MDWF Database Management Tool
Complete Command Reference

Contents

1 Lattice QCD Parameters

The MDWF database tracks Domain Wall Fermion lattice QCD ensembles with the following physics parameters:

1.1 Required Parameters

- **beta**: Gauge coupling parameter (e.g., 6.0)
- **b**: Domain wall height parameter (e.g., 1.8)
- **Ls**: Domain wall extent in 5th dimension (e.g., 24)
- **mc**: Charm quark mass (e.g., 0.8555)
- **ms**: Strange quark mass (e.g., 0.0725)
- **ml**: Light quark mass (e.g., 0.02)
- **L**: Spatial lattice size (e.g., 32)
- **T**: Temporal lattice size (e.g., 64)

1.2 Directory Structure

Ensembles are organized in a hierarchical directory structure:

```
TUNING/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64/
ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.0195/L32/T64/
```

Listing 1: Directory Structure

Each ensemble directory contains:

- `cnfg/`: Configuration files
- `jlog/`: Job logs
- `log_hmc/`: HMC logs
- `slurm/`: SLURM scripts

2 Database Management Commands

2.1 init-db: Initialize Database

Initialize a new MDWF database and directory structure.

2.1.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `--base-dir BASE_DIR`: Root directory for TUNING/ and ENSEMBLES/ (default: current)

2.1.2 Example Usage

```
$ mdwf_db init-db
Ensured directory: /Users/wyatt/Development/mdwf_db
Ensured directory: /Users/wyatt/Development/mdwf_db/TUNING
Ensured directory: /Users/wyatt/Development/mdwf_db/ENSEMBLES
init_database returned: True

$ mdwf_db init-db --base-dir /scratch/lattice
Ensured directory: /scratch/lattice
Ensured directory: /scratch/lattice/TUNING
Ensured directory: /scratch/lattice/ENSEMBLES
init_database returned: True
```

Listing 2: Initialize Database

2.2 add-ensemble: Add New Ensemble

Add a new ensemble to the database with physics parameters.

2.2.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-p PARAMS, --params PARAMS`: Space-separated key=val pairs (required)
- `-s \{TUNING,PRODUCTION\}, --status`: Ensemble status (required)
- `-d DIRECTORY, --directory`: Explicit directory path (optional)
- `-b BASE_DIR, --base-dir`: Root directory (default: current)
- `--description DESCRIPTION`: Free-form description (optional)

2.2.2 Example Usage

```
# Basic TUNING ensemble
$ mdwf_db add-ensemble \
  -p "beta=6.0 b=1.8 Ls=24 mc=0.8555 ms=0.0725 ml=0.02 L=32 T=64" \
  -s TUNING
Ensemble added: ID=1

# PRODUCTION ensemble with description
$ mdwf_db add-ensemble \
  -p "beta=6.0 b=1.8 Ls=24 mc=0.8555 ms=0.0725 ml=0.0195 L=32 T=64" \
  -s PRODUCTION \
  --description "Production run with lighter quark masses"
Ensemble added: ID=2
Marked PRODUCTION in DB: OK

# Custom directory path
$ mdwf_db add-ensemble \
  -p "beta=6.0 b=1.8 Ls=24 mc=0.8555 ms=0.0725 ml=0.01 L=32 T=64" \
  -s TUNING \
  -d ./custom/path/to/ensemble
Ensemble added: ID=3

# With custom base directory
$ mdwf_db add-ensemble \
  -p "beta=6.0 b=1.8 Ls=24 mc=0.8555 ms=0.0725 ml=0.02 L=32 T=64" \
  -s TUNING \
  -b /scratch/lattice
Ensemble added: ID=4
```

Listing 3: Add Ensemble Examples

2.3 query: Query Ensemble Information

List ensembles or show detailed information for a specific ensemble.

