内置AI的BMC创新技术
AI-Embedded BMC Innovation

王魁英，阿里云BMC架构师，OpenBMC 多个模块管理员
Kwin Wang，Alibaba Cloud Architect，OpenBMC Maintainer

Content

01

Why BMC-CLI

IPMI 协议 `Legacy Protocol`

NetFn / Command 二进制协议

二进制帧格式，厂商私有扩展 OEM NetFn 各不相同

同一命令在不同平台返回字段顺序、编码方式存在差异

云端服务需为每个厂商单独维护适配层

无法直接被 LLM 理解，需要大量 prompt engineering

Redfish REST API `Vendor Fragmented`

DMTF DSP0266 标准

标准定义宽泛，厂商实现差异巨大（OEM 扩展字段泛滥）

URI 路径结构因平台而异，无法统一编程

JSON Schema 庞大复杂，LLM 上下文窗口消耗极高

认证方式、TLS 配置、错误码各厂商自定义

数百人天	3–6月	∞	高
厂商适配累计投入	新平台适配周期	LLM 直接调用难度	跨厂商维护风险

bmc-cli 的答案

统一抽象层

bmc-cli 提供统一的命令语义，云端服务只需对接一套接口，彻底消除厂商差异

AI 原生设计

IPMI 和 Redfish 均非为 LLM 设计，bmc-cli 的结构化输出 + 自描述 schema 让 AI 可以零配置直接调用

降低适配风险

新平台上线只需验证 bmc-cli 的 D-Bus 接口，无需重新适配云端所有运维服务，风险收敛到单点

02

What BMC-CLI

bmc-cli 是一个运行在 OpenBMC 上的统一命令行工具，为 AI 系统和云端运维服务提供标准化的 BMC 硬件管理接口。

基于 **C++20 + sdbusplus** 构建，通过 D-Bus 抽象层屏蔽厂商差异，对上提供一致的命令语义。

统一命令行接口

CLI

将 IPMI、Redfish、D-Bus 等多种底层协议封装为一套一致的 CLI 命令体系，运维人员只需掌握一套语法

AI 原生设计

AI-Native

内置 OpenAI 兼容的 function-calling schema 生成，LLM 可零配置直接发现并调用全部 57 条 BMC 管理命令

结构化输出

Structured

所有命令返回强类型 CommandResult，原生支持 JSON / YAML / Table / Human 四种输出格式，机器可读

插件可扩展

Extensible

OEM 厂商通过 .so 插件注册差异化命令，无需修改主线代码，新命令自动出现在 AI tool schema 中

定位架构

快速上手

```
$ bmc-cli power status
```

查询电源状态

```
$ bmc-cli sensor list --type temperature
```

列出温度传感器

```
$ bmc-cli describe --format openai-tools
```

生成 AI tool schema

02

What BMC-CLI . Spec规范

DSP-BMC-CLI v1.0.0

- April 2026 · 阿里云
- 52 个域 · 164 条标准命令
- 涵盖所有 IPMI 命令
- C++20 + sdbusplus 实现

命令层次结构

- bmc-cli <domain> <action> [options]
- 两层子命令路由（domain → action）
- CommandRegistry 全局注册表
- ICommand 接口统一抽象

AI 原生设计

- bmc-cli describe --format openai-tools
- 自动生成 OpenAI function-calling schema
- CommandMeta 编译期携带完整元数据
- destructive:true 防幻觉误操作标注

插件扩展机制

- BMC_CLI_PLUGIN_INIT 宏注册入口
- dlopen 运行时动态加载 .so
- /usr/lib/bmc-cli/plugins/ 插件目录
- OEM 差异化命令零侵入主线

03

How BMC-CLI

L4 调用者层
任何调用者，无需了解底层协议
AI / LLM Agent | Python 运维脚本 | 人工 CLI 操作 | CI/CD 自动化

L3 bmc-cli 命令层
统一命令语义，自描述 schema，结构化输出
CommandRegistry 命令注册 | CommandMeta 元数据 / Schema | ICommand 接口抽象
JSON / Table / YAML 输出

