

# nProtect AppGuard Server-Side Authentication 적용 가이드

AppGuard SDK Developer Guide

(주)잉카인터넷

DV-191104

Ver. 3.0.2

업체 제공용

## 저작권

이 문서는 (주)잉카인터넷의 비밀 자산으로 nProtect AppGuard 를 Mobile Application 에 적용하기 위한 표준 가이드와 관련 정보를 제공하기 위하여 작성되었으며, (주)잉카인터넷의 승인 없이 제 3 자에게 제공하거나 배포할 수 없습니다.



## 최소 요구 사항

AppGuard Module No.	[18010] Platform11.3 Build:a4.4a.1a9 (36.65)
AppGuard SDK	appguard-sdk-r15.jar

## 문서 개요

이 문서는 SDK 형태로 제공되는 nProtect AppGuard for Mobile 의 API 함수 사용법을 설명한 프로그래밍 가이드입니다.

## 문서 목차

1. AppGuard 서버인증 소개 .....	4
1.1 서버인증 시스템 .....	4
1.2 서버인증으로 보호 가능한 공격 유형 .....	4
1.3 서버인증 흐름도 .....	4
2. AppGuard 서버인증 적용 방법 .....	6
2.1. 동기화(Synchronous) 적용 방식 .....	6
2.1.1 동기화 방식의 흐름도 .....	6
2.1.2 동기화 방식의 예상되는 문제점 .....	7
2.2. 비동기화(Asynchronous) 적용 방식 .....	7
2.2.1 비동기화 방식의 흐름도 .....	8
2.2.2 비동기화 방식의 예상되는 문제점 .....	8
2.3 커스터마이징시 유의사항 .....	9
3. 서버인증 세션 식별 아이디 생성 메커니즘 .....	10
3.1 아이디(Unique ID) 생성 규칙 .....	10
3.2 아이디 자동 발급 방법 .....	11

4. 서버인증 적용 상세.....	13
4.1 앱 서버에서 인증용 아이디 생성하기 .....	13
4.2 앱 클라이언트에서 서버인증 요청하기.....	13
4.3 앱 클라이언트에서 서버인증 콜백 적용하기 .....	15
4.4 앱 서버에서 인증 결과 확인하기 .....	17

## 1. AppGuard 서버인증 소개

### 1.1. 서버인증 시스템

AppGuard 서버인증 (Server-Side Authentication) 시스템은 앱 보호를 위하여 모바일 클라이언트 단에서 동작하는 AppGuard 엔진(Engine) 모듈이 앱과 함께 배포되어 정상적으로 실행되었으며 안전하게 초기화 되었음을 인증하는 메커니즘을 제공합니다.

모든 인증은 앱(App or Game) 서버에서 생성한 유일 식별 아이디(Unique Client ID)를 바탕으로 동작합니다.

AppGuard 보안 모듈은 정상적으로 엔진이 초기화되고 필수 보안 검사에서 무결성이 확인된 상태에서만 동작하도록 설계된 암호화 통신 메커니즘을 가지고 있으며, 해당 메커니즘을 이용하여 안전하게 서버인증을 수행합니다.

