(s) the entry belongs to.

Building a Configuration File

The configuration file is constructed by first defining a set of global entries that represent those properties of the iDAR that are common to all client groups.

Examples of such properties are:

- the host name and port number iDAR is going to listen on for incoming connections
- the maximum number of simultaneous clients connections accepted
- SSL / TLS configuration (see also “Configuring Security,” on page 53.)
- the search base for the group, property, rule and action configuration entries

A complete description of all the global entry object classes and attributes can be found in “Global Configuration,” on page 59.

Next, the property entries should be defined for all properties to be utilized within the configuration. For example, if the iDAR is going to be configured for load balancing and automatic failover and failback then a load balancing entry should be defined using the `ids-proxy-sch-LoadBalanceProperty` object class. A complete description of all the property entry object classes and attributes can be found in “Properties Configuration,” on page 81.

The action entries should be defined next. iDAR currently supports the `ids-proxy-sch-ChangeGroupAction` object class that allows iDAR to be configured to change a client from one access group to another based on the evaluation of a rule. This is typically used to support mobile users where the rule might be `ids-proxy-sch-OnBindSuccessRule`. A mobile user could connect to the iDAR with a dynamic IP address and drop into a “default” access group. The “default” access group would have the `ids-proxy-sch-OnBindSuccessRule` rule specified that only evaluates to TRUE if the bind credentials provided by the mobile user are authenticated. This rule would also have the `ids-proxy-sch-ChangeGroupAction` specified to change the mobile user's access group from the “default” to the access group the mobile user is usually assigned to when accessing iDAR with a static IP address. Actions need to be defined before rules since the rule definition requires you to specify a previously defined action. A complete description of all the action entry object classes and attributes can be found in Chapter 9 on page 95.

Rule entries should be defined next, following the definition of the action entries. iDAR currently supports the `ids-proxy-sch-OnBindSuccessRule` and the `ids-proxy-sch-OnSSLEstablishedRule` object classes. If the iDAR administrator wishes to change the access group of a client based on the evaluation of one of these two rules then both an `ids-proxy-sch-ChangeGroupAction` action entry and one of the two rule object classes need to be defined. The rule entry must include the name of the action entry as part of the value of the `ids-proxy-con-execute` attribute. A complete description of all the rule entry object classes and attributes can be found in Chapter 8 on page 91.

Having defined global, property, action and rule entries you are now at the point to define the group entries. Group entries are the core entries of the iDAR configuration file since they identify the different LDAP client groups and determine the routing, access controls, request filtering, response filtering etc. that iDAR applies to LDAP operations initiated by those clients. There are currently over forty different attributes for configuration options within the `ids-proxy-sch-Group` and `ids-proxy-sch-NetworkGroup` object classes. These can be classified into the following categories:

- group object attributes
- client identification
- including rules and properties
- SSL policy (see also “Configuring Security,” on page 53)
- LDAP bind request validation
- control of forwarding different LDAP operation types

- controlling search, compare and modify requests and responses
- referral following policy
- controlling the directory server load
- hiding a subtree of entries

A complete description of all the group entry object classes and attributes can be found in Chapter 6 on page 67.

Group entries are constructed by first considering the client communities you need to support or differentiate between. Identify the clients making up a community and create a group for them by first identifying them in the group entry by correctly defining the `ids-proxy-con-client` multi-valued attribute. If you have previously defined property and rule entries that are appropriate for this group then include them by configuring the `ids-proxy-con-include-property` and/or the `ids-proxy-con-include-rule` attributes. Using the attributes defined for the group object classes, configure additional controls for this group as appropriate. Then move onto the next group definition.

Configuration File Build Tool

To help iDAR administrators to get started with configuring the iDAR, iDAR ships with a command line configuration tool that allows you to build a number of different configuration files. A complete description of the iDAR configuration file build tool can be found in Chapter 3 on page 41.

The command line tool, `tailor.tcl` allows the user to specify command line parameters that the tool then uses to automatically generate a working iDAR configuration file in LDIF format. Note that there are limitations to what the command line tool can generate so in most cases it should be considered a tool for building a basic working configuration file that can then be modified to provide further customization.

Currently, the build tool supports generating configuration files for the following customer scenarios:

- The bare minimum, “out-of-the-box” configuration to get the user up and running with a working iDAR configuration immediately. This configuration is a simple “passthru” from the LDAP client to a single LDAP directory server via iDAR. There is no filtering of requests or responses and no additional access controls. Only one client group will be generated.

- A more sophisticated configuration that incorporates load balancing and automatic failover and failback into the basic configuration mentioned above. The user can specify any number of LDAP directory servers and load balance the client load across them all. Only one client group will be generated.
- The third scenario is probably the most complex. This scenario will allow the LDAP client to bind to the LDAP directory server either anonymously or with bind credentials and treat the two differently. The `tailor.tcl` tool will generate two groups: one that is employed when a client binds with bind credentials and another group that acts as a "default" group. All clients that bind anonymously will fall into the "default" group and will be limited to search operations only. Those clients that bind with bind credentials will initially drop into the "default" group. The "default" group has the `ids-proxy-sch-OnBindSuccessRule` rule included in it. This rule has the `ids-proxy-sch-ChangeGroupAction` action associated with it. So clients that bind with bind credentials that are subsequently authenticated cause the rule to evaluate to `TRUE` and the action to get executed that changes the group the client belongs to from the "default" group to a group with more access rights. The concept behind these two groups is to grant more access rights to those clients binding with bind credentials and to restrict the access rights of clients binding anonymously.

The `tailor.tcl` tool can be helpful in generating LDIF configuration files for the deployment scenarios described previously. For example, `tailor.tcl` could be used to generate a configuration file for the typical customer deployment of an internal high availability configuration described on page 19. You would use `tailor.tcl` load balancing scenario with the following syntax:

```
tailor.tcl -server ldaphost-1 -server ldaphost-2 -server ldaphost-3\
           -server ldaphost-4
```

No further customization of the configuration file generated by `tailor.tcl` is necessary. The firewall deployed in this scenario should ensure there is no external access to either the iDAR or the LDAP directory server. However, for additional peace of mind the user could customize the configuration file by identifying "internal" LDAP clients via the `ids-proxy-con-Client` multi-valued attribute in the `ids-proxy-sch-NetworkGroup` object class. Any client that attempts to connect to the iDAR but hasn't been identified by the `ids-proxy-con-Client` multi-valued attribute will be rejected.

Don't forget to use TCP wrapper or equivalent program on UNIX platforms and TCP/IP port filtering on Windows NT to ensure access to the LDAP directory servers is only permitted for the iDARs you deploy.

iDAR Decision Functions

This section describes the flow of control in iDAR for some specific functionalities.

Establishing Group on Connection

When a client makes a connection to iDAR, it checks the `ids-proxy-con-Client` attribute (page 68) in `ids-proxy-sch-NetworkGroup` object entries until it finds a match. The `ids-proxy-sch-NetworkGroup` objects are tried in highest to lowest priority defined by the `ids-proxy-con-priority` attribute (page 68). iDAR places the client in the first group whose `ids-proxy-con-client` attribute matches the IP address of the client. If no matching groups are found, the connection is closed.

Change Group on Bind

When the client initially connects it is placed in a group based on its IP address. The client can be moved to a different group with different access controls when it binds to a directory. To accomplish this, the initial group object must include a rule object that is evaluated on a successful bind operation. If the rule evaluates to TRUE, the change group action is taken to move the client to a different group. Figure 2-2 illustrates this functionality.

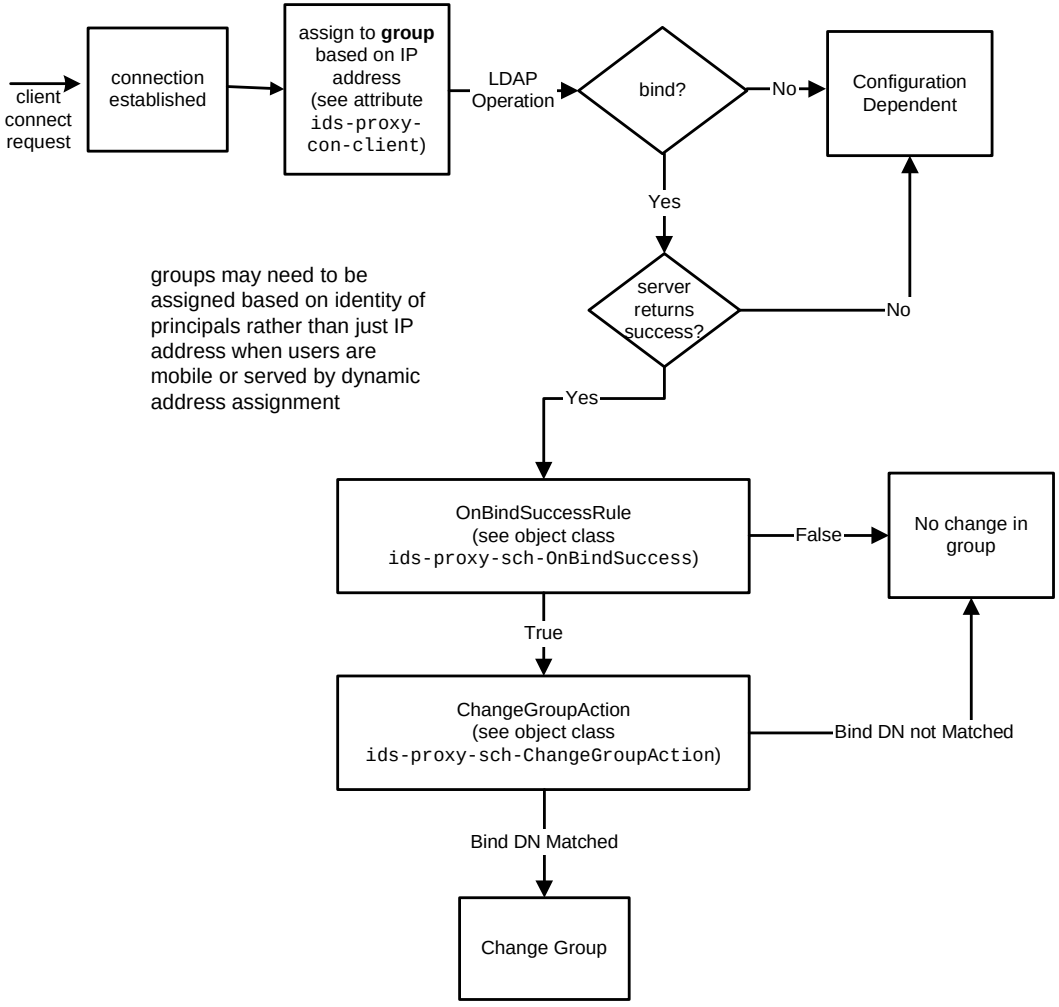

Figure 2-2 Change Group on Bind Flow Chart

Change Group on Establishment of TLS

Similar to the change group on bind mechanism a client can change group when it successfully establishes an TLS session. The SSL Established rule is evaluated when the client establishes TLS, following which the Change Group action follows. This functionality is illustrated in Figure 2-3.

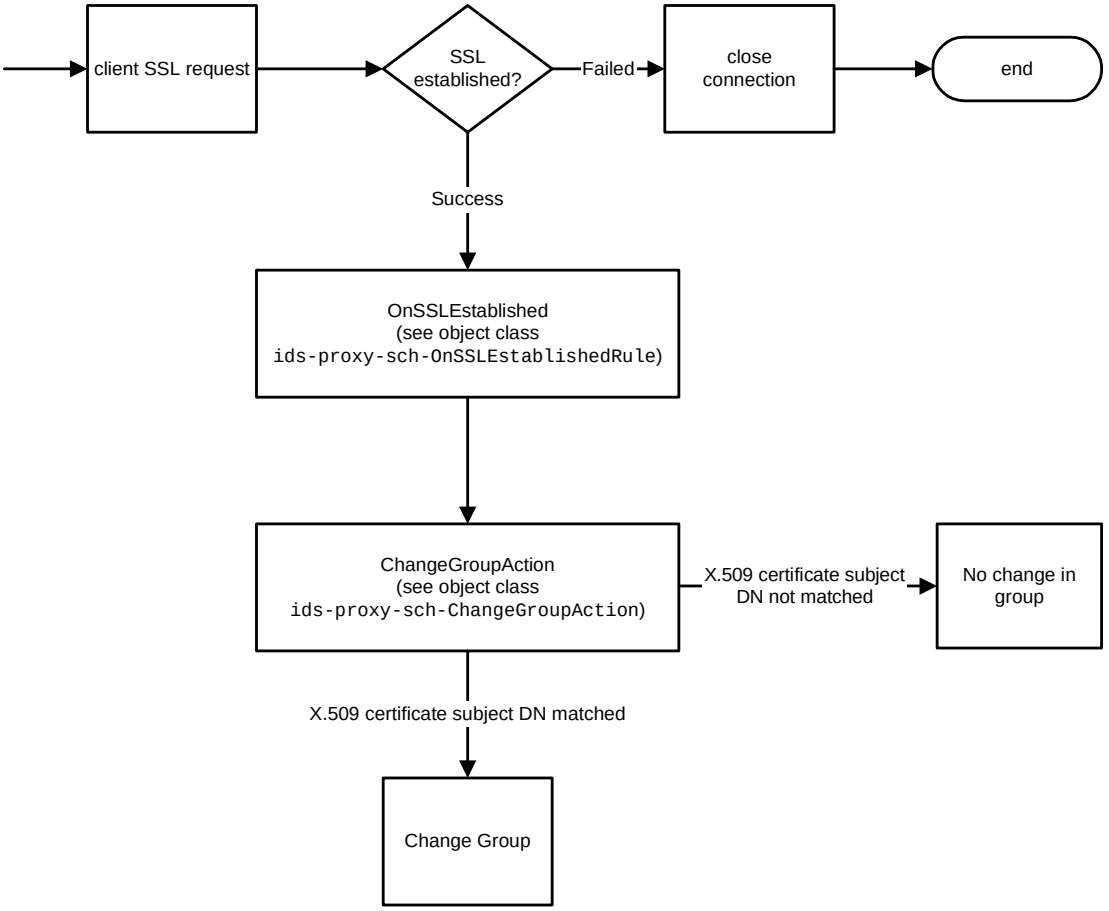

Figure 2-3 Change Group on Establishment of TLS Flow Chart

High Availability Setup

If you have configured more than one backend directory server, then you can set up iDAR to load balance across these and fail over to another if one of the backend servers go down. In order to do this, you must create a Load Balance Property (see page 82) and include it in the Group object for which you want to Load Balance. You will also need to create LDAP Server Properties (see page 86) for each of your backend servers and include it in the Load Balance Property. You must specify the amount of load in percentage of total load each of you backend servers should handle in the Load Balance Property object. With this setup, iDAR will redistribute load across its backend directory servers if one of them goes down. It will fail over clients from one server to another in case the former goes down. iDAR will also fail over if the network link between itself and the LDAP server is down or if the LDAP server becomes unresponsive (see `ids-proxy-con-keepalive-interval` on page 88). Note that iDAR 2.1 is unable to fail over if the client was bound using a SASL mechanism.

Following Referrals

iDAR can be setup to follow referrals for LDAPv2 clients that cannot do so on their own. Your backend LDAP directory server must be capable of sending referrals, i.e., it must support LDAP v3 standards. Configure iDAR to use LDAP v3 between itself and the backend LDAP server in order for iDAR to receive referrals form the directory server (see `ids-proxy-con-use-version` on page 87). Then set your referral (see “Controlling the Return of Referrals,” on page 78) and continuation referral policy (see `ids-proxy-con-search-reference` on page 77) for the group of clients you want referrals to be followed.

Chapter 3

Configuration Generation Tool

iPlanet Directory Access Router's operation requires the creation of two independent sets of configuration information. The main iDAR configuration is large and can be unwieldy, (in contrast to the startup configuration,) and can be enumerated within a text file or it can be stored in an LDAP server. The advantage of storing the main configuration in an LDAP server is that allows central administration of distributed set of iDAR deployments.

The startup configuration file possesses only the information necessary for iDAR to find the main configuration. Unlike the main configuration, which could be in an LDAP server, the startup configuration must be a text file on a file system accessible by the machine hosting the iDAR process.

Configuration Assistance

To simplify the generation of these two configuration files, iDAR is packaged with a command line tool, `tailor.tcl`, that will emit working configuration files from a minimal set of user supplied parameters. The tool has been created to ease the chore of expressing the relation between iDAR, its clients, and its LDAP servers. However, the tool does not attempt to support some of iDAR's more complex configuration subtleties such as attribute renaming, subtree hiding, etc. Complex configuration such as those are best accomplished by using the tool as the starting point and then completing the tailoring process via a text editor of your choice.

Configuration Tool's Parameters

The configuration tool `tailor.tcl` assumes it is being run on the environment that will host the iDAR that's being configured. In non-Windows platforms, it also verifies that it can resolve any and all hostnames you stipulate.

