

# use\_mfgtool

## mfgtool 制造信息工具 — 使用手册

版本: 1.0.0

作者: mengfei.wu

日期: 2026-02-28

适用平台: AGX Orin (AI-BOX) / 508 (RK3568) / 轨交 (Rail)

安装目录: `/opt/equip` (离线部署) 或 `pip install .` (开发环境)

零第三方依赖: 仅使用 Python 标准库

## 目录

- [1. 快速开始](#)
- [2. 全局选项](#)
- [3. 操作命令](#)
  - [3.1 write — 写入制造信息](#)
  - [3.2 show — 读取制造信息](#)
  - [3.3 update — 更新单个属性](#)
  - [3.4 verify — 校验数据完整性](#)
- [4. 平台说明](#)
- [5. 字段定义表](#)
- [6. MAC 地址处理](#)
- [7. 三级存储架构](#)
- [8. 配置文件](#)
- [9. JSON 批量导入](#)
- [10. 常见问题](#)
- [附录: 完整命令速查](#)

## 1. 快速开始

```
# 查看帮助
sudo mfgtool --help
```

```
# 写入单个字段
```

```
sudo mfgtool write --key Board_Mfg_Date --value "2026-03-15"

# 写入 MAC 地址（自动同步烧写到网卡硬件）
sudo mfgtool write --key Device_MAC1 --value "50:D3:3B:00:49:51"

# 从 JSON 文件批量导入
sudo mfgtool write --json input.json

# 查看所有制造信息
sudo mfgtool show --all

# 查看单个字段
sudo mfgtool show --key Board_Serial

# 更新已有字段
sudo mfgtool update --key Product_Serial --value "210235KXXXXXXX"

# 校验数据完整性
sudo mfgtool verify

# 指定平台
sudo mfgtool --platform 508 show --all
```

## 2. 全局选项

以下全局选项可在任何子命令前使用：

选项	缩写	默认值	说明
--platform PLATFORM	-p	自动检测	目标平台： agx / 508 / rail
--config CONFIG	-c	内置 config.json	配置文件路径
--verbose	-v	关闭	显示详细调试信息

**平台优先级：** 命令行 --platform > 配置文件 default\_platform > 自动检测

**自动检测规则：**

检测条件	判定平台
存在 vendor_storage 命令	508
存在 /sys/class/net/enP1p1s0f0 网口	agx

检测条件	判定平台
存在 <code>/sys/bus/i2c/devices/0-0050/eeprom</code>	rail
以上均不满足	agx (默认)

## 3. 操作命令

### 3.1 write — 写入制造信息

支持两种模式：单个字段写入和 JSON 文件批量导入。

#### 单字段写入

```
sudo mfgtool write --key <字段名> --value <值> [--force]
```

参数	必填	说明
<code>--key</code>	是（与 <code>--json</code> 二选一）	写入的属性键名
<code>--value</code>	是（配合 <code>--key</code> ）	写入的属性值
<code>--force</code> / <code>-f</code>	否	强制覆盖已存在的数据
<code>--script</code>	否	指定 MAC 更新脚本路径（默认： <code>/etc/network/update_mac.sh</code> ）

#### 示例：

```
# 写入生产日期
sudo mfgtool write --key Board_Mfg_Date --value "2026-03-15"

# 写入生产厂家
sudo mfgtool write --key Board_Mfg --value "CNIT"

# 写入 MAC 地址（支持多种格式）
sudo mfgtool write --key Device_MAC1 --value "50:D3:3B:00:49:51"
sudo mfgtool write --key Device_MAC1 --value "50D33B004951"
sudo mfgtool write --key Device_MAC1 --value "50-D3-3B-00-49-51"

# 强制覆盖已存在的值
sudo mfgtool write --key Board_Mfg --value "CNIT" --force
```

行为说明：

- 如果字段已存在且值相同 → 跳过，返回成功
- 如果字段已存在且值不同 → 提示使用 `--force` 或 `update` 命令
- 写入 `Device_MAC*` 字段时 → 自动同步烧写到网卡硬件
- 每次写入后自动重新计算 MD5 校验和

