# 《芯祥联科技NMS SDK用户开发说明书》

## 关于芯祥联科技

芯祥联科技是专注于SNMP（简单网络管理协议）相关开发套件研发与服务的科技企业，核心产品涵盖NMS SDK（网络管理系统开发套件）、Agent SDK（代理开发套件）等，可为用户提供从协议栈调用、定制化开发到跨平台适配的全流程技术支持。公司致力于降低SNMP相关应用的开发门槛，助力企业快速构建稳定、高效的设备管理系统。如需业务合作，可通过合作邮箱hezuo@xxltech.cn或官网www.xxltech.cn联系。

## 文档概述与说明

本说明书是芯祥联科技NMS SDK的专属用户开发指南，旨在为开发人员提供全面的SDK使用参考。文档系统梳理了NMS SDK的核心接口能力、开发流程、定制化扩展方法及典型应用场景，重点解析了上层APP通过nms_snmp_api.h接口调用SNMP协议栈的实现逻辑，并配套样例说明与注意事项。本说明书适用于具备基础编程能力的开发人员，无论是快速上手SDK开发、实现多Agent管理等基础功能，还是进行跨平台适配等定制化开发，均可从中获取针对性指导。

# 二、接口核心定位与作用

nms_snmp_api.h是NMS SDK（SNMP网络管理系统开发套件）对外暴露的核心接口头文件，其核心定位是作为上层应用程序（APP）与NMS SNMP协议栈之间的“桥梁”。该接口封装了SNMP协议栈的底层复杂逻辑，向上层APP提供简洁、标准化的调用入口，使开发人员无需深入理解SNMP协议（如SNMPv1/v2的报文格式、协议交互流程等）细节，即可快速实现对SNMP Agent的管理能力，从而聚焦于上层自定义业务场景的开发。

通过该接口，上层APP可直接复用NMS SDK内置的协议解析、网络通信、数据处理等核心能力，大幅降低SNMP相关应用的开发门槛，提升开发效率与系统稳定性。

# 二、核心数据类型与枚举定义

`nms_snmp_api.h`定义了SDK开发的基础数据类型与枚举常量，是接口调用的核心依赖，开发需重点掌握以下关键定义（基于实际API规范）：

## 2.1 基础枚举类型

- **SNMP版本**：支持v1（API_SNMP_VERSION_1）、v2c（API_SNMP_VERSION_2C）、v3（API_SNMP_VERSION_3，需商业合作开通），覆盖demo中v1/v2c的实际使用场景；

- **数据类型（BER类型）**：含整数（INTEGER）、无符号整数（UINTEGER32）、字符串（OCTET_STRING）、时间戳（TIMETICKS）等，对应demo中OID值的解析逻辑（如系统运行时间转换为秒）；
- **安全算法（v3专属）**：认证算法支持MD5/SHA1/SHA256，加密算法支持DES/AES128-AES256，为进阶安全需求预留；
- **命令类型**：包含GET/GET-NEXT/SET/GET-BULK/TRAP等，与demo中实现的所有SNMP操作完全对应；
- **错误码**：覆盖参数无效、超时、网络错误、认证失败等30+场景，demo中通过`api_snmp_error_to_str`将错误码转换为可读信息，开发需强制处理接口返回的错误码。

### 2.2 核心数据结构

- **变量绑定（ApiSnmpVarBind）**：存储OID与对应值的关联关系，是GET/SET等请求的核心参数，demo中通过`api_snmp_create_get_varbind`/`api_snmp_create_set_varbind`构造该结构；
- **Agent配置（ApiSnmpAgentConfig）**：封装设备SNMP版本、共同体名等信息，demo中创建Agent时必传该配置；
- **响应数据（ApiSnmpResponse）**：包含请求ID、错误信息、变量绑定列表，是接口异步回调的核心数据载体；
- **Trap消息（ApiSnmpTrapMessage）**：存储设备主动上报的通知数据，含版本、安全参数、变量列表，对应demo的TRAP监听功能。

