**Preparedness**

- *Remove or downgrade version 2 text indexes* (page 874) before downgrading MongoDB 2.6 to 2.4.
- *Remove or downgrade version 2 2dsphere indexes* (page 875) before downgrading MongoDB 2.6 to 2.4.
- *Downgrade 2.6 User Authorization Model* (page 872). If you have upgraded to the 2.6 user authorization model, you must downgrade the user model to 2.4 before downgrading MongoDB 2.6 to 2.4.

**Procedures** Follow the downgrade procedures:

- To downgrade sharded clusters, see *Downgrade a 2.6 Sharded Cluster* (page 876).
- To downgrade replica sets, see *Downgrade a 2.6 Replica Set* (page 875).
- To downgrade a standalone MongoDB instance, see *Downgrade 2.6 Standalone mongod Instance* (page 875).

**Downgrade 2.6 User Authorization Model** If you have upgraded to the 2.6 user authorization model, you **must first** downgrade the user authorization model to 2.4 **before** before downgrading MongoDB 2.6 to 2.4.

**Considerations**

- For a replica set, it is only necessary to run the downgrade process on the *primary* as the changes will automatically replicate to the secondaries.
- For sharded clusters, although the procedure lists the downgrade of the cluster's authorization data first, you may downgrade the authorization data of the cluster or shards first.
- You *must* have the `admin.system.backup_users` and `admin.system.new_users` collections created during the upgrade process.
- **Important**. The downgrade process returns the user data to its state prior to upgrading to 2.6 authorization model. Any changes made to the user/role data using the 2.6 users model will be lost.

**Access Control Prerequisites** To downgrade the authorization model, you must connect as a user with the following *privileges*:

```
{ resource: { db: "admin", collection: "system.new_users" }, actions: [ "find", "insert", "update" ]
{ resource: { db: "admin", collection: "system.backup_users" }, actions: [  "find" ] }
{ resource: { db: "admin", collection: "system.users" }, actions: [ "find", "remove", "insert"] }
{ resource: { db: "admin", collection: "system.version" }, actions: [ "find", "update" ] }
```

If no user exists with the appropriate *privileges*, create an authorization model downgrade user:

**Step 1: Connect as user with privileges to manage users and roles.** Connect and authenticate as a user with `userAdminAnyDatabase` (page 412).

**Step 2: Create a role with required privileges.** Using the `db.createRole` method, create a *role* (page 325) with the required privileges.

```
use admin
db.createRole(
  {
    role: "downgradeAuthRole",
    privileges: [
      { resource: { db: "admin", collection: "system.new_users" }, actions: [ "find", "insert", "upda
```