CHAPTER 3

Client and RDU Communication

This chapter details the communication between the client library and the RDU and describes on how to establish, maintain, and close the connection between the client library and the RDU.

This chapter contains the following sections:

Overview

The Cisco BAC API communicates with the RDU in a Cisco BAC deployment over TCP/IP.

The API client library initiates the connection between the API and the RDU. The RDU does not try to establish a connection between itself and the API.

Though the client library initiates and establishes connectivity between the API and the RDU, information flows in both directions, with the client library submitting requests to the RDU, and the RDU responding to those requests. The bilateral heartbeat messages enable the API client and the RDU to maintain a bidirectional connection.

The network administrator must ensure that:

- IP connectivity exists between the client and the RDU.
- The TCP port that the RDU listens on is opened through a firewall between the client API and the RDU. The default TCP port that the RDU listens on is 49187. The RDU uses this TCP port to bind itself to all network interfaces.

Establishing a Connection

The client establishes a connection with the RDU by passing the following parameters:

- Hostname of the RDU; for example, rdu.mso.com
- Port for communication with the RDU; the default port is 49187.
- Administrator username; the default administrator username is **bacadmin**.
- Administrator password; the default administrator password is **changeme**.

You can use the following code to establish a connection between the RDU and the client library:

```
final PACEConnection connection =
   PACEConnectionFactory.newInstance(
         "rdu.mso.com", 49187, "bacadmin", "changeme");
```

The connection between the client library and RDU is maintained until it is explicitly closed. See Closing a Connection, page 3-2 on how to close a connection.

Maintaining a Connection

The client library automatically maintains the connection between the client and RDU. In case the connection breaks in the network layer because of congestion, routing problems, or other issues, the client library automatically reconnects to the RDU. The client library tries to reconnect to the RDU until the connectivity is restored.

The reconnection process is automatic and does not impact your code while the RDU interacts with the library. For example, a synchronous call to submit a batch blocks the thread and returns the results when the results are available as usual; even if the client library had to automatically reconnect to the RDU.

Connection Concurrency

The client library maintains a single TCP connection to the RDU. This connection can be used for any number of requests and responses. Multiple threads can use the same single connection object.

While there is only a single underlying TCP connection, many Provisioning API Command Engine (PACE) connection instances can be created. If there is a need for multiple BAC users in a single client, then multiple PACE connections are required.

Closing a Connection

The connection between the RDU and the client library is maintained until you explicitly close the connection. You can use the following code to close the connection:

```
connection.releaseConnection();
```