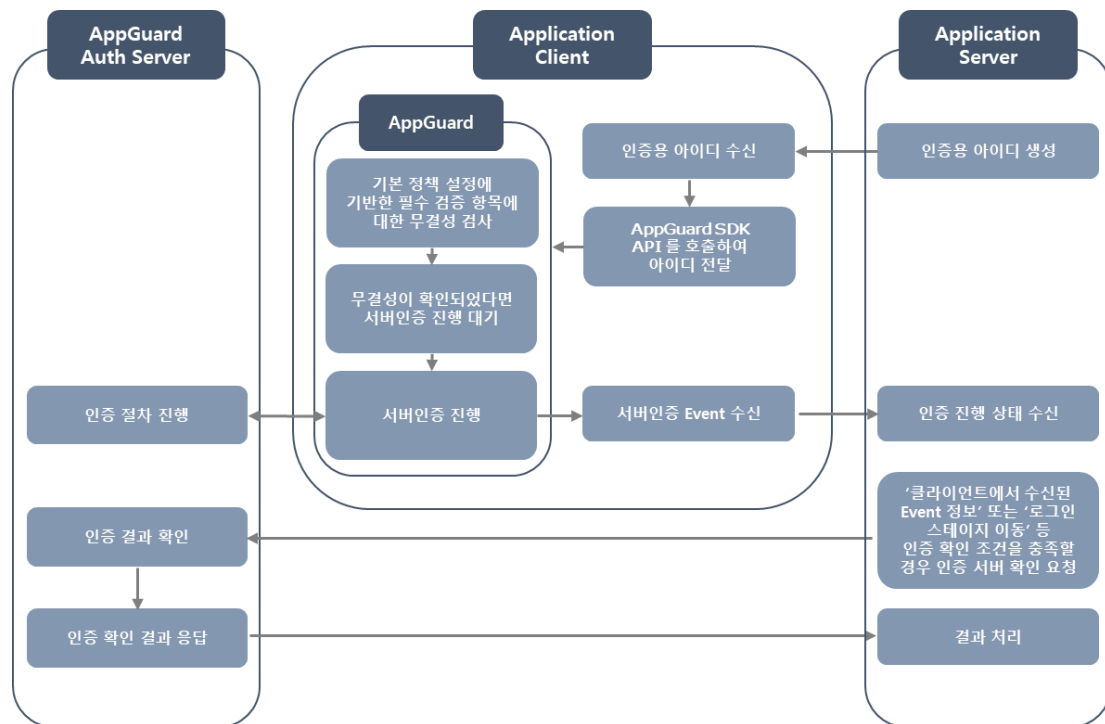
AppGuard 는 앱 클라이언트로부터 전달받은 인증용 아이디를 기반으로 인증 서버에 앱의 기본 무결성 및 보안 엔진의 정상 실행 유무를 인증 받으며, 앱 서버는 인증 서버로 자신이 발급한 인증용 아이디로 해당 아이디에 대한 AppGuard 보안 엔진의 정상 동작을 확인하는 행위를 통하여 해당 클라이언트의 정상 동작 유무를 판단합니다.

### 1.2. 서버인증으로 보호 가능한 공격 유형

AppGuard 서버인증은 기본적으로 AppGuard 보안 시스템을 구성하는 보안 모듈이 정상적으로 메모리에 적재되고 실행되었는지 유무를 검사하는 시스템으로서 MOD(Modified)와 같이 보안 모듈을 제거 또는 무력화 시키고 원본 실행 파일 및 데이터를 추출하여 재패키징(Re-Packing)하여 배포하는 류의 공격에 대응하기 위하여 사용됩니다.

### 1.3. 서버인증 흐름도

다음은 AppGuard 와 함께 패키징된 앱 클라이언트의 정상 동작 유무를 판단하는 서버 인증의 기본 흐름도를 보입니다.



[그림 1] 서버인증 기본 흐름도

위 내용에 대한 서버인증 과정을 자세히 살펴보면 아래와 같습니다.