2.3.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE, --ensemble`: Ensemble ID, path, or "." for current directory
- `--detailed`: Show physics parameters and operation counts in list mode

2.3.2 Example Usage

```
# List all ensembles (basic)
$ mdwf_db query
[1] (PRODUCTION) /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
[2] (PRODUCTION) /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.0195/L32/T64

# List all ensembles with details
$ mdwf_db query --detailed
[1] (PRODUCTION) /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
    Parameters: L=32, Ls=24, T=64, b=1.8, beta=6.0, mc=0.8555, ml=0.02, ms=0.0725
    Operations: 2
    Description: Example ensemble for documentation

[2] (PRODUCTION) /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.0195/L32/T64
    Parameters: L=32, Ls=24, T=64, b=1.8, beta=6.0, mc=0.8555, ml=0.0195, ms=0.0725
    Operations: 0

# Show detailed info for specific ensemble
$ mdwf_db query -e 1
ID          = 1
Directory   = /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
Status      = PRODUCTION
Created     = 2025-06-26T13:37:33.638950
Description = Example ensemble for documentation
Parameters:
    L = 32
    Ls = 24
    T = 64
    b = 1.8
    beta = 6.0
    mc = 0.8555
    ml = 0.02
    ms = 0.0725

=== Operation history ===
Op 1: HMC_TUNE [RUNNING]
  Created: 2025-06-26T13:38:04.948828
  Updated: 2025-06-26T13:38:04.948828
    config_end = 50
    config_start = 0
    slurm_job = 12345

Op 2: PROMOTE_ENSEMBLE [COMPLETED]
  Created: 2025-06-26T13:38:32.123456
  Updated: 2025-06-26T13:38:32.123456

# Query by directory path
$ mdwf_db query -e ./ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
```

```
# Query current directory (when inside ensemble)
$ cd ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
$ mdwf_db query -e .
```

Listing 4: Query Examples

2.4 promote-ensemble: Promote to Production

Move a TUNING ensemble to PRODUCTION status and directory.

2.4.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE, --ensemble`: Ensemble ID, path, or "." (required)
- `--base-dir BASE_DIR`: Root directory (default: current)
- `--force`: Skip confirmation prompt

2.4.2 Example Usage

```
# Promote with confirmation
$ mdwf_db promote-ensemble -e 1
Promote ensemble 1:
  from /Users/wyatt/Development/mdwf_db/TUNING/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
    to /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
Continue? [y/N]: y
Created operation 2: Created
Promotion OK

# Promote without confirmation
$ mdwf_db promote-ensemble -e 1 --force
Promote ensemble 1:
  from /Users/wyatt/Development/mdwf_db/TUNING/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
    to /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
Created operation 2: Created
Promotion OK

# Promote by directory path
$ mdwf_db promote-ensemble -e ./TUNING/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64

# Promote current directory
$ cd TUNING/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
$ mdwf_db promote-ensemble -e . --force
```

Listing 5: Promote Ensemble Examples

2.5 clear-history: Clear Operation History

Clear all operation history for an ensemble while preserving the ensemble record.

2.5.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE, --ensemble`: Ensemble ID, path, or "." (required)
- `--force`: Skip confirmation prompt

2.5.2 Example Usage

```
# Clear history with confirmation
$ mdwf_db clear-history -e 1
Clear all operation history for ensemble 1?
This will remove all operations but preserve the ensemble record.
Continue? [y/N]: y
Cleared 2 operations for ensemble 1

# Clear history without confirmation
$ mdwf_db clear-history -e 1 --force
Cleared 2 operations for ensemble 1

# Clear by directory path
$ mdwf_db clear-history -e ./ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64

# Clear current directory
$ cd ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
$ mdwf_db clear-history -e . --force
```

Listing 6: Clear History Examples

2.6 remove-ensemble: Remove Ensemble

Remove an ensemble and all its operations from the database.

2.6.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE, --ensemble`: Ensemble ID, path, or "." (required)
- `--force`: Skip confirmation prompt
- `--remove-directory`: Also delete the on-disk directory tree

2.6.2 Example Usage

```
# Remove from database only (preserve files)
$ mdwf_db remove-ensemble -e 1
Remove ensemble 1 from database?
This will delete all ensemble and operation records.
Continue? [y/N]: y
Removed ensemble 1 from database

# Remove database record and delete files
$ mdwf_db remove-ensemble -e 1 --remove-directory --force
Removed ensemble 1 from database
Deleted directory: /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0
    .0725/ml0.02/L32/T64

# Remove by directory path
$ mdwf_db remove-ensemble -e ./ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64

# Remove current ensemble
$ cd ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
$ mdwf_db remove-ensemble -e . --force
```

Listing 7: Remove Ensemble Examples

3 Job Script Generation Commands

3.1 glu-input: Generate GLU Input Files

Generate input files for the GLU gauge field utility program.