The tool has the following syntax:

```
tailor.tcl [-startup outFile] [-ldif outFile] [-listen port]
           [-suffix dn] [-objectclass class]
           [-bind_dn dn -bind_pw password -url [ldap://]host[:port]]
           [-onbind [[ldap://]host[:port] [name] [secureport]]]
           [-server [ldap://]host[:port] [name] [secureport]]
           [-ssl_key file -ssl_cert file [-ssl_version number] [-ldaps port]]
           [-remember [outFile]] [-recall [file]]
```

The parameters follow a fixed pattern: a hyphenated key word / option followed by one or more values to associate with the option.

The recognized options are (grouped by subject):

`-ldif outFile`

The `-ldif` option informs the configuration tool where to store the textual representation of the subsequently generated configuration file. The resulting text file can either be loaded directly by iDAR or imported into an LDAP server with certain provisos (i.e., suffix is acceptable to the server and the server has supporting iDAR attributes and class configuration information).

The supplied `outFile` should be stipulated as a complete and absolute path to the destination file. An absolute path is required. The configuration tool will verify that you have sufficient permissions to create or replace the given file.

If this option is omitted, the tool will name the output file `tailor.ldif` and place it in the `etc` subdirectory subordinate to the installation directory. No attempt is made by the tool to import the resulting configuration into an LDAP environment. That step requires end user manual intervention.

`-bind_pw password`
`-bind_dn dn`
`-url [ldap://]host[:port]`

These options are used in the case were iDAR is to retrieve its main configuration from an LDAP server. The `-bind_dn` and `-bind_pw`, `-url` options are recorded into iDAR's startup configuration where they'll be used by iDAR when it contacts the configuration's host server. Note: all three of these options are required if any one of them is stipulated.

`-startup outFile`

This option instructs the tool where to place iDAR's startup configuration file.

Corollary to the `-ldif` option, the supplied `outFile` here should also be a complete and absolute path to the destination file. Again, an absolute path is required. The configuration tool will verify that you have sufficient permissions to create or replace the given file.

If this option is omitted, the tool will name the output file `tailor.txt` and place it in the `etc` subdirectory subordinate to the installation directory.

`-listen port`

This option provides the facility to dictate which TCP port number on which iDAR will listen for new connections. This option sets the value stipulated for `ids-proxy-con-listen-port` enumerated for the `ids-proxy-sch-GlobalConfiguration` objectclass entry.

The tool assumes a value of 389 for this option when it is not stipulated.

```
-onbind  [[ldap://]host[:port] [name] [secureport]]]
-server  [ldap://]host[:port] [name] [secureport]]
```

The `-server` option is used to enumerate the hosts that iDAR can contact, thus more than one `-server` could be specified. The optional `name` and `secureport` values are used to associate a name to the given host and to indicate which TCP port (if any) the host expects secured connections. Those two optional values may be stipulated in any order. In addition, either or both the `name` and the `secureport` directives may be omitted. The name given will be used to form part of the distinguished name in the properties section of the resulting configuration; thus care should be exercised to avoid duplicating names. Note: the tool generates a default name in the case where one isn't given. Those default names begin with "server-" and are appended with a sequentially generated number to avoid duplication.

If the tool is given more than one `-server` option, the tool will emit a configuration that reflects load balancing directives. Note: iDAR assumes that a set of servers that are being load balanced are of equal content and of equal functionality. If the configuration tool isn't presented a `-server` option, it will act as if `-server ldap://localhost:389` has been stipulated.

The `-onbind` directive informs the configuration tool that it is to establish two distinct groups within the configuration: one group for anonymous binds and another for non-anonymous binds. The `-onbind` option's syntax is similar to `-server`'s syntax with the notable exception that it does not require the specification of a host name. If the host name is omitted then `-onbind` will assume that the set of hosts to contact on successful bind is the same set as

stipulated by the -server option(s). The key to recognize here is that connections that fall within -onbind's purgative will still contact the same machines but the connections will have an expanded set of available operations. To the contrary, stipulating one or more -onbind hosts removes the possibility of supporting the rebinding of a SASL connection.

If you use -onbind to enumerate more than one host, then the configuration tool will set up additional properties to accommodate load balancing across the set of -onbind hosts.

```
-ssl_cert file
-ssl_key file
-ssl_version number
-ldaps port
```

These options enable iDAR to support encrypted connections from its clients. These options do not effect iDAR's ability to encrypt connections between itself and its corresponding servers. The -ssl_cert parameter specifies the absolute path to the file that possess the certificate that iDAR should employ. The -ssl_key option identifies the file that possesses the key iDAR should use in conjunction with the given certificate. These options map their values to the ids-proxy-sch-GlobalConfiguration objectclass' ids-proxy-con-ssl-key and ids-proxy-con-ssl-cert attributes.

If both -ssl_key and -ssl_cert are specified then the tool will also allow the specification of -ssl_version and/or -ldaps.

The -ssl_version must be one of the following values: 20, 23, 30, or 31. A value of 20 configures iDAR for SSL version 2 only, 30 selects SSL version 3 only, 23 permits either SSL version 2 or version 3 connections and 31 stipulates TLS version 1 only. This option maps its value to the ids-proxy-sch-GlobalConfiguration objectclass' ids-proxy-con-ssl-version attribute.

Setting a value for -ldaps instructs iDAR to listen on the given TCP port (in addition to the TCP port specified by the -listen directive) with the subtle difference being that only encrypted connections will be accepted on the -ldaps specified TCP port. This option maps its value to the ids-proxy-sch-GlobalConfiguration objectclass' ids-proxy-con-ldaps-port attribute.

`-suffix dn`

The tool will append the given suffix to all of the distinguished names used internally within the generated configuration. This option adds value to the resulting configuration if and only if the intent is to store the configuration in an LDAP server. Otherwise, the suffix just serves to clutter the distinguished names. The tool does assume that the given suffix already exists in the destined LDAP server.

`-objectclass class`

If your production deployment model calls for iDAR to keep its configuration in an LDAP server and you are employing schema checking, then you will probably find that you will need to supplement the configuration's entries with one or more objectclasses. By default, the configuration tool will tag each entry it creates with an objectclass of `ids-proxy-top`, but by stipulating this option (one or multiple times) you can specify the objectclass(es) to use instead.

`-remember outFile`
`-recall file`

The `-remember` and `-recall` options provide a mechanism to support the recording of the command line parameters and their subsequent reuse. The `-remember` and `-recall` directives can be used simultaneously or independently of each other. When used together, the given file names may or may not be the same depending if you wish to overlay the given input (same names) or evolve the input (different names). The `-remember` and `-recall` options can be used in conjunction with any other options as the tool merely merges the current command line with the recalled values.

Configuration Examples

First, let's define the pseudo environment we have at our disposal.

- Servers:

```
ldap://red.iplanet.com:10389
ldap://blue.iplanet.com:389
ldap://pink.iplanet.com:10205
```

- iDAR:

```
ldap://bitter.iplanet.com:389
```

- Encrypted iDAR:

```
ldap://bitter.iplanet.com:636
```

Observation: since `bitter.iplanet.com` is declared as supporting encryption then the iDAR stationed at `bitter.iplanet.com` will support “Start TLS” operations over its non-secured port as well.

Now let's further assume that DNS can resolve the aforementioned machines merely by their hostname, i.e., `red`, `blue`, `green`, etc. And the last assumption: the subsequent examples assume that the tool is being run from a Unix environment. Note: Windows users must prepend the `tailor.tcl` command line with `iDARtcl.exe` that is found in the `bin` subdirectory subordinate to iDAR’s installation directory.

Given these assumptions, we are now in a position to show you how the example configurations shown in Appendix A, “Examples: tailor.ldif,” on page 105 were generated.

The tool is capable of generating four different classes of configurations: straight through, load balancing, binding based operation filtering, and load balancing with binding based operation filtering.

Straight Through Configuration

In the straight through configuration, iDAR is not adding much value to the overall process, rather, it is just relaying the information from its clients to a server. The general expectation is that the straight through configuration would not have much application in a production deployment but it does have value when it comes to building confidence that your iDAR is inter-operating with your clients and your LDAP server. In this particular case, the command line tool is given only information regarding what TCP port number the iDAR process should listen on and what single server should iDAR contact.

```
tailor.tcl -listen 389 -server ldap://pink.iplanet.com:10205
```

This will yield the first example configuration file found Appendix A, “Examples: tailor.ldif,”on page 105. The output files were named `/opt/iDAR/etc/tailor.txt` and `/opt/iDAR/etc/tailor.ldif` as we didn't supply the `-startup` and `-ldif` options.

Load Balancing Configuration

This scenario expands upon the straight through configuration via the stipulation of more than one server to contact. The configuration tool will emit directives specifying that the given set of servers is to divide the connection load among themselves.

```
tailor.tcl -server pink:10205 -server red:10389 -server blue
```

Here the tool generates a configuration where `pink.iplanet.com` will receive 34 percent, `red.iplanet.com` receives 33 percent and `blue.iplant.com` receives 33 percent of the connections. The tool will apportion the set of servers equally based on the cardinality of the set stipulated. To provide 100 percent coverage, any remaining percentage is added to the first enumerated server.

Again you'll find the resulting configuration files as one of the examples in Appendix A on page 105.

Binding Based Operation Filtering

Upon examination of the generated configuration files from the previous scenarios (see Appendix A) you'll note that the configuration tool generated only one group. (Note: all connections through iDAR resolve to a group through which permissions, actions and rules are derived.) In this scenario, the tool will generate two groups: one that is employed when a connection utilizes a non-anonymous bind and another group that acts as a catch all.

The idea behind these two groups is to extend permissions to non-anonymous binding connections and to restrict the permissions to anonymous binding connections; specifically anonymous clients will only be given search capabilities.

```
tailor.tcl -server blue -onbind
```

Load Balancing and Binding Based Operation Filtering

Creating a configuration that capitalizes on these two types of functions merely requires combining their parameter lists.

```
tailor.tcl -server pink:10205 -server red:10389 -server blue -onbind
```

The emitted configuration will instruct iDAR to load balance over the `pink`, `red`, and `blue` servers and anonymous binding connections will only possess search privileges.

See Appendix A for a listing of the resulting configuration file.

Startup Configuration File

The `tailor.txt` file contains the bootstrap information iDAR needs to locate its main configuration. The directives in this file dictate if iDAR will utilize an additional file for its main configuration or if iDAR will solicit its main configuration from an LDAP server. By default, iDAR expects to find the startup configuration file, `tailor.txt`, in the `etc` subdirectory of the installation directory. Note: via the use of the command line parameter `-t`, iDAR can be instructed to use an alternate file as its startup configuration file.

As an aide in supporting high availability configurations, a startup configuration file may list several contact points for the main configuration's retrieval. Contact points are delineated within the startup configuration file by the use of two keywords: `Begin` and `End`. iDAR will process the contact information one by one in the order given. iDAR's actions on each contact point depends on the type of the given contact point (which is either an LDAP URL or an absolute path name to a file).

For LDAP URLs based contact points, iDAR will attempt to contact the given host. If the host is unwilling or unable to return a configuration then iDAR will proceed to its next contact point (if any). If the host returns a configuration then iDAR will edit the contents returned and will then either begin following the main configuration's directives or end its execution if the configuration was deemed invalid.

For file based contact points, iDAR will attempt to load the given file as its main configuration. If the specified configuration is missing or is deemed invalid, iDAR will end its execution. iDAR will not attempt to move to the next contact point once it encounters a file based contact point.

In the case where iDAR is retrieving its main configuration from an LDAP host, iDAR can bind to the host using one of three methods: anonymous, simple or using SASL.

Anonymous binding is accomplished by omitting the `configuration_bind_pw` and `configuration_bind_dn` directives. In other words, your startup configuration's contact information would only specify a `configuration_url` directive and nothing else.

Simple binding is supported through the use of both the `configuration_bind_pw` and the `configuration_bind_dn` directives.

SASL binding requires the specification of the `sasl_bind_mechanism`, `conguration_bind_pw` and one (and only one) of the following directives: either `configuration_bind_dn` or `configuration_username`.

Startup configuration's keywords

Each enumerated contact point uses the keyword `Begin` to signify the start of a contact point entry. Conversely, each contact point entry is terminated by the keyword `End`. Every directive stipulated in a startup configuration file is expressed on a line by itself. Line continuation within the startup configuration is not recognized nor supported. The configuration's options are specified via an option, colon, and value triplet.

configuration_url

The `configuration_url` option specifies either an LDAP directory server and the distinguished name of the entry in that directory where the iDAR configuration is stored, or a local file in LDIF format. For example, if the iDAR configuration is stored in an LDAP directory on host `ldap.innosoft.com` with the LDAP service running on port `389` and the distinguished name of the iDAR entry is "`ids-proxy-con-Server-Name=iDAR`", then the following should be added to the configuration file:

```
Begin
configuration_url:
ldap://ldap.innosoft.com:389/ids-proxy-con-Server-Name=iDAR
End
```

If the configuration is to be kept in an LDAP server, you would probably need to specify a suffix following the `ids-proxy-con-Server-Name=iDAR` in order to maintain compatibility with the host directory's naming context. For example:

```
Begin
configuration_url:
ldap://ldap.innosoft.com:389/ids-proxy-con-Server-Name=iDAR,
ou=services, dc=iPlanet, dc=com
End
```

Each startup configuration directive should be specified as one contiguous line within the configuration file. In other words: do not interpret the line wrapping in the aforementioned `configuration_url` examples as an instruction to insert a line break into your configuration file.

In the case where the configuration is stored in a LDIF formatted file, i.e., `/opt/iDAR/etc/tailor.ldif`, the following should be added to the configuration file:

```
Begin
configuration_url:
file:///opt/iDAR/etc/tailor.ldif#ids-proxy-con-Server-Name=iDAR
End
```

configuration_bind_dn

The `configuration_bind_dn` option specifies the distinguished name to use when iDAR binds to the LDAP server specified in the `configuration_url` option. If this option is not specified, and an LDAP directory is specified as the value of the `configuration_url` option, then iDAR binds anonymously to the directory. If it is specified, iDAR will perform a simple bind with this distinguished name and the value of `configuration_bind_pw` as the password. For example:

```
Begin
configuration_url:
ldap://ldap.innosoft.com:389/ids-proxy-con-Server-Name=iDAR
configuration_bind_dn: cn=Directory Manager
configuration_bind_pw: secret
End
```

The `configuration_bind_dn` option is not needed and ignored if `configuration_url` is of the “file” form. Note: the `configuration_bind_dn` and `configuration_username` directives are mutually exclusive.

configuration_bind_pw

The `configuration_bind_pw` option is used to specify the password to use when binding to the LDAP directory. The directive is used to specify the password to use for either simple or SASL based binding. In order to preserve security, the configuration file must be protected against unauthorized reading. The `configuration_bind_pw` option is not needed and ignored if `configuration_url` is of the “file” form. (See `configuration_bind_dn` for an example.)

configuration_username

The `configuration_username` option specifies the username to use when iDAR binds to the LDAP server specified in the `configuration_url` option. This option is only used if SASL bind mechanism is used. Note: the `configuration_bind_dn` and `configuration_username` directives are mutually exclusive.

```
Begin
configuration_url:
ldap://ldap.innosoft.com:389/ids-proxy-con-Server-Name=iDAR
configuration_username: administrator
configuration_bind_pw: secret
sasl_bind_mechanism: CRAM-MD5
End
```

sasl_bind_mechanism

The `sasl_bind_mechanism` option can be set to either `CRAM-MD5` or `DIGEST-MD5` depending on which SASL bind mechanism you want iDAR to use. iDAR will perform either a simple bind or an anonymous bind if this option is absent. `DIGEST-MD5` provides a higher level of security than `CRAM-MD5` but `DIGEST-MD5` has not been as widely adopted as `CRAM-MD5`.

Chapter 4

Configuring Security

iPlanet Directory Access Router supports SSL/TLS for secure communication between its clients and backend directory servers. It also supports SASL bind mechanisms for client authentication. The following sections describe how to configure SSL/TLS functionality in iDAR and what kinds of SASL mechanisms are supported.

Configuring TLS/SSL in iDAR

Transport Layer Security (TLS) is the internet standard protocol for secure communication between two end points. TLS/SSL is widely used with the HTTP protocol on the Web. Because of US government restrictions on export of cryptography, TLS is supported in the following ways in iDAR:

- For US, Canadian, and customers in specific industry segments, iDAR supports 168-bit encryption.
- iDAR supports 56-bit encryption for export purposes to all other customers.

iDAR supports TLS/SSL for encryption purposes only. An encrypted TLS/SSL connection between clients and iDAR will protect the data transmitted over that connection. iDAR can also be configured to always create a SSL session between itself and the backend directory server if the network between itself and the backend LDAP server is not safe, and the directory server supports TLS/SSL.

iDAR cannot act as a “passthru” for an “end-to-end” TLS/SSL connection between the client and a backend directory server because iDAR has to be configured to bind to a directory server with one set of bind credentials and does not use the bind credentials presented by the client. Furthermore, a compliant TLS/SSL implementation would treat the iDAR as performing a “man-in-the-middle” attack, since proxying a client's credentials is not allowed in strong authentication.

iDAR can be configured to establish TLS/SSL between itself and the client and itself and the server in many different combinations. It supports both, the “Start TLS” method where clients establish TLS/SSL session over the standard LDAP port and the “alternate port 636” method where clients connect to an alternate secure port if they want to establish TLS/SSL.