1. 앱 클라이언트에 포함된 AppGuard는 앱 실행 시 가장 먼저 실행되며 보안 모듈 초기화를 비동기적으로 수행합니다.
2. AppGuard는 보안 정책에 기반하여 필수 검증 항목에 대한 무결성 검사를 수행합니다.
3. 무결성이 확인되었다면 서버인증을 수행하기 위한 대기 모드로 진입하며, 만약 무결성이 손상되었다면 해당 정보를 로그 서버로 전송 후 서버인증 대기 모드로 진입하지 않고 더 이상 인증을 시도하지 않습니다.
4. 앱 서버에서 생성된 인증용 아이디를 수신한 앱 클라이언트는 AppGuard SDK 통해 제공된 `AppGuardClient::setUniqueClientID` 메서드(for Android) 또는 `APPGUARD_SET_CERTIFICATION_ID` 함수(for iOS)를 사용하여 AppGuard에 전달합니다.
5. AppGuard는 지정된 인증용 아이디를 사용하여 인증서버와 인증 작업을 수행합니다.
6. AppGuard는 인증 작업 진행 상황에 대한 정보를 약속된 콜백(Callback)을 사용하여 앱 클라이언트에 전달합니다. (상태: Success, Retry, Fail)
7. 앱 서버는 '클라이언트에서 수신된 서버인증 진행 결과 이벤트가 Success 또는 Fail' 경우 또는 '로그인, 스테이지 이동, 채널 변경, 클라이언트 결과 저장 등' 인증 확인 조건을 충족하였을 시 인증 서버에게 인증 결과 확인을 요청합니다.
8. 인증 서버는 앱 서버의 질의에 대하여 Valid, Not Yet, Invalid로 구성된 상태 정보 중의 하나를 응답합니다.
9. 앱 서버는 확인된 상태에 따라 세션 유지(인증 확인), 인증 질의 재시도, 세션 종료와 같은 결과에 대한 행위를 결정합니다.

## 2. AppGuard 서버인증 적용 방법

AppGuard 서버인증 시스템은 인증이 완료된 후 앱을 진행하는 동기화 방식과 앱을 먼저 진행 시키고 추후 인증 여부를 확인하는 비동기화 방식으로 구성할 수 있습니다.

