北向接口文档v2

|  |  |  |
| --- | --- | --- |
| **时间** | **编撰人** | **编撰说明** |
| 2023-2-16 | 徐泽威 | 初稿 |
| 2023-3-6 | 徐泽威 | 按评审建议修改 |
| 2023-3-13 | 徐泽威 | 按二次评审建议修改 |
| 2023-3-23 | 徐泽威 | 按三次评审建议修改 |

# 概述

从数据分类来看，北向接口大致分为四大数据：资产、告警、配置、日志。其中，配置的接口占比较多，日志接口较少，资产和告警的接口数量中等，因为除了资产、告警、日志，剩下的大部分接口几乎都可以理解为配置接口。配置涉及数据源、组织架构、网段、策略中心、系统配置等大的功能项，以及数据面板配置、资产查询条件等小的功能点。

# 接口URI命名规则

从整体看，首先注意大小写问题，统一使用小驼峰，其次注意版本化。格式如下。

|  |
| --- |
| /north/{接口分组(名词)}/{版本}/{接口操作(动词)} |

正确示例如下。

|  |
| --- |
| /north/alarm/v1/get  /north/alarm/v2/queryPagedList  /north/asset/resource/basicInfo/v1/getPagedList  /north/asset/v2/queryPagedList  /north/support/v2/auth |

错误示例如下。

|  |
| --- |
| /north/alarm/V1/get 不能用大写的V1  /north/alarm/v2 缺少接口操作（动词）  /north/asset/resource/basicInfo/v1/GetPagedList 不能用大驼峰命名法  /north/asset/resource/basic\_info/v2/getPagedList 不能用下划线命名法  /north/asset/resource/basic-info/v2/getPagedList 不能用连字符命名法  /support/v2/auth 没有添加统一的/north前缀 |

从细节看，“接口分组”等于接口的命名空间，用来区分接口的层级结构，支持三级结构，细节如下。

|  |
| --- |
| /north/{一级接口分组(名词)}/{二级接口分组(名词)}/{三级接口分组(名词)}/{版本}/{接口操作(动词)} |

有asset（资产）、alarm（告警）、log（日志）、config（配置）、support（支撑）等5个一级接口分组。

# 状态码

状态码有两种，一是HTTP响应状态码，二是业务状态码，两者是递进关系。前者是标准的HTTP响应状态码，200表示成功，其他均为失败。后者是自定义的业务状态码，仅在前者成功时有意义。也就是说，只有HTTP响应状态码等于200，才需要进一步检查业务状态码，如果HTTP响应状态码不等于200，不必进一步检查业务状态码即可确认本次请求已经失败。

# 分页排序

v2版定义了pageIndex、pageSize、pageSorts这3个明确、无歧义的header请求头通用参数，页码从0开始，支持多字段排序。

从总体看，接口的请求参数有三个位置，一是当做URI的一部分，放在query-string中，常见于GET类型的请求；二是放在body请求体中，常见于POST类型的请求；三是放到header请求头中，所有类型的请求都会用到。从实际看，前两者互斥二选一，GET类型的请求参数有两个位置，query-string和header请求头；POST类型的请求参数有两个位置，body请求体和header请求头。

注意header请求头只能放置token鉴权、timestamp时间戳、以及分页排序等几乎所有接口都会用到的通用参数，不能放置业务参数，防止header请求头的参数数量无限制膨胀。