The following matrix summarizes some of the possible configuration scenarios. A complete matrix can be found in Appendix B, “TLS/SSL Scenario Matrix,” on page 123. The columns represent connections between client and iDAR, and the rows represent connections between the iDAR and the backend LDAP directory server.

Table 4-1 TLS/SSL Configuration Scenarios

		CLIENT	
	iDAR	No TLS	TLS using alternate port 636
SERVER	No TLS	X	X
	TLS using alternate port 636	X	X

Steps to Configure TLS/SSL Support

1. Generate a certificate request and a private key for iDAR using the supported `certreq` utility (see “Generating a TLS Key Pair,” on page 56).
2. Send the proxy server's certificate request generated by `certreq` to a registered Certification Authority (CA) of your choice, e.g., Thawte, and obtain a certificate from the CA.
3. In the `ids-proxy-sch-GlobalConfiguration` object entry, set the `ids-proxy-con-ldaps-port` attribute to the port number on which to listen for LDAPS (LDAP over TLS/SSL) connections. This is optional and is necessary for clients that use the alternative port 636 method to establish TLS/SSL.
4. Make the following changes to the `ids-proxy-con-GlobalConfiguration` object entry.
   a. Set the `ids-proxy-con-ssl-cert` attribute to the location pathname of the file on disk containing the server's own certificate signed by the CA.

b. If you are configuring iDAR to communicate with a directory server over TLS/SSL and the directory server in question supports receiving certificates, set the `ids-proxy-con-send-cert-as-client` attribute to `TRUE`. Otherwise allow it to default to `FALSE`.

c. Set the version of TLS that iDAR will use for establishing TLS with its clients or the backend directory server by setting the value of `ids-proxy-con-server-ssl-version` and/or `ids-proxy-con-client-ssl-version` to the appropriate version number. (see `ids-proxy-con-server-ssl-version`, `ids-proxy-con-client-ssl-version` on page 64). The defaults interoperate with iPlanet Directory products.

d. If you want the iDAR to verify the certificates presented to it you must set the `ids-proxy-con-ssl-cafile` to the file containing the root certificates of CA's that iDAR can trust. This file must contain the list of certificates in PEM format.

e. If you want your clients to always present a certificate when they initiate a TLS session with iDAR, set `ids-proxy-con-ssl-cert-required` attribute to `TRUE`.

5. For each Group definition configured, set the `ids-proxy-con-ssl-policy` attribute in the `ids-proxy-sch-NetworkGroup` object entry as appropriate depending on whether you want to force the client to start a TLS session before sending any LDAP operation, leave the decision to the client, or disallow the client to start a TLS session. See `ids-proxy-con-ssl-policy` on page 70.

6. Make the following changes to each of the `ids-proxy-sch-LDAPServer` object entry.

a. Set `ids-proxy-con-link-security-policy` to appropriate value (see `ids-proxy-con-link-security-policy` on page 88) so that iDAR will always establish SSL/TLS to the backend server, never establish TLS/SSL to the backend server, or only establish SSL/TLS with the backend server when the client does the same to iDAR.

b. Set `ids-proxy-con-x509cert-subject` to the DN of the backend server that iDAR must receive as value of the `subject` attribute in the X509 certificate it will receive from the backend server. iDAR will accept any name if this attribute is not set.

Generating a TLS Key Pair

In order for TLS to operate, the server must have a private key and a certificate. The `certreq` program provided in the `sbin` directory can be used to generate a private key and a certificate request for the server.

In order to have iDAR's certificate certified by a Certification Authority (CA). You should send the server's certificate request generated by `certreq` to the CA, and receive back a certificate for the server signed by the CA.

Generating Files with certreq

The `certreq` program is invoked as follows:

```
certreq -dn dn -reqout filename -keyout filename
[-dsaparms filename] [-bits bits]
```

The arguments are:

`-reqout filename`

`certreq` writes a certificate request to this file. This file should be sent to a Certification Authority to be signed.

`-keyout filename`

`certreq` writes the private key to this file. Make sure that only the administrator has read permission for this file.

`-dn dn`

The distinguished name of the server or client.

`-dsaparms filename`

A Digital Signature Algorithm (DSA) key is based on a set of shared parameters. Two parameter sets are provided in the lib directory: `dsa512.pem` for 512 bit keys and `dsa1024.pem` for 1024 bit keys. This argument is not needed if an RSA key is to be generated.

`-bits bits`

The bits field should contain the number of bits of key to generate for the RSA algorithm, such as 512 or 1024.

For example, to create a RSA key in the file `k` and a certificate request in the file `r`:

```
certreq -reqout r -keyout k -dn "dc=iPlanet, dc=com" -bits 1024
```

To create a DSA key in the file `k` and the certificate request in the file `r`:

```
certreq -reqout r -keyout k -dn "dc=iPlanet, dc=com"
-dsaparms ../lib/dsa512.pem
```

Key File

When RSA is being used, the file specified by the `-keyout` parameter will contain the RSA private key similar to the following:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDrYc78Q9PnU8Q5d0SfFyNXI84sRtGP9NXgP70XxY6Wdg3xoQAx
Z/xWyE/0dx0xRR8bzCpi25eTVyYbyQMxY6yu2OxvsywDYwkAN2bRBgePUMVSlx2j
Zi62Fr2CzsAaajbOO0yqXFP/gyjXphYorbOvyG78Xp3vFIesWYl6GgzglwIDAQAB
AoGBAOIm1E9N/+/XbMXl0Nmlyn+z2Ch0Vm6gx0kxFEYduvTXMmiAzwWpKipbRW7V
bwfiqJP1opfe8hBPgD7b8CRo5wQziQlypp7JnFnDjL7U/QARzRATUax/t8RdCxQ4
PrZ45At/amZpkkWCozYfXA+57LlhW535KxLcstMTNpNu7YPhAkEA+PgUO09eSzMs
YyAfvVlctmaLDmcapMLRlNp5oAnUYvwK17l0mCrATTgT3w1rQ1h2zKixlUSooqy5
W3fTN19ZMQJBAPIHgJvjGGKTvDVWM4kKOQgUm1KqaUgwBqfBnpSfA+kgJenSgRs9
aX1OM6xa0w+V3zwMSeF5VBLZ4X94kgsEZEcCQA/2iAWNfzQ/IbdxVdekJSekx4Gy
5qhtvVZX87hpKO73zhIq1+jxxMaus8d3ass0ntlcb5Zsgot7m57bvfUs7eECQC/s
nmG/wQdb+4uQKxo6pPpdojfnOHurztWO+EiziAG0dO1s2lW7flTqlD7PqTVP1uk8
AbEc5jHpZMZp6Hk4AGcCQQC2PtFQ4jykO/VdXETM3jUGPbyavxA7prdb8ut1fPfi
lzPtow1Lcn4uCK/sXecS+RaXEh3wPKJkaoTlUMJpivbO
-----END RSA PRIVATE KEY-----
```

This is the base 64 encoding of the RSA private key. The private key file must be protected against theft, tampering, or viewing.

Certificate request file

The file specified by `-reqout` will contain a certificate request. This should be sent to the CA or discarded; the iDAR does not make use of the certificate request. An example file is:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBbTCB1wIBADAuMRMwEQYKCZImiZPyLGQBGRYDY29tMRcwFQYKCZImiZPyLGQB
GRYHaVBsYW5ldDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA62HO/EPT51PE
OXdEnxcjVyPOLEbRj/TV4D+9F8WOlnYN8aEAMWf8VshP9HcdMUUfG8wqYtuXk1cm
G8kDMWOsrtjsb7MsA2MJADdm0QYHj1DFUpcdo2Yutha9gs7AGmo2zjtMqlxT/4Mo
16YWKK2zr8hu/F6d7xSHrFmJehoM4JcCAwEAAaAAMA0GCSqGSIb3DQEBBAUAA4GB
AHcBoLa3Bi3o+HblCIkD6Rx29gShLwVK+QyzrPHrC9iGgrOuZBcrJSqSVXx5U/iR
8ia14YKkZDqPJmEIe/eoxhKWgfxUQyi3D0DUovzg/+M0EIWxoCrwbJhl/1DvUgQl
eTLj+mPeW3cj4KSQdWa5TJxj6Zf2PvIMppVB0wmbkPfj
-----END CERTIFICATE REQUEST-----
```

The contents of the certificate request file are a base 64 encoding of a value of the ASN.1 type defined in PKS#10

```
SIGNED { SEQUENCE {
version Version,
subject Name,
subjectPublicKeyInfo SubjectPublicKeyInfo,
attributes [0] IMPLICIT Attributes } }
```

Supported SASL Mechanisms

iDAR supports clients that bind to the directory using the SASL mechanism. However, not all SASL mechanisms are supported. The following matrix summarizes the current support.

Table 4-2 Supported SASL Mechanisms

CRAM-MD5	Yes
DIGEST-MD5 (auth only)	Yes
DIGEST-MD5 (connection protection)	No
EXTERNAL	No
GSSAPI	No

Chapter 5

Global Configuration

The iPlanet Directory Access Router configuration can be represented as entries either in an LDAP directory or in a file in LDIF format. This section describes the schema for storing the iDAR global configuration information.

Throughout the remainder of this document, all configuration attributes are assumed to have a syntax of a case-insensitive UTF-8 string (`cis`) unless explicitly stated otherwise. Attribute names follow one of the following two conventions: `ids-proxy-sch-*` attributes are those that are part of iDAR configuration schema but do not have a direct effect on iDAR functionality; `ids-proxy-con-*` attributes directly effect iDAR functional behavior.

ids-proxy-sch-LDAPProxy object class

The `ids-proxy-sch-LDAPProxy` object contains information about the name of the iDAR server and the distinguished name (DN) of the `ids-proxy-sch-GlobalConfiguration` object. The iDAR finds this object using the DN specified in the `configuration_url` attribute in the startup configuration file. This object contains the following mandatory attributes.

ids-proxy-sch-Global-Config-Dn

The `ids-proxy-sch-Global-Config-Dn` attribute specifies the distinguished name of the `ids-proxy-sch-Global-Configuration` object entry. This attribute must be present and has `dn` syntax.

ids-proxy-con-Server-Name

The `ids-proxy-con-Server-Name` attribute specifies the string name of the server. This attribute must be present, as it is used to form the Relative Distinguished Name (RDN) of entries of this class, as well as to locate group and property entries.

Locating group, property, rule, and action objects

ids-proxy-sch-Group-Base, ids-proxy-sch-Property-Base, ids-proxy-sch-Rule-Base, ids-proxy-sch-Action-Base

These attributes specify the DN of the entries under which the group and property object entries reside. iDAR performs a one-level search with the respective attribute value as the base of the search and one of the following filters.

Object	LDAP Filter

group	`(&(objectclass=ids-proxy-sch-Group)`
	`(ids-proxy-sch-enable=TRUE)`
	`(ids-proxy-sch-belongs-to=<`*`Server-Name`*`>))`

property	`(&(objectclass=ids-proxy-sch-Property)`
	`(ids-proxy-sch-enable=TRUE)`
	`(ids-proxy-sch-belongs-to=<`*`Server-Name`*`>))`

rule	`(&(objectclass=ids-proxy-sch-Rule)`
	`(ids-proxy-sch-enable=TRUE)`
	`(ids-proxy-sch-belongs-to=<`*`Server-Name`*`>))`

action	`(&(objectclass=ids-proxy-sch-Action)`
	`(ids-proxy-sch-enable=TRUE)`
	`(ids-proxy-sch-belongs-to=<`*`Server-Name`*`>))`

`<Server-Name>` is the name of the server as specified in the `ids-proxy-con-Server-Name` attribute. The `ids-proxy-sch-Group-Base` attribute must be present in every entry of this class. The other three attributes are optional and may be omitted if no entries of that type are used.

ids-proxy-sch-GlobalConfiguration object class

The `ids-proxy-sch-GlobalConfiguration` object contains various global configuration attributes. Each configuration must have one entry of this object class.

ids-proxy-con-Config-Name

The `ids-proxy-con-config-Name` attribute specifies the string name of the configuration. This attribute must be present, as it is used to form the RDN of entries of this class.

ids-proxy-con-listen-port

The `ids-proxy-con-listen-port` attribute takes a value of the port number, 1 through 65535, on which to listen for incoming connections. The default, if this attribute is not present, is TCP port 389, as specified for LDAP. This port number must be different from that used by any other LDAP server running on the same host. On UNIX platforms the server must be started as root to listen on a port number below 1024.

For example, to have iDAR listen on port 10389, the following attribute / value pair should be in the `ids-proxy-sch-GlobalConfiguration` object entry:

```
ids-proxy-con-listen-port: 10389
```

ids-proxy-con-listen-host

The `ids-proxy-con-listen-host` attribute takes a value of the name or the IP address of the host interface on which the iDAR will listen for connections. This attribute is needed only if there are multiple network interfaces on the host running iDAR. If this attribute is not present or has the value “`localhost`”, iDAR will listen on all available network interfaces.

ids-proxy-con-max-conns

The `ids-proxy-con-max-conns` attribute takes a value (greater than zero) of the maximum number of simultaneous client connections that iDAR will accept. For example, if you only want to allow ten simultaneous connections, the following attribute / value pair should be in the `ids-proxy-sch-GlobalConfiguration` object entry:

```
ids-proxy-con-max-conns:   10
```

To allow an unlimited number of simultaneous connections, do not specify this attribute.

ids-proxy-con-listen-backlog

The `ids-proxy-con-listen-backlog` attribute takes a value (greater than zero) of the maximum number of outstanding connections in the listening socket's queue. The default is `128` if the attribute is absent. This attribute should be present if more than 128 clients could attempt to contact iDAR every second.

The maximum value depends on the underlying operating system configuration.

ids-proxy-con-ldaps-port

The `ids-proxy-con-ldaps-port` attribute takes as a value the port number on which to listen for `LDAPS` (LDAP over SSL) connections. By default, iDAR does not listen for connections from `LDAPS` clients. This attribute must be present to enable `LDAPS` connections from clients using this nonstandard function, with a value such as `636`. This value must be different from the `ids-proxy-con-listen-port` value. This option also requires TLS / SSL configuration (see Chapter 4 on page 53).

ids-proxy-con-userid

This option specifies the userid under which iDAR will run. If iDAR was run as `root` then it will change its `uid` to the one specified here. The default is to switch to `nobody`. This option is not applicable on Windows NT.

ids-proxy-con-working-dir

This attribute specifies the directory that iDAR should run from. iDAR will change its working directory to the directory specified as value for this attribute on startup. The default is `/tmp`. This attribute only takes effect on platforms other than Windows NT.

ids-proxy-con-include-logproperty

This attribute specifies the value of the `ids-proxy-con-name` attribute of the `ids-proxy-sch-LogProperty` property entry. This property should be defined by an entry if you want to customize logging. Refer to “Logging,” on page 97 for details.

TLS/SSL Configuration Attributes

The `ids-proxy-con-ssl-cert` and `ids-proxy-con-ssl-key` attributes must be present in the `ids-proxy-sch-GlobalConfiguration` object entry if the Start TLS extended operation or `ids-proxy-con-ldaps-port` option is to be supported. The TLS policy of each group can be configured using the group attribute `ids-proxy-con-ssl-policy`.

ids-proxy-con-ssl-key

The `ids-proxy-con-ssl-key` attribute specifies the pathname of the file on disk containing the private key associated with the server's certificate. Refer to the section “Generating a TLS Key Pair,” on page 56 for more details.

ids-proxy-con-ssl-cert

The `ids-proxy-con-ssl-cert` attribute specifies the location pathname of the file on disk containing the server's own certificate signed by the CA. The certificate in the file must conform to the Public Key Cryptography Standards (PKCS#10) format. Refer to the section “Generating a TLS Key Pair,” on page 56 for more details.

ids-proxy-con-send-cert-as-client

This attribute has a boolean syntax and specifies whether iDAR will send its server certificate when making a TLS connection to an LDAP directory server. Set it to `TRUE` if you want iDAR to send its certificate to the backend LDAP directory server when making a TLS connection. The default is `FALSE`.

ids-proxy-con-server-ssl-version, ids-proxy-con-client-ssl-version

These options specify the version of SSL that iDAR will use. `ids-proxy-con-server-ssl-version` specifies the version of SSL it will use for connections between itself and backend LDAP servers while `ids-proxy-con-client-ssl-version` specifies the version between itself and the clients. Permitted values are `20` for SSL version 2 only, `30` for SSL version 3 only, `23` for SSL version 2 or SSL version 3, and `31` for TLS version 1 only. The default for both is `23`.

ids-proxy-con-ssl-cert-required

If the `ids-proxy-con-ssl-cert-required` attribute is set to `TRUE`, iDAR will require all clients that establish an SSL session to submit a certificate chain. iDAR will close the connection if a certificate chain is not submitted. Note that this option does not effect SSL sessions between iDAR and the backend servers. The default value for this attribute is `FALSE`.

ids-proxy-con-ssl-cafile

The ids-proxy-con-ssl-cafile attribute specifies the path of the file on disk containing trusted root certificates. These certificates must be in the PEM format.