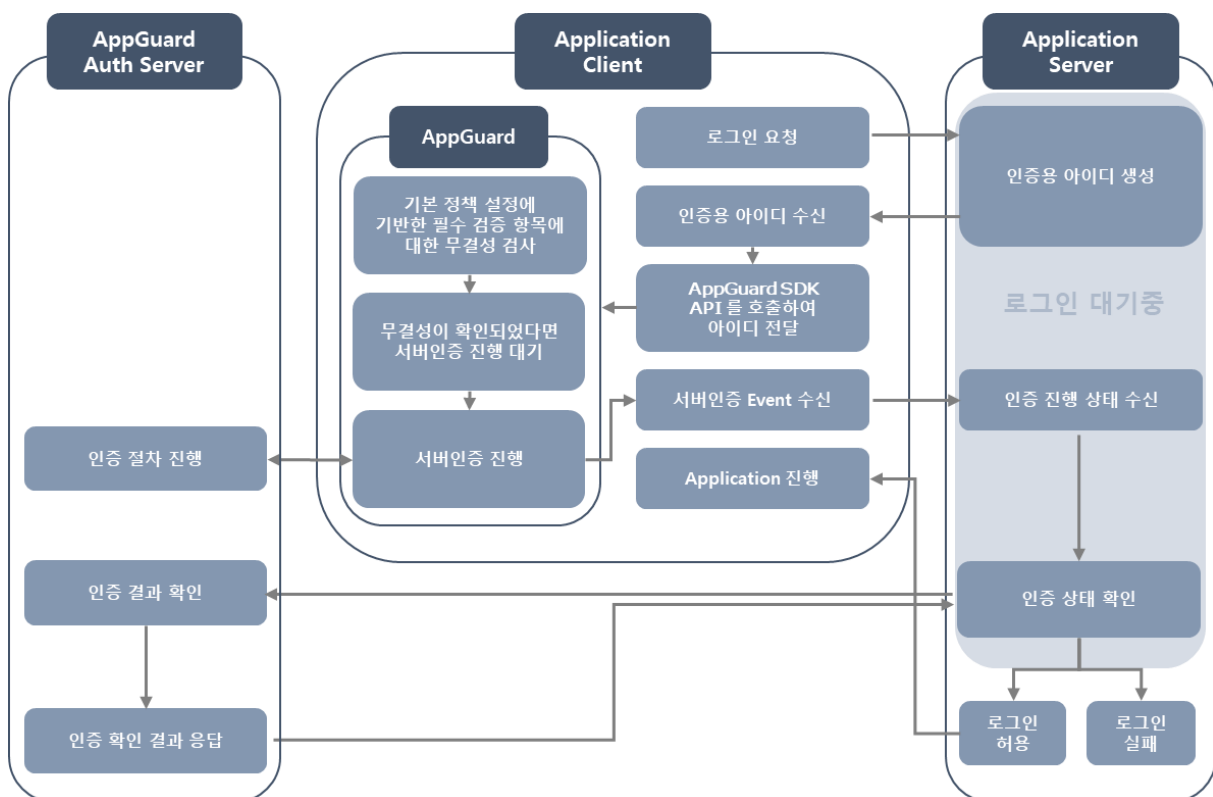
### 2.1. 동기화(Synchronous) 적용 방식

동기화 방식은 기본적으로 모든 인증 확인 과정이 정상적으로 완료되었을 경우 앱의 실제 서비스 콘텐츠 영역으로 진입을 허가하는 방식으로 적용하기 위하여 사용됩니다.

일반적으로 앱의 경우 앱 콘텐츠 영역으로 진입 전에 사용자 인증을 통한 로그인 과정이 선행되며, 해당 로그인 확인 프로세스에 서버인증 확인 절차를 적용하여 인증 성공이 확인된 경우에만 콘텐츠 영역으로 진입하도록 허용하는 방식으로 구성됩니다

#### 2.1.1 동기화 방식의 흐름도

다음은 동기화 방식으로 서버인증을 적용 시 흐름도를 보입니다.



[그림 2] 동기화 방식의 흐름도

## 2.1.2 동기화 방식의 예상되는 문제점

동기화 방식은 로그인 과정에서 서버인증의 성공을 확인 후 앱 콘텐츠로의 진입 허용을 전제로 하기 때문에 만약 클라이언트 네트워크 환경이 불안정한 경우 인증 확인 지연에 영향을 받을 수 있습니다.

앱 클라이언트에서 서버인증 수행을 위하여 호출하는 AppGurd SDK 에 포함된 API 인 AppGuardClient::setUniqueClientID() 메소드(for Android)의 구조는 타임아웃(timeout)값을 지정 가능하도록 되어 있으며, 최대 180 초(3 분)을 지정할 수 있습니다. 만약 타임아웃값을 180 초로 지정하였다면 클라이언트 자체 네트워크 환경 또는 클라이언트에서 인증서버까지의 네트워크 불안정이 지속되는 최악의 경우 로그인 과정이 최대 3 분동안 지연될 수 있다는 것을 의미합니다

## 2.2. 비동기화(Asynchronous) 적용 방식

비동기화 방식은 기본적으로 일정 기준 시간(기본 3 분) 동안 MOD 공격자가 앱을 진행하는 것을 허용하는 전제 조건하에서 서버인증 과정으로 인한 앱의 어떠한 지연 가능성도 배재하는 것을 목적으로 하는 방식으로 적용하기 위하여 사용됩니다.