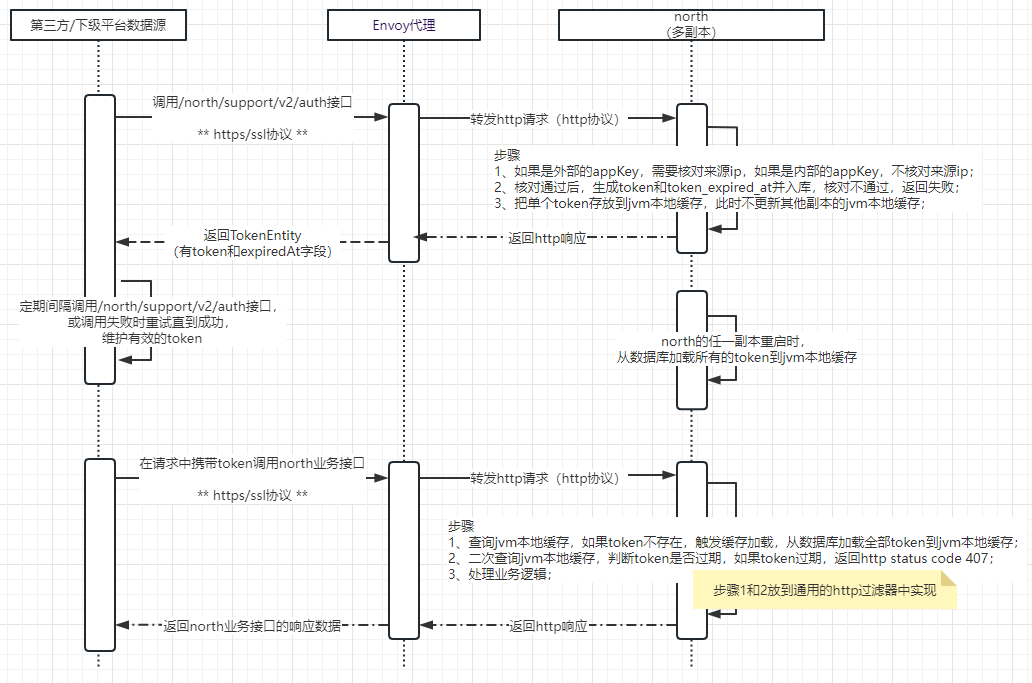
|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **描述** | **是否必选** | **示例** |
| pageIndex | 页码，从0开始递增，0表示第一页（优先级最高，覆盖query-string或body中的分页参数） | 否 | 0 |
| pageSize | 分页大小，每页最大条数（优先级最高，覆盖query-string或body中的分页参数） | 否 | 50 |
| pageSorts | 分页排序，用英文逗号分隔多个排序条件，前导+表示正序，前导+可省略，前导-表示倒序（优先级最高，覆盖query-string或body中的排序参数） | 否 | ip,+groupCode,-name |