3.1.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE_ID, --ensemble-id`: ID of the ensemble (required)
- `-o OUTPUT_FILE, --output-file`: Path for GLU input file (required)
- `-g GLU_PARAMS, --glu-params`: Space-separated key=val pairs (optional)
- `-t \{smearing,gluon_props,other\}, --type`: Calculation type (default: smearing)

3.1.2 GLU Parameters

GLU parameters use flat names (no dots) and include:

- `CONFNO`: Configuration number (default: 24)
- `DIM_0, DIM_1, DIM_2`: Spatial dimensions (auto-set from ensemble)
- `DIM_3`: Temporal dimension (auto-set from ensemble)
- `SMEARTYPE`: Smearing algorithm (default: STOUT)
- `SMITERS`: Number of smearing iterations (default: 8)
- `ALPHA1`: Primary smearing parameter (default: 0.75)
- `ALPHA2`: Secondary smearing parameter (default: 0.4)
- `ALPHA3`: Tertiary smearing parameter (default: 0.2)
- `GFTYPE`: Gauge fixing type (default: COULOMB)
- `GF_TUNE`: Gauge fixing tuning (default: 0.09)
- `ACCURACY`: Gauge fixing accuracy (default: 14)
- `MAX_ITERS`: Maximum iterations (default: 650)

3.1.3 Example Usage

```
# Basic smearing input with defaults
$ mdwf_db glu-input -e 1 -o smear.in
Generated GLU input file: smear.in

# View generated file
$ cat smear.in
MODE = SMEARING
HEADER = NERSC
    DIM_0 = 32
    DIM_1 = 32
    DIM_2 = 32
    DIM_3 = 64
CONFNO = 24
RANDOM_TRANSFORM = NO
SEED = 0
GFTYPE = COULOMB
```

```
    GF_TUNE = 0.09
    ACCURACY = 14
    MAX_ITERS = 650
CUTTYPE = GLUON_PROPS
FIELD_DEFINITION = LINEAR
    MOM_CUT = CYLINDER_CUT
    MAX_T = 7
    MAXMOM = 4
    CYL_WIDTH = 2.0
    ANGLE = 60
    OUTPUT = ./
SMEARTYPE = STOUT
    DIRECTION = ALL
    SMITERS = 8
    ALPHA1 = 0.75
    ALPHA2 = 0.4
    ALPHA3 = 0.2
U1_MEAS = U1_RECTANGLE
    U1_ALPHA = 0.07957753876221914
    U1_CHARGE = -1.0
CONFIG_INFO = 2+1DWF_b2.25_TEST
    STORAGE = CERN
BETA = 6.0
    ITERS = 1500
    MEASURE = 1
    OVER_ITERS = 4
    SAVE = 25
    THERM = 100

# Custom smearing parameters
$ mdwf_db glu-input -e 1 -o custom_smear.in \
  -g "CONFNO=168 SMITERS=50 ALPHA1=0.1"
Generated GLU input file: custom_smear.in

# Gauge fixing input
$ mdwf_db glu-input -e 1 -o gauge_fix.in -t other \
  -g "CONFNO=100 GFTYPE=LANDAU ACCURACY=16"
Generated GLU input file: gauge_fix.in

# Gluon properties calculation
$ mdwf_db glu-input -e 1 -o gluon_props.in -t gluon_props \
  -g "CONFNO=200 MAXMOM=6 MAX_T=10"
Generated GLU input file: gluon_props.in
```

Listing 8: GLU Input Examples

3.2 smear-script: Generate Smearing Scripts

Generate complete SLURM scripts for configuration smearing using GLU.

3.2.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE_ID, --ensemble-id`: ID of the ensemble (required)
- `-j JOB_PARAMS, --job-params`: SLURM job parameters (required)
- `-g GLU_PARAMS, --glu-params`: GLU smearing parameters (optional)
- `-o OUTPUT_FILE, --output-file`: Output script path (auto-generated if not specified)

3.2.2 Job Parameters

Required job parameters:

- `mail_user`: Email address for job notifications
- `config_start`: First configuration number to smear
- `config_end`: Last configuration number to smear

Optional job parameters with defaults:

- `account`: SLURM account (default: m2986_g)
- `constraint`: Node constraint (default: gpu)
- `queue`: SLURM partition (default: regular)
- `time_limit`: Job time limit (default: 06:00:00)
- `nodes`: Number of nodes (default: 1)
- `cpus_per_task`: CPUs per task (default: 16)
- `gpus`: GPUs per node (default: 4)
- `gpu_bind`: GPU binding (default: none)
- `ranks`: MPI ranks (default: 4)
- `bind_sh`: CPU binding script (default: bind.sh)