Connection Pool Configuration Attributes

If the connection pool is enabled, iDAR will try to reuse existing connections to the backend LDAP servers. Switching on this option can give significant performance gain if the backend server is on a Wide Area Network (WAN).

ids-proxy-con-connection-pool

The `ids-proxy-con-connection-pool` attribute enables or disables the connection pool module. If this attribute is present with the value `TRUE` iDAR will preconnect to the directory servers. The default is `FALSE`.

ids-proxy-con-connection-pool-interval

The `ids-proxy-con-connection-pool-interval` attribute takes an integer value greater or equal to one. It specifies the interval in seconds at which iDAR will sample the incoming requests to anticipate future activity. The default is `15`.

ids-proxy-con-connection-pool-timeout

The `ids-proxy-con-connection-pool-timeout` attribute takes an integer value greater or equal to zero. It specifies the period of time in seconds after which an idle connection to an LDAP server will be terminated. If a zero value is specified, no timeout will be applied. The default is `30`. This value should be less than the idle connection timeout value of the backend LDAP server.

Debugging

This attribute is useful when debugging iDAR. It is only available on UNIX platforms.

ids-proxy-con-foreground

The `ids-proxy-con-foreground` attribute specifies whether the iDAR should remain in the foreground or detach from the controlling terminal. The default is `FALSE`, and the iDAR will detach. This attribute should not be specified in the `ids-proxy-sch-GlobalConfiguration` object entry except when debugging.

Chapter 6

Groups Configuration

iPlanet Directory Access Router groups describe how to identify an LDAP client, and the restrictions to enforce for clients that match that group. Clients are initially identified into a group based on the network address they connect from. They may change their group after a successful bind. (See sections `ids-proxy-sch-OnBindSuccessRule` object class on page 92 and `ids-proxy-sch-ChangeGroupAction` object class on page 96 for more details.)

Groups are tested in the descending order of priority, specified by the `ids-proxy-con-priority` attribute. The order of evaluation of groups with equal priority is undefined. If no groups are found to match a client, the client's request will be rejected. There must be at least one group entry in the configuration specification.

ids-proxy-sch-Group object class

This object class contains the group object attributes.

ids-proxy-con-Name

The `ids-proxy-con-Name` attribute specifies the name of the group or property. This attribute value must be unique within the set of groups. This attribute must be present as it forms the RDN of entries of this class.

ids-proxy-con-Priority

The `ids-proxy-con-Priority` attribute specifies the order of evaluation. It takes an integer value greater than or equal to zero. If this attribute is not present its value is assumed to be 0. If the priority of two objects are the same, their order of evaluation is undefined. The object with the highest value is evaluated first. Thus, for example, the most generic group should have the lowest `ids-proxy-con-Priority` value. This is a single valued attribute.

ids-proxy-sch-Enable

The `ids-proxy-sch-Enable` attribute specifies if this object entry is part of the configuration or not. For an entry to be part of iDAR configuration, the `ids-proxy-sch-Enable` attribute must be present with the value `TRUE`. If the `ids-proxy-sch-Enable` attribute is not present or has the value `FALSE`, the object entry is ignored. This is a single valued attribute.

ids-proxy-sch-belongs-to

This is a multi-valued attribute that specifies which server configuration this object entry belongs to. The value of this attribute should be the value of the `ids-proxy-con-server-name` attribute in the `ids-proxy-sch-LDAPProxy` object entry. If it does not match, the entry is ignored.

ids-proxy-sch-NetworkGroup object class

This object class is sub-classed from the `ids-proxy-sch-Group` object class. Clients are identified to belong to this object class based on their IP address and/or domain name. Each of the `ids-proxy-sch-NetworkGroup` object entries must have at least one value for the `ids-proxy-con-Client` attribute.

ids-proxy-con-Client

The `ids-proxy-con-Client` attribute specifies the domain names or IP addresses of the hosts that match the `ids-proxy-sch-NetworkGroup` object entry. If a group is only used for clients to switch to when they bind, this attribute should be present with value `0.0.0.0`.

The value of the ids-proxy-con-Client attribute can be one of the following:

- A domain name, e.g., foo.iplanet.com. Note: iDAR does not assume any domain suffix by default; hence, complete domain names must be provided.

- A domain name suffix with a leading period, e.g., .iplanet.com. All hosts with domain names that end in that suffix are considered to match.

- A single IP address in dotted decimal form, e.g., 198.214.11.1.

- An IP network mask, in the form of <network number>/<mask bits>, e.g., 198.241.11.0/24. The first half is the network number and the second half indicates the number of bits of the network number necessary for matching.

- An IP network mask, in the form of a pair of dotted decimal quads, e.g., 198.241.11.0/255.255.255.128. The first half is a network number, the second half indicates the bits of the network number necessary for matching. For example, 198.214.11.0/255.255.255.128 will match a host with IP address 198.214.11.63 but not one with IP address 198.214.11.191.

- If a group is only used for clients to switch to when they bind, this attribute should be present with value 0.0.0.0.

- The string ALL (must be in all caps) matches all hosts.

Since the groups are checked in the order of priority, the most specific group should have the highest value. Generic groups, such as those with the attribute ids-proxy-con-Client: ALL, should have the lowest value, as no further groups will be checked.

Note: use of domain names or domain name suffixes requires ids-proxy-con-reverse-dns-lookup to be set to TRUE.

ids-proxy-con-include-property

The ids-proxy-con-include-property attribute is used to specify the property object entry to include in a group. The value of this attribute should be the value of the ids-proxy-con-Name attribute in the property object you want to include. This attribute is used to include more specialized configuration properties into a group.

For example, to include a property with ids-proxy-con-Name:Load-Balance you must specify the following:

ids-Proxy-con-include-property:Load-Balance

ids-proxy-con-include-rule

The `ids-proxy-con-include-rule` attribute is used to specify the rule object entry to include in a group. The value of this attribute should be the value of the `ids-proxy-con-Name` attribute in the rule object you want to include.

This attribute is used to include rules that can be evaluated upon completion of predetermined processes and actions can be taken on the outcome of the evaluation.

For example, the `ids-proxy-sch-OnSuccessfulBindRule` is evaluated on the successful completion of a bind operation. If the rule evaluates to `true` then a client may be able to change its group. For more information see “Rules Configuration,” on page 91.

ids-proxy-con-ssl-policy

The `ids-proxy-con-ssl-policy` attribute can have one of the following three values: `ssl_required`, `ssl_optional` or `ssl_unavailable`. If `ssl_optional` is specified, then a TLS/SSL connection will be established if the client requests TLS/SSL. If `ssl_required` is specified, then the client must establish a TLS/SSL session before performing any operation. iDAR will not permit a TLS/SSL session if `ssl_unavailable` is specified. This is the default.

For example, to make sure that all clients in a group establish a TLS/SSL session before performing any operation, specify the following:

```
ids-proxy-con-ssl-policy: ssl_required
```

For more information on setting up TLS, see “Configuring Security,” on page 53.

ids-proxy-con-tcp-no-delay

If the value of this option is set to `TRUE` then iDAR will disable the Nagle Algorithm for connections between itself and clients that fall into this group. This option should only be set to `FALSE` if the network bandwidth between iDAR and clients is small. Setting this option to false may create substantial performance degradation. The default is `TRUE`.

ids-proxy-con-allow-multi-ldapv2-bind

If this attribute is set to TRUE, iDAR will allow a LDAP v2 client to bind multiple times over a single session. The LDAP v2 specification (RFC 1777) does not allow a client to bind multiple times in one session. However, some clients expect this functionality. This attribute can be used to interoperate with these clients. If you do set ids-proxy-con-allow-multi-ldapv2-bind to TRUE, you must set ids-proxy-con-use-version attribute in the ids-proxy-sch-LDAPServer objectclass to the value 3. This will prevent iDAR from sending any illegal operations to the backend server(s). The default value for this attribute is FALSE.

ids-proxy-con-reverse-dns-lookup

If ids-proxy-con-reverse-dns-lookup is set to FALSE, then iDAR will not perform a reverse DNS lookup to find the domain name of the connecting client. Setting this attribute to FALSE can sometimes significantly improve iDAR performance. If you have used a domain name or a domain name suffix as a value to ids-proxy-con-client attribute, you must not set ids-proxy-con-reverse-dns-lookup to FALSE, otherwise iDAR will not function properly. The default is TRUE.

ids-proxy-con-timeout

The ids-proxy-con-timeout attribute in an ids-proxy-sch-NetworkGroup defines a period of client inactivity, after which iDAR may close the connection to the client. The value is a number of seconds, typically 600 or more. The default, if this attribute is not present in an ids-proxy-sch-NetworkGroup object entry, is to not timeout connections. If TCP keepalives are not enabled, this attribute must be present to keep iDAR from being clogged by lost client connections.

Forwarding Binds

Once iDAR has accepted a connection from the client and matched a group, it will wait for the client to send the LDAP bind request. iDAR uses the ids-proxy-con-bind-name, ids-proxy-con-permit-auth-none, ids-proxy-con-permit-auth-simple and ids-proxy-con-permit-auth-sasl attributes to determine whether to pass the bind request to the server, or reject the bind request and close the client's connection.

If the client's bind passes these tests, iDAR will forward it to the server. If the server accepts the bind, the connection is established. If, however, the server returns an error indication for the bind request, iDAR will forward the error indication to the client, and then close the connection to the client, if the client was using LDAPv2.

ids-proxy-con-bind-name

The `ids-proxy-con-bind-name` attribute takes as a value a distinguished name. If a client provides a distinguished name in its bind that is not subordinate to a value in a `ids-proxy-con-bind-name` attribute, the bind will be rejected. This attribute must be present at least once in a group in order to allow any access. To allow all access for clients that match the `ids-proxy-con-client` attribute/value pair, regardless of bind name, this attribute should be present, with a zero-length value, as in

```
ids-proxy-con-bind-name:
```

This attribute has the `dn` syntax.

ids-proxy-con-permit-auth-none

The `ids-proxy-con-permit-auth-none` attribute specifies whether anonymous binds, when the client has not supplied a password, are permitted. The default is `TRUE`, if this attribute is not present in a group. This attribute may be changed to `FALSE` to forbid anonymous bind requests.

ids-proxy-con-permit-auth-simple

The `ids-proxy-con-permit-auth-simple` attribute specifies whether simple binds are permitted, when the client has supplied a password in the clear. The default is `TRUE`, if this attribute is not present in a group. This attribute may be changed to `FALSE` to forbid clear text password authenticated bind requests.

ids-proxy-con-permit-auth-sasl

The `ids-proxy-con-permit-auth-sasl` attribute specifies whether SASL binds are permitted. The default is `TRUE`, if this attribute is not present in a group. This attribute may changed to `FALSE` to forbid SASL authentication.

Controlling Which Operations are Forwarded

After a successful bind, iDAR by default forwards search and compare requests. iDAR also recognizes the unbind request, and closes the connection to the server.

ids-proxy-con-permit-op-search

If the `ids-proxy-con-permit-op-search` attribute is present in a group, with the value `FALSE`, iDAR will not forward search requests to the server. The default is `TRUE`.

ids-proxy-con-permit-op-compare

If the `ids-proxy-con-permit-op-compare` attribute is present in a group, with the value `FALSE`, iDAR will not forward compare requests to the server. The default is `TRUE`, however, few clients use this operation.

ids-proxy-con-permit-op-add, ids-proxy-con-permit-op-delete, ids-proxy-con-permit-op-modify, ids-proxy-con-permit-op-modrdn, ids-proxy-con-permit-op-extended

By default, iDAR does not forward Add, Modify, Delete, ModifyRDN, or Extended operations requests. To permit forwarding of these operations, the attributes `ids-proxy-con-permit-op-add`, `ids-proxy-con-permit-op-delete`, `ids-proxy-con-permit-op-modify`, `ids-proxy-con-permit-op-modrdn`, and `ids-proxy-con-permit-op-extended` must be present in the `ids-proxy-sch-NetworkGroup` object entry, with the value `TRUE` for the appropriate operation to be allowed.

Please note that you must set `ids-proxy-con-permit-op-extended` to `TRUE` if you want your clients to be able to negotiate Start TLS.

Hiding a Subtree of Entries with ids-proxy-con-forbidden-subtree

The `ids-proxy-con-forbidden-subtree` attribute specifies the distinguished name of the base of a subtree of entries to be excluded. This is a multi-valued attribute. The default, if this attribute is not present in a group, is to allow access to all entries in the directory.

Operations that request entries at or below a forbidden subtree will be rejected with an insufficient access error. Entries that match a search filter and are inside a forbidden subtree are dropped. Please note that this option does not remove DN syntax attributes whose values fall under the subtree from entries that are being returned as part of the result.

This attribute has dn syntax.

Attributes Controlling Search Requests

The `ids-proxy-con-filter-inequality` and `ids-proxy-con-min-substring-size` attributes are used to prevent certain kinds of searches from reaching the LDAP server. If the client's search request falls under either restriction, iDAR will return an insufficient access error to the client.

ids-proxy-con-filter-inequality

The `ids-proxy-con-filter-inequality` attribute specifies whether clients are permitted to request searches that contain inequality filters (`attr>=value`) and (`attr<=value`). The default is `TRUE` if this attribute is not present. This attribute should be present in the `ids-proxy-sch-NetworkGroup` object entry with the value `FALSE` if you do not wish to permit inequality searches to be performed.

ids-proxy-con-min-substring-size

The `ids-proxy-con-min-substring-size` attribute specifies the minimum permissible length of a substring in a search filter. The value is a number greater than one. The default, if this attribute is not present, is to allow any size of substring in a search filter. This attribute should be present in the `ids-proxy-sch-NetworkGroup` object entry if you wish to restrict the kinds of searches that may be performed by web robots, etc. For example, a value of 2 will block searches like`(cn=A*)`.

Controlling Search and Compare Requests

The `ids-proxy-con-forbidden-compare` and `ids-proxy-con-permitted-compare` attributes are used to prevent certain kinds of search and compare operations from reaching the LDAP server. If the client's request falls under this restriction, iDAR will return an insufficient access error to the client.

ids-proxy-con-forbidden-compare

The `ids-proxy-con-forbidden-compare` attribute specifies the name of an attribute that cannot be used by a client in a search filter or compare request. This is a multi-valued attribute. The default, if this attribute is not present in a group, and the `ids-proxy-con-permitted-compare` attribute is not present in the group either, is to permit all attributes to be used for search filters and comparisons.

For example, to allow clients to search on all attributes except `telephoneNumber` and `payrollCode`, add the following:

```
ids-proxy-con-forbidden-compare: telephoneNumber
ids-proxy-con-forbidden-compare: payrollCode
```

ids-proxy-con-permitted-compare

The `ids-proxy-con-permitted-compare` attribute specifies the name of an attribute that may be used in a search filter or compare request. This is a multi-valued attribute.

If there is one or more `ids-proxy-con-permitted-compare` attributes values present in an `ids-proxy-sch-NetworkGroup` object entry, and a compare does not match one of these, the request will be rejected by iDAR. If there are no `ids-proxy-con-permitted-compare` attributes present in a group, and an attribute does not match any `ids-proxy-con-forbidden-compare` attributes, then it may be used by clients.

For example, if you only want the `cn`, `dn`, and `mail` attributes to be searchable by the client, add the following to that `ids-proxy-sch-NetworkGroup` object entry:

```
ids-proxy-con-permitted-compare: cn
ids-proxy-con-permitted-compare: dn
ids-proxy-con-permitted-compare: mail
```

Attributes Modifying Search Requests

The `ids-proxy-con-minimum-base`, `ids-proxy-con-max-scope`, and `ids-proxy-con-max-timelimit` attributes may modify the search request before it is forwarded to the server.

ids-proxy-con-minimum-base

The `ids-proxy-con-minimum-base` attribute specifies the base of a subtree for all operations. This is a single valued attribute and has `dn` syntax. If this attribute is not present, then there is no restriction to a minimum base.

Operations whose target entry is at or below the minimum base entry are not affected by this attribute. If the target entry is superior to the minimum base entry, and the operation is a subtree search, then the query will be rewritten before being sent to the server, to change the target entry to be the minimum base. If the target entry is not below the minimum base or a superior of it, the request will be rejected with a no such object error.

For example, if the `ids-proxy-con-minimum-base` is set as:

```
ids-proxy-con-minimum-base: o=iPlanet, st=California, c=US
```

and a subtree search of `st=California, c=US` is received, the search will be rewritten such that the server performs a subtree search of `o=iPlanet, st=California, c=US`.

ids-proxy-con-max-scope

The `ids-proxy-con-max-scope` attribute specifies the maximum scope that a client may specify in a search request. This attribute may take one of the following values:

0: only `baseObject` searches may be requested
1: `baseObject` and `oneLevel` searches may be requested
2: `baseObject`, `oneLevel` and `subtree` may be requested

The default, if this attribute is not present, is to permit all search scopes.

ids-proxy-con-max-timelimit

The `ids-proxy-con-max-timelimit` attribute specifies a maximum time limit in seconds for search operations. If the client specifies a time limit that is larger than the value given in this attribute, this attribute will override the client's request. The default, if this attribute is not present in a group, is to allow the client to set any time limit, including no limit.