JSON 批量导入

```
sudo mfgtool write --json <文件路径> [--force]
```

参数	必填	说明
--json	是 (与 --key 二选一)	JSON 文件路径
--force / -f	否	强制覆盖已存在的数据

示例：

```
# 首次批量导入
sudo mfgtool write --json input.json

# 覆盖已有数据
sudo mfgtool write --json input.json --force
```

| [JSON 文件格式参见 第9 节](#)。

3.2 show — 读取制造信息

```
sudo mfgtool show [--key <字段名>] [--all]
```

参数	必填	说明
--key	否	显示指定字段的值
--all	否	显示所有字段（默认行为）

示例：

```
# 显示所有制造信息（JSON 格式）
sudo mfgtool show --all

# 显示特定字段
sudo mfgtool show --key Device_MAC1
# 输出: Device_MAC1: 50D33B004951 (50:D3:3B:00:49:51)

sudo mfgtool show --key Board_Serial
# 输出: Board_Serial: 02K1HLG90012345678
```

### 行为说明：

- 数据为空时提示先执行 `write` 命令
- 读取时自动验证 MD5 校验和，不匹配则输出警告
- MAC 字段同时显示原始格式和冒号分隔格式

## 3.3 update — 更新单个属性

仅更新已存在的字段。如果字段不存在，会报错并建议使用 `write` 命令。

```
sudo mfgtool update --key <字段名> --value <新值>
```

参数	必填	说明
<code>--key</code>	是	要更新的属性名（必须已存在）
<code>--value</code>	是	新的属性值
<code>--script</code>	否	指定 MAC 更新脚本路径

### 示例：

```
# 更新序列号
sudo mfgtool update --key Product_Serial --value "210235KXXXXXXX"

# 更新 MAC 地址
sudo mfgtool update --key Device_MAC2 --value "AA:BB:CC:DD:EE:02"
```

### write 与 update 的区别：

行为	write	update
字段不存在时	✅ 创建新字段	❌ 报错
字段已存在时	需要 <code>--force</code> 才覆盖	✅ 直接更新
适用场景	首次录入	修改已有数据

### 3.4 verify — 校验数据完整性

计算存储数据的 MD5 并与记录的校验和比对，检测数据是否被篡改。

```
sudo mfgtool verify
```

无需任何参数。

输出示例：

```
# 校验通过
Checksum valid (MD5: a1b2c3d4e5f6...)

# 校验失败
Checksum invalid: MD5 不匹配
  存储值: a1b2c3d4e5f6...
  计算值: f6e5d4c3b2a1...
  可能被修改的字段: Board_Serial, Device_MAC1
```

## 4. 平台说明

mfgtool 支持三个硬件平台，各平台的存储配置和 MAC 处理方式不同：

项目	AGX (AI-BOX)	508 (NX)	Rail (轨交)
平台标识	agx	508	rail
操作系统	Ubuntu 22.04	Debian 10	Ubuntu 22.04
EEPROM I2C	Bus 0, Addr 0x51	Bus 3, Addr 0x50	Bus 1, Addr 0x51
MAC 处理工具	ethtool -E (I350)	vendor_storage	Motorcomm_NIC_Burn_Helper

项目	AGX (AI-BOX)	508 (NX)	Rail (轨交)
MAC 端口数	4 口	1 口	2 口 (1 口烧写)
系统工具依赖	i2c-tools , ethtool	i2c-tools , vendor_storage	i2c-tools , Motorcomm_NIC_Burn_Helper

AGX 平台 MAC 端口映射

字段	网口接口	丝印标识	I350 Port	EEPROM 偏移
Device_MAC1	enP1p1s0f0	LAN4	Port 0	0x0000
Device_MAC2	enP1p1s0f1	LAN2	Port 1	0x0100
Device_MAC3	enP1p1s0f2	LAN5	Port 2	0x0180
Device_MAC4	enP1p1s0f3	LAN3	Port 3	0x0200