## 二、核心数据类型与枚举定义

`nms_snmp_api.h`定义了SDK开发所需的核心数据类型与枚举常量，是接口调用的基础，开发人员需

## 三、核心调用逻辑与开发流程

结合demo的实现逻辑与API调用规范，上层APP的开发流程需遵循“环境配置-初始化-核心操作-资源释放”闭环，具体步骤细化如下：

### 1. 开发环境搭建

- **依赖配置**：除引入`nms_snmp_api.h`与对应平台库文件外，需关联`platform.h`（跨平台接口）、`type.h`（基础类型），demo中已明确包含这些依赖；
- **平台适配细节**：demo通过条件编译（WIN32/LINUX）实现跨平台兼容，Windows需管理员权限运行（占用162端口），Linux需处理SIGINT信号，文档补充该实操细节。

### 2. 定制化开发扩展（基于开源目录）

NMS SDK的开源特性支持开发人员基于现有目录代码，针对自身业务场景定制化扩展核心能力，重点可围绕硬件适配、网络接口、日志系统三大方向进行开发，具体结合目录结构的实现方式如下：

- **跨平台硬件适配层开发（基于os/目录）**：开源os/目录下的platform.h与platform.c提供了跨平台接口框架，可在此基础上开发目标硬件的适配层。需针对特定硬件平台（如ARM、STM32等）实现platform.h中声明的底层接口（如线程创建、内存分配、定时器驱动等），通过新增平台分支代码（如platform_arm.c）并关联至主逻辑，完成硬件相关的资源调度与控制适配，使SDK能在自定义硬件上稳定运行。demo中`platform_get_tick_ms`、`platform_delay_ms`等接口即源于此层；
- **自定义网络接口开发（基于network/目录）**：network/目录为网络接口扩展预留了标准化框架，当前以以太网（eth）为基础实现，net_es.c是为AFDX网络预留的接口文件。开发人员可根据场景需求（如使用4G/5G、LoRa等通信方式），在该目录下新增网络适配文件（如net_4g.c、net_lora.c），实现网络初始化、数据收发、链路检测等自定义逻辑，通过接口注册机制接入SNMP协议栈，替代或扩展原有网络能力；
- **日志系统定制（基于comm/目录）**：comm/目录包含日志基础组件，可结合API的`api_snmp_set_log_level`接口配置日志级别（DEBUG/INFO/WARN/ERROR），支持自定义输出方式（控制台/文件/云端），demo中默认输出至控制台，开发可通过该接口对接业务日志系统。

## 3. 协议栈初始化

上层APP启动后，需首先完成SNMP协议栈的配置与启动，核心操作结合demo与API规范细化为：

- 配置SNMP版本（demo中混用v1/v2c，如SET用v1、GET用v2c）；
- 设置通信参数（Agent IP/端口、超时时间，demo中默认超时5000ms）；
- 初始化平台依赖（demo中`PLATFORM_INIT`完成Windows WSA初始化或Linux信号注册）；
- 调用协议栈初始化接口，完成资源分配（demo通过`safe_snmp_init`实现单例初始化，避免重复调用）。

## 4. 核心业务功能调用

协议栈初始化后，基于API接口实现核心业务，结合demo场景可分为三大类接口调用，覆盖实际开发核心需求：

### 4.1 Agent管理接口

实现与被管理设备的连接管理，demo中通过创建/移除Agent完成通信链路管控：

- 创建Agent：传入设备IP、端口及配置（共同体名/版本），建立通信链路；
- 移除Agent：释放指定设备的连接资源，避免内存泄漏；
- 状态检查：通过接口判断Agent是否已存在，避免重复创建（demo中未显式调用，但为最佳实践）。

### 4.2 数据交互接口

是SDK核心能力，对应demo中所有SNMP操作，按功能分为三类：

- **数据获取类**：GET（读取指定OID值，如系统描述）、GET-NEXT（遍历MIB树节点）、GET-BULK（批量获取节点数据，提升效率），demo中均实现了对应命令；
- **数据设置类**：SET（修改OID值，如demo中控制LED状态触发Trap/Inform），需传入目标OID、数据类型及值；