# 限流限速

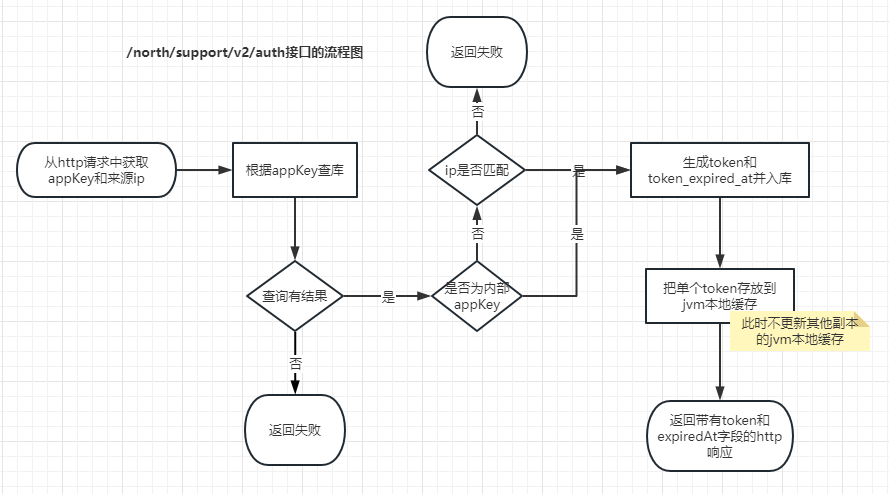
限制北向接口的所有的接口的整体流量，不区分接口。使用Envoy实现。

# 验证授权

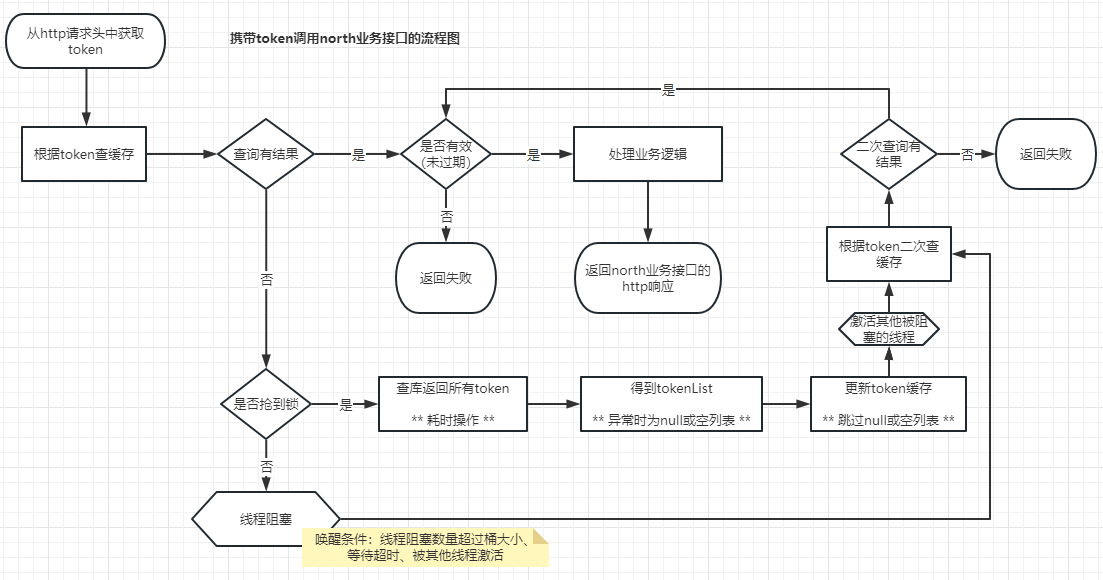
时序图



/north/support/v2/auth接口的流程图



携带token调用north业务接口的流程图



建表sql语句

|  |
| --- |
| CREATE TABLE `north\_port\_peer` (  `id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT '自增id',  `peer\_ip` VARCHAR(15) NULL DEFAULT NULL COMMENT '对端IP地址' COLLATE 'utf8mb4\_bin',  `app\_key` VARCHAR(64) NOT NULL COMMENT '平台唯一标识,appKey和identificationKey统一称为appKey,通过appKeyType区分外部和内部' COLLATE 'utf8mb4\_bin',  `app\_key\_type` CHAR(1) NOT NULL DEFAULT '0' COMMENT '平台唯一标识的类型,0外部1内部',  `desc` VARCHAR(128) NULL DEFAULT NULL COMMENT '备注信息' COLLATE 'utf8mb4\_bin',  `enable` CHAR(1) NOT NULL DEFAULT '0' COMMENT '是否启用,0禁用1启用',  `token` VARCHAR(64) NULL DEFAULT NULL COMMENT 'token,每个ip只有一个token' COLLATE 'utf8mb4\_bin',  `token\_expired\_at` BIGINT NULL DEFAULT NULL COMMENT 'token过期时间戳',  `create\_time` datetime DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',  `update\_time` datetime DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT '更新时间',  PRIMARY KEY (`id`),  UNIQUE INDEX `index\_token` (`token`) ) COMMENT='北向接口的对端信息' COLLATE='utf8mb4\_bin' ENGINE=InnoDB;  外部appKey需要在平台的web页面申请，内部appKey只有一个，不需申请，提前预置在一级和二级的数据库中 |