Attributes Restricting Search Responses

The `ids-proxy-con-max-result-size`, `ids-proxy-con-forbidden-return`, `ids-proxy-con-permitted-return`, and `ids-proxy-con-search-reference` attributes in a group describe restrictions that are applied to search results being returned by the server, before they are forwarded to the client.

ids-proxy-con-max-result-size

The `ids-proxy-con-max-result-size` attribute specifies the maximum number of result entries that may be returned to a client at one time from a single search operation. The value may be any number greater than zero, and if reached, will cause an `administrativeLimitExceeded` error to be indicated to the client and subsequent entries will be discarded. The default, if this attribute is not present, is to not discard entries.

ids-proxy-con-forbidden-return

The `ids-proxy-con-forbidden-return` attribute specifies the name of an attribute that is to be excluded from search result entries. This is a multi-valued attribute. The default, if this attribute is not present in the `ids-proxy-sch-NetworkGroup` object entry, and the `ids-proxy-con-permitted-return` attribute (discussed below) is not present in the group, is to permit all attributes to be returned.

ids-proxy-con-permitted-return

The `ids-proxy-con-permitted-return` attribute specifies the name of an attribute that may be returned from a search result, if present. This is a multi-valued attribute.

If there are `ids-proxy-con-permitted-return` attributes values present in an `ids-proxy-sch-NetworkGroup` object entry, and an attribute does not match one of those listed, it is not returned. If there are no `ids-proxy-con-permitted-return` attributes values present in an `ids-proxy-sch-NetworkGroup` object entry, and no value matches any `ids-proxy-con-forbidden-return` attribute values, then it is returned.

For example, if you do not want the `homePhone` attribute to be returned in the search result, add the following attribute / value pair to the `ids-proxy-sch-NetworkGroup` object entry:

```
ids-proxy-con-forbidden-return: homePhone
```

ids-proxy-con-search-reference

The `ids-proxy-con-search-reference` attribute specifies what to do if a search continuation reference is generated during a search. It can be one of the following values: `forward` (forward the reference to the client), `follow` (follow the reference and return the result to the client), or `discard`. The default is `discard`.

A search continuation referral is a special case of a referral whereby part of the query has been satisfied by the original directory server queried but that directory server has a reference to another directory server with more data satisfying the query.

This attribute can be used to hide the part of your Directory Information Tree whose naming context is mastered by another LDAP server. It also prevents clients from finding out the network address and port on which this server runs.

Controlling the Return of Referrals

ids-proxy-con-reference

Each group can have an attribute `ids-proxy-con-reference` that specifies whether to forward, follow, or discard referrals returned by the server. It takes the following values: `forward`, `follow`, or `discard`. The default is `forward`. It should be noted that a client that does not implement LDAPv3 will not understand forwarded referrals.

The use of this attribute is similar to the `ids-proxy-con-search-reference` attribute. This attribute applies to all referrals except for search continuation referrals.

ids-proxy-con-referral-ssl-policy

This attribute is similar to `ids-proxy-con-ssl-policy`. If `ssl_required` is specified, and an operation is referred, then a TLS/SSL session will be initiated before the operation is forwarded to the referred server. If `ssl_unavailable` is specified, TLS/SSL will not be initiated. If `ssl_optional` is specified, TLS/SSL will be initiated only if the client already has TLS/SSL session established with iDAR. The default, if this attribute is not present, is `ssl_unavailable`.

ids-proxy-con-referral-bind-policy

The `ids-proxy-con-referral-bind-policy` controls the bind policy when an operation is referred and the referral is being followed. The following three values can be specified: `bind_anonymous` (always bind anonymous), `bind_any` (use simple bind if the client had used password-based bind else bind as anonymous), or `bind_required` (reject the referred operation if the client is not password-based bound). The default is `bind_any`.

Please note that iDAR cannot replay binds for clients bound using a SASL mechanism. Thus the referral operation will be rejected if `bind_required` is specified and the client used a SASL mechanism to bind.

ids-proxy-con-max-refcount

The `ids-proxy-con-max-refcount` takes an integer value greater or equal to zero. It limits the maximum number of references that will be followed for a single operation. The default is 15. A value of zero indicates that no limit will be applied.

Controlling Server Load

The `ids-proxy-con-max-simultaneous-operations-per-connection` and `ids-proxy-con-max-operations-per-connection` attributes can be used to control the number of simultaneous operations and total number of operations a client can request on one connection. The `ids-proxy-con-max-conns` and `ids-proxy-con-max-simultaneous-conns-from-ip` can be used to restrict the number of connections for a group.

ids-proxy-con-max-simultaneous-operations-per-connection

The `ids-proxy-con-max-simultaneous-operations-per-connection` attribute limits the number of simultaneous operations iDAR will process per connection in that group. The value is an integer greater than zero. If this attribute is not present then no limit is enforced. For example, if you set:

```
ids-proxy-con-max-simultaneous-operations-per-connection: 1
```

all the clients in that group will be forced to perform synchronous LDAP operations. Additional simultaneous requests, except for requests to abandon an operation, will fail with `Server Busy` error.

ids-proxy-con-max-operations-per-connection

The `ids-proxy-con-max-operations-per-connection` limits the total number of operations that iDAR will allow per connection in a group. The value is an integer greater than zero. If a client exceeds the maximum number of operations allowed for its group on one connection, that connection will be closed by iDAR. If this attribute is not present, then no limit is set.

ids-proxy-con-max-conns

This `ids-proxy-con-max-conns` attribute is the same attribute as the global attribute described in `ids-proxy-con-max-conns` on page 62. If specified within an `ids-proxy-sch-NetworkGroup`, then this attribute applies only to that group entry.

ids-proxy-con-max-simultaneous-conns-from-ip

The `ids-proxy-con-max-simultaneous-conns-from-ip` attribute can be used to restrict the number of simultaneous connections clients can make from a single IP address. The default is `0`, which means any number of connections are allowed.

Properties Configuration

Property objects are used to specify more specialized restrictions that the clients must follow. These properties can then be included in other entries where the restrictions need to be applied.

ids-proxy-sch-Property object class

This object class defines the property object attributes. It contains the following attributes:

ids-proxy-con-Name

The `ids-proxy-con-Name` attribute specifies the name of the property. This attribute value must be unique within the set of properties. This attribute must be present as it forms the RDN of entries of this class.

ids-proxy-con-Priority

The `ids-proxy-con-Priority` attribute is reserved for future versions of this product.

ids-proxy-sch-Enable

The `ids-proxy-sch-Enable` attribute specifies if this object entry is part of the configuration or not. For an entry to be part of an iDAR configuration, the `ids-proxy-sch-Enable` attribute must be present with a value `TRUE`. If the `ids-proxy-sch-Enable` attribute is not present or has the value `FALSE`, the entry is ignored.

ids-proxy-sch-belongs-to

This is a multi-valued attribute that specifies which server configurations this object entry belongs to. Each value of this attribute should be the value of the `ids-proxy-con-server-name` attribute in the `ids-proxy-sch-LDAPProxy` object entry, defined in the section `ids-proxy-con-Server-Name` on page 60.

ids-proxy-sch-LoadBalanceProperty object class

This object class is sub-classed from the `ids-proxy-sch-property` object class and can be used to tell iDAR to load balance across multiple LDAP servers. Either this property can be included in a group using the `ids-proxy-con-include-property` attribute, or the group can have the `ids-proxy-con-server` attribute, but not both. The following attribute is mandatory for this object class.

ids-proxy-con-Server

The `ids-proxy-con-Server` attribute is used to specify the server name and the percentage of client connections to be served by it. Its value should have the following syntax:

`<server-Property>[#<percentage>]`

The `server-Property` is the value of `ids-proxy-con-Name` of an `ids-proxy-sch-LDAPServer` property object entry.

The `percentage` part is the percentage of total connections that this server should be sent.

The sum of all the `percentage` for the different servers specified must be 100. If the `percentage` is not specified, it is assumed to be zero. A server with zero percentage will only be used as a failover server.

For example, to load balance equally across three LDAP servers with `ids-proxy-con-name`, `ldap1`, `ldap2`, and `ldap3`, this attribute should have the following three values:

```
ids-proxy-con-Server: ldap1#33
ids-proxy-con-Server: ldap2#33
ids-proxy-con-Server: ldap3#34
```

ids-proxy-sch-SizeLimitProperty object class

This object class is sub-classed from the `ids-proxy-sch-property` object class. The `ids-proxy-sch-SizeLimitProperty` object class can be used to apply size limits based on the search base and search scope. If neither the search base nor search scope specified in this property object entry match a given search, the size limit defaults to the value of `ids-proxy-con-max-result-size` group attribute. This property class has the following attributes.

ids-proxy-con-Size-Limit

The `ids-proxy-con-Size-Limit` attribute takes an integer value, specifying the size limit to enforce. This is a single valued attribute. The value of `-1` implies that no size limit will be enforced. This attribute is mandatory. Note that this is different from the LDAP protocol.

ids-proxy-con-Dn-One

If the DN of the search base of a one level search matches one of the distinguished names specified for the `ids-proxy-con-Dn-One` attribute, the value of `ids-proxy-con-Size-Limit` is enforced as the size limit of that search. This attribute has `dn` syntax.

ids-proxy-con-Dn-Sub

If the DN of the search base of a subtree search matches on the distinguished names specified for the `ids-proxy-con-Dn-Sub` attribute, the value of `ids-proxy-con-Size-Limit` is enforced as the size limit of that search. This attribute has `dn` syntax.

For example, to enforce a size limit of 50 for subtree searches with search base `dc=iplanet,dc=com` and one level searches with search base `ou=Development, dc=iplanet,dc=com`, the following attribute / value pairs should be placed in the `ids-proxy-sch-SizeLimitProperty` object entry:

```
ids-proxy-con-Size-Limit: 50
ids-proxy-con-Dn-One: dc=iPlanet, dc=com
ids-proxy-con-Dn-Sub: ou=Development, dc=iPlanet, dc=com
```

ids-proxy-sch-RenameAttributeProperty object class

This object class is sub-classed from the `ids-proxy-sch-property` object class. The `ids-proxy-sch-RenameAttributeProperty` object class can be used to rename attributes in client queries and in server results. This property has two mandatory single valued attributes:

ids-proxy-con-server-attr-name

The `ids-proxy-con-server-attr-name` attribute is used to specify the name of the attribute known to the server.

ids-proxy-con-client-attr-name

The `ids-proxy-con-client-attr-name` attribute is used to specify the name of the attribute known to the client.

If a client request contains an attribute name specified in `ids-proxy-con-client-attr-name`, it will be transformed to the value of `ids-proxy-con-server-attr-name`. Similarly, if a result sent by the server contains an attribute name specified in `ids-proxy-con-server-attr-name`, it will be transformed to the value of `ids-proxy-con-client-attr-name`.

ids-proxy-sch-ForbiddenEntryProperty object class

This object class is sub-classed from the `ids-proxy-sch-property` object class. The `ids-proxy-sch-ForbiddenEntryProperty` can be used to specify a list of entries or attributes in those entries that you want to hide. It has the following attributes.

ids-proxy-con-dn-exact

This is a multi-valued attribute that can be used to specify a list of DN's of entries or attributes in those entries that you want to hide.

ids-proxy-con-dn-regexp

This is a multi-valued attribute that can be used to specify a list of regular expressions of DN's of entries, or attributes in those entries that you want to hide. The regular expression of the DN should be specified in the normalized form, i.e., there should be no spaces between RDN components and the '=' sign, attribute names must be in all capital letters and if the attribute values have the `cis` syntax then they should also be capitalized.

For example, to match any DN with a RDN component of "`ou=internal`", you must specify the following:

```
ids-proxy-con-dn-regexp: .*OU=INTERNAL.*
```

If there are `ids-proxy-con-permitted-return` attributes present in the `ids-proxy-sch-ForbiddenEntryProperty` object entry, and an attribute does not match one of those listed, it is not returned. If there are no `ids-proxy-con-permitted-return` attribute values are present in an `ids-proxy-sch-ForbiddenEntryProperty` object entry, and no attributes match any `ids-proxy-con-forbidden-return` attribute values, then it is returned.

The following book can be used as a reference on regular expressions: *Mastering Regular Expressions*, by Friedl and Oram, published by O'Reilly, ISBN: 1565922573.

ids-proxy-con-ava

The `ids-proxy-con-ava` is a multi-valued attribute that can be used to specify a list of attribute name / value pairs. If an entry has an attribute name / value pair that matches any of the specified attribute name / value then that entry or some of its content will be hidden. The `ids-proxy-con-ava` attribute has the following syntax:

`<attribute-name>:<attribute-value>`

For example if you want to restrict all entries which have either "`ou=internal`" or "`secret=yes`" as one if its attributes, then you can specify the following:

```
ids-proxy-con-ava: ou:internal
ids-proxy-con-ava: secret:yes
```

ids-proxy-con-forbidden-return

The `ids-proxy-con-forbidden-return` attribute specifies the name of an attribute that is to be excluded from the entry that has matched any of the above specifications. This is a multi-valued attribute. If this attribute is not present and the `ids-proxy-con-permitted-return` attribute is also not present in the `ids-proxy-sch-ForbiddenEntryProperty`, then the whole entry is hidden.

ids-proxy-con-permitted-return

The `ids-proxy-con-permitted-return` attribute specified the name of an attribute that may be returned as part of the entry that has matched any of the above specifications. This is a multi valued attribute.

ids-proxy-sch-LDAPServer object class

This object class is sub-classed from the `ids-proxy-sch-property` object class. The `ids-proxy-sch-LDAPServer` property is used to define the backend LDAP servers. In order to reduce DNS lookup overhead, iDAR caches the DNS entry of the LDAP server host. Thus, if the IP address of your backend server changes, you must either restart iDAR or send it a HUP signal (UNIX only) so that it rereads it configuration.

ids-proxy-con-host

The `ids-proxy-con-host` attribute specifies the full DNS or IP address name of the host where the backend LDAP server is running. This attribute is mandatory.

ids-proxy-con-port

The `ids-proxy-con-port` attribute specifies the port number on which the backend LDAP server is running. The default port used if this attribute is absent is `389`.

ids-proxy-con-sport

The `ids-proxy-con-sport` attribute takes as a value the port number on which the backend server listens for `LDAPS` (LDAP over SSL) connections. Note that this is the non standard mechanism for supporting LDAP over SSL. Do not set any value for this attribute if the backend LDAP server does not support `LDAPS`.

ids-proxy-con-supported-version

The `ids-proxy-con-supported-version` specifies if the server defined by this entry supports only LDAP version 2 (`2`) or both LDAP version 2 and LDAP version 3 (`23`). The default is `23`.

ids-proxy-con-use-version

This attribute tells iDAR the preferred LDAP protocol version to use when talking to the backend server this entry defines. Permitted values are:

`2` Only use LDAPv2
`3` Only use LDAPv3
`23` Use the version that the client is using. This is also the default.

This option is useful when you have an LDAPv2 client for which iDAR needs to follow referrals. In this case iDAR itself needs to connect as LDAPv3 client to the backend server in order for the backend server to send referrals back to it. This attribute value must be set to `3` if the attribute `ids-proxy-con-allow-multi-ldapv2-bind` in object class `ids-proxy-sch-GlobalConfiguration` is set to `TRUE`.

ids-proxy-con-tcp-no-delay

If the `ids-proxy-con-tcp-no-delay` attribute is set to `FALSE` then iDAR will enable the Nagel Algorithm on connections to this server. This option must be set to `FALSE` only if the network bandwidth between iDAR and the server defined by this object entry is very limited. The default is `TRUE`.

ids-proxy-con-link-security-policy

The `ids-proxy-con-link-security-policy` defines the security policy for connections between iDAR and the backend server. If it is set to `ssl_always`, iDAR will establish a TLS session with the backend server before performing any operation. If it is set to `ssl_never` iDAR will not establish a TLS session with the backend server. Finally, if `ssl_optional` is specified, iDAR will only establish a TLS session with the backend server when the client establishes one with iDAR. This is the default behavior.

ids-proxy-con-x509cert-subject

The `ids-proxy-con-x509cert-subject` specifies the name of the LDAP server as specified by the `subject` field of its certificate. This attribute has a `dn` syntax. If this attribute is specified, iDAR will match it with the value of `subject` attribute present in the server certificate. iDAR will reject a TLS session with the LDAP server in case of a mismatch.

This attribute allows iDAR to authenticate the LDAP server it is connecting to.

ids-proxy-con-keepalive-interval

The value of `ids-proxy-con-keepalive-interval` is used by iDAR to determine if the network link to an LDAP directory server is down or if the LDAP directory server has become unresponsive. If the client connected to iDAR has pending operations and if iDAR has not received any data from the connection's LDAP server for the number of seconds specified by `ids-proxy-con-keepalive-interval` attribute, then iDAR will test the availability of the LDAP server by opening another communication channel to it. If iDAR is unsuccessful in doing so it will fail over to another LDAP server if available. If it is successful is establishing a second channel, iDAR will wait for another interval to receive data from the LDAP server. If iDAR has still not received any data from the LDAP directory server, it will assume that the LDAP

server is in an unresponsive state and fail over to another directory server if available. The default value for this attribute is 180 seconds. It is recomended that you increase this value if the LDAP server is not on the same local network as iDAR.

ids-proxy-sch-LogProperty object class

This object class is sub-classed from the `ids-proxy-sch-property` object class. The `ids-proxy-sch-LogProperty` object class can be used to define the logging characteristics of iDAR. This property can only be referred by the `ids-proxy-sch-GlobalConfiguration` object entry, using the `ids-proxy-con-include-logproperty` attribute.

