DataMigratePro — Parameter Quick Reference & Sample Calls

Version: 2026-04 | **Language:** English & Deutsch

---

1. Quick Syntax Overview

Basic Command Structure

```
DataMigratePro [global options] -t <TASK> [task options]
```

Many features work without `-t` (service, installation, queue handling, mapping build).

Interactive Mode (No Arguments)

When started without arguments, DataMigratePro enters **guided interactive mode** with step-by-step prompts:

- **Action selection:** choose from the numbered command list (or type the action name)
- **Prompt workflow:** required parameters are requested one after another
- **Exit:** press `Esc` (or type `exit` in line-based prompts)

German: Das Tool bietet automatische Vorschläge und Parameter-Validierung vor der Ausführung.

---

2. Global Connection & Environment Options

These options apply to all tasks and are persisted in `settings.json`.

Parameter	Type	Meaning	Example
`--registration <GUID>`	Global	License/registration key (requires `--tenantid`)	`--registration "abc123-def456"`
`--tenantid <id>`	Global	Azure AD Tenant ID (SaaS)	`--tenantid "12345678-1234-..."`
`--clientid <id>`	Global	Azure AD App ID (SaaS)	`--clientid "app-id-value"`
`--clientsecret <str>`	Global	Azure AD App Secret (SaaS)	`--clientsecret "secret-value"`
`--sqlconnection <str>`	Global	Primary SQL connection string	`"Data Source=SQL01;Initial Catalog=NAV;..."`
`--sqlconnectionmigration <str>`	Global	Migration DB SQL connection (helpers/transfer)	`"Data Source=SQL01;Initial Catalog=MIGRATION;..."`
`--sourcedatabase <str>`	Global	Source database name (query generation)	`BC_PROD`
`--mappingdatabase <str>`	Global	Mapping database name	`BC_MAPPING`
`--migrationdatabase <str>`	Global	Migration DB (helper tables)	`BC_MIGRATION`
`--sourcecompany <str>`	Global	Source NAV/BC company name	`CRONUS AG`
`--destinationcompany <str>`	Global	Target BC company name	`CRONUS AG`
`-w <EndpointURL>`	Global	BC OData Upload_LoadData endpoint	`https://api.businesscentral.dynamics.com/.../Upload_LoadData`
`--environment <name>`	Global	Named BC environment (for OData config)	`Production`
`--sqlcommandtimeout <seconds>`	Global	SQL command timeout (0 = unlimited)	`--sqlcommandtimeout 300`

Parameter	Type	Meaning	Example
`--verbose 1`	Global	Enable verbose logging	`--verbose 1`

German Hinweis: Die Parameter `--tenantid`, `--clientid`, und `--clientsecret` sind erforderlich für Azure AD-Authentifizierung in SaaS-Umgebungen. Für On-Premise-Systeme verwenden Sie SQL-Authentifizierung oder integrierte Sicherheit.

---

3. Common Task Options

Used by most migration tasks to specify source/destination tables and data filtering.

Parameter	Applies To	Meaning	Example
`-s <TableNo>`	putdata, getdata, deletedata, execute, generate*	Source table number in BC	`-s 18` (Customer table)
`-d <TableNo>`	putdata, getdata, deletedata	Destination table override	`-d 349` (map source 347 to dest 349)
`-m <Path>`	putdata, getdata, generate*	Mapping file path (or directory for batch)	`-m mapping\Dimension.json`
`-i <SQL or Path>`	putdata, getdata, getalldata	SQL query / stored proc / JSON file or directory	`-i "SELECT * FROM ..."` or `-i chunks/`
`-n <TableName>`	getdata (SQL output)	Destination table name when writing to SQL	`-n [CRONUS$Customer]`
`-v <FilterStr>`	getdata, deletedata	BC filter/view syntax	`-v "WHERE(Active=1)"`
`-o <Path>`	putdata, loadcfg, savecfg	Output file path (instead of sending to BC)	`-o out\export.json`
`-f <Format>`	putdata (with `-o`), JSON input	Output format: `j` (JSON) `csv` `xls`	`-f csv`
`-r <Range>`	putdata, putalldata, generate*	Filter Entry No. range or table filter	`-r 1..100000` or `-r 5..10|20..30`
`--ojsonformat <old|new>`	putdata (JSON output)	JSON format variant (default: old)	`--ojsonformat new`
`--testrun`	putdata, putalldata	Send only first record per table (validation)	`--testrun`

Deutsch Erklärung: Der Parameter `-m` akzeptiert entweder eine einzelne Mapping-Datei oder ein Verzeichnis mit mehreren Mappings für Batch-Operationen. `-i` kann SQL-Statements, SQL-Prozeduraufrufe oder Pfade zu JSON-Dateien entgegennehmen.