L2 D-Bus 抽象层
OpenBMC D-Bus 总线，零硬编码服务名
ObjectMapper 服务发现 | sdbusplus 类型安全绑定 | Properties / Methods / Signals
异步超时 & 错误处理

L1 OpenBMC 服务层
OpenBMC 各功能服务，通过 D-Bus 暴露接口
phosphor-power | phosphor-hwmon | phosphor-network | phosphor-user-manager

L0 硬件层
物理硬件与底层协议，对上层完全透明
BMC SoC (AST2700/AST2600) | IPMI KCS / LAN | Redfish REST | I2C / UART / GPIO

核心设计决策

ObjectMapper 动态发现
通过 ObjectMapper 查询服务名，无需硬编码

sdbusplus 类型安全
C++20 模板绑定，编译期保证接口类型正确

CommandResult 结构化
所有命令返回 {exitCode, data, error}

插件 dlopen 扩展
OEM 差异化命令以 .so 插件注入，主线零修改

03

How BMC-CLI . Dbus调用链

01 **bmc-cli**

```
bmc-cli sensor get CPU0_Temp --format json
```

用户或 AI 发起命令，CommandRegistry 路由到 SensorGetCommand

02 **ObjectMapper**

```
GetObject("/xyz/openbmc_project/sensors/temperature/CPU0_Temp")
```

调用 ObjectMapper 动态查询持有该路径的服务名，返回 xyz.openbmc_project.Hwmon-xxxx

03 **D-Bus Call**

```
org.freedesktop.DBus.Properties.Get(interface, "Value")
```

sdbusplus 向目标服务发起 Properties.Get 调用，类型安全绑定，自动处理 variant 解包

04 **phosphor-hwmon**

```
return variant<double>(45.125)
```

OpenBMC hwmon 服务从内核 sysfs 读取传感器值，通过 D-Bus 返回强类型数据

05 **CommandResult**

```
{"exitCode":0,"data":{"name":"CPU0_Temp","value":45.125,"unit":"°C"}}
```

bmc-cli 将原始 D-Bus 数据封装为 CommandResult，序列化为 JSON 输出

- xyz.openbmc_project.Sensor.Value
  Value: double
- xyz.openbmc_project.State.Host
  CurrentHostState: string
- xyz.openbmc_project.Network.IP
  Address / PrefixLength
- xyz.openbmc_project.Software.Version
  Version / Purpose
- xyz.openbmc_project.User.Attributes
  UserEnabled / UserGroups
- xyz.openbmc_project.Logging.Entry
  Message / Severity / Timestamp

结构化错误码

ERR_DBUS_TIMEOUT
D-Bus 调用超时（默认 30s）

ERR_SERVICE_NOT_FOUND
ObjectMapper 未找到服务

ERR_PERMISSION_DENIED
D-Bus policy 权限拒绝

ERR_INVALID_ARGUMENT
参数类型不匹配

03

How BMC-CLI . OpenBMC集成

OPENBMC 集成架构

AI Agent / 运维工程师
调用者

bmc-cli Service（新增…
唯一新增组件
NEW

D-Bus / ObjectMapper
现有基础设施，无需修改

phosphor-power · phosphor…
现有 OpenBMC 服务，无需修改

BMC 硬件（AST2700 / AST260…
硬件层，无需修改

完全独立
单一二进制 + 单一 service，不依赖任何现有服务的修改

零侵入集成
不修改任何 phosphor-* 服务代码，不改动现有 D-Bus 接口

可随时回滚
systemctl disable bmc-cli 即可完全移除，不留任何副作用

快速上线
从 bitbake bmc-cli 到生产可用，整个集成过程不超过 1 天

01 添加 BitBake Recipe | 02 独立 systemd Service | 03 配置 D-Bus Policy | 04 集成完成，立即可用

STEP 01 添加 BitBake Recipe

在 Yocto 层中添加 bmc-cli 的 BitBake recipe，无需修改任何现有服务

meta-bmc-cli/recipes-core/bmc-cli/bmc-cli_1.0.0.bb

```
# meta-bmc-cli/recipes-core/bmc-cli/bmc-cli_1.0.0.bb
SUMMARY = "BMC Command-Line Interface"
LICENSE = "Apache-2.0"
SRC_URI = "git://github.com/bmc-cli/bmc-cli.git"

inherit meson pkgconfig systemd

SYSTEMD_SERVICE:${PN} = "bmc-cli.service"

RDEPENDS:${PN} = "sdbusplus"
```