3.2.3 Example Usage

```
# Basic smearing job
$ mdwf_db smear-script -e 1 \
  -j "mail_user=user@example.com config_start=100 config_end=200"
Generated smearing script: glu_smear_STOUT8_100_200.sh

# Custom smearing and job parameters
$ mdwf_db smear-script -e 1 \
  -j "mail_user=user@example.com config_start=100 config_end=200 time_limit=12:00:00 nodes=2
    " \
  -g "SMITERS=10 ALPHA1=0.8 SMEARTYPE=APE"
Generated smearing script: glu_smear_APE10_100_200.sh

# Specify output file
$ mdwf_db smear-script -e 1 -o custom_smear.sh \
  -j "mail_user=user@example.com config_start=100 config_end=200"
Generated smearing script: custom_smear.sh

# High-precision smearing
$ mdwf_db smear-script -e 1 \
  -j "mail_user=user@example.com config_start=50 config_end=100 time_limit=24:00:00" \
  -g "SMITERS=20 ALPHA1=0.1 ALPHA2=0.05 ACCURACY=16"
Generated smearing script: glu_smear_STOUT20_50_100.sh

# Custom account and queue
$ mdwf_db smear-script -e 1 \
  -j "mail_user=user@example.com config_start=1 config_end=50 account=lattice_qcd queue=
    debug time_limit=02:00:00"
Generated smearing script: glu_smear_STOUT8_1_50.sh
```

Listing 9: Smear Script Examples

3.3 hmc-script: Generate HMC Scripts

Generate HMC XML parameters and SLURM batch scripts for gauge configuration generation.

3.3.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE_ID, --ensemble-id`: ID of the ensemble (required)
- `-a ACCOUNT, --account`: SLURM account name (required)
- `-m \{tepid,continue,reseed\}, --mode`: HMC run mode (required)
- `--base-dir BASE_DIR`: Root directory (default: current)
- `-x XML_PARAMS, --xml-params`: HMC XML parameters (optional)
- `-j JOB_PARAMS, --job-params`: SLURM job parameters (optional)
- `-o OUTPUT_FILE, --output-file`: Output script path (auto-generated if not specified)

3.3.2 HMC Modes

- `tepid`: Initial thermalization run (TepidStart)
- `continue`: Continue from existing checkpoint (CheckpointStart)
- `reseed`: Start new run with different seed (CheckpointStartReseed)

3.3.3 Job Parameters

Required job parameter:

- `cfg_max`: Maximum configuration number to generate

Optional job parameters with defaults:

- `constraint`: Node constraint (default: gpu)
- `time_limit`: Job time limit (default: 17:00:00)
- `cpus_per_task`: CPUs per task (default: 32)
- `nodes`: Number of nodes (default: 1)
- `gpus_per_task`: GPUs per task (default: 1)
- `gpu_bind`: GPU binding (default: none)
- `mail_user`: Email notifications (from environment)
- `queue`: SLURM partition (default: regular)
- `exec_path`: Path to HMC executable (auto-detected)
- `bind_script`: CPU binding script (auto-detected)

3.3.4 XML Parameters

Available HMC XML parameters:

- `StartTrajectory`: Starting trajectory number (default: 0)
- `Trajectories`: Number of trajectories to generate (default: 50)
- `MetropolisTest`: Perform Metropolis test (true/false, default: true)
- `NoMetropolisUntil`: Trajectory to start Metropolis (default: 0)
- `PerformRandomShift`: Perform random shift (true/false, default: true)
- `StartingType`: Start type (auto-set by mode)
- `Seed`: Random seed (for reseed mode)
- `MDsteps`: Number of MD steps (default: 2)
- `trajL`: Trajectory length (default: 1.0)

