# API Gateway HMAC 인증 가이드 (for C++)

NHN 웹플랫폼개발랩 / 2013.04.02

## 목적

API Gateway는 NHN 콘텐츠 데이터를 단일화 된 채널로 외부에 제공하기 위하여 인증, 쿼터제한, 통계, 포맷변환 등의 기능을 제공하는 플랫폼이다.

본 문서에서는 API Gateway의 인증 방법 중 하나인 HMAC 인증을 통하여 API를 호출하는 방법에 대한 가이드를 UNIX/Linux와 Windows 버전으로 나누어 제공한다.

## 인증 방법

HMAC 인증을 통해 NHN 서비스 API를 호출하기 위해서는 호출할 URL에 대해 Encrypt된 URL 값을 얻어야 한다. EncryptedUrl은 각 업체에 발급된 제휴업체 Key와 API URL의 조합으로 생성된다.

* 제휴업체 Key: NHN에서 발급한 업체별로 고유한 Key 업체별로 파일로 제공된다.
* API URL: API 수행을 위하여 호출할 HTTP URL

API Gateway는 수신한 URL의 Hash 값을 확인하여 해당 요청이 유효한지 여부를 검사한다.

## 설치

### UNIX/Linux

* 전달 받은 파일의 압축을 해제한다. (제공되는 파일 목록은 **6. 제공 파일** 항목 참고)
* libApiGateway-MAC.so 를 작성 중인 프로그램이 접근 가능한 경로에 저장한다.  
  예) $PROGRAM\_ROOT/lib/libApiGateway-MAC.so
* MACManager.h 를 작성 중인 프로그램이 접근 가능한 경로에 저장한다.  
  예) $PROGRAM\_ROOT/lib/include/MACManager.h
* NHNAPIGatewayKey.properties를 작성 중인 프로그램이 접근 가능한 경로에 저장한다. MACManager::getKeyFromFile()은 기본적으로 $PROGRAM\_ROOT에서 NHNAPIGatewayKey.properties 파일을 읽는다. 만약 NHNAPIGatewayKey.properties 파일이 다른 곳에 있다면, MACManager::getKeyFromFile(const std::string& filePath) 메소드를 사용해야 한다.
* 작성 중인 프로그램 빌드에 libApiGateway-MAC.so 를 링크한다.  
  예) $PROGRAM\_ROOT/lib/libApiGateway-MAC.so 가 있을 경우,   
   **g++ $(OPTIONS) –Wl,-rpath=lib –Llib –lApiGateway-MAC $(SOURCES) –o $(TARGET)**

### Windows

* 전달 받은 파일의 압축을 해제한다. (제공되는 파일 목록은 **6. 제공 파일** 항목 참고)
* MACManager.dll 을 작성 중인 프로그램이 접근 가능한 경로에 저장한다.  
  예) **$(SolutionDir)\DLL\Debug\libApiGateway-MAC.dll   
   $(SolutionDir)\DLL\Release\libApiGateway-MAC.dll**  
  또한, 아래와 같이 프로그램 빌드 시, dll 파일을 실행 파일이 위치한 경로로 복사할 수 있다.  
  예) Project Properties -> Build Events -> Post-Build Event -> Command Line  
   **copy /Y $(SolutionDir)DLL\$(Configuration)\libApiGateway-MAC.dll $(SolutionDir)$(Configuration)\**
* MACManager.h 를 프로젝트 내 헤더 파일이 위치한 경로에 저장한 후, 아래와 같이 프로젝트를 설정한다.  
  예) Project Properties -> C/C++ -> General -> Additional Include Directories : **$(SolutionDir)include**
* libGApiGateway-MAC.lib 를 프로젝트 내 라이브러리 폴더에 저장한 후, 아래와 같이 프로젝트를 설정한다.  
  예) Project Properties -> Linker -> General -> Additional Library Directories : **$(SolutionDir)lib** Project Properties -> Linker -> Input -> Additional Dependencies : **libApiGateway-MAC.lib**

## 사용 방법

### UNIX/Linux, Windows 동일

라이브러리의 사용 방법은 \*NIX 시스템과 Windows 시스템 간의 차이가 거의 없다. 아래의 예제 코드는 \*NIX 시스템에서의 코드를 바탕으로 작성되었으며, Windows 시스템에서도 기본적인 코드는 다르지 않다.  
단, 라이브러리의 위치나 개발 환경에 따라 include 경로나 pragma comment 등의 명령어의 사용이 다를 수 있기 때문에, 아래의 예제를 바탕으로 개발 환경에 맞게 코딩을 해야 한다.

* 호출할 URL을 설정한다. 호출할 URL이란 업체별로 제공된 API URL을 의미한다.

#include “lib/include/MACManager.h”

const std::string URL = “http://dev.apis.naver.com/제휴 업체 ID/hmac/hmactest.xml”;

* Key 값을 설정한다. Key는 업체별로 제공된 NHNAPIGatewayKey.properties 안에 정의된 값을 의미한다.

// 소스에서 key 값을 직접 지정하는 경우

const std::string KEY = “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”;

// 디폴트 경로인 $PROGRAM\_ROOT의 NHNAPIGatewayKey.properties 파일에서 key 값을 읽는 경우

const std::string KEY = MACManager::getKeyFromFile();

// 커스텀 경로인 resources/NHNAPIGatewayKey.properties 파일에서 key 값을 읽는 경우

const std::string filePath = “resources/NHNAPIGatewayKey.properties”;

const std::string KEY = MACManager::getKeyFromFile(filePath);

* 호출할 URL의 encrypt된 값을 얻기 위해 MACManager의 getEncryptUrl(const std::string& url) 메소드를 호출한다.  
  getEncryptUrl 메소드의 url 파라미터는 호출할 URL과, get 방식으로 넘겨줄 파라미터들을 **모두 포함한 상태**여야 하며, getEncryptUrl 메소드를 통해 생성된 encryptedUrl에는 msgpad와 md라는 파라미터가 **추가로** 더 붙게 된다.