Please refer to `ids-proxy-sch-LogProperty object class` on page 97 for more details of this property.

Chapter 8

Rules Configuration

Rule objects are used to specify conditions which are evaluated by iPlanet Directory Access Router at predetermined states. If these conditions are satisfied, then iDAR could optionally take certain actions as defined in the next chapter.

ids-proxy-sch-Rule object class

This object class defines the rule object attributes. It contains the following attributes.

ids-proxy-con-Name

The `ids-proxy-con-Name` attribute specifies the name of the rule. This attribute value must be unique within the set of rules. This attribute must be present as it forms the RDN of entries of this class.

ids-proxy-con-Priority

The `ids-proxy-con-Priority` attribute specifies the order of evaluation is reserved for future versions of this product.

ids-proxy-sch-Enable

The `ids-proxy-sch-Enable` attribute specifies if this object entry is part of the configuration or not. For an entry to be part of iDAR configuration, the `ids-proxy-sch-Enable` attribute must be present with a value `TRUE`. If the `ids-proxy-sch-Enable` attribute is not present or has the value `FALSE`, the entry is ignored.

ids-proxy-sch-belongs-to

This is a multi-valued attribute that specifies which server configurations this object entry belongs to. Each value of this attribute should be the value of the `ids-proxy-con-server-name` attribute in the `ids-proxy-sch-LDAPProxy` object entry.

ids-proxy-con-execute

The `ids-proxy-con-execute` is a multi-valued attribute that specifies the actions to take if the rule evaluates to `TRUE`. If this action does not exist then no actions will be taken. The `ids-proxy-con-execute` attribute has the following syntax:

`<action-name>#<precedence>`

The `action-name` part is the value of the `ids-proxy-con-Name` attribute in the `ids-proxy-sch-Action` object entry that is to be executed.

The `precedence` part is a positive integer. This integer determines the order of execution of the actions. Actions with higher precedence are executed first. If two actions have the same precedence, their order of execution is undefined.

ids-proxy-sch-OnBindSuccessRule object class

This object class is sub-classed from the `ids-proxy-sch-Rule` object class. This rule is evaluated when a client successfully completes a bind operation.

ids-proxy-con-ssl-required

If the `ids-proxy-con-ssl-required` attribute is set to `TRUE` then the `ids-proxy-sch-OnBindSuccessRule` rule will evaluate to true only if the client has established an SSL session with iDAR. The default is `FALSE`.

ids-proxy-con-bind-anonymous

If the `ids-proxy-con-bind-anonymous` attribute is set to `TRUE` then the `ids-proxy-sch-OnBindSuccessRule` rule will evaluate to true when the condition in `ids-proxy-con-ssl-required` is satisfied and the client has just completed a successful anonymous bind.

ids-proxy-con-bind-simple

If the `ids-proxy-con-bind-simple` attribute is set to `TRUE` then the `ids-proxy-sch-OnBindSuccessRule` rule will evaluate to true when the condition in `ids-proxy-con-ssl-required` is satisfied and the client has just completed a successful password based bind.

ids-proxy-con-bind-sasl

If the `ids-proxy-con-bind-sasl` attribute is set to `TRUE` then the `ids-proxy-sch-OnBindSuccessRule` rule will evaluate to true when the condition in `ids-proxy-con-ssl-required` is satisfied and the client has just completed a successful bind using any SASL mechanism.

ids-proxy-sch-OnSSLEstablishedRule object class

This object class is sub-classed from the `ids-proxy-sch-Rule` object class. This rule is evaluated when a client successfully establishes an SSL session. This rule does not have any attributes.

Chapter 9

Actions Configuration

Action objects are used to specify actions to take if a rule evaluates to TRUE.

ids-proxy-sch-Action object class

This object class defines the action object attributes. It contains the following attributes.

ids-proxy-con-Name

The `ids-proxy-con-Name` attribute specifies the name of the Action. This attribute value must be unique within the set of actions. This attribute must be present as it forms the RDN of entries of this class.

ids-proxy-con-Priority

The `ids-proxy-con-Priority` attribute is reserved for future versions of this product.

ids-proxy-sch-Enable

The `ids-proxy-sch-Enable` attribute specifies if this object entry is part of the configuration or not. For an entry to be part of the iDAR configuration, the `ids-proxy-sch-Enable` attribute must be present with a value `TRUE`. If the `ids-proxy-sch-Enable` attribute is not present or has the value `FALSE`, the entry is ignored.

ids-proxy-sch-belongs-to

This is a multi-valued attribute that specifies which server configurations this object entry belongs to. Each value of this attribute should be the value of the `ids-proxy-con-server-Name` attribute in the `ids-proxy-sch-LDAPProxy` object entry.

ids-proxy-sch-ChangeGroupAction object class

This object class is sub-classed from the `ids-proxy-sch-Action` object class. It can be used to specify the groups that a client can change to. It has the following multivalued attribute.

ids-proxy-con-to-group

The `ids-proxy-con-to-group` specifies the group the client can change to. It has the following syntax:

```
( <group-name>|NONE )
#<normal-DN-regexp>#<precedence>
```

`group-name`: The value of `ids-proxy-con-Name` attribute from a `ids-proxy-sch-Group` object entry. The special value `NONE` indicates no change is to be made.

`normal-DN-regexp`: The clients DN must match this regular expression. This regular expression must be normalized, i.e., there should be no spaces in between RDN components and "=" sign, all attribute names must be capitalized, and attribute values with `cis` syntax must be capitalized.

`precedence`: This must be a positive integer. This determines the order in which the change of group will be tried. `ids-proxy-con-to-group` value with higher value will be tried first.

For example, if you want all clients who have `cn=manager` as their least significant RDN component in their bind DN, into the manager group and not do anything with other bind credentials, you must specify the following:

```
ids-proxy-con-to-group: manager#CN=MANAGER,.*#100
ids-proxy-con-to-group: NONE#.*#10
```

The following book can be used as a reference on regular expressions: *Mastering Regular Expressions*, by Friedl and Oram, published by O'Reilly, ISBN: 1565922573.

Chapter 10

Logging

Logging is configured in iPlanet Directory Access Router by the `ids-proxy-sch-LogProperty` object class.

ids-proxy-sch-LogProperty object class

This object class is sub-classed from the `ids-proxy-sch-property` object class. The `ids-proxy-sch-LogProperty` object class can be used to define the logging characteristics of iDAR. This property can only be referred by the `ids-proxy-sch-GlobalConfiguration` object entry, using the `ids-proxy-con-include-logproperty` attribute.

Before iDAR reads this property, log messages are either written to a file on disk or optionally on platforms other than Windows NT, iDAR can send them to the `syslog` daemon. You cannot send the log messages to both a file and to the `syslog` daemon at the same time. This file defaults to `"[Program Files]\iPlanet\iDAR\logs\fwd.log"` on Windows NT and `/opt/iDAR/logs/fwd.log` on other platforms. If the environment variable `IDAR_ROOT` is defined then logs are written to the file `"IDAR_ROOT"/logs/fwd.log`. The `"IDAR_ROOT"/logs` directory must exist with write permissions to iDAR.

On platforms other than Windows NT, you can optionally send these initial log messages to the `syslog` daemon by specifying the `-s` flag on startup. iDAR uses the `daemon` facility, with the `warning`, `info` and `debug` priorities. If you choose this route, make sure that your `syslogd` is properly configured. For example, to have all the messages written to a particular file `/var/adm/messages` the following line must be added to the file `/etc/syslog.conf`.

```
daemon.warning;daemon.info;daemon.debug   /var/adm/messages
```

If the ids-proxy-sch-LogProperty object entry is not specified in a configuration iDAR continues to log in the default location. This object class has the following attributes.

ids-proxy-con-log-level

The ids-proxy-con-log-level attribute specifies the amount of logging iDAR will do. It can have any one of the following values in order of detail.

```
critical
exception
warning
notice
trace
detail_trace
```

A more detailed log level automatically implies all levels. For example, if the value of warning is specified, which is also the default, then exception, and critical level messages will also be logged.

ids-proxy-con-stat-level

The ids-proxy-con-stat-level attribute specifies the different kinds of statistics iDAR will log. The value of this attribute is a '|' separated list of the following options.

Table 10-1 Logging Levels

Option	Description
none	No statistics will be logged.
mods	Statistics about operations that write to the directory like add, modify, delete will be logged.
op	Statistics about all LDAP operations will be logged.
conn	Statistics about network connections will be logged.
stat	General statistics such as how many clients are connected will be logged.
audit	Audit information such as DN of client who just completed a bind / unbind will be logged.
full	All of the above except none.

For example, if you want to log audit and connection statistics only you must specify the following in the `ids-proxy-sch-LogProperty` object entry.

```
ids-proxy-con-stat-level: conn|audit
```

The default is none.

ids-proxy-con-log-syslog

The `ids-proxy-con-log-syslog` attribute specifies the syslog facility code if logging is being done through the `syslogd` daemon. The attribute is not available on Windows NT. The default is `LOG_DAEMON`. On all platforms other than Windows NT, and if the environment variable `IDAR_ROOT` is not defined, logging is done throughout the `syslogd` daemon by default. If `IDAR_ROOT` is defined, logging is done to the file `$(IDAR_ROOT)/logs/fwd.log` by default. See “Starting the Directory Access Router,” on page 101 for more information.

ids-proxy-con-log-file

The `ids-proxy-con-log-file` takes as value the full path name of the file you want the log messages to go to. Only one of the `ids-proxy-con-log-syslog` and `ids-proxy-con-log-file` attributes must be specified. On Windows NT, if this attribute is not specified then the log messages are sent to `"[Program Files]\iPlanet\iDAR\log\fwd.log"` by default.

ids-proxy-con-audit-syslog

The `ids-proxy-con-audit-syslog` attribute is essentially the same as the `ids-proxy-con-log-syslog` attribute. However, this attribute only effects the audit log messages. If this attribute is absent, all audit messages are logged with other log messages specified by the `ids-proxy-con-log-syslog` or `ids-proxy-con-log-file` attributes.

ids-proxy-con-audit-file

The `ids-proxy-con-audit-file` attribute is essentially the same as the `ids-proxy-con-log-file` attribute. However, this attribute only effects the audit log messages. Only one of `ids-proxy-con-audit-syslog` or `ids-proxy-con-audit-file` attributes can be specified. If this attribute is absent, all audit messages are logged with other log messages specified by the `ids-proxy-con-log-syslog` or `ids-proxy-con-log-file` attributes.

Starting the Directory Access Router

The iPlanet Directory Access Router program runs as a UNIX daemon process or an NT service normally started during system boot time.

On UNIX platforms, the iDAR program resides at:

`/opt/iDAR/sbin/ldapfwd`

The startup configuration file resides at:

`/opt/iDAR/etc/tailor.txt`

On Windows NT the location of the startup configuration file is:

`[Program Files]\iPlanet\iDAR\etc\tailor.txt`

The Windows NT Service Manager should be used to start and stop iDAR.

If you want the initial log messages to go to a location other than the default location or if you are using the `-M` option, define the environment variable `IDAR_ROOT`.

On platforms other than Windows NT, iDAR will only produce a `core` image in case of a crash if its effective user ID is same as its real user ID. Hence, if you want iDAR to produce a core you must set the `ids-proxy-con-userid` attribute in object class `ids-proxy-sch-GlobalConfiguration` to the same user who starts iDAR process. By default, if iDAR is run by `root`, it changes its userid to `nobody`.

The following flags are supported:

`-d`

When this flag present, iDAR will only handle a single incoming connection at a time, and will send more detailed internal tracing information to the log file. This flag should not be used during normal operation, since it will prevent the iDAR daemon from detaching from the controlling terminal.

`-D`

This flag tells iDAR to send more detailed tracing information to the log file. iDAR will still handle multiple client connections and run as a daemon. One of `-d` or `-D` must be used.

`-I`

This flag is only applicable on Windows NT, and is ignored on all other platforms. This flag must be used when starting iDAR program `ldapfwd` from the command line. The `-d` flag should be used in conjunction with the `-I` flag on Windows NT to get the debug output.

`-t <startup configuration file>`

This option can be used to specify an alternate startup configuration file.

`-s`

This option tells iDAR to send the initial log messages to the `syslogd` using the `LOG_DAEMON` facility. This flag is ignored on Windows NT. This is the default if the environment variable `IDAR_ROOT` is not defined.

`-M`

If this flag is specified, iDAR will spawn another process to monitor itself. In the case where iDAR exits ungracefully, the monitor process restarts iDAR after waiting for 30 seconds. This is not available on Windows NT.

`-v`

This flag prints the version information for iDAR. On Windows NT, this flag should only be used from the command line.

Chapter 12

iDAR Configuration Reload

On UNIX platforms, iPlanet Directory Access Router can be sent a `SIGHUP` signal to make it re-read its configuration. If the configuration is re-read successfully, iDAR will use this new configuration for future connections. Client connections that are already established will continue to use the old configuration until the clients disconnect.

For example, use the following command to send the `HUP` signal to iDAR where `Kill-hup Pid` is the process ID of `ldapfwd`.

Some attribute values cannot be changed using the `HUP` signal facility. For changes to the following configuration parameters, iDAR will have to be shut down and started again. These attributes include:

```
ids-proxy-con-listen-port
ids-proxy-con-listen-host
ids-proxy-con-ldaps-port
ids-proxy-con-foreground
ids-proxy-con-listen-backlog
ids-proxy-con-ssl-cert
ids-proxy-con-ssl-key
```

Also, the logging properties `ids-proxy-sch-LogProperty` cannot be changed using this facility.

Appendix A

Examples: tailor.ldif

Code Example A-1 Straight Through Configuration's LDIF

```
version: 1

dn: ou=actions
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: actions

dn: ou=global
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: global

dn: ou=groups
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: groups

dn: ou=properties
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: properties

dn: ou=rules
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: rules

#
# This bootstrap entry defines:
#
# 1) the name of the iDAR server set is: "iDAR"
# 2) the DN of the ids-proxy-sch-GlobalConfiguration object
# 3) the base DN of the Properties definitions
# 4) the base DN of the Groups definitions
#
# Note: iDAR learns this entry's DN via the startup configuration (tailor.txt).
```

Code Example A-1 Straight Through Configuration's LDIF *(Continued)*

```
#
dn: ids-proxy-con-Server-Name=iDAR
objectclass: ids-proxy-sch-LDAPProxy
objectclass: ids-proxy-top
ids-proxy-con-Server-Name: iDAR
ids-proxy-sch-Global-Config-Dn: ids-proxy-con-Config-Name=config,ou=global
ids-proxy-sch-Group-Base: ou=groups
ids-proxy-sch-Property-Base: ou=properties
ids-proxy-sch-Rule-Base: ou=rules

#
# This Global properties entry declares:
#
# 1) which port the proxy will listen on for connections
#
dn: ids-proxy-con-Config-Name=config,ou=global
objectclass: ids-proxy-sch-GlobalConfiguration
objectclass: ids-proxy-top
ids-proxy-con-Config-Name: config
ids-proxy-con-connection-pool: FALSE
ids-proxy-con-connection-pool-interval: 15
ids-proxy-con-connection-pool-timeout: 30
ids-proxy-con-include-logproperty: LogProperty
ids-proxy-con-listen-backlog: 128
ids-proxy-con-listen-host: localhost
ids-proxy-con-listen-port: 389
ids-proxy-con-one-thread-only: FALSE
ids-proxy-con-userid: nobody
ids-proxy-con-working-dir: /tmp
ou: global

#
# Here we stipulate to iDAR where to send its log file as well
# as what logging level to use.
#
dn: ids-proxy-con-Name=LogProperty,ou=properties
objectclass: ids-proxy-sch-LogProperty
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: LogProperty
ids-proxy-con-log-file: /opt/iDAR/logs/fwd.log
ids-proxy-con-log-level: warning
ids-proxy-con-stat-level: none
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# iDAR finds its group properties by essentially performing a one level search
# beneath the ids-proxy-sch-LDAPProxy stipulated ids-proxy-sch-Group-Base.
# (This is in contrast with all other entries which are "found" by an exact DN
# to utilize.)  Since there could be more than one entry beneath the
# ids-proxy-sch-Group-Base, each group entry has an ids-proxy-con-Priority
# value to indicate their evaluation order. In addition, the
```