3.3.5 Example Usage

```
# Basic HMC script for new ensemble
$ mdwf_db hmc-script -e 1 -a m2986 -m tepid -j "cfg_max=100"
Generated HMC script: hmc_tepid.sh
Wrote HMCparameters.xml to /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0
    .8555/ms0.0725/ml0.02/L32/T64

# Continue existing run
$ mdwf_db hmc-script -e 1 -a m2986 -m continue \
  -j "cfg_max=200 time_limit=24:00:00" \
  -x "StartTrajectory=100 Trajectories=100"
Generated HMC script: hmc_continue.sh

# Custom parameters and output file
$ mdwf_db hmc-script -e 1 -a m2986 -m tepid -o custom_hmc.sh \
  -j "cfg_max=50 nodes=2 time_limit=12:00:00 mail_user=user@example.com" \
  -x "MDsteps=4 trajL=0.75 Seed=12345"
Generated HMC script: custom_hmc.sh

# Reseed mode with custom seed
$ mdwf_db hmc-script -e 1 -a lattice_qcd -m reseed \
  -j "cfg_max=150 constraint=gpu time_limit=20:00:00" \
  -x "Seed=98765 StartTrajectory=100 Trajectories=50"
Generated HMC script: hmc_reseed.sh

# Debug run with short time limit
$ mdwf_db hmc-script -e 1 -a m2986 -m tepid \
  -j "cfg_max=10 time_limit=01:00:00 queue=debug" \
  -x "Trajectories=10 MetropolisTest=false"
Generated HMC script: hmc_tepid.sh
```

Listing 10: HMC Script Examples

3.4 hmc-xml: Generate HMC XML Files

Generate standalone HMC parameters XML files for an ensemble.

3.4.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE_ID, --ensemble-id`: ID of the ensemble (required)
- `-m \{tepid,continue,reseed\}, --mode`: Run mode (required)
- `-b BASE_DIR, --base-dir`: Root directory (default: current)
- `-x XML_PARAMS, --xml-params`: XML parameter overrides (optional)

3.4.2 Example Usage

```
# Basic XML generation
$ mdwf_db hmc-xml -e 1 -m tepid
Wrote HMCparameters.xml to /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0
    .8555/ms0.0725/ml0.02/L32/T64

# View generated XML
$ cat ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64/HMCparameters.xml
<?xml version="1.0" ?>
<grid>
  <HMCparameters>
    <StartTrajectory>0</StartTrajectory>
    <Trajectories>50</Trajectories>
    <MetropolisTest>false</MetropolisTest>
    <NoMetropolisUntil>0</NoMetropolisUntil>
    <PerformRandomShift>false</PerformRandomShift>
    <StartingType>TepidStart</StartingType>
    <Seed>973655</Seed>
    <MD>
      <name>
        <elem>OMF2_5StepV</elem>
        <elem>OMF2_5StepV</elem>
        <elem>OMF4_11StepV</elem>
      </name>
      <MDsteps>2</MDsteps>
      <trajL>1.0</trajL>
    </MD>
  </HMCparameters>
</grid>

# Custom XML parameters
$ mdwf_db hmc-xml -e 1 -m continue \
  -x "StartTrajectory=100 Trajectories=100 MetropolisTest=true"
Wrote HMCparameters.xml to /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0
    .8555/ms0.0725/ml0.02/L32/T64

# Reseed mode with custom seed
$ mdwf_db hmc-xml -e 1 -m reseed \
  -x "Seed=42 StartTrajectory=50 Trajectories=25"
Wrote HMCparameters.xml to /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0
    .8555/ms0.0725/ml0.02/L32/T64

# Custom base directory
$ mdwf_db hmc-xml -e 1 -m tepid -b /scratch/lattice \
  -x "Trajectories=200 MDsteps=4"
Wrote HMCparameters.xml to /scratch/lattice/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0
    .02/L32/T64
```

Listing 11: HMC XML Examples

4 Operation Tracking Commands

4.1 update: Record Operations

Create or update operation records in the database for tracking job execution.

4.1.1 Options

- `--db-file DB_FILE`: Path to SQLite database (auto-discovered)
- `-e ENSEMBLE_ID, --ensemble-id`: ID of the ensemble (required)
- `-o OPERATION_TYPE, --operation-type`: Type of operation (required)
- `-s \{RUNNING,COMPLETED,FAILED\}, --status`: Operation status (required)
- `-i OPERATION_ID, --operation-id`: ID of existing operation to update (optional)
- `-p PARAMS, --params`: Operation parameters (optional)

4.1.2 Operation Types

Common operation types:

- `HMC_TUNE`: HMC tuning run
- `HMC_PRODUCTION`: HMC production run
- `GLU_SMEAR`: Configuration smearing
- `PROMOTE_ENSEMBLE`: Ensemble promotion

4.1.3 Operation Parameters

Common parameters:

