CHAPTER 18

RDU and DPE Connection Management

This chapter deals with RDU TCP Connection Management, RDU Batch Concurrency and RDU DPE synchronization. This chapter includes the following sections:

TCP Connection Management

Communication between the RDU and its clients, such as DPEs, API, occurs over TCP. Each client creates a connection to the RDU that is maintained as long as the client desires.

Over this connection, clients submit batches to the RDU while the RDU replies with the batch results and additionally sends events. In order to maintain the connection, a heart beat is used when no other traffic is available. Timeouts are used to prevent the RDU and the clients from creating unwanted traffic.

Heart Beat

Heart beat messages are initiated by the clients. The client sends a heart beat when a read timeout has occurred and it is in the process of receiving a message. On receiving this heart beat, the RDU echoes it. The client receives this echo and recognizes that the connection is alive. This cycle repeats itself until the connection is lost.

If the client fails to receive a reply after sending the heart beat (echo message), it closes its connection and attempts to open a new one.

The RDU will close the connection if no traffic is sent or received for two consecutive read timeouts. To change the number of read timeouts before closing the connection, you must set the following property in the RDU property file and restart the application. You can configure this property in the BPR_HOME/rdu/rdu.properties file.

/messaging/connection/numReadTimeouts=<value>

RDU Batch Concurrency

In the Cisco BAC system, a provisioning client submits API requests to the RDU in the form of batches containing single or multiple commands. A command represents an operation that can be performed on an object in the RDU's database, for example, changing a device's Class of Service. Depending on the commands contained in the batch, the RDU executes the batch in one of two modes: concurrent or non-concurrent.

The commands contained within a batch determine the mode it is to be executed with. When a batch is received at the RDU, its commands are checked and the batch is marked with the mode with which it must be executed with. During the execution of a concurrent-mode batch, other concurrent-mode batches are allowed to be executed in parallel. The majority of batches executed are run in the concurrent mode.

If the batch is marked as non-concurrent, it is executed only when all the currently executing batches are completed in the RDU. The non-concurrent batch is run alone. After the batch completes running, the RDU processes the queued batches in the mode that they have been marked.

The reason concurrent and non-concurrent modes exist is to provide higher throughput at the RDU without losing data integrity. There are only a few commands that cause a batch to run in non-concurrent mode. They are mainly concerned with system configuration operations that are not called very often.

Non-concurrent Commands

Along with Server Registration, if any of the following commands are present in a batch, it is executed in non-concurrent mode.

Configuration Interface

- addLicenseKey
- addUser
- addCustomPropertyDefinition
- addClassOfService
- addFile
- changeClassOfServiceProperties
- changeDPEDefaults
- changeExtensionPointSettings
- changeRDUDefaults
- changeSystemDefaults
- changeFileProperties
- changeProvGroupProperties
- changeUser
- deleteClassOfService
- deleteFile
- deleteLicenseKey
- removeCustomPropertyDefinition
- replaceFile

IPDevice Interface

- addDeviceType
- changeDefaults
- deleteDeviceType
- regenConfigs
- IPDevice.addNode
- IPDevice.changeNodeProperties
- IPDevice.deleteNode
- IPDevice.addNodeType
- IPDevice.changeNodeTypePriority
- IPDevice.changeNodeTypeProperties
- IPDevice.deleteNodeType

Long-Running Batch

A long-running batch is one that requires more than two seconds to execute. The only batch that fits into this category is the synchronization batch submitted by the DPEs. Additional batches can become long running batches if custom extension points are used to interface with external systems (e.g. publishing to an external database).

Long-running batches, interleaved with non-concurrent batches, can create an impression that the RDU is not responding or is frozen. If a long-running batch is in progress and the next batch in the queue is a non-concurrent batch, the RDU waits for all current batches to finish before it starts to execute the non-concurrent batch. It also will not start any other batches. At those times, the RDU will seem to not respond. However, as soon as the long-running batch is completed, the RDU resumes executing the queued batches. At this point, the RDU will respond again.

DPE-RDU Synchronization

Cisco BAC supports multiple DPEs. The DPEs communicate with devices and the RDU. During installation, you must configure the following for each DPE:

- Name of the provisioning group to which this DPE belongs; this name determines the logical group of devices serviced by this DPE.
- The IP address and port number of the RDU.

The RDU generates the instructions for the device and sends the new instructions to all DPEs that service the provisioning group of the given device. The RDU will regenerate the instructions for a device when certain provisioning API calls are made at the RDU (such as changing the Class of Service of the device). The DPE stores the instructions and uses it to service subsequent requests.

The DPE persists all device instructions that it receives from the RDU to disk. Each instruction includes an identifier (device identifier or file name) and a revision number which is incremented every time the instruction is regenerated. In addition to events (dynamic notifications) fired by the RDU to all DPEs when a given configuration changes, there is also an automatic synchronization process which is used to bring DPE up to date with the RDU.

The DPE-RDU synchronization is a process of automatically updating the DPE cache to be consistent with the RDU. The DPE cache comprises the instruction cache, with instructions for devices, and the file cache, with files required for devices.

Under normal conditions, the RDU generates the events with instruction updates and sends them to all relevant DPEs to keep them up to date. Synchronization is needed if the DPE is missing some events due to connection loss. Such loss could be due to a network issue, the DPE server going down for administrative purposes, or a failure. Synchronization also covers the special case when the RDU database is restored from backup. In this case, the DPE cache database must be returned to an older state to be consistent with the RDU.

Note The RDU and DPE synchronization process is automatic and requires no administrative intervention. Throughout the synchronization process, the DPE is still fully capable of performing provisioning and management operations on the CPE.

The DPE triggers the synchronization process every time it establishes a connection with the RDU.

When the DPE first starts up, it establishes the connection to the RDU and registers with the RDU to receive updates of instruction changes. The DPE and RDU then monitor the connection by using heartbeat message exchanges.

When the DPE determines that it had lost its connection to the RDU, it automatically attempts to re-establish it. It continues attempting to do this with a backoff-retry interval until it is successful. The RDU also detects the lost connection and stops sending events to this DPE. Since the DPE may miss the update events from the RDU when the connection is down, the DPE performs synchronization every time it establishes a connection with the RDU.

During the process of connection establishment and registration with the RDU, the DPE is in the *Registering* state.

Note To manually trigger the synchronization process for testing purpose, you can use the DPE CLI command *dpe reload* or break the connection by disconnecting and reconnecting the Ethernet cable.

The DPE requests a list of all the instructions it should have from the RDU. This list contains the identifiers for instructions and revision numbers, but not the actual instruction content. By using this list, the DPE determines which instructions in its store are inconsistent (wrong revision number), which ones are missing, and which ones to delete. Throughout the process of obtaining the synchronization list and comparing it with its store, the DPE is in the *Synchronizing* state.

As soon as the DPE finishes determining what to obtain from the RDU, it starts obtaining the instructions from the RDU. The DPE only obtains missing or out-of-date instructions. During this process, the DPE is in the *Populating* state.

The DPE populates at a fixed rate to ensure that the RDU is not overloaded with its requests. If multiple DPEs in the provisioning group are populating, the population time may be decreased as the requested instructions are sent to all DPEs in the provisioning group. After the DPE finishes populating, it is in the *Ready* state and fully synchronized with the RDU.

You can view the DPE state from the administrator user interface (see Viewing Device Provisioning Engines, page 16-23) or from the DPE CLI (by using the **show dpe** command).