上层APP基于nms_snmp_api.h调用NMS SNMP协议栈的开发流程遵循“环境配置-初始化-功能调用-资源释放”的标准逻辑，具体步骤如下：

## 1. 开发环境搭建

- **接口与库依赖配置**：将nms_snmp_api.h头文件引入上层APP的开发工程，并链接NMS SDK对应的库文件（Windows平台为lib_nms_sdk_windows.lib等，具体需匹配开发环境）；
- **平台适配检查**：开源版NMS SDK的协议LIB仅支持Windows平台，因此需先确认开发环境为Windows。若后续有其他平台（如Linux、嵌入式系统）的扩展需求，可选择以下方案：①先在Windows完成业务软件的基础开发与调试，再联系芯祥联科技（合作邮箱：hezuo@xxltech.cn，官网：www.xxltech.cn）获取专属平台适配方案；②提前对接公司获取平台服务，直接在目标平台完成开发；③直接获取芯祥联科技提供的一体化定制服务解决方案，实现全流程高效落地。

## 2. 定制化开发扩展（基于开源目录）

NMS SDK的开源特性支持开发人员基于现有目录代码，针对自身业务场景定制化扩展核心能力，重点可围绕硬件适配、网络接口、日志系统三大方向进行开发，具体结合目录结构的实现方式如下：

- **跨平台硬件适配层开发（基于os/目录）**：开源os/目录下的platform.h与platform.c提供了跨平台接口框架，可在此基础上开发目标硬件的适配层。需针对特定硬件平台（如ARM、STM32等）实现platform.h中声明的底层接口（如线程创建、内存分配、定时器驱动等），通过新增平台分支代码（如platform_arm.c）并关联至主逻辑，完成硬件相关的资源调度与控制适配，使SDK能在自定义硬件上稳定运行。
- **自定义网络接口开发（基于network/目录）**：network/目录为网络接口扩展预留了标准化框架，当前以以太网（eth）为基础实现，net_es.c是为AFDX网络预留的接口文件。开发人员可根据场景需求（如使用4G/5G、LoRa等通信方式），在该目录下新增网络适配文件（如net_4g.c、net_lora.c），实现网络初始化、数据收发、链路检测等自定义逻辑，通过接口注册机制接入SNMP协议栈，替代或扩展原有网络能力。
- **日志系统定制（基于comm/目录）**：comm/目录包含日志基础组件，可基于此开发符合业务需求的日志模块。支持自定义日志输出方式（如打印至控制台、写入本地文件、上传至云端日志系统）、日志级别（DEBUG/INFO/WARN/ERROR）过滤规则及日志格式（包含时间戳、模块名、接口信息等），通过修改agent_log.h相关接口实现，使日志信息更贴合问题定位与业务审计需求。

## 3. 协议栈初始化

上层APP启动后，需首先通过nms_snmp_api.h中的初始化接口完成SNMP协议栈的配置与启动，核心操作包括：

- 设置协议版本（如指定对接SNMPv1或SNMPv2）；
- 配置本地通信参数（如监听端口、超时时间、重试次数等）；
- 初始化日志模块（支持对接APP自定义日志系统，需通过接口指定日志输出方式）；
- 调用协议栈初始化接口，完成协议栈核心资源的分配与启动，返回初始化结果（成功/失败及错误码）。

### 3. 核心业务功能调用

协议栈初始化完成后，上层APP可根据自定义场景需求，调用nms_snmp_api.h中对应的接口实现业务逻辑，核心功能接口涵盖Agent管理、数据交互、用户配置等多个维度，是实现“多Agent管理”等场景的核心环节。

### 4. 资源释放与退出

当上层APP关闭或无需使用SNMP功能时，需调用nms_snmp_api.h中的协议栈资源释放接口，完成协议栈占用的内存、端口、线程等资源的回收，避免内存泄漏或资源残留。

## 四、典型自定义场景应用实现

nms_snmp_api.h接口的核心价值在于支撑上层APP的个性化场景开发，其中“多Agent管理”是最具代表性的场景之一，此外还可扩展实现数据采集、权限管控等功能，具体场景及接口调用方式如下：

### 1. 多Agent管理场景

该场景下，上层APP可通过接口实现对多个SNMP Agent（即被管理设备）的集中化管控，核心操作及对应接口能力包括：

- **Agent信息管理**：通过接口添加Agent（传入设备IP、SNMP版本、共同体名等信息）、删除Agent（按设备ID或IP定位）、修改Agent配置（如更新超时时间）、查询Agent列表及状态（在线/离线、通信延迟等）；
- **批量通信操作**：支持对指定Agent或Agent组发起批量SNMP请求（如批量获取设备CPU利用率、内存使用率等指标），接口内部优化了请求调度逻辑，提升多设备并发管理效率；
- **状态监控与告警**：通过注册回调接口，实时接收Agent状态变化通知（如设备离线、通信异常），上层APP可基于此实现告警弹窗、日志记录等自定义处理逻辑。

