# API GUIDE

# Table of contents

API 개요 > API 호출 및 인증 > 기본 API

# Ncloud API

Classic/VPC 환경에서 이용 가능합니다 .

NAVER Cloud Platform API는 RESTful API 방식으로 제공되며, JSON 및 XML 형식으로 응답합니다. 액션에 따라 파라미터 값을 입력하고 등록, 수정, 삭제, 조회할 수 있으며, 서비스 및 운영 도구 자동화에 활용할 수 있습니다. HTTP 방식으로 메서드를 호출하며, 호출 실패 시 오류 코드와 메시지를 반환합니다. Ncloud API에는 호환 API와 연동 API를 제외한 네이버 클라우드 플랫폼의 모든 API가 해당됩니다.

> 참고
>
> Ncloud API의 일부 내용은 호환 API와 연동 API에도 해당되므로 서비스별 API 가이드를 참조하여 사용해 주십시오.

# API 인증키

네이버 클라우드 플랫폼 API는 권한을 가진 사용자만 호출할 수 있도록 사용자 식별 도구인 API 인증키를 `Access key` 와 `Secret key` 한 쌍으로 구성하여 계정별로 발급하고 있습니다. 이는 API 호출을 위한 인증 시 전달되는 파라미터로 사용합니다. 따라서 NAVER Cloud Platform API를 사용하려면 우선 인증키를 생성해야 합니다.
인증키 생성과 관리는 네이버 클라우드 플랫폼의 포털에서 진행합니다. 네이버 클라우드 플랫폼 포털에서 인증키를 생성하고 관리하는 방법은 다음과 같습니다.



1. 네이버 클라우드 플랫폼 포털에 로그인해 주십시오.
2. 마이페이지 > 계정관리 > 인증키관리 메뉴를 차례대로 클릭해 주십시오.
3. **API** 인증키 관리에서 기본 제공되는 **Access key ID**와 **Secret key**를 확인해 주십시오.
4. 필요한 작업을 진행해 주십시오.
    - API 인증키 추가 발급
        - **[**신규 **API** 인증키 생성**]** 버튼 클릭
    - API 인증키 사용 중지
        - **[**사용 중지**]** 버튼 클릭
        - 사용 중지한 인증키를 다시 사용하려면 **[**사용**]** 버튼 클릭
    - API 인증키 삭제
        - **[**삭제**]** 버튼 클릭
        - 사용 중지한 인증키만 삭제 가능

참고

- NAVER Cloud Platform API 인증키는 네이버 클라우드 플랫폼 계정 생성 시 자동으로 1개가 생성됩니다. 자동 생성된 인증키 외에 사용자가 포털에서 직접 생성할 수도 있기 때문에 사용자당 최대 2개까지 인증키를 발급받을 수 있습니다.
- 인증키를 사용 중지하거나 삭제하면 유효하지 않은 키로 인식되어 더 이상 인증에 사용할 수 없습니다.

# API 공통 설정

Ncloud API에서 공통으로 사용하는 요청 형식과 응답 형식을 설명합니다.

## 요청

공통 요청 형식을 설명합니다.

### API URL

요청 API URL은 서비스별 API 가이드에서 확인해 주십시오.

### 시그니처 생성



시그니처는 공통 요청 헤더 `x-ncp-apigw-signature-v2` 필드에 들어가는 값으로, 요청 정보를 API 인증키( `AccessKey` 에 맵핑되는 `SecretKey` )로 암호화한 후 Base64로 인코딩하여 생성하는 서명값입니다.
암호화에는 HMAC 암호화 알고리즘(HmacSHA256)이 사용됩니다.
시그니처를 생성하는 언어별 예제 코드는 다음과 같습니다.

Java Copy

```java
public String makeSignature() {
    String space = " ";                 // one space
    String newLine = "\n";                  // new line
    String method = "GET";                  // method
    String url = "/photos/puppy.jpg?query1=&query2";    // url
(include query string)
    String timestamp = "{timestamp}";           // current timestamp
(epoch)
    String accessKey = "{accessKey}";           // access key id
(from portal or Sub Account)
    String secretKey = "{secretKey}";

    String message = new StringBuilder()
        .append(method)
        .append(space)
        .append(url)
        .append(newLine)
        .append(timestamp)
        .append(newLine)
        .append(accessKey)
        .toString();

    SecretKeySpec signingKey = new
SecretKeySpec(secretKey.getBytes("UTF-8"), "HmacSHA256");
    Mac mac = Mac.getInstance("HmacSHA256");
 mac.init(signingKey);

    byte[] rawHmac = mac.doFinal(message.getBytes("UTF-8"));
    String encodeBase64String = Base64.encodeBase64String(rawHmac);

  return encodeBase64String;
}
```



JavaScript Copy

```javascript
/*
https://code.google.com/archive/p/crypto-js/
https://storage.googleapis.com/google-code-archive-
downloads/v2/code.google.com/crypto-js/CryptoJS%20v3.1.2.zip
*/

/*
```

```
CryptoJS v3.1.2
code.google.com/p/crypto-js
(c) 2009-2013 by Jeff Mott. All rights reserved.
code.google.com/p/crypto-js/wiki/License
*/
<script type="text/javascript" src="./CryptoJS/rollups/hmac-
sha256.js"></script>
<script type="text/javascript" src="./CryptoJS/components/enc-
base64.js"></script>

function makeSignature() {
    var space = " ";                // one space
    var newLine = "\n";             // new line
    var method = "GET";             // method
    var url = "/photos/puppy.jpg?query1=&query2";   // url (include
query string)
    var timestamp = "{timestamp}";          // current timestamp
(epoch)
    var accessKey = "{accessKey}";          // access key id (from
portal or Sub Account)
    var secretKey = "{secretKey}";          // secret key (from
portal or Sub Account)

    var hmac = CryptoJS.algo.HMAC.create(CryptoJS.algo.SHA256,
secretKey);
 hmac.update(method);
 hmac.update(space);
 hmac.update(url);
 hmac.update(newLine);
 hmac.update(timestamp);
 hmac.update(newLine);
 hmac.update(accessKey);

    var hash = hmac.finalize();

    return hash.toString(CryptoJS.enc.Base64);
}
```

Python | Copy

```python
import sys
import os
import hashlib
import hmac
import base64
import requests
import time

def make_signature():
 timestamp = int(time.time() * 1000)
 timestamp = str(timestamp)

 access_key = "{accessKey}"             # access key id (from portal
or Sub Account)
 secret_key = "{secretKey}"             # secret key (from portal or
Sub Account)
 secret_key = bytes(secret_key, 'UTF-8')

 method = "GET"
 uri = "/photos/puppy.jpg?query1=&query2"

 message = method + " " + uri + "\n" + timestamp + "\n"
    + access_key
 message = bytes(message, 'UTF-8')
 signingKey = base64.b64encode(hmac.new(secret_key, message,
digestmod=hashlib.sha256).digest())
    return signingKey
```

Bash | Copy

```bash
function makeSignature() {
 nl=$'\\n'

 TIMESTAMP=$(echo $(($(date +%s%N)/1000000)))
 ACCESSKEY="{accessKey}"                # access key id (from portal or Sub Account)
 SECRETKEY="{secretKey}"                # secret key (from portal or Sub Account)

 METHOD="GET"
 URI="/photos/puppy.jpg?query1=&query2"

 SIG="$METHOD"' '"$URI"${nl}
 SIG+="$TIMESTAMP"${nl}
 SIG+="$ACCESSKEY"

 SIGNATURE=$(echo -n -e "$SIG"|iconv -t utf8 |openssl dgst -sha256 -hmac $SECRETKEY -binary|openssl enc -base64)
}
```



## 요청 헤더

NAVER Cloud Platform API의 공통 헤더에는 인증과 관련된 필드가 포함됩니다. 요청 헤더에 대한 설명은 다음과 같습니다.

> 참고
>
> 서비스별 API의 헤더 구성은 상이할 수 있으므로 서비스별 API 가이드를 참조해 주십시오.

| 필드 | 필수 여부 | 설명 |
|---|---|---|
| `x-ncp-apigw-timestamp` | Required | 1970년 1월 1일 00:00:00 협정 세계시(UTC)부터의 경과 시간 (밀리초)<br>API Gateway 서버와 5분 이상 시간차 발생 시 유효하지 않은 요청으로 간주 |
| `x-ncp-iam-access-key` | Required | 네이버 클라우드 플랫폼에서 발급받은 Access Key<br>Access Key 발급 및 확인: 인증키 생성 참조<br>서브 계정의 Access Key 발급 및 확인: 서브 계정 생성 참조 |
| `x-ncp-apigw-signature-v2` | Required | 네이버 클라우드 플랫폼에서 발급받은 Access Key에 맵핑되는 Secret Key 및 HMAC 암호화 알고리즘(HmacSHA256)으로 요청 정보를 암호화한 후 Base64로 인코딩한 서명<br>Secret Key 발급 및 확인: 인증키 생성 참조<br>서명 생성: 시그니처 생성 참조 |

주의

`x-ncp-apigw-timestamp` 값은 시그니처 생성 시 입력한 timestamp 값과 반드시 동일한 값이어야 합니다.

참고

NAVER Cloud Platform API의 V1 버전에서는 계정별 Key를 발급받아 인증에 활용하는 API Key 인증 방식을 사용했습니다. 그렇기 때문에 API 호출 시 API Key를 헤더에 반드시 포함해야 했지만 2022년 1월 1일부터 V1 버전 서비스를 종료하고 API Key 적용이 필요 없는 V2 버전을 제공하고 있습니다. V2 버전은 오퍼레이션 및 기본 호출 방식이 V1 버전과 동일하지만 API 호출 시 API Key를 헤더에 포함시킬 필요가 없어 API Gateway 이용 신청이 필요하지 않아 이용이 간편하다는 장점이 있습니다. 단, 일부 서비스에서는 API Key를 통한 인증 방식이 반드시 필요한 경우가 있으므로 서비스별 API 가이드에서 API Key의 필수 적용 여부를 확인한 후 사용해 주십시오.

- API Key는 네이버 클라우드 플랫폼의 API Gateway에서 생성할 수 있습니다. 생성 방법은 **API Gateway** 사용 가이드를 참조해 주십시오.
  - API Gateway에서 API key 발급하는 것에 대해서는 별도의 과금은 발생하지 않습니다.

## 요청 예시



인증키와 공통 헤더를 이용한 요청 예시는 다음과 같습니다.

Bash | Copy

```
curl --location --request GET
'https://example.apigw.ntruss.com/photos/puppy.jpg?query1=&query2' \
--header 'x-ncp-apigw-timestamp: {Timestamp}' \
--header 'x-ncp-iam-access-key: {Access Key}' \
--header 'x-ncp-apigw-signature-v2: {API Gateway Signature}' \
```

# 응답

공통 응답 형식을 설명합니다.

참고

서비스별 API의 응답 형식은 상이할 수 있으므로 서비스별 API 가이드를 참조해 주십시오.

## 응답 상태 코드

응답 상태 코드에 대한 설명은 다음과 같습니다.

| HTTP 상태 코드 | 코드 | 메시지 | 설명 |
|---|---|---|---|
| 200 | - | OK | 요청 성공 |
| 201 | - | Created | 요청의 결과로 새로운 리소스가 생성됨 |
| 202 | - | Accepted | 요청이 수락되었지만 아직 완료되지 않았음 |
| 400 | 100 | Bad Request Exception | protocol(https), endocing(UTF-8) 등 Request 오류 |
| 401 | 200 | Authentication Failed | 인증 실패 |
| 401 | 210 | Permission Denied | 권한 없음 |
| 404 | 300 | Not Found Exception | 권한 없음 |
| 429 | 400 | Quota Exceeded | Quota 초과 |
| 429 | 410 | Throttle Limited | Rate 초과 |
| 429 | 420 | Rate Limited | Rate 초과 |
| 413 | 430 | Request Entity Too Large | 요청 엔티티 크기 초과 |
| 503 | 500 | Endpoint Error | 엔드포인트 연결 오류 |
| 504 | 510 | Endpoint Timeout | 엔드포인트 연결 시간 초과 |
| 500 | 900 | Unexpected Error | 예외 처리가 안된 오류 |

## 응답 예시

응답 예시는 다음과 같습니다.



- 성공( `Content-type: application/json` )

JSON Copy

```
{
  "status": {
    "code": "20000",
    "message": "OK"
  },
  "result": {...}
}
```

- 성공( `Content-type: application/xml` )

XML Copy

```
<Message>
   <status>
       <code>20000</code>
       <message>OK</message>
   </status>
   <result> ... </result>
</Message>
```

- 실패( `Content-type: application/json` )

```JSON
{
 "error":{
    "errorCode":"210",
    "message":"Permission Denied"
  }
 }
```

- 실패( `Content-type: application/xml` )

```XML
<?xml version='1.0' encoding='UTF-8' ?>
<Message>
    <error>
     <errorCode>210</errorCode>
     <message>Permission Denied</message>
 </error>
</Message>
```

참고



응답 형식 기본값은 JSON입니다.

# API 보안 설정

API는 인증키가 제3자에게 유출되는 경우, 리소스를 임의로 변경하거나 조회할 수 있어 심각한 보안 문제가 발생할 수 있으므로 적절한 사전 대비와 대응이 필요합니다.

## API 인증키 재발급

API 인증키를 사용하지 않거나 도용이 의심된다면 발급한 인증키를 사용 중지하거나 삭제 후 재발급해 주십시오.

## 서브 계정 인증키 사용

모든 권한을 갖고 있는 메인 계정이 아닌 서브 계정에서 API 인증키를 발급하여 API를 호출하고, 인증키 유출에 대비하여 주기적으로 교체할 수도 있습니다. 서브 계정은 역할에 맞는 권한만 부여 받아 사용할 수 있으며, IP 대역 뿐 아니라 VPC까지 API Access Source의 지정이 가능하기 때문에 보안상 좀 더 안전하게 API를 사용할 수 있습니다.
서브 계정은 네이버 클라우드 플랫폼의 Sub Account 서비스에서 생성합니다. 서브 계정 및 API 인증키 생성 방법은 Sub Account 사용 가이드를 참조해 주십시오.

# API 접근 제한 설정

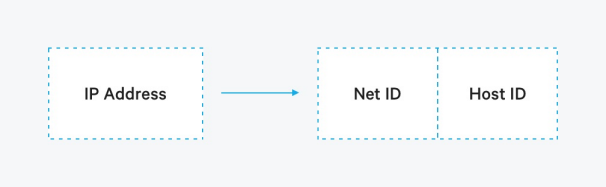
허용되지 않은 위치에서 API Gateway 접속을 방지하기 위해 특정 IP 대역에서만 사용이 가능하도록 설정할 수 있습니다. 특정 IP 대역에서만 API 사용이 가능하도록 설정하는 방법은 다음과 같습니다.

1. 네이버 클라우드 플랫폼 포털에 로그인해 주십시오.
2. 마이페이지 > 계정관리 > 인증키관리 메뉴를 차례대로 클릭해 주십시오.
3. **API** 접근 제한 설정의 **IP Address**에 접근을 허용 IP 대역을 입력해 주십시오.
    - CIDR 표기법을 참고하여 입력해 주십시오.
    - 0.0.0.0~32 대역은 보안상 입력할 수 없습니다.
    - 별도로 등록하지 않으면 모든 접속을 허용합니다.
4. **[**저장**]** 버튼을 클릭해 주십시오.
    - 접근 허용 IP 대역을 추가하려면 **[**추가**]** 버튼을 클릭해 주십시오.

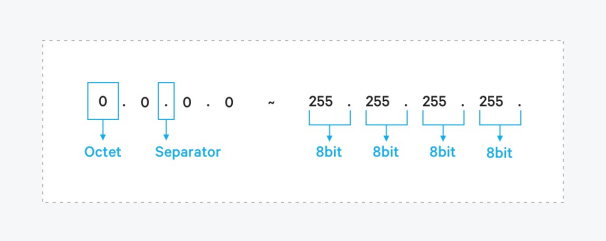
## CIDR 표기법



IP Address는 192.168.0.1 ~ 192.168.0.255 형식으로 접속을 허용할 IP 대역을 표기합니다. 네이버 클라우드 플랫폼은 네트워크를 관리할 때 주로 사용되는 CIDR(Classless Inter-Domain Routing) 표기법을 사용하여 입력하는 방법도 지원합니다.




CIDR은 연속된 IP Address 범위를 표기하는 방법 중 하나입니다. CIDR 표기법은 IP Address를 Net ID(Network ID)와 Host ID로 구분하고, Net ID 하위에 속하는 IP 대역(Host ID 범주)을 그룹화하여 표현할 수 있습니다. 이를 Address Aggregation/Supernetting이라고도 합니다.




일반적으로 사용하는 IP Address는 다음과 같이 4옥텟(Octet)으로 이루어져 있습니다.

| CIDR | Octet | Octet | Octet | Octet | Range | IP Address Band | Hosts |
|---|---|---|---|---|---|---|---|
| 192.168.0.0/24 | 192 | 168 | 0 | 0 | 24 | 192.168.0.0 ~ 192.168.0.255 | 256 |
| 192.168.10.23/30 | 192 | 168 | 10 | 23 | 30 | 192.168.10.20 ~ 192.168.10.23 | 4 |
| 192.168.23.11/32 | 192 | 168 | 23 | 11 | 32 | 192.168.23.11 ~ 192.168.23.11 | 1 |

CIDR 표기법은 IP 대역을 4 Octet으로 이루어진 IP Address와 빗금(/) 뒤에 Range 값 1개를 덧붙여 표기합니다. Range 값은 0~32, 즉 32 비트까지 사용할 수 있습니다. CIDR 표기법에서는 표기된 Range 값 이후에 올 수 있는 비트 수를 사용할 수 있다는 의미로 이해할 수 있습니다.
따라서 IP 대역은 IP Address로 표기된 4 Octet 값에서 Range 값 이후에 올 수 있는 비트 수까지의 범위를 IP 대역으로 계산합니다. 이때 IP Address의 4 Octet 값 ~ Range 값만큼의 비트 수를 고정 영역인 Net ID로 간주하고 나머지 비트를 사용할 수 있는 Host ID 대역으로 계산합니다.

> 참고
>
> Range의 입력 범위는 24~32입니다.

Range 값에 의한 Host의 IP Address 수를 정리하면 다음과 같습니다.

| Range | Class | Hosts |
|---|---|---|
| /32 | 1/256 C | 1 |
| /31 | 1/128 C | 2 |
| /30 | 1/64 C | 4 |
| /29 | 1/32 C | 8 |
| /28 | 1/16 C | 16 |
| /27 | 1/8 C | 32 |
| /26 | 1/4 C | 64 |
| /25 | 1/2 C | 128 |
| /24 | 1 C | 256 |



> 주의
>
> 입력 행을 추가하면 Range는 Default/32 (단일 IP Address)로 설정되며, 수정이 가능해지므로 입력 시 유의해 주십시오.

# 지원 API 및 SDK

네이버 클라우드 플랫폼에서 지원하는 API와 SDK 정보를 안내합니다.

## API

서비스별 지원하는 API 목록은 서비스별 API 가이드에서 확인해 주십시오.

# SDK

서비스별 지원하는 SDK 목록과 다운로드 링크는 다음과 같습니다.

| 서비스 | **SDK** 다운로드 |
|---|---|
| Server | ncloud_server.zip |
| Load Balancer | ncloud_loadbalancer.zip |
| Auto Scaling | ncloud_autoscaling.zip |
| Monitoring | ncloud_monitoring.zip |
| Security | ncloud_security.zip |
| GeoLocation | ncloud_geolocation.zip |
| CDN+ | ncloud_cdn_v2.zip |
| Cloud DB | ncloud_clouddb_v2.zip |
| Cloud Outbound Mailer | 다운로드 |