---

4. Transfer Behavior & Performance Options

Control how data flows, what data transfers, and how fast.

Parameter	Meaning	Impact	Example
`--automapping <value>`	Auto-generate field mappings	Saves manual mapping work; semantics vary by task	`--automapping 18` (putdata) or `--automapping true` (generateallquery)
`--changesonly <0|1>`	Send only new/changed records (delta)	Default 0 = full export; 1 = delta only	`--changesonly 1`
`--tablegroup "grp[,grp]"`	Filter which table groups process	Default "1,2" (master data); use "3", "4", etc. for transaction data	`--tablegroup "1,2,3"`
`--limitmemory <MB>`	Max memory per batch (SQL-sourced flow)	Limits peak memory usage during SQL generation	`--limitmemory 500`
`--buffered`	Disable streaming, buffer in memory	**Increases memory**; use only if streaming fails	`--buffered`
`--benchmark`	Record performance metrics (putdata/putalldata)	Outputs JSON log to `logs/perf/<runId>.json`	`--benchmark`

Parameter	Meaning	Impact	Example
`--benchmarkmode <label>`	Label for benchmark comparison	Helps distinguish runs in analysis	`--benchmarkmode "full-delta-v2"`

Wichtig (Important): Without `--changesonly 1`, each run sends **all** records matching the source query. Use delta mode (`--changesonly 1`) for repeatable syncs.

---

5. Tasks Reference

DataMigratePro supports these core tasks (specified via `-t <TASK>`):

Data Transfer Tasks

Task	Purpose	Requires	Typical Use
`putdata`	Send single table to BC	`-s`, `-i` (SQL or JSON), `-m` or `--automapping`	Migrate one customer/vendor/dimension table
`putalldata`	Send all configured tables to BC	`--tablegroup`, `--automapping` or `-m`	Full migration run or delta sync
`getdata`	Export BC table to SQL	`-s`, `-n`, optional `-v` filter	Verify BC data or extract for reporting
`deletedata`	Delete records from BC table	`-s`, optional `-v` filter `-d`	Clean test/sandbox data before re-migration

Query & Analysis Tasks

Task	Purpose	Requires	Typical Use
`generatequery`	Build SQL query for single table	`-s`, optional `-m` (to save mapping)	Preview what will be extracted from source
`generatetransferquery`	Generate SQL + mapping for transfer	`-s`, optional `-m`	Plan custom transfer without execution
`generateallquery`	Build queries for all configured tables	`--tablegroup`, `-m` (directory)	Batch SQL generation for all tables
`countrecords`	Count records in source tables	(configured tables)	Validate data volume before migration

Configuration & Setup Tasks

Task	Purpose	Requires	Typical Use
`loadconfiguration`	Pull configuration from BC to local DB	`--tenantid`, `--environment`	Bootstrap local migration environment
`saveconfiguration`	Push local config back to BC	`--tenantid`, `--environment`	Deploy updates after changes
`exportsource`	Export source data structure	(configured source DB)	Document source schema before migration

Database Management Tasks

Task	Purpose	Requires	Typical Use
`createazuresql`	Create Azure SQL staging database	`--tenantid`, Azure credentials	Set up remote staging for cloud migrations
`removeazuresql`	Drop Azure SQL staging database	`--tenantid`, Azure credentials	Clean up after migration completes

Service & Automation Tasks

Task	Purpose	Requires	Typical Use
`listen`	Start Service Bus listener (unattended sync)	`--registration`, `--tenantid`, ServiceBus config	Continuous sync daemon for repeated jobs
`execute`	Call BC codeunit from CLI	`-s` (codeunit ID)	Trigger custom post-migration logic

---

6. Service & Installation Options

Run DataMigratePro as a Windows service or web service for unattended operation.