508 平台 MAC 端口映射

字段	网口接口	丝印标识	处理方式
Device_MAC1	eth0	LAN1	vendor_storage -w VENDOR_LAN_MAC_ID

轨交平台 MAC 端口映射

字段	网口接口	丝印标识	处理方式
Device_MAC1	enP8p1s0	GB2	Motorcomm_NIC_Burn_Helper (烧写)
Device_MAC2	enP1p1s0	GB1	仅记录，不烧写

5. 字段定义表

所有条目必须符合以下长度和格式要求。写入/更新时工具自动验证。

字段名	最大长度	格式	必填	说明
Board_Mfg_Date	10 字节	日期		单板测试日期（格式：YYYY-MM-DD）

字段名	最大长度	格式	必填	说明
Board_Mfg	10 字节	字符串	☑	生产厂家
Board_Part_Number	20 字节	字符串	☑	0302 编码或板名
Board_Serial	20 字节	字符串	☑	PCBA 条码
BOM_Ver	5 字节	字符串	☑	BOM 版本（从 PDM 获取）
Device_MAC1	20 字节	MAC	—	AGX: enP1p1s0f0 (LAN4) / 508: eth0 / 轨交: enP8p1s0 (GB2)
Device_MAC2	20 字节	MAC	—	AGX: enP1p1s0f1 (LAN2) / 轨交: enP1p1s0 (GB1)
Device_MAC3	20 字节	MAC	—	AGX: enP1p1s0f2 (LAN5)
Device_MAC4	20 字节	MAC	—	AGX: enP1p1s0f3 (LAN3)
Product_Mfg_Date	10 字节	日期	☑	整机测试时间（格式：YYYY-MM-DD）
Product_Manufacturer	10 字节	字符串	☑	整机生产厂商
Product_Part_Number	20 字节	字符串	☑	整机 0235 编码或产品名称
Product_Serial	20 字节	字符串	☑	0235 序列号
Product_Version	5 字节	字符串	☑	产品版本
Model_Serial	20 字节	字符串	—	选配模组序列号
SSD_Serial	20 字节	字符串	—	选配固态序列号
WIFI_Serial	20 字节	字符串	—	选配 WIFI 序列号
5G_Serial	20 字节	字符串	—	选配 5G 序列号
PCIE_Serial	20 字节	字符串	—	选配 PCIE 板卡序列号
Reserved	20 字节	字符串	—	预留空间



## 6. MAC 地址处理

### 输入格式

MAC 地址支持以下三种输入格式，工具会自动标准化为 **12 位大写十六进制字符串**：

输入格式	示例	标准化结果
纯十六进制	50D33B004951	50D33B004951
冒号分隔	50:D3:3B:00:49:51	50D33B004951
横线分隔	50-D3-3B-00-49-51	50D33B004951

### 校验规则

规则	说明
长度	必须是 6 字节（12 位十六进制字符）
单播	第一字节最低位必须为 0（拒绝组播地址）
非全零	不允许 000000000000
非广播	不允许 FFFFFFFFFF

### 写入流程

当写入 `Device_MAC*` 字段时，工具自动执行：

- 1. 格式清洗 → 标准化为 12 位大写十六进制
- 2. 有效性校验 → 拒绝组播/全零/广播地址
- 3. 存储写入 → 保存到 EEPROM/磁盘/文件系统
- 4. 硬件同步 → 调用平台对应工具烧写到网卡
  - └─ AGX: `ethtool -E enP1p1s0f0 magic 0x15218086 ...`
  - └─ 508: `vendor_storage -w VENDOR_LAN_MAC_ID ...`
  - └─ Rail: `Motorcomm_NIC_Burn_Helper ...`

**注意：**MAC 地址烧写到网卡 EEPROM 后，需要 **重启系统** 才能生效。

## 7. 三级存储架构

mfgtool 按照配置的 `storage_priority` 顺序探测并使用第一个可用的存储后端：

优先级 1: EEPROM /sys/bus/i2c/devices/{bus}-{addr}/eeprom

优先级 2: 磁盘预留分区 /dev/mmcblk0p10 → /mnt/mfg\_info/mfg\_info.conf

优先级 3: 文件系统 /etc/mfg\_info.conf

存储后端	路径	特点
EEPROM	I2C 设备 (如 AT24C32)	最可靠, 硬件级持久化, 容量 4KB
磁盘分区	EMMC 预留分区	系统升级不受影响, 需挂载
文件系统	/etc/mfg_info.conf	兜底方案, 可能被系统升级覆盖