通过上述接口组合，上层APP可快速构建类似iReasoning MIB Browser的设备管理核心能力，结合GUI开发即可实现可视化的集中化管理工具。

### 2. 自定义OID数据采集与交互

结合芯祥联Agent SDK时，上层APP可通过nms_snmp_api.h接口实现自定义OID的管理与数据交互：

- 调用接口注册自定义OID（关联设备、数据类型、读写权限等）；
- 发起SNMP Get/Set请求，获取自定义OID对应的设备数据或下发配置指令；
- 对接自定义数据解析逻辑，将协议栈返回的原始数据转换为APP所需的业务格式（如将字节数转换为“GB”单位的存储容量）。

## 3. 多用户权限管理

上层APP可通过接口实现基于用户角色的权限管控，例如：

- 添加不同权限的用户（如管理员可执行Agent配置修改，普通用户仅能查看数据）；
- 调用接口验证用户身份，基于权限过滤可执行的操作（如拒绝普通用户调用Agent删除接口）。

## 4. 上层样例参考：snmp_api_demo.c的核心实现

examples/目录下的snmp_api_demo.c是上层APP开发的核心参考样例，其以“单窗口交互+后台监听”为核心架构，完整实现了SNMPv1/v2c的常用操作，开发人员可直接复用框架快速落地业务。以下是样例的核心流程与关键实现步骤，聚焦实操性无需关注细粒度类型定义：

### 4.1 样例核心架构：单窗口双能力融合

样例创新性实现“前台命令交互+后台TRAP监听”的单窗口模式，无需多终端切换，核心依赖两大模块：

- **主交互模块**：接收用户输入命令（如get、set-led1），触发对应的SNMP操作；
- **后台监听线程**：独立运行负责接收设备上报的TRAP/INFORM消息，收到消息后自动刷新界面并恢复输入提示符，不阻塞前台操作。

### 4.2 样例核心流程：从启动到操作的完整链路

1. **启动与环境初始化（主函数入口）**
   - 平台适配：通过`PLATFORM_INIT`完成系统依赖初始化（Windows初始化WSA、Linux配置输出缓冲）；
   - 退出信号注册：Windows注册控制台关闭信号、Linux注册SIGINT信号，确保退出时能清理资源；
   - 模式选择：无参数启动进入交互模式，输入“-help”打印命令说明。

2. **核心交互模式：命令驱动的SNMP操作**
   样例提供7类常用命令，覆盖开发核心需求，命令对应流程如下：
   命令功能描述核心实现步骤start-listen启动TRAP监听1. 创建后台线程；2. 初始化SNMP（单例模式避免重复初始化）；3. 注册TRAP回调函数；4. 启动162端口监听stop-listen停止TRAP监听1. 置位停止标记；2. 等待后台线程退出；3. 释放线程资源getSNMPv2c GET请求1. 初始化SNMP；2. 注册响应回调；3. 创建Agent连接（配置IP、端口、共同体名）；4. 构造系统信息OID（如系统描述、运行时间）；5. 发送请求并等待响应set-led1/0SNMPv1 SET操作1. 构造LED控制OID及值（1=开启触发Inform，0=关闭触发Trap）；2. 用私有共同体名创建Agent；3. 下发SET指令；4. 回调返回操作结果exit退出程序1. 停止TRAP监听；2. 销毁SNMP资源；3. 执行平台清理（如Windows WSACleanup）
3. **关键技术点：开发可复用的核心设计**
   - 单例SNMP初始化：通过`safe_snmp_init`函数确保初始化仅执行一次，避免资源冲突；
   - 回调函数复用：响应回调（处理GET/SET结果）与TRAP回调（接收设备通知）分离，逻辑清晰；
   - 跨平台兼容：通过条件编译适配Windows/Linux/STM32，开发时只需修改平台相关宏定义即可移植。