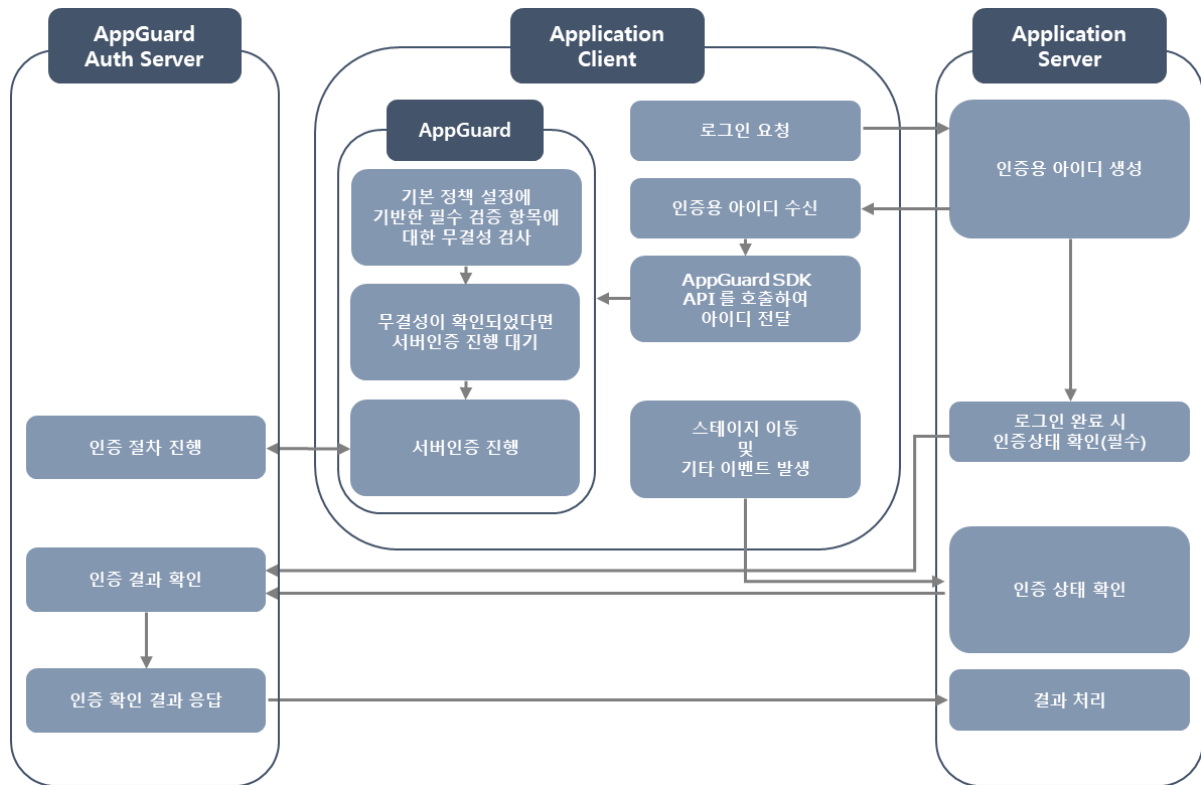
일반적으로 앱의 경우 클라이언트와 서버가 정보를 교환하는 여러가지 포인트를 가지고 있으며 중요 포인트마다 서버에서 인증서버에게 해당 세션에 대한 인증 유무를 질의하여 최종적으로 'Invalid' 상태로 확인되었을 경우 해당 세션을 종료처리하는 방식으로 구성됩니다.

예를 들면 앱에서는 로그인 과정이 필수적이며, 스테이지 이동, 맵(map)에서의 지역 이동, 채널 변경 등의 이벤트가 발생할 경우 인증서버에 인증 질의를 수행함으로 비동기적으로 서버인증 확인을 수행할 수 있습니다.

일반적으로 AppGuard 에서 제안하는 방법은 앱 서버에서 인증서버로의 질의 시점은 사용자 로그인 시점을 최초 질의 시점으로 하고 3 분~5 분 사이에 1~2 회 추가 질의를 할 수 있는 이벤트 시점을 선택하시는 것입니다

## 2.2.1 비동기화 방식의 흐름도

다음은 비동기화 방식으로 서버인증을 적용 시 흐름도를 보입니다.



[그림 3] 비동기화 방식의 흐름도

## 2.2.2 비동기화 방식의 예상되는 문제점

비동기화 방식 적용 시 하나의 세션에 대한 인증 확인을 위하여 만약 너무 잦은 질의 요청이 있을 경우 AppGuard 인증 서버의 