修改现有服务： 0 个　新增文件： 1 个 .bb　集成风险： 极低　回滚方式： systemctl disable

04

BMC嵌入AI 展示．AI运维

运维工程师

"服务器好像有点热，帮我查一下"

AI 运维助手 function_call

```
function_call: bmc-cli_sensor_list(type="temperature")
```

CPU0_Temp: 92°C ⚠ CPU1_Temp: 45°C Inlet: 28°C

AI 运维助手 function_call

```
function_call: bmc-cli_sensor_list(status="critical")
```

Fan0: 800RPM ❌ (critical_low: 1000RPM)

AI 运维助手 function_call

```
function_call: bmc-cli_log_list(severity="Warning", count=5)
```

[2026-04-15 14:23] Fan0 below minimum speed threshold

诊断结论

Fan0 故障导致 CPU0 散热不足（92°C，warning:85°C，critical:95°C），建议立即更换 Fan0

核心优势

- 零脚本编写
- 零命令记忆
- 自动关联分析
- 结构化结论

AI 调用链路

describe → 获取 schema
sensor_list() → 温度数据
sensor_list() → 风扇数据
log_list() → 告警记录
→ 自动生成结论

安全保障

查询操作：直接执行
变更操作：需确认
破坏操作：需 no_confirm

04

BMC嵌入AI 展示．在线模型

在线模型调用链路

- AI Agent / 运维工程师
- 互联网 / 公网
- 云端 AI 大模型
- bmc-cli
- BMC 硬件

特点

- 响应慢（网络 + 推理延迟）
- 费用高，按 Token 计费
- 需要 API Key / Token
- 依赖外网连接
- 数据上传云端，安全风险

如果每分钟访问一次命令，费用将是天文数字 （1,440次/天 × ¥0.25/次）

¥0.25	¥360	¥13.1万	¥13.1亿
单次查询	每天（1次/分钟）	每年（单台）	1万台/年
本地: ¥0	本地: ¥0	本地: ¥0	本地: ¥0

极端场景：1万台机器 × 1次/分钟 × 365天

= 5,256,000,000 次调用

云端 AI 年度总费用	BMC 本地 NPU 年度总费用
¥13.1亿	¥0

*按单次查询 ¥0.25 计算，实际费用因模型和 Token 用量而异

04

BMC嵌入AI 展示．本地模型

本地 NPU 调用链路

- AI Agent / 运维工程师
- SSH / 本地调用
- BMC NPU 本地模型
- bmc-cli
- BMC 硬件

特点

- 快 10 倍（本地推理，无网络）
- 完全免费，零 Token 费用
- 无需 API Key / Token
- 完全离线，断网可用
- 数据不出机器，安全合规

响应延迟

云端	本地
~2000ms	~200ms

快10倍

单次费用

云端	本地
¥0.25	¥0

100%节省

万台年费用

云端	本地
¥13.1亿	¥0

节省13亿

NPU 专用推理芯片

BMC SoC 内置 NPU，专为 AI 推理优化，功耗极低，无需额外硬件

数据安全合规

敏感硬件数据在本地处理，不上传云端，满足数据主权和安全合规要求

无限高频调用

无 Token 限制，无 Rate Limit，支持每秒多次调用，适合自动驾驶运维

bmc-cli 标准接口

本地 AI 通过 bmc-cli 调用 BMC，与云端方案完全相同的接口，无缝切换

BMC 本地 NPU 是规模化 AI 运维的唯一可持续方案

快 10 倍 · 零成本 · 离线可用 · 数据不出机器 = 自动驾驶运维基础设施

¥0

万台年费用

谢谢

谢谢

谢谢

谢谢