示例java代码

|  |
| --- |
| public class Token {     String token;     long expiredAt;     String peerIp;     String identificationKey;     // getter }  public class TokenCacheKit {     /\*      \* jvm本地缓存      \*/     private static final Cache<String, Token> tokenCache = Caffeine.newBuilder()         .initialCapacity(10)//初始大小         .maximumSize(Integer.MAX\_VALUE)//最大数量         .expireAfterWrite(30, TimeUnit.MINUTES)//过期时间         .build();     /\*      \* 加载token的业务函数, 例如从数据库加载      \*/     private static final TokenCacheLoadFunction tokenCacheLoadFunction;     /\*      \* token加载器, 多线程安全, 可能会阻塞线程, 参数为“线程阻塞超时毫秒数”和“允许阻塞的最大线程数”      \*/     private static final TokenCacheLoader tokenCacheLoader = new TokenCacheLoader(10 \* 1000, new Semaphore(50));      public static void put(Token token) {         tokenCache.put(token.getToken(), token);     }      public static void putAll(List<Token> tokenList) {         if (null == tokenList) {             return;         }         tokenList.forEach(x -> tokenCache.put(x.getToken(), x));     }      /\*\*      \* 查询token, 自动加载, 多线程安全      \*/     public static Token get(String token) {         Token tokenObject = tokenCache.get(token);         if (tokenObject != null) {             return tokenObject;         }         tokenCacheLoader.loadAll(tokenCacheLoadFunction);         return tokenCache.get(token);     }      public static void setTokenCacheLoadFunction(TokenCacheLoadFunction tokenCacheLoadFunction) {         TokenCacheKit.tokenCacheLoadFunction = tokenCacheLoadFunction;     }      public static class TokenCacheLoader {         private ReentrantLock lock = new ReentrantLock();         private Condition wait = lock.newCondition();         private long timeoutMillis; // 线程阻塞超时毫秒数         private Semaphore semaphore; // 允许阻塞的最大线程数          public TokenCacheLoader(long timeoutMillis, Semaphore semaphore) {             this.timeoutMillis = timeoutMillis;             this.semaphore = semaphore;         }          public void loadAll(TokenCacheLoadFunction tokenCacheLoadFunction) {             if (lock.tryLock()) {                 try {                     TokenCacheKit.putAll(tokenCacheLoadFunction.loadAll());                 } catch (Exception e) {                     // 忽略                 } finally {                     wait.signalAll();                     lock.unlock();                 }             } else {                 await();             }         }          private void await() {             if (semaphore.tryAcquire()) {                 wait.await(timeoutMillis, TimeUnit.MILLISECONDS);                 semaphore.release();             }         }     } }  @FunctionalInterface public interface TokenCacheLoadFunction {     List<Token> loadAll(); } |

示例Envoy配置

|  |
| --- |
| static\_resources:    listeners:    - name: listener\_https      address:        socket\_address: {address: 0.0.0.0, port\_value: 443}      filter\_chains:      - filters:        - name: envoy.filters.network.http\_connection\_manager          typed\_config:            "@type": type.googleapis.com/envoy.extensions.filters.network.http\_connection\_manager.v3.HttpConnectionManager            codec\_type: AUTO            stat\_prefix: ingress\_http            route\_config:              name: local\_route              virtual\_hosts:              - name: local\_service                domains: ["\*"]                routes:                - match: {prefix: "/north/support/v2/auth"}                  route: {cluster: service\_north}                  typed\_per\_filter\_config:                    envoy.filters.http.local\_ratelimit:                      "@type": type.googleapis.com/envoy.extensions.filters.http.local\_ratelimit.v3.LocalRateLimit                      stat\_prefix: http\_local\_rate\_limiter                      token\_bucket:                        max\_tokens: 10                        tokens\_per\_fill: 1                        fill\_interval: 10m                      filter\_enabled:                        runtime\_key: local\_rate\_limit\_enabled                        default\_value:                          numerator: 100                          denominator: HUNDRED                      filter\_enforced:                        runtime\_key: local\_rate\_limit\_enforced                        default\_value:                          numerator: 100                          denominator: HUNDRED                      response\_headers\_to\_add:                      - append\_action: OVERWRITE\_IF\_EXISTS\_OR\_ADD                        header:                          key: x-local-rate-limit                          value: 'true'                      local\_rate\_limit\_per\_downstream\_connection: 'true'                - match: {prefix: "/"}                  route: {cluster: service\_north}                  typed\_per\_filter\_config:                    envoy.filters.http.bandwidth\_limit:                      "@type": type.googleapis.com/envoy.extensions.filters.http.bandwidth\_limit.v3.BandwidthLimit                      stat\_prefix: bandwidth\_limiter\_custom\_route                      enable\_mode: REQUEST\_AND\_RESPONSE                      limit\_kbps: 500                      fill\_interval: 0.1s            http\_filters:            - name: envoy.filters.http.local\_ratelimit              typed\_config:                "@type": type.googleapis.com/envoy.extensions.filters.http.local\_ratelimit.v3.LocalRateLimit                stat\_prefix: http\_local\_rate\_limiter            - name: envoy.filters.http.bandwidth\_limit              typed\_config:                "@type": type.googleapis.com/envoy.extensions.filters.http.bandwidth\_limit.v3.BandwidthLimit                stat\_prefix: bandwidth\_limiter\_default            - name: envoy.filters.http.router              typed\_config:                "@type": type.googleapis.com/envoy.extensions.filters.http.router.v3.Router        transport\_socket:          name: envoy.transport\_sockets.tls          typed\_config:            "@type": type.googleapis.com/envoy.extensions.transport\_sockets.tls.v3.DownstreamTlsContext            common\_tls\_context:              tls\_certificates:              - certificate\_chain:                  filename: certs/servercert.pem                private\_key:                  filename: certs/serverkey.pem    clusters:    - name: service\_north      type: STRICT\_DNS      lb\_policy: ROUND\_ROBIN      load\_assignment:        cluster\_name: service\_north        endpoints:        - lb\_endpoints:          - endpoint:              address:                socket\_address: {address: north.uap, port\_value: 80}  admin:    address:      socket\_address: {address: 0.0.0.0, port\_value: 9901} |