Parameter	Purpose	Typical Use
`--service <port>`	Run as local SOAP-like web service on port	`--service 8080` → listen on `http://localhost:8080`
`--install <Name>`	Install as Windows Service (with `-t listen` only)	`--install "DataMigrate PRO Synch-Service"`
`--uninstall <Name>`	Uninstall Windows Service	`--uninstall "DataMigrate PRO Synch-Service"`
`--queuehandling`	Process queue once (no listener loop)	Trigger one-time queue processing without daemon
`--buildmapping`	Build mappings & apply BC configuration	Apply config changes from local DB to BC

---

7. Practical Sample Commands

Quick Start: Single Table with Auto-Mapping

```
DataMigratePro.exe -t putdata `
  -s 18 `
  -i "SELECT * FROM [CRONUS$Customer]" `
  --automapping 18
```

English: Migrate Customer table (ID 18) using automatic field mapping detection.
Deutsch: Kundentabelle migrieren mit automatischer Feldmapping-Erkennung.

Test Run: Validate Mapping Without Full Transfer

```
DataMigratePro.exe -t putdata `
  -s 18 `
  -m mapping\Customer.mapping.json `
  -i "SELECT TOP 100 * FROM [CRONUS$Customer]" `
  --testrun
```

Use Case: Check mapping and BC endpoint before running full volume.

Full Table Group Migration: All Master Data

```
DataMigratePro.exe -t putalldata `
  --tablegroup "1,2" `
  --automapping `
  --changesonly 0
```

English: Transfer all tables in groups 1–2 (master data) with full export.
Deutsch: Vollständiger Transfer aller Mastertabellen (Gruppen 1–2).

Delta Migration: Only Changed Records

```
DataMigratePro.exe -t putalldata `
  --tablegroup "1,2,3,4" `
  --automapping `
  --changesonly 1
```

English: Send only new/updated/deleted records for a repeatable sync cycle.
Deutsch: Nur Änderungen seit dem letzten Durchlauf synchronisieren.

JSON Input: Migrate from Pre-Generated Chunks

```
DataMigratePro.exe -t putdata `
  -s 347 `
  -m mapping\Dimension.mapping.json `
  -f json `
  -i in\chunks\
```

English: Read JSON chunk files from directory and send to BC (streaming).
Deutsch: JSON-Chunk-Dateien aus Verzeichnis lesen und zu BC übertragen.

Generate Output Files (No BC Send)

CSV Output:

```
DataMigratePro.exe -t putdata `
  -s 18 `
  -m mapping\Customer.mapping.json `
  -i "SELECT * FROM [CRONUS$Customer]" `
  -o out\customer_export.csv `
  -f csv
```

Excel Output:

```
DataMigratePro.exe -t putdata `
  -s 18 `
  -m mapping\Customer.mapping.json `
  -i "SELECT * FROM [CRONUS$Customer]" `
  -o out\customer_export.xlsx `
  -f xls
```

English: Export data to CSV or Excel without sending to BC (review/audit step).
Deutsch: Daten exportieren zur Überprüfung vor BC-Upload.

Configuration Export/Import Workflows

Export BC Configuration to Local ZIP:

```
DataMigratePro.exe -t loadconfiguration `
  -o configuration_backup.zip
```

Import ZIP to BC:

```
DataMigratePro.exe -t saveconfiguration `
  -i configuration_backup.zip
```

English: Use for configuration version control and migration environment replication.
Deutsch: Konfiguration sichern und auf andere Umgebungen übertragen.

Advanced: Range-Filtered Migration

Migrate **only** records with Entry No. between 1 and 100,000:

```
DataMigratePro.exe -t putdata `
  -s 18 `
  -m mapping\Customer.mapping.json `
  -i "SELECT * FROM [CRONUS$Customer]" `
  -r 1..100000
```

Use Case: Staged migration of large tables in smaller batches to manage memory/API load.

Advanced: Table Group Filter by Range

Process **only** tables 5 through 10, excluding table 7:

```
DataMigratePro.exe -t putalldata `
  -r "5..10&<>7" `
  --automapping `
  --changesonly 1
```

English: Business Central filter syntax for precise table selection.
Deutsch: BC-Filtersyntax unterstützt komplexe Bereichs- und Ausschlusslogik.

Monitor Performance: Benchmark Mode

```
DataMigratePro.exe -t putalldata `
  --tablegroup "1,2" `
  --automapping `
  --benchmark `
  --benchmarkmode "full-run-20260408"
```

Output: JSON metrics in `logs/perf/<runId>.json` → analyze throughput, latency, row counts.