样例的核心价值在于提供“即拿即用”的框架，开发人员可直接替换OID、Agent配置、回调处理逻辑，快速适配自身设备的管理需求。

- **环境与配置加载**：样例首先加载SNMP协议配置（指定协议版本为v1或v2、共同体名、目标Agent的IP与端口），以及自定义OID的相关参数（如OID标识、数据类型、初始值），为后续操作奠定基础。
- **协议栈快速初始化**：调用协议栈初始化接口完成协议栈启动，同时初始化日志模块（默认输出至控制台），并校验初始化结果，确保后续接口调用的合法性。
- **自定义OID核心操作**：样例重点实现了两类核心操作——SNMP Get操作（获取目标Agent上自定义OID对应的设备数据，如传感器采集值）与SNMP Set操作（向自定义OID下发配置指令，如控制设备执行开关动作），并将操作结果（成功/失败信息、返回数据）打印输出。
- **资源清理与退出**：操作完成后调用协议栈资源释放接口释放资源，避免内存泄漏，为上层APP提供规范的资源管理示范。

该样例代码结构清晰，可作为开发模板，通过修改配置参数、扩展OID操作逻辑，快速适配自身的Agent管理场景。需要特别说明的是，snmp_api_demo.c仅展示了接口交互的基础模式，NMS SDK的接口能力远不止于此——上层用户APP可基于现有接口组合，灵活构建更复杂的专业级应用，例如实现类似iReasoning MIB Browser的综合性网络管理软件。

## 4.3 基于demo拓展：实现专业级MIB管理软件的核心思路

iReasoning MIB Browser的核心价值在于“MIB树可视化+多维度设备管控+协议兼容性”，这些功能均可通过nms_snmp_api.h接口与demo基础框架结合实现，具体拓展方向与接口调用逻辑如下：

- **MIB树解析与可视化呈现**：demo已实现OID基础操作，在此基础上可通过`api_snmp_mib_load`接口加载标准MIB文件（如RFC1213-MIB），解析MIB树的层级结构与节点属性（名称、数据类型、读写权限），结合GUI框架（如Qt、MFC）将OID节点组织为树形结构展示，实现“点击节点自动填充OID”“节点属性悬停提示”等便捷功能；
- **多协议版本与批量操作增强**：demo以v1/v2c为基础，可扩展集成v3版本能力（联系芯祥联科技获取授权），通过`api_snmp_user_add`接口配置v3用户的认证/加密参数；针对批量管理需求，基于demo的Agent管理逻辑，通过循环调用`api_snmp_agent_create`批量添加设备，结合`api_snmp_get_bulk`接口实现多设备多OID的批量数据采集，效率优于单设备循环查询；
- **高级功能拓展**：参考iReasoning的特色功能，可通过接口组合实现：①Trap消息的过滤与告警（注册Trap回调后，按OID、设备IP、消息类型设置过滤规则）；②MIB节点的读写权限校验（结合用户角色权限，调用接口前判断是否允许执行SET操作）；③数据导出与报表生成（将`api_snmp_get`获取的设备数据按CSV/Excel格式封装，通过文件操作接口输出）；
- **界面与交互优化**：基于demo的“命令交互+后台监听”架构，扩展为“可视化操作面板+实时数据面板+告警日志面板”三区域界面，通过多线程封装接口调用逻辑——前台GUI负责用户操作输入，后台线程处理SNMP请求与Trap接收，避免界面卡顿，提升用户体验。

上述拓展场景的核心逻辑，均以demo验证的“初始化-接口调用-回调处理-资源释放”为基础，仅需结合业务需求扩展接口调用组合与上层交互逻辑，即可实现从“基础工具”到“专业软件”的升级。

## 5. NMS SDK与Agent SDK配套实现网络设备统一管理

NMS SDK（网络管理端开发套件）与芯祥联科技自主研发的Agent SDK（设备代理端开发套件）为配套产品，两者协同工作可构建完整的SNMP管理闭环，实现对各类网络设备的简单、统一化管理，是芯祥联科技面向网络设备管理场景的核心解决方案。

### 5.1 配套工作核心逻辑

Agent SDK部署于被管理的网络设备端（如路由器、交换机、工业控制器等），负责采集设备硬件状态、业务数据并转换为SNMP可识别的OID节点；NMS SDK部署于管理端（如监控中心服务器、上位机），通过标准化API接口与各设备的Agent进行通信，完成数据采集、指令下发与状态监控，最终实现“管理端-代理端-设备”的三层统一管控架构。