# 调用方式

请求URI组成结构为{URI-scheme} :// {Endpoint} / {Path} ? {query-string}。

|  |  |
| --- | --- |
| **参数** | **参数描述** |
| URI-scheme | 表示用于传输请求的协议，当前所有API均采用HTTPS协议。 |
| Endpoint | 指定服务端的域名或IP:端口，即统一安全监测与管理系统的登录IP地址与API开放端口，当前端口固定为31000。 |
| Path | API访问路径，从具体API的基本信息中获取，例如“创建查询条件”接口的Path为/north/asset/queryCondition/v2/create。 |
| query-string | 查询参数，是可选部分，并非每个接口都有查询参数。查询参数前需加一个?，形式为“参数名=参数取值”，例如?param1=val1&param2=val2，表示传递了param1和param2这两个参数。 |

例如您需要新增一个快捷查询条件，假定当前系统的IP地址为196.168.1.1，在接口文档中您可以查看其Path为/north/asset/queryCondition/v2/create，URI拼接后如下所示。

|  |
| --- |
| https://196.168.1.1:31000/north/asset/queryCondition/v2/create |

由于URI-scheme固定为https，同一个系统的Endpoint相同，后文每个具体接口将只提供Path和Method部分。

请求消息头：

附加请求头字段，主要用于指定的URI和HTTP方法等所要求的字段，例如定义消息体类型的请求头“Content-Type”等。详细的公共请求消息头字段汇总如下。

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **描述** | **是否必选** | **示例** |
| Content-Type | 消息体的类型（格式）。POST固定使用application/json，GET无请求体，无需设置此请求头，有其他取值时将会在具体接口中说明。 | 参考接口详情 | application/json |
| appKey | 在统一安全监测与管理系统控制台创建北向接口时分配的唯一授权凭证。 | 参考接口详情 | 以下仅为appKey示例片段  78a91eb443…924a67f3d5 |
| token | v2接口的验证方式，先调用/north/support/v2/auth接口,通过appKey获取动态的token,后续调用其他v2接口均带上token。 | 参考接口详情 |  |
| timestamp | 13位时间戳。 | 参考接口详情 | 1675043839423 |
| pageIndex | 页码，从0开始递增，0表示第一页（优先级最高，覆盖query-string或body中的分页参数） | 参考接口详情 | 0 |
| pageSize | 分页大小，每页最大条数（优先级最高，覆盖query-string或body中的分页参数） | 参考接口详情 | 50 |
| pageSorts | 分页排序，用英文逗号分隔多个排序条件，前导+表示正序，前导+可省略，前导-表示倒序（优先级最高，覆盖query-string或body中的排序参数） | 参考接口详情 | ip,+groupCode,-name |