---

8. Edge Cases & Validation

Missing Source Table ID

Error: Most tasks require `-s` (source table number).

Fix: Supply `-s <TableNo>` before running.

```
# ❌ Wrong: Missing -s
DataMigratePro.exe -t putdata -i "SELECT ..." -m mapping.json

# ✅ Correct: Include -s
DataMigratePro.exe -t putdata -s 18 -i "SELECT ..." -m mapping.json
```

Incorrect Mapping Path

Error: "Mapping file not found" or null reference during transfer.

Fix: Verify file exists and path is correctly quoted (especially paths with spaces).

```
# ❌ Wrong: Path with spaces, no quotes
DataMigratePro.exe -t putdata -s 18 -m my mapping\file.json

# ✅ Correct: Quoted path
DataMigratePro.exe -t putdata -s 18 -m "my mapping\file.json"
```

AutoMapping Semantics

`--automapping` has **different** meanings depending on task:

- **putdata:** Pass the table ID number → auto-detect field mappings
- **putalldata:** Pass `true`/`false` as validation toggle
- **generate tasks:** Pass `true`/`false` for validation

```
# putdata: automapping = table number
DataMigratePro.exe -t putdata -s 18 --automapping 18
```

```
# generateallquery: automapping = boolean
DataMigratePro.exe -t generateallquery --automapping true
```

Service Installation Boundary

Only `-t listen` or `--service <port>` can be installed as Windows Service.

```
# ❌ Wrong: Cannot install putdata as service
DataMigratePro.exe -t putdata --install "MyService"

# ✅ Correct: Install listener for unattended sync
DataMigratePro.exe -t listen --install "DataMigrate PRO Sync"
```

---

9. Common Troubleshooting Map

Symptom	Likely Cause	Quick Fix
Connection refused	BC endpoint unreachable or wrong URL	Verify `-w` parameter and network connectivity
OAuth fails (SaaS)	Invalid `--tenantid`, `--clientid`, `--clientsecret`	Re-check Azure AD app in BC tenant; confirm secrets are current
Service stops immediately	ServiceBus not configured or unreachable	Check `ServiceBus` section in `settings.json`; verify namespace/connection string
No queue processing	Queue names wrong or permissions insufficient	Verify queue names in settings; check Azure RBAC roles
Timeout during putdata	SQL query too large or BC endpoint slow	Increase `--sqlcommandtimeout` or `HttpClientTimeoutSeconds` in settings.json
Unmapped fields	Mapping file incomplete or wrong structure	Run `--testrun` first to validate mapping; check JSON syntax

---

10. Next Steps

1. **For your first migration:** Start with the quick-start single-table command in §7, using `--testrun` to validate before full run.
2. **For repeated syncs:** Adopt delta mode (`--changesonly 1`) and full table groups with `--automapping`.
3. **For production service:** Install as Windows Service (`-t listen --install`) with Service Bus configuration.
4. **For detailed configuration:** See `docs/settings.json.md` for all persistence and timeout semantics.
5. **For scenario-based examples:** See `docs/DataMigratePro-CLI-Szenarien.md` (German, comprehensive workflow guides).

Quick Reference Cheat Sheet

Goal	Command Pattern
Single table test	`DataMigratePro -t putdata -s <ID> -i "SELECT ..." -m mapping.json --testrun`
Single table full	`DataMigratePro -t putdata -s <ID> -i "SELECT ..." -m mapping.json`
All master data	`DataMigratePro -t putalldata --tablegroup "1,2" --automapping`
Delta sync	`DataMigratePro -t putalldata --tablegroup "1,2" --automapping --changesonly 1`
Export to file	`DataMigratePro -t putdata -s <ID> -i "SELECT ..." -o output.csv -f csv`
BC to SQL read	`DataMigratePro -t getdata -s <ID> -n [Company$Table]`
Install sync service	`DataMigratePro -t listen --install "ServiceName" --registration <KEY> --tenantid <ID>`

Version 2026-04 | Bilingual Reference | For DataMigratePro ≥ v2026.02

IO Integrated GmbH & Co. KG

Experts in Business Central Migration & Data Integration

Contact:

- Email: kontakt@ioi.gmbh
- Phone: +49 (0)214 8402 3000
- Web: https://ioi.gmbh/