Code Example A-1 Straight Through Configuration's LDIF *(Continued)*

```
# ids-proxy-sch-belongs-to attribute allows entries at this level to exclude
# or include themselves based on if their ids-proxy-sch-belongs-to value
# matches the value asserted by the ids-proxy-con-Server-Name value found
# in the ids-proxy-sch-LDAPProxy objectclass entry.
#
# The following entry defines a group called "default" that:
#
# 1) assigns an evaluation priority of 33 (evaluates before 32, 31, 30, etc.)
# 2) declares the server configuration this entry belongs to as "iDAR"
# 3) allows all clients to be a member of the group as con-Client is ALL
# 4) allows all clients regardless of bind name (since it isn't enumerated)
# 5) allows all operations
# 6) stipulates that referrals are returned to the proxy's client
# 7) stipulates that the backend server for this group is described by the
#    ids-proxy-sch-LDAPServer objectclass entry named server-1.
#
dn: ids-proxy-con-Name=default,ou=groups
objectclass: ids-proxy-sch-Group
objectclass: ids-proxy-sch-NetworkGroup
objectclass: ids-proxy-top
ids-proxy-con-Client: ALL
ids-proxy-con-Name: default
ids-proxy-con-Priority: 33
ids-proxy-con-Server: server-1
ids-proxy-con-allow-multi-ldapv2-bind: FALSE
ids-proxy-con-filter-inequality: TRUE
ids-proxy-con-max-refcount: 15
ids-proxy-con-max-scope: 2
ids-proxy-con-max-simultaneous-conns-from-ip: 0
ids-proxy-con-permit-auth-none: TRUE
ids-proxy-con-permit-auth-sasl: TRUE
ids-proxy-con-permit-auth-simple: TRUE
ids-proxy-con-permit-op-add: TRUE
ids-proxy-con-permit-op-compare: TRUE
ids-proxy-con-permit-op-delete: TRUE
ids-proxy-con-permit-op-extended: TRUE
ids-proxy-con-permit-op-modify: TRUE
ids-proxy-con-permit-op-modrdn: TRUE
ids-proxy-con-permit-op-search: TRUE
ids-proxy-con-reference: forward
ids-proxy-con-referral-bind-policy: bind_any
ids-proxy-con-referral-ssl-policy: ssl_unavailable
ids-proxy-con-search-reference: follow
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-con-timeout: 120
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: groups

#
# server enumeration
#
dn: ids-proxy-con-Name=server-1,ou=properties
objectclass: ids-proxy-sch-LDAPServer
```

Code Example A-1 Straight Through Configuration's LDIF *(Continued)*

```
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: server-1
ids-proxy-con-host: pink.iplanet.com
ids-proxy-con-link-security-policy: ssl_optional
ids-proxy-con-port: 10205
ids-proxy-con-supported-version: 23
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties
```

Code Example A-2 Load Balancing Configuration's LDIF

```
version: 1

dn: ou=actions
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: actions

dn: ou=global
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: global

dn: ou=groups
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: groups

dn: ou=properties
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: properties

dn: ou=rules
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: rules

#
# This bootstrap entry defines:
#
# 1) the name of the iDAR server set is: "iDAR"
# 2) the DN of the ids-proxy-sch-GlobalConfiguration object
# 3) the base DN of the Properties definitions
# 4) the base DN of the Groups definitions
```

Code Example A-2 Load Balancing Configuration's LDIF *(Continued)*

```
#
# Note: iDAR learns this entry's DN via the startup configuration (tailor.txt).
#
dn: ids-proxy-con-Server-Name=iDAR
objectclass: ids-proxy-sch-LDAPProxy
objectclass: ids-proxy-top
ids-proxy-con-Server-Name: iDAR
ids-proxy-sch-Global-Config-Dn: ids-proxy-con-Config-Name=config,ou=global
ids-proxy-sch-Group-Base: ou=groups
ids-proxy-sch-Property-Base: ou=properties
ids-proxy-sch-Rule-Base: ou=rules

#
# This Global properties entry declares:
#
# 1) which port the proxy will listen on for connections
#
dn: ids-proxy-con-Config-Name=config,ou=global
objectclass: ids-proxy-sch-GlobalConfiguration
objectclass: ids-proxy-top
ids-proxy-con-Config-Name: config
ids-proxy-con-connection-pool: FALSE
ids-proxy-con-connection-pool-interval: 15
ids-proxy-con-connection-pool-timeout: 30
ids-proxy-con-include-logproperty: LogProperty
ids-proxy-con-listen-backlog: 128
ids-proxy-con-listen-host: localhost
ids-proxy-con-listen-port: 389
ids-proxy-con-one-thread-only: FALSE
ids-proxy-con-userid: nobody
ids-proxy-con-working-dir: /tmp
ou: global

#
# Here we stipulate to iDAR where to send its log file as well
# as what logging level to use.
#
dn: ids-proxy-con-Name=LogProperty,ou=properties
objectclass: ids-proxy-sch-LogProperty
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: LogProperty
ids-proxy-con-log-file: /opt/iDAR/logs/fwd.log
ids-proxy-con-log-level: warning
ids-proxy-con-stat-level: none
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# iDAR finds its group properties by essentially performing a one level search
# beneath the ids-proxy-sch-LDAPProxy stipulated ids-proxy-sch-Group-Base.
# (This is in contrast with all other entries which are "found" by an exact DN
# to utilize.)  Since there could be more than one entry beneath the
```

Code Example A-2 Load Balancing Configuration's LDIF *(Continued)*

```
# ids-proxy-sch-Group-Base, each group entry has an ids-proxy-con-Priority
# value to indicate their evaluation order. In addition, the
# ids-proxy-sch-belongs-to attribute allows entries at this level to exclude
# or include themselves based on if their ids-proxy-sch-belongs-to value
# matches the value asserted by the ids-proxy-con-Server-Name value found
# in the ids-proxy-sch-LDAPProxy objectclass entry.
#
# The following entry defines a group called "default" that:
#
# 1) assigns an evaluation priority of 33 (evaluates before 32, 31, 30, etc.)
# 2) declares the server configuration this entry belongs to as "iDAR"
# 3) allows all clients to be a member of the group as con-Client is ALL
# 4) allows all clients regardless of bind name (since it isn't enumerated)
# 5) allows all operations
# 6) stipulates that referrals are returned to the proxy's client
# 7) stipulates that this group is to use load balancing.
#
dn: ids-proxy-con-Name=default,ou=groups
objectclass: ids-proxy-sch-Group
objectclass: ids-proxy-sch-NetworkGroup
objectclass: ids-proxy-top
ids-proxy-con-Client: ALL
ids-proxy-con-Name: default
ids-proxy-con-Priority: 33
ids-proxy-con-allow-multi-ldapv2-bind: FALSE
ids-proxy-con-filter-inequality: TRUE
ids-proxy-con-include-property: load-balance
ids-proxy-con-max-refcount: 15
ids-proxy-con-max-scope: 2
ids-proxy-con-max-simultaneous-conns-from-ip: 0
ids-proxy-con-permit-auth-none: TRUE
ids-proxy-con-permit-auth-sasl: TRUE
ids-proxy-con-permit-auth-simple: TRUE
ids-proxy-con-permit-op-add: TRUE
ids-proxy-con-permit-op-compare: TRUE
ids-proxy-con-permit-op-delete: TRUE
ids-proxy-con-permit-op-extended: TRUE
ids-proxy-con-permit-op-modify: TRUE
ids-proxy-con-permit-op-modrdn: TRUE
ids-proxy-con-permit-op-search: TRUE
ids-proxy-con-reference: forward
ids-proxy-con-referral-bind-policy: bind_any
ids-proxy-con-referral-ssl-policy: ssl_unavailable
ids-proxy-con-search-reference: follow
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-con-timeout: 120
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: groups

#
# The following entry defines the property used to stipulate which servers
# will be contacted. It also apportions the servers.
#
```

Code Example A-2 Load Balancing Configuration's LDIF *(Continued)*

```
# Note: it is assumed that the load balanced servers are equal in content
# and in capability.
#
dn: ids-proxy-con-Name=load-balance,ou=properties
objectclass: ids-proxy-sch-LoadBalanceProperty
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: load-balance
ids-proxy-con-Server: server-1#34
ids-proxy-con-Server: server-2#33
ids-proxy-con-Server: server-3#33
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# server enumeration
#
dn: ids-proxy-con-Name=server-1,ou=properties
objectclass: ids-proxy-sch-LDAPServer
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: server-1
ids-proxy-con-host: pink
ids-proxy-con-link-security-policy: ssl_optional
ids-proxy-con-port: 10205
ids-proxy-con-supported-version: 23
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# server enumeration
#
dn: ids-proxy-con-Name=server-2,ou=properties
objectclass: ids-proxy-sch-LDAPServer
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: server-2
ids-proxy-con-host: red
ids-proxy-con-link-security-policy: ssl_optional
ids-proxy-con-port: 10389
ids-proxy-con-supported-version: 23
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# server enumeration
#
dn: ids-proxy-con-Name=server-3,ou=properties
objectclass: ids-proxy-sch-LDAPServer
```

Code Example A-2 Load Balancing Configuration's LDIF *(Continued)*

```
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: server-3
ids-proxy-con-host: blue
ids-proxy-con-link-security-policy: ssl_optional
ids-proxy-con-port: 389
ids-proxy-con-supported-version: 23
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties
```

Code Example A-3 Binding Based Operation Filtering's LDIF

```
version: 1

dn: ou=actions
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: actions

dn: ou=global
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: global

dn: ou=groups
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: groups

dn: ou=properties
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: properties

dn: ou=rules
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: rules

#
# This bootstrap entry defines:
#
# 1) the name of the iDAR server set is: "iDAR"
# 2) the DN of the ids-proxy-sch-GlobalConfiguration object
# 3) the base DN of the Properties definitions
# 4) the base DN of the Groups definitions
```

Code Example A-3 Binding Based Operation Filtering's LDIF *(Continued)*

```
#
# Note: iDAR learns this entry's DN via the startup configuration (tailor.txt).
#
dn: ids-proxy-con-Server-Name=iDAR
objectclass: ids-proxy-sch-LDAPProxy
objectclass: ids-proxy-top
ids-proxy-con-Server-Name: iDAR
ids-proxy-sch-Action-Base: ou=actions
ids-proxy-sch-Global-Config-Dn: ids-proxy-con-Config-Name=config,ou=global
ids-proxy-sch-Group-Base: ou=groups
ids-proxy-sch-Property-Base: ou=properties
ids-proxy-sch-Rule-Base: ou=rules

#
# This Global properties entry declares:
#
# 1) which port the proxy will listen on for connections
#
dn: ids-proxy-con-Config-Name=config,ou=global
objectclass: ids-proxy-sch-GlobalConfiguration
objectclass: ids-proxy-top
ids-proxy-con-Config-Name: config
ids-proxy-con-connection-pool: FALSE
ids-proxy-con-connection-pool-interval: 15
ids-proxy-con-connection-pool-timeout: 30
ids-proxy-con-include-logproperty: LogProperty
ids-proxy-con-listen-backlog: 128
ids-proxy-con-listen-host: localhost
ids-proxy-con-listen-port: 389
ids-proxy-con-one-thread-only: FALSE
ids-proxy-con-userid: nobody
ids-proxy-con-working-dir: /tmp
ou: global

#
# Here we stipulate to iDAR where to send its log file as well
# as what logging level to use.
#
dn: ids-proxy-con-Name=LogProperty,ou=properties
objectclass: ids-proxy-sch-LogProperty
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: LogProperty
ids-proxy-con-log-file: /opt/iDAR/logs/fwd.log
ids-proxy-con-log-level: warning
ids-proxy-con-stat-level: none
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# note that we change group on bind
#
dn: ids-proxy-con-Name=successfulBind,ou=rules
```

Code Example A-3 Binding Based Operation Filtering's LDIF *(Continued)*

```
objectclass: ids-proxy-sch-OnBindSuccessRule
objectclass: ids-proxy-sch-Rule
objectclass: ids-proxy-top
ids-proxy-con-Name: successfulBind
ids-proxy-con-bind-anonymous: FALSE
ids-proxy-con-bind-sasl: TRUE
ids-proxy-con-bind-simple: TRUE
ids-proxy-con-execute: change#100
ids-proxy-con-ssl-required: FALSE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: rules

#
# this is the group we changeto on successful bind
#
dn: ids-proxy-con-Name=change,ou=actions
objectclass: ids-proxy-sch-Action
objectclass: ids-proxy-sch-ChangeGroupAction
objectclass: ids-proxy-top
ids-proxy-con-Name: change
ids-proxy-con-to-group: onbind#.*#5
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: actions

#
# iDAR finds its group properties by essentially performing a one level search
# beneath the ids-proxy-sch-LDAPProxy stipulated ids-proxy-sch-Group-Base.
# (This is in contrast with all other entries which are "found" by an exact DN
# to utilize.)  Since there could be more than one entry beneath the
# ids-proxy-sch-Group-Base, each group entry has an ids-proxy-con-Priority
# value to indicate their evaluation order. In addition, the
# ids-proxy-sch-belongs-to attribute allows entries at this level to exclude
# or include themselves based on if their ids-proxy-sch-belongs-to value
# matches the value asserted by the ids-proxy-con-Server-Name value found
# in the ids-proxy-sch-LDAPProxy objectclass entry.
#
# The following entry defines a group called "onbind" that:
#
# 1) assigns an evaluation priority of 10 (evaluates before 9, 8, 7, etc.)
# 2) declares the server configuration this entry belongs to as "iDAR"
# 3) uses a con-Client value that FAILS matching during priority scanning
# 4) allows all clients regardless of bind name (since it isn't enumerated)
# 5) allows all operations
# 6) stipulates that referrals are returned to the proxy's client
# 7) stipulates that the backend server for this group is described by the
#    ids-proxy-sch-LDAPServer objectclass entry named server-1.
#
dn: ids-proxy-con-Name=onbind,ou=groups
objectclass: ids-proxy-sch-Group
objectclass: ids-proxy-sch-NetworkGroup
objectclass: ids-proxy-top
ids-proxy-con-Client: 0.0.0.0
```

Code Example A-3 Binding Based Operation Filtering's LDIF *(Continued)*

```
ids-proxy-con-Name: onbind
ids-proxy-con-Priority: 10
ids-proxy-con-Server: server-1
ids-proxy-con-allow-multi-ldapv2-bind: FALSE
ids-proxy-con-filter-inequality: TRUE
ids-proxy-con-max-refcount: 15
ids-proxy-con-max-scope: 2
ids-proxy-con-max-simultaneous-conns-from-ip: 0
ids-proxy-con-permit-auth-none: TRUE
ids-proxy-con-permit-auth-sasl: TRUE
ids-proxy-con-permit-auth-simple: TRUE
ids-proxy-con-permit-op-add: TRUE
ids-proxy-con-permit-op-compare: TRUE
ids-proxy-con-permit-op-delete: TRUE
ids-proxy-con-permit-op-extended: TRUE
ids-proxy-con-permit-op-modify: TRUE
ids-proxy-con-permit-op-modrdn: TRUE
ids-proxy-con-permit-op-search: TRUE
ids-proxy-con-reference: forward
ids-proxy-con-referral-bind-policy: bind_any
ids-proxy-con-referral-ssl-policy: ssl_unavailable
ids-proxy-con-search-reference: follow
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-con-timeout: 120
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: groups

#
# The following entry defines a group called "default" that:
#
# 1) assigns an evaluation priority of 33 (evaluates before 32, 31, 30, etc.)
# 2) declares the server configuration this entry belongs to as "iDAR"
# 3) allows all clients to be a member of the group as con-Client is ALL
# 4) allows all clients regardless of bind name (since it isn't enumerated)
# 5) disallows add, compare, delete, modify, modrdn and extended operations
# 6) stipulates that referrals are returned to the proxy's client
# 7) stipulates that the backend server for this group is described by the
#    ids-proxy-sch-LDAPServer objectclass entry named server-1.
#
dn: ids-proxy-con-Name=default,ou=groups
objectclass: ids-proxy-sch-Group
objectclass: ids-proxy-sch-NetworkGroup
objectclass: ids-proxy-top
ids-proxy-con-Client: ALL
ids-proxy-con-Name: default
ids-proxy-con-Priority: 33
ids-proxy-con-Server: server-1
ids-proxy-con-allow-multi-ldapv2-bind: FALSE
ids-proxy-con-filter-inequality: TRUE
ids-proxy-con-include-rule: successfulBind
ids-proxy-con-max-refcount: 15
ids-proxy-con-max-scope: 2
ids-proxy-con-max-simultaneous-conns-from-ip: 0
```

Code Example A-3 Binding Based Operation Filtering's LDIF *(Continued)*

```
ids-proxy-con-permit-auth-none: TRUE
ids-proxy-con-permit-auth-sasl: TRUE
ids-proxy-con-permit-auth-simple: TRUE
ids-proxy-con-permit-op-add: FALSE
ids-proxy-con-permit-op-compare: FALSE
ids-proxy-con-permit-op-delete: FALSE
ids-proxy-con-permit-op-extended: FALSE
ids-proxy-con-permit-op-modify: FALSE
ids-proxy-con-permit-op-modrdn: FALSE
ids-proxy-con-permit-op-search: TRUE
ids-proxy-con-reference: forward
ids-proxy-con-referral-bind-policy: bind_any
ids-proxy-con-referral-ssl-policy: ssl_unavailable
ids-proxy-con-search-reference: follow
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-con-timeout: 120
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: groups

#
# server enumeration
#
dn: ids-proxy-con-Name=server-1,ou=properties
objectclass: ids-proxy-sch-LDAPServer
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: server-1
ids-proxy-con-host: blue
ids-proxy-con-link-security-policy: ssl_optional
ids-proxy-con-port: 389
ids-proxy-con-supported-version: 23
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties
```

Code Example A-4 Load Balancing and Binding Based Filtering's LDIF

```
version: 1

dn: ou=actions
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: actions

dn: ou=global
objectclass: ids-proxy-top
```