- `config_start`: First configuration number
- `config_end`: Last configuration number
- `exit_code`: Job exit code
- `runtime`: Job runtime in seconds
- `slurm_job`: SLURM job ID
- `host`: Execution hostname

4.1.4 Example Usage

```
# Record new HMC operation
$ mdwf_db update -e 1 -o HMC_TUNE -s RUNNING \
  -p "config_start=0 config_end=100 slurm_job=12345"
Created operation 1: Created

# Update operation to completed
$ mdwf_db update -e 1 -o HMC_TUNE -s COMPLETED -i 1 \
  -p "config_start=0 config_end=100 exit_code=0 runtime=3600"
Updated operation 1: Updated

# Record failed operation
$ mdwf_db update -e 1 -o GLU_SMEAR -s FAILED \
  -p "config_start=100 config_end=200 exit_code=1 slurm_job=12346"
Created operation 2: Created
```

```
# Record smearing operation
$ mdwf_db update -e 1 -o GLU_SMEAR -s RUNNING \
  -p "config_start=100 config_end=200 slurm_job=12347 host=gpu-node-01"
Created operation 3: Created

# Update with runtime information
$ mdwf_db update -e 1 -o GLU_SMEAR -s COMPLETED -i 3 \
  -p "config_start=100 config_end=200 exit_code=0 runtime=7200"
Updated operation 3: Updated
```

Listing 12: Update Operation Examples

5 Workflow Examples

5.1 Complete Ensemble Workflow

This section demonstrates a complete workflow from initialization to production.

```
# 1. Initialize database
$ mdwf_db init-db
Ensured directory: /Users/wyatt/Development/mdwf_db
Ensured directory: /Users/wyatt/Development/mdwf_db/TUNING
Ensured directory: /Users/wyatt/Development/mdwf_db/ENSEMBLES
init_database returned: True

# 2. Add tuning ensemble
$ mdwf_db add-ensemble \
  -p "beta=6.0 b=1.8 Ls=24 mc=0.8555 ms=0.0725 ml=0.02 L=32 T=64" \
  -s TUNING \
  --description "Tuning run for new parameters"
Ensemble added: ID=1

# 3. Generate HMC script for thermalization
$ mdwf_db hmc-script -e 1 -a m2986 -m tepid -j "cfg_max=100"
Generated HMC script: hmc_tepid.sh

# 4. Record HMC operation start
$ mdwf_db update -e 1 -o HMC_TUNE -s RUNNING \
  -p "config_start=0 config_end=100 slurm_job=12345"
Created operation 1: Created

# 5. Update operation when completed
$ mdwf_db update -e 1 -o HMC_TUNE -s COMPLETED -i 1 \
  -p "exit_code=0 runtime=14400"
Updated operation 1: Updated

# 6. Generate smearing script
$ mdwf_db smear-script -e 1 \
  -j "mail_user=user@example.com config_start=50 config_end=100"
Generated smearing script: glu_smear_STOUT8_50_100.sh

# 7. Record smearing operation
$ mdwf_db update -e 1 -o GLU_SMEAR -s RUNNING \
  -p "config_start=50 config_end=100 slurm_job=12346"
Created operation 2: Created

# 8. Check ensemble status
$ mdwf_db query -e 1
ID          = 1
Directory   = /Users/wyatt/Development/mdwf_db/TUNING/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0
    .02/L32/T64
Status      = TUNING
Parameters:
    L = 32, Ls = 24, T = 64, b = 1.8, beta = 6.0
    mc = 0.8555, ml = 0.02, ms = 0.0725
```

```
=== Operation history ===
Op 1: HMC_TUNE [COMPLETED]
Op 2: GLU_SMEAR [RUNNING]

# 9. Promote to production
$ mdwf_db promote-ensemble -e 1 --force
Promote ensemble 1:
  from /Users/wyatt/Development/mdwf_db/TUNING/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
    to /Users/wyatt/Development/mdwf_db/ENSEMBLES/b6.0/b1.8Ls24/mc0.8555/ms0.0725/ml0.02/L32/T64
Created operation 3: Created
Promotion OK

# 10. Continue production run
$ mdwf_db hmc-script -e 1 -a m2986 -m continue \
  -j "cfg_max=500" -x "StartTrajectory=100 Trajectories=400"
Generated HMC script: hmc_continue.sh
```

Listing 13: Complete Workflow Example