### 5.2 配套协同的核心优势

- **协议与数据兼容性保障**：两者均由芯祥联科技自主研发，深度适配SNMPv1/v2协议规范，且在数据格式、OID编码规则、通信交互逻辑上完全对齐，可避免第三方套件对接时的兼容性问题，减少开发调试成本。
- **功能互补形成管理闭环**：Agent SDK提供设备端数据采集、OID注册、协议响应等核心能力，NMS SDK提供管理端多设备接入、批量管控、数据聚合等能力，两者结合实现“数据采集-传输-管控-反馈”的全流程覆盖，满足网络设备管理的完整需求。

- **支持自定义扩展与标准化管理平衡**：Agent SDK支持设备自定义OID开发，可适配不同厂商、不同型号网络设备的个性化数据采集需求；NMS SDK则通过统一API将分散的设备数据聚合，提供标准化的管理视图，实现“设备个性化适配”与“管理端统一管控”的平衡。

### 5.3 统一管理实现步骤

1. **设备端部署Agent SDK**：在目标网络设备上集成Agent SDK，开发硬件适配层对接设备接口（如读取CPU、内存、端口流量等数据），注册标准及自定义OID节点，完成Agent初始化与网络配置（指定管理端IP、通信端口等）。
2. **管理端集成NMS SDK**：在管理端应用中集成NMS SDK，通过API接口配置Agent接入信息（设备IP、SNMP版本、共同体名等），建立与各设备Agent的通信链路。
3. **实现统一管控功能**：基于NMS SDK接口开发统一管理模块，包括：批量获取各设备的标准OID数据（如系统信息、接口状态）与自定义OID数据（如设备专属运行参数）；通过SNMP Set指令向Agent下发配置指令，实现设备参数远程调整；实时监控Agent上报的设备状态，异常时触发告警机制。
4. **数据聚合与可视化呈现**：通过NMS SDK将多设备数据汇总，结合上层APP开发统计分析与可视化模块（如设备状态仪表盘、数据趋势图），实现网络设备管理的直观化与高效化。

通过上述配套方案，可快速构建轻量、高效的网络设备统一管理系统，适用于中小型网络环境、工业控制网络等各类场景，降低多设备管理的复杂度。

## 五、接口调用注意事项

- **接口调用顺序规范**：必须先执行协议栈初始化，再调用其他功能接口；协议栈资源释放接口需在所有功能接口调用结束后执行，避免出现资源访问异常。
- **错误处理机制**：所有接口调用均返回状态码及错误信息，上层APP需强制处理这些返回结果（如初始化失败时提示用户检查库依赖或端口占用），避免忽略错误导致程序崩溃。
- **平台兼容性限制**：开源版接口依赖的NMS SDK协议LIB仅支持Windows平台，若上层APP需部署至Linux、嵌入式等其他硬件平台，建议先在Windows环境完成业务软件的开发调试，实现基本逻辑功能，后续可联系公司获取NMS SDK其他平台服务后，再进行进一步的开发调试与优化。
- **第三方对接建议**：接口支持与第三方SNMPv1/v2设备对接，但为保障兼容性与功能完整性，建议优先搭配芯祥联科技的Agent SDK使用，减少协议适配成本。
- **资源占用控制**：管理大量Agent（如数百台设备）时，需通过接口合理配置并发数与超时时间，避免因频繁请求导致CPU或网络资源占用过高，影响APP整体性能。
- **进阶需求合作**：若存在SNMPV3协议支持、TRAP/INFORM消息处理、安全算法集成及其他硬件平台移植等进阶需求，可联系芯祥联科技进一步洽谈合作。

## 六、总结

nms_snmp_api.h作为NMS SDK的核心对外接口，为上层APP提供了“低门槛、高灵活”的SNMP协议栈调用能力。开发人员通过遵循“初始化-功能调用-资源释放”的标准流程，结合接口封装的多Agent管理、数据交互等核心能力，可快速实现符合自身业务场景的上层应用。对于其他硬件平台的开发，建议先在Windows环境完成业务软件的基础逻辑开发与调试，再对接芯祥联科技的平台适配服务。若存在SNMPV3、TRAP/INFORM、安全算法等进阶需求，也可随时通过合作邮箱（hezuo@xxltech.cn）或官网（www.xxltech.cn）联系，获取专业技术支持与合作方案。

（注：文档部分内容可能由 AI 生成）