Code Example A-4 Load Balancing and Binding Based Filtering's LDIF *(Continued)*

```
objectclass: organizationalUnit
ou: global

dn: ou=groups
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: groups

dn: ou=properties
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: properties

dn: ou=rules
objectclass: ids-proxy-top
objectclass: organizationalUnit
ou: rules

#
# This bootstrap entry defines:
#
# 1) the name of the iDAR server set is: "iDAR"
# 2) the DN of the ids-proxy-sch-GlobalConfiguration object
# 3) the base DN of the Properties definitions
# 4) the base DN of the Groups definitions
#
# Note: iDAR learns this entry's DN via the startup configuration (tailor.txt).
#
dn: ids-proxy-con-Server-Name=iDAR
objectclass: ids-proxy-sch-LDAPProxy
objectclass: ids-proxy-top
ids-proxy-con-Server-Name: iDAR
ids-proxy-sch-Action-Base: ou=actions
ids-proxy-sch-Global-Config-Dn: ids-proxy-con-Config-Name=config,ou=global
ids-proxy-sch-Group-Base: ou=groups
ids-proxy-sch-Property-Base: ou=properties
ids-proxy-sch-Rule-Base: ou=rules

#
# This Global properties entry declares:
#
# 1) which port the proxy will listen on for connections
#
dn: ids-proxy-con-Config-Name=config,ou=global
objectclass: ids-proxy-sch-GlobalConfiguration
objectclass: ids-proxy-top
ids-proxy-con-Config-Name: config
ids-proxy-con-connection-pool: FALSE
ids-proxy-con-connection-pool-interval: 15
ids-proxy-con-connection-pool-timeout: 30
ids-proxy-con-include-logproperty: LogProperty
ids-proxy-con-listen-backlog: 128
ids-proxy-con-listen-host: localhost
ids-proxy-con-listen-port: 389
```

Code Example A-4 Load Balancing and Binding Based Filtering's LDIF *(Continued)*

```
ids-proxy-con-one-thread-only: FALSE
ids-proxy-con-userid: nobody
ids-proxy-con-working-dir: /tmp
ou: global

#
# Here we stipulate to iDAR where to send its log file as well
# as what logging level to use.
#
dn: ids-proxy-con-Name=LogProperty,ou=properties
objectclass: ids-proxy-sch-LogProperty
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: LogProperty
ids-proxy-con-log-file: /opt/iDAR/logs/fwd.log
ids-proxy-con-log-level: warning
ids-proxy-con-stat-level: none
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# note that we change group on bind
#
dn: ids-proxy-con-Name=successfulBind,ou=rules
objectclass: ids-proxy-sch-OnBindSuccessRule
objectclass: ids-proxy-sch-Rule
objectclass: ids-proxy-top
ids-proxy-con-Name: successfulBind
ids-proxy-con-bind-anonymous: FALSE
ids-proxy-con-bind-sasl: TRUE
ids-proxy-con-bind-simple: TRUE
ids-proxy-con-execute: change#100
ids-proxy-con-ssl-required: FALSE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: rules

#
# this is the group we changeto on successful bind
#
dn: ids-proxy-con-Name=change,ou=actions
objectclass: ids-proxy-sch-Action
objectclass: ids-proxy-sch-ChangeGroupAction
objectclass: ids-proxy-top
ids-proxy-con-Name: change
ids-proxy-con-to-group: onbind#.*#5
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: actions

#
# iDAR finds its group properties by essentially performing a one level search
# beneath the ids-proxy-sch-LDAPProxy stipulated ids-proxy-sch-Group-Base.
```

Code Example A-4 Load Balancing and Binding Based Filtering's LDIF *(Continued)*

```
# (This is in contrast with all other entries which are "found" by an exact DN
# to utilize.)  Since there could be more than one entry beneath the
# ids-proxy-sch-Group-Base, each group entry has an ids-proxy-con-Priority
# value to indicate their evaluation order. In addition, the
# ids-proxy-sch-belongs-to attribute allows entries at this level to exclude
# or include themselves based on if their ids-proxy-sch-belongs-to value
# matches the value asserted by the ids-proxy-con-Server-Name value found
# in the ids-proxy-sch-LDAPProxy objectclass entry.
#
# The following entry defines a group called "onbind" that:
#
# 1) assigns an evaluation priority of 10 (evaluates before 9, 8, 7, etc.)
# 2) declares the server configuration this entry belongs to as "iDAR"
# 3) uses a con-Client value that FAILS matching during priority scanning
# 4) allows all clients regardless of bind name (since it isn't enumerated)
# 5) allows all operations
# 6) stipulates that referrals are returned to the proxy's client
# 7) stipulates that this group is to use load balancing.
#
dn: ids-proxy-con-Name=onbind,ou=groups
objectclass: ids-proxy-sch-Group
objectclass: ids-proxy-sch-NetworkGroup
objectclass: ids-proxy-top
ids-proxy-con-Client: 0.0.0.0
ids-proxy-con-Name: onbind
ids-proxy-con-Priority: 10
ids-proxy-con-allow-multi-ldapv2-bind: FALSE
ids-proxy-con-filter-inequality: TRUE
ids-proxy-con-include-property: load-balance
ids-proxy-con-max-refcount: 15
ids-proxy-con-max-scope: 2
ids-proxy-con-max-simultaneous-conns-from-ip: 0
ids-proxy-con-permit-auth-none: TRUE
ids-proxy-con-permit-auth-sasl: TRUE
ids-proxy-con-permit-auth-simple: TRUE
ids-proxy-con-permit-op-add: TRUE
ids-proxy-con-permit-op-compare: TRUE
ids-proxy-con-permit-op-delete: TRUE
ids-proxy-con-permit-op-extended: TRUE
ids-proxy-con-permit-op-modify: TRUE
ids-proxy-con-permit-op-modrdn: TRUE
ids-proxy-con-permit-op-search: TRUE
ids-proxy-con-reference: forward
ids-proxy-con-referral-bind-policy: bind_any
ids-proxy-con-referral-ssl-policy: ssl_unavailable
ids-proxy-con-search-reference: follow
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-con-timeout: 120
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: groups

#
# The following entry defines a group called "default" that:
```

Code Example A-4 Load Balancing and Binding Based Filtering's LDIF *(Continued)*

```
#
# 1) assigns an evaluation priority of 33 (evaluates before 32, 31, 30, etc.)
# 2) declares the server configuration this entry belongs to as "iDAR"
# 3) allows all clients to be a member of the group as con-Client is ALL
# 4) allows all clients regardless of bind name (since it isn't enumerated)
# 5) disallows add, compare, delete, modify, modrdn and extended operations
# 6) stipulates that referrals are returned to the proxy's client
# 7) stipulates that this group is to use load balancing.
#
dn: ids-proxy-con-Name=default,ou=groups
objectclass: ids-proxy-sch-Group
objectclass: ids-proxy-sch-NetworkGroup
objectclass: ids-proxy-top
ids-proxy-con-Client: ALL
ids-proxy-con-Name: default
ids-proxy-con-Priority: 33
ids-proxy-con-allow-multi-ldapv2-bind: FALSE
ids-proxy-con-filter-inequality: TRUE
ids-proxy-con-include-property: load-balance
ids-proxy-con-include-rule: successfulBind
ids-proxy-con-max-refcount: 15
ids-proxy-con-max-scope: 2
ids-proxy-con-max-simultaneous-conns-from-ip: 0
ids-proxy-con-permit-auth-none: TRUE
ids-proxy-con-permit-auth-sasl: TRUE
ids-proxy-con-permit-auth-simple: TRUE
ids-proxy-con-permit-op-add: FALSE
ids-proxy-con-permit-op-compare: FALSE
ids-proxy-con-permit-op-delete: FALSE
ids-proxy-con-permit-op-extended: FALSE
ids-proxy-con-permit-op-modify: FALSE
ids-proxy-con-permit-op-modrdn: FALSE
ids-proxy-con-permit-op-search: TRUE
ids-proxy-con-reference: forward
ids-proxy-con-referral-bind-policy: bind_any
ids-proxy-con-referral-ssl-policy: ssl_unavailable
ids-proxy-con-search-reference: follow
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-con-timeout: 120
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: groups

#
# The following entry defines the property used to stipulate which servers
# will be contacted. It also apportions the servers.
#
# Note: it is assumed that the load balanced servers are equal in content
# and in capability.
#
dn: ids-proxy-con-Name=load-balance,ou=properties
objectclass: ids-proxy-sch-LoadBalanceProperty
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
```

Code Example A-4 Load Balancing and Binding Based Filtering's LDIF *(Continued)*

```
ids-proxy-con-Name: load-balance
ids-proxy-con-Server: server-1#34
ids-proxy-con-Server: server-2#33
ids-proxy-con-Server: server-3#33
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# server enumeration
#
dn: ids-proxy-con-Name=server-1,ou=properties
objectclass: ids-proxy-sch-LDAPServer
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: server-1
ids-proxy-con-host: pink
ids-proxy-con-link-security-policy: ssl_optional
ids-proxy-con-port: 10205
ids-proxy-con-supported-version: 23
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# server enumeration
#
dn: ids-proxy-con-Name=server-2,ou=properties
objectclass: ids-proxy-sch-LDAPServer
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: server-2
ids-proxy-con-host: red
ids-proxy-con-link-security-policy: ssl_optional
ids-proxy-con-port: 10389
ids-proxy-con-supported-version: 23
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties

#
# server enumeration
#
dn: ids-proxy-con-Name=server-3,ou=properties
objectclass: ids-proxy-sch-LDAPServer
objectclass: ids-proxy-sch-Property
objectclass: ids-proxy-top
ids-proxy-con-Name: server-3
ids-proxy-con-host: blue
ids-proxy-con-link-security-policy: ssl_optional
ids-proxy-con-port: 389
ids-proxy-con-supported-version: 23
```

Code Example A-4 Load Balancing and Binding Based Filtering's LDIF *(Continued)*

```
ids-proxy-con-tcp-no-delay: TRUE
ids-proxy-sch-Enable: TRUE
ids-proxy-sch-belongs-to: iDAR
ou: properties
```

TLS/SSL Scenario Matrix

iPlanet Directory Access Router can be configured to establish TLS/SSL between itself and the client and itself and the server in many different combinations. It supports both, the “Start TLS” method where clients establish TLS/SSL session over the standard LDAP port and the “alternate port 636“ method where clients connect to an alternate secure port if they want to establish TLS/SSL.

The following matrix summarizes the possible configuration scenarios. The columns represent connections between client and iDAR, and the rows represent connections between the iDAR and the backend LDAP directory server.

Table B-1 TLS/SSL Configuration Scenarios

		CLIENT		
	iDAR	No TLS	TLS using alternate port 636	TLS using START TLS mechanism
SERVER	No TLS	X	X	X
	TLS using alternate port 636	X	X	X
	TLS using START TLS mechanism	X	X	X

Appendix C

iDAR Error Guide

These are the error numbers and text logged by iPlanet Directory Access Router if some exception condition occurs. Only those errors that warrant an explanation are listed.

Unnumbered Errors

These errors are unnumbered and will only occur if something bad happened even before iDAR could initialize itself.

`Registry settings are not present`

Installation problem or the Registry entries deleted by the Administrator.

`Could not open log file %s`

On Windows NT it means `%logpath%\fwd.log` is not accessible where logpath is the variable in the registry.

On Unix platforms, iDAR was not started with `-s` option and `$PROXY_IDAR_ROOT/logs/fwd.log` is not accessible or `/opt/IIIlps/logs/fwd.log` is not accessible.

```
RegisterServiceCtrlHandler call failed
StartServiceCtrlDispatcher call failed
```

Only on Windows NT. Administrator did not reboot after installation. Or, bad installation. Or Administrator tried to run iDAR from console without the `-I` flag.

`Syslog is not available on Windows NT platform`

Administrator tried to configure syslog (in `ids-proxy-sch-LogProperty`) on Windows NT.

Numbered Errors

```
100203 Missing second quote in RDN AVA
100301 Too many AVAs in RDN (%d)
100302 Problem decoding RDN component
100303 Extra quote in RDN
100304 Problem decoding RDN component
```

Badly formed RDN component.

```
110201 Could not open %s
```

Could not open the initial tailoring file.

```
110202 Line %d too long/missing newline
```

Each line in the initial tailoring file must be terminated with a newline, or the line is more than 1024 characters long.

```
110203 Missing colon in line %d
```

Each option / value pair in the initial tailoring file must be separated by a colon.

```
110204 Unrecognized keyword in line %d (%s)
110205 Problem parsing line %d (%s)
```

Bad option name in initial tailoring file.

```
110207 Unknown configuration_url type %s
```

The protocol selector must be either `ldap` or `file` for `configuration_url` in the initial tailoring file.

```
110303 Problem parsing (%s)
```

One of the attribute value in the `ids-proxy-sch-GlobalConfiguration` entry was bad.

```
110401 Unknown property object %s referenced from global entry
110402 Unknown property object %s referenced from global entry
```

The property being referenced is incorrectly specified. This could mean either he property entry is incorrectly specified or it is not being included correctly. Run iDAR in debug mode and make sure all the properties are added. Make sure the spelling of the property name is correct at the point of inclusion.

```
110403 Unknown property type %s referenced from %s
110404 Unknown property type %s referenced from %s
```

Only allowed properties from `ids-proxy-sch-GlobalConfiguration` object class is `ids-proxy-sch-LogProperty`.

```
130108 ids-proxy-con-ava has attribute name specified twice in %s in %s
```

An attribute name was specified twice.

```
150201 SetServiceStatus call failed
```

Windows NT service manager configuration problem.

```
150304 Failed to Reread Configuration... Aborted
```

iDAR tried to reread its config files which failed.

```
150403 iDAR process %d has exited, attempting restart in 10 seconds
```

The monitor has detected that iDAR has exited abnormally and will attempt to restart it.

```
170802 Could not decode sasl cred
```

iDAR could not decode the credentials received from the client as part of a SASL bind.

```
170910 Operation received while SASL bind in progress
```

No operations are allowed when a SASL bind is in progress. The client sent a request other than “bind” while SASL bind was in progress.

`170911 Duplicate message id received: %d`

The client sent two simultaneous requests with the same message id.

`170912 Operation received between SSL negotiation`

No operations are allowed when the START TLS is in progress. The client initiated START TLS but send other operations before completing it.

`171012 Could not establish SSL for referral server`

SSL could not be established with a LDAP server to which iDAR was referred.

`171027 OnBindSuccess rule failed... closing connection`

An error occurred while evaluating `OnBindSuccess` rule. Note: This error message does not mean `OnBindSuccess` rule evaluated to `FALSE`.

`190201 Wrong number of elements in NOT filter`

Bad filter.

`190402 First tag was not a SEQUENCE: source(%s, %d)`
`190403 First element was not a MessageID: source(%s, %d)`

A incorrectly formatted PDU was received from the client or the server.

`210302 ber_flush failed: Destination(%s, %d)`
`210303 ber_flush failed: Destination(%s, %d)`

iDAR failed to send data to the server or the client because either the connection was reset or other network problems.

`240101 Failed to tailor entry %s for attribute %s a bad attribute was found in a Group entry.`

`240102 Either (or both) LoadBalanceProperty or ids-proxy-con-server are present in %s`

One of the above are allowed.

`240201 A Property failed check from group %s`

One of the properties included in this group was inconsistent with the Group.

`240701 Could not find property %s referenced from %s`

There can be several reasons for this. You must reference a property by its name, i.e. the value of the attribute `ids-proxy-con-Name` in the property entry you are referencing. Check if the spelling is correct. Next thing to check is if the referred property is one level under the value of `ids-proxy-sch-PropertyBase` attribute in the `ids-proxy-sch-LDAPProxy` object entry. Then make sure if the referred property belongs to `ids-proxy-sch-belongs-to` the correct server instance and is enabled (`ids-proxy-sch-enable`).

`240703 Wrong property type for ids-proxy-con-server in %s`

The property type can only be `ids-proxy-sch-LDAPServer`.

`280205 Could not find property %s referenced from %s`

See 240701.

`280302 Check failed of one of the server properties`

One of the properties included in this property was inconsistent with the Property.

`300301 Connection from %s has IP options present`
`300302 Could not clear IP options from %s`
`300402 Failed for %s,can’t verify hostname`

iDAR could not resolve a hostname.

```
300403 Host name/name mismatch %s
300404 Host name/address mismatch %s
```

A reverse lookup of the host yielded a different host name for the client.

```
301101 Failed
```

iDAR failed to send data to a client/server.

```
330207 Failed
330211 Failed
```

Call to listen for failed.

```
340101 Failed to open connection, #times tried %d
```

Could not open connection to configuration LDAP server.

```
340901 No file name specified
```

No configuration file name was found in startup configuration file.

```
340902 No Server Entry DN Specified
```

The DN for the server's entry (`object class ids-proxy-sch-LDAPProxy`) was missing from the startup configuration file.

```
340912 No proper type for Server Entry %s
```

The server entry must have an objectclass attribute with value `ids-proxy-sch-LDAPProxy`.

```
380101 Failed
380103 Failed
```

A thread could not be created.

```
380201 Failed
380202 Failed
```

A mutex lock could not be acquired.

`510103 No group %s as referenced in %s found`

See explanation of 240701. Similar logic applies to Groups.

`520112 Contradiction between ids-proxy-con-use-version`

The LDAP protocol version that iDAR is configured to use with a backend LDAP server must be supported by that backend server.

Index

I

K

L

M

N

O

P

R

S

T

U

V