请求消息体：

请求消息体为可选项，请求消息体通常以结构化格式发出，与请求消息头中Content-type对应，传递除请求消息头之外的内容。若请求消息体中参数支持中文，则中文字符必须为UTF-8编码。

每个接口的请求消息体内容不同，且并非每个接口都需要有请求消息体（或消息体为空），例如GET操作类型的接口不需要消息体，消息体具体内容需要根据具体接口而定。

返回数据：

请求成功发送后，您会收到返回的响应数据，包含响应消息头和响应消息体。响应消息头中包含标准的HTTP响应状态码，200表示成功，其他表示失败。响应消息体中包含业务状态码和业务详情。

HTTP状态码：

是标准的HTTP响应状态码，200表示成功，其他表示失败。

业务状态码：

业务状态码是一组数字代码，业务状态码表示请求响应的状态，0表示成功，其他表示失败，失败详情请参见错误码。

业务详情：

每个接口的返回详情不同，具体信息将在对应接口中介绍。正确详情示例（有code字段，且值为0）如下。

|  |
| --- |
| {    "code": 0,    "message": "查询成功",    "result": [      {        "assetId": "5cf8d52a948c44deaf578ca9a4d4838d",        "brand": "华为",        "deviceType": "交换机",        "group": "研发部门",        "groupCode": "00000000",        "interfaceName": "Ten-GigabitEthernet1/0/36",        "ip": "172.18.5.84",        "mac": "00:50:56:aa:b3:33",        "onOffline": "0",        "switchIp": "172.18.100.1"      }    ]  } |

错误详情（无code字段，或值不为0）示例如下。

|  |
| --- |
| {    "code": 10007,    "message": "参数非法"  } |
| {    "timestamp": 1662621549876,    "status": 400,    "error": "Bad Request",    "message": "Required request body is missing: public java.lang.String org.fzgang.northboundinterface.modules.asset.controller.AssetController.getPageList(org.fzgang.northboundinterface.modules.asset.entity.SourceBasicInfoEntity)",    "path": "/north/asset/resource/basicInfo/v1/getPagedList"  } |

# 接口列表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **{{apiDefinitions}}接口名称**  **(描述)** | **Path** | **方法** |
| [ext.number] | **[name]**  接口描述：[comment] | [ext.uri] | [ext.method] |

# 接口详情

{{?apiDefinitions}}

## {{ext.number}}. {{name}}

##### 基本信息

**Path：**{{ext.uri}}

**Method：**{{ext.method}}

**接口描述：**{{comment}}

##### 请求参数

**Headers**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **{{requestHeaderDefinition.fieldDefinitions}}描述** | **是否必选** | **示例** |
| [name] | [comment] | [required] | [example] |

**Query**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **{{requestDefinition.fieldDefinitions}}描述** | **是否必选** | **示例** |
| [name] | [comment] | [required] | [example] |

##### 返回数据

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **类型** | **是否必选** | **{{responseDefinition.fieldDefinitions}}备注** |
| [name] | [javaType] | [required] | [comment] |

{{/apiDefinitions}}

# 实体定义

{{?modelDefinitions}}

{{ext.number}}. {{name}}

*{{comment}}*

|  |  |  |
| --- | --- | --- |
| **字段名称 & 类型** | **是否必选** | **{{fieldDefinitions}}备注** |
| **[name]** [javaType] | [required] | [comment] |

{{/modelDefinitions}}

# 错误码

错误码按范围区分为：系统级50000~50999，业务级51000~51999

|  |  |
| --- | --- |
| **错误码** | **含义说明** |
| 0 | 成功 |
| 50000 | 系统错误 |
| 50001 | 接口不存在 |
| 50002 | 缺少必须的参数 |
| 50003 | 非法的参数 |
| 50004 | 数据库错误 |
| 51000 | 业务错误 |
| 51001 | 用户名或密码错误 |