数据格式

存储内容为 UTF-8 编码的 JSON 字符串, 尾部包含 MD5 校验和:

```
{
  "Board_Mfg_Date": "2026-03-15",
  "Board_Mfg": "CNIT",
  "Board_Part_Number": "02K1HS6256V00002",
  "Board_Serial": "02K1HLG90012345",
  "Device_MAC1": "50D33B004951",
  "checksum": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6"
}
```

8. 配置文件

加载优先级

1. 命令行参数 --config /path/to/config.json
2. 包内默认配置 mfgtool/config/config.json
3. 系统配置 /etc/mfgtool.conf

关键配置项

配置项	说明	示例
default_platform	默认平台, 免去每次输入 --platform	"agx"
log_path	日志文件路径	"/var/log/mfgtool.log"

配置项	说明	示例
storage_priority	存储后端优先级数组	["eeprom", "disk_partition", "filesystem"]
fields	字段定义（格式、长度、是否必填）	见字段定义表
platforms	各平台的 EEPROM/MAC 配置	见配置文件

## 修改默认平台

```
# 直接编辑配置文件
vi /opt/equip/.venv/lib/python3.10/site-packages/mfgtool/config/config.json
# 将 "default_platform": "agx" 改为目标平台

# 或使用命令行覆盖
sudo mfgtool --platform 508 show --all
```

# 9. JSON 批量导入

## 文件格式

准备一个 JSON 文件，包含所有需要写入的字段（不含 checksum，工具自动计算）：

```
{
  "Board_Mfg_Date": "2026-03-15",
  "Board_Mfg": "CNIT",
  "Board_Part_Number": "02K1HS6256V00002",
  "Board_Serial": "02K1HLG90012345678",
  "BOM_Ver": "BOMA",
  "Device_MAC1": "50:D3:3B:00:49:51",
  "Device_MAC2": "50:D3:3B:00:49:52",
  "Device_MAC3": "50:D3:3B:00:49:53",
  "Device_MAC4": "50:D3:3B:00:49:54",
  "Product_Mfg_Date": "2026-03-20",
  "Product_Manufacturer": "CNIT",
  "Product_Part_Number": "IGWB02",
  "Product_Serial": "210235KXXXXXXX",
  "Product_Version": "10000",
  "Model_Serial": "V1.00.00",
  "SSD_Serial": "SNSSD123456789991",
  "WIFI_Serial": "SNWIFI1212121299",
  "5G_Serial": "SN5G123456654321",
```

```
"PCIE_Serial": "SNPCIE1234566611",  
"Reserved": ""  
}
```

## 导入命令

```
# 首次导入  
sudo mfgtool write --json input.json  
  
# 覆盖已有数据  
sudo mfgtool write --json input.json --force
```

## 导入流程

1. 读取 JSON 文件 → 解析所有字段
2. 逐字段验证 → 格式、长度校验
3. MAC 地址标准化 → 统一为 12 位大写
4. 计算 MD5 校验和 → 追加 checksum 字段
5. 持久化写入 → 存储到最高优先级后端
6. MAC 同步 → 逐个烧写到网卡硬件

---

## 10. 常见问题

### Q1: 运行命令提示 `Permission denied` ?

读写 EEPROM 需要硬件权限。请使用 `sudo` 执行：

```
sudo mfgtool show --all
```

或将用户加入 `i2c` 用户组：

```
sudo usermod -aG i2c $USER
```

### Q2: 如何在没有硬件的电脑上测试？

修改 `config.json`，将 `storage_priority` 设置为 `["filesystem"]`，并将文件路径指向临时目录：

```
"storage_priority": ["filesystem"],  
"filesystem": { "path": "/tmp/mfg_test.conf" }
```

### Q3: 提示找不到 `vendor_storage` 或 `ethtool` ?

这些是系统级工具，需根据平台安装：

```
# AGX 平台
sudo apt install -y i2c-tools ethtool

# 508 平台
sudo apt install -y i2c-tools
which vendor_storage # 确认工具存在

# 轨交平台
sudo cp Motorcomm_NIC_Burn_Helper /usr/local/bin/
sudo chmod +x /usr/local/bin/Motorcomm_NIC_Burn_Helper
```

### Q4: `write` 和 `update` 有什么区别？

- `write` 用于 **首次写入**，字段不存在时创建，已存在时需 `--force` 覆盖
- `update` 用于 **修改已有字段**，字段必须已存在，否则报错

### Q5: MAC 地址写入后立即生效吗？

MAC 地址会立即写入到网卡 EEPROM / `vendor_storage`，但需要 **重启系统** 后网卡才会加载新的 MAC 地址。

### Q6: 如何查看日志？

```
cat /var/log/mfgtool.log

# 或使用 --verbose 实时查看详细输出
sudo mfgtool -v show --all
```

---

## 附录：完整命令速查

```
# ===== 写入 =====
# 写入单个字段
sudo mfgtool write --key Board_Mfg_Date      --value "2026-03-15"
sudo mfgtool write --key Board_Mfg           --value "CNIT"
sudo mfgtool write --key Board_Part_Number   --value "02K1HS6256V00002"
sudo mfgtool write --key Board_Serial        --value "02K1HLG90012345678"
sudo mfgtool write --key BOM_Ver             --value "BOMA"
```

# 写入 MAC 地址（自动烧写到网卡，重启生效）

```
sudo mfgtool write --key Device_MAC1      --value "50:D3:3B:00:49:51"
sudo mfgtool write --key Device_MAC2      --value "50:D3:3B:00:49:52"
sudo mfgtool write --key Device_MAC3      --value "50:D3:3B:00:49:53"
sudo mfgtool write --key Device_MAC4      --value "50:D3:3B:00:49:54"
```

# 写入产品信息

```
sudo mfgtool write --key Product_Mfg_Date  --value "2026-03-20"
sudo mfgtool write --key Product_Manufacturer --value "CNIT"
sudo mfgtool write --key Product_Part_Number --value "IGWB02"
sudo mfgtool write --key Product_Serial    --value "210235KXXXXXXX"
sudo mfgtool write --key Product_Version   --value "10000"
```

# 写入选配信息

```
sudo mfgtool write --key Model_Serial      --value "V1.00.00"
sudo mfgtool write --key SSD_Serial        --value "SNSSD123456789991"
sudo mfgtool write --key WIFI_Serial       --value "SNWIFI1212121299"
sudo mfgtool write --key 5G_Serial         --value "SN5G123456654321"
sudo mfgtool write --key PCIE_Serial       --value "SNPCIE1234566611"
```

# 强制覆盖

```
sudo mfgtool write --key Board_Mfg        --value "CNIT" --force
```

# JSON 批量导入

```
sudo mfgtool write --json input.json
sudo mfgtool write --json input.json --force
```

# ===== 读取 =====

# 显示所有

```
sudo mfgtool show --all
```

# 显示单个字段

```
sudo mfgtool show --key Board_Serial
sudo mfgtool show --key Device_MAC1
```

# 详细模式

```
sudo mfgtool -v show --all
```

# ===== 更新 =====

```
sudo mfgtool update --key Product_Serial --value "210235KNEW0001"
sudo mfgtool update --key Device_MAC1    --value "AA:BB:CC:DD:EE:01"
```

# ===== 校验 =====

```
sudo mfgtool verify
```

# ===== 指定平台 =====

```
sudo mfgtool --platform agx show --all
sudo mfgtool --platform 508 write --key Device_MAC1 --value "AA:BB:CC:DD:EE:01"
sudo mfgtool --platform rail show --key Device_MAC1

# ===== 指定配置文件 =====
sudo mfgtool --config /etc/mfgtool.conf show --all
```