Upgrading

Upgrading an engine is as simple as restarting it with the new version. Unless otherwise specified, active tasks can be allowed to run during the restart. A new engine is compatible with older blades; restarting blades is only required to take advantage of new blade functionality.

Some sites may wish to back up their job data before upgrading. The benefits of doing this are twofold: a backup guards against losing historical job data should an unforeseen problem happen during upgrade (e.g. software or hardware failure); also, the system can be restored to its pre-upgrade state should a downgrade be required. Backups and downgrades are discussed below.

Down Time

The time to perform an upgrade is affected primarily by operations required to modify the database schema. Documentation for a specific upgrade will advise when there may be significant time (e.g. over one minute) to perform the upgrade. During the upgrade, active tasks can continue to run, however no new work will be scheduled, no new jobs can be submitted, and Dashboard, command-line, and API communication with the engine will block.

Backing Up

Before backing up a job database, ensure that the tractor engine and the postgresql database have stopped. The engine will automatically shut down the database server unless a site has set DBShutdown to False in its site db.config file, or if the engine has difficulty communicating with the database server. t ractor-dbctl --status can be used to verify that the database server has stopped. If it is still running, it can be stopped with tractor-dbctl --stop.

To back up the job database, run tractor-dbctl --backup <path-to-backup-file>. Ensure there is enough space to store the backup file. The backup file is a compressed tarball of the engine data directory's psql/ subdirectory, so it will only need a fraction of the space used by that directory.

Downgrading

When downgrading, it may not be possible to use the existing database if it has undergone a schema change. In this case, it will be necessary to start with a new job database or to restore from a backup. Please contact Customer Support if it is necessary to preserve job submissions and state changes that occured after the upgrade.

To perform a downgrade, stop the existing engine and database server as described above in *Backing Up*. Then destroy the job database with tractor-dbctl --destroy.

To restore the job database from a backup, use tractor-dbctl --restore <path-to-backup-file> and start with the applicable version of the engine; job IDs will resume from the last used job ID when the backup was created. Jobs that were queued when the backup was taken will be rescheduled when the system is restored to the backup. Jobs submitted after the backup will not appear in the restored job database.

If a the job database is not restored from a backup, it will simply be an empty job database with job IDs starting at 1.

Note that if a job ID is reused, its log file output will be appened to the log files of the former job of the same ID. Thus, it is advised to remove log files associated with the job IDs that will be reused.

The job ID sequence can resume from a different value in order to avoid reusing job IDs. First start the database server with tractor-dbctl --start-for-engine and then run tractor-dbctl --set-job-counter <#>.

To summarize, the steps to downgrade are as follows:

> 1. *Pause dispatching in the Dashboard admin panel.*
> 2. *Retry all active tasks: tq --force --yes retry active*
> 3. *Kill the engine.*
> 4. *Destroy the job database: tractor-dbctl --destroy*
> 5. *Optionally restore the backed up database: tractor-dbctl --restore <path-to-backup-file>*
> 6. *Start the database server: tractor-dbctl --start-for-engine*
> 7. *Note the last used job ID with tractor-dbctl --show-params and remove any log files from jobs with a higher job ID.*
> 8. *Optionally set the job counter: tractor-dbctl --set-job-counter <#>*
> 9. *Start the applicable engine.*

Note that all of the tractor-dbctl commands described here will require the site configuration directory to be specified with the --configdir option.

Multiple Engines

It is possible to simultaneoulsy run different Tractor engines with different versions, such as to test a new version before deployment into production. Multiple engines may run on the same host; however, they must have distinct ports, configuration directories, data directories, and database directories. Even if engines are not running on the same host, if they use the same network-sharable volumes for any of the above directories, corruption is possible. Configuration settings of importance here are TractorDataDirectory in tractor.configand DBDataDir in db.config. You will also need to manage distinct locations for log files and for pointing blades to the appropriate engine.

Additional Information

Please refer to the following documents for version-specific upgrade information.

Upgrading from 1.x

Upgrading to 2.1

Upgrading to 2.2