// Stack에 객체 생성

MACManager manager(KEY);

std::string encryptedUrl = manager.getEncryptUrl(URL);

// Heap에 객체 생성

MACManager\* pManager = new MACManager(KEY);

std::string encryptedUrl = pManager->getEncryptUrl(URL);

delete pManager;

* 위에 생성된 encryptedUrl을 이용하여 APIGW 서버에 데이터를 요청한다.

## 호출 결과

API 호출 후 HMAC 인증 성공 여부에 따라 아래 형태의 값이 반환된다.

### 인증 성공

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<result>

<authentication>true</authentication>

</result>

### 인증 실패 (예)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<result>

<message>HMAC Authentication failed</message>

<error\_code>022</error\_code>

</result>

### 에러 코드

|  |  |
| --- | --- |
| 에러코드 | 메시지와 설명 |
| 000 | System error  시스템 에러 |
| 010 | Your query request count is over the limit  쿼리요청 제한을 초과하였습니다. 일반적으로는 25,000개 이상의 쿼리에 대하여 이 에러 메시지가 발생되나, 쿼리 제한이 다르게 설정된 경우에는 이에 준하여 발생됩니다. |
| 011 | Incorrect query request  잘못된 쿼리 요청입니다. 쿼리(query=) 필드 자체가 없는 경우 발생하는 에러 메시지 입니다.  ...&query=& 식으로 입력하는 경우 정상적인 쿼리 요청으로 보고, 에러 메시지를 발생하지 않습니다. |
| 020 | Unregistered key  등록되지 않은 키입니다. |
| 021 | Your key is temporary unavailable  사용할 수 없는 키입니다. OpenAPI에 등록되었으나, 일시적으로 사용 중지된 키를 통하여 쿼리를 검색하는 경우 발생하는 에러입니다. |
| 022 | Authentication Fail  인증을 실패했습니다. |
| 030 | UnsupportedFormatException  지원하지 않는 리턴 format입니다. |
| 040 | NoSuchApiException  존재하지 않는 API입니다. |
| 100 | Invalid target value  부적절한 Target 값입니다. 필드 설명에 없는 target 이름을 사용한 경우, 또는 target 필드가 없는 경우에 발생하는 메시지입니다. |
| 101 | Invalid display value  부적절한 display 값입니다. display 필드 자체가 없는 경우에는 정상적인 쿼리 요청으로 보고 에러 메시지를 발생하지 않으며, 기본값인 display=10으로 처리됩니다.  display 값이 100을 초과하는 경우나 0 이하의 값일 때, 혹은 ...&display=& 와 같이 값을 입력하지 않은 경우에는 에러 메시지를 발생합니다. |
| 102 | Invalid start value  부적절한 start 값입니다. start 값이 없는 경우에는 정상적인 쿼리 요청으로 보고 에러 메시지를 발생하지 않으며, 기본값인 start=1으로 처리됩니다. start 값이 1000을 초과하거나 0 이하의 경우, 또 ...&start=&와 같이 값을 입력하지 않은 경우에는 에러 메시지를 발생합니다. |
| 110 | Undefined sort value  정의되지 않은 sort 값입니다. sort 필드 자체가 없는 경우에는 정상적인 쿼리 요청으로 보고 에러메시지를 발생하지 않으며, sort 옵션을 사용하는 서비스 API인 경우 기본 값인 sort=sim으로 처리됩니다.  sort 값이 해당하는 서비스의 정렬 옵션 이외의 값을 가지는 경우, 혹은 ...sort=& 와 같이 값을 입력하지 않은 경우 에러 메시지를 발생합니다.(예, 지식iN 검색 쿼리 요청에서 쇼핑 검색의 정렬 옵션인 sort=asc를 사용하는 경우) |
| 200 | Reserved |
| 900 | Undefined error occurred  정의되지 않은 오류가 발생하였습니다. |

## 제공 파일

### UNIX/Linux

|  |  |
| --- | --- |
| 파일명 | 설명 |
| libApiGateway-MAC.so | 인증을 위해 NHN에서 제공하는 library 파일 |
| MACManager.h | 인증 library에서 제공하는 헤더 파일 |
| NHNAPIGatewayKey.properties | 제휴사별로 발급되는 Key가 저장된 파일 |

### Windows

|  |  |
| --- | --- |
| 파일명 | 설명 |
| libApiGateway-MAC.dll | 인증을 위해 NHN에서 제공하는 library 파일 |
| libApiGateway-MAC.lib | Load-time linking을 위한 link library 파일 |
| MACManager.h | 인증 library에서 제공하는 헤더 파일 |
| NHNAPIGatewayKey.properties | 제휴사별로 발급되는 Key가 저장된 파일 |

## 기타 필요 라이브러리

### UNIX/Linux

OpenSSL

대부분의 \*NIX 시스템에는 이미 컴파일된 OpenSSL 패키지가 설치되어 있다. 만약 시스템에 OpenSSL이 설치되어 있지 않다면 yum install openssl로 설치할 수 있다.

참고: <http://www.openssl.org/>