Client and RDU Communication

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

Overview

The Prime Cable Provisioning API communicates with the RDU in a Prime Cable Provisioning deployment over TCP/IP. The TCP/IP based interactions are secured using the Secured Socket Layer (SSL) protocol. For client and RDU communication, SSL secures the following interactions:

- Clients using the Prime Cable Provisioning API to interact with the RDU.
- Client interaction with the PWS interface.

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.

When the RDU Java client library initiates and establishes connectivity between API and RDU, the information flows in both the directions; with the RDU Java 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.

Note 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 is 49187 for nonsecured communication with RDU and 49188 for secured communication with RDU. The RDU uses these TCP ports to bind itself to all network interfaces.
- For secure communication, the SSL function must be established between the Prime Cable provisioning components. The SSL secures all inbound communication within Prime Cable Provisioning components through the use of Secure Socket Layer (SSL) 3.0 or and TLS 1.0 protocols. The SSL facilitates encryption of default and custom properties and supports both signed and unsigned certificates. For more information on SSL, see the *Cisco Prime Cable Provisioning 6.0 User Guide*.

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
- Selected mode of communication with the RDU; non-secured or secured. For secured communication, you must configure the SSL function. For information on how to configure SSL, see the *Cisco Prime Cable Provisioning 6.0 User Guide*.
- Port for communication with the RDU; the default non-secured port is 49187 and secured port is 49188.
- Administrator username; the default administrator username is **admin**.
- Administrator password; the default administrator password is **changeme**.

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

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

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

Maintaining a Connection

The RDU Java 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 RDU Java client library automatically reconnects to the RDU. The RDU Java 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 RDU Java client library had to automatically reconnect to the RDU.

Connection Concurrency

The RDU Java 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 Prime Cable Provisioning users in a single client, then multiple PACE connections are required.

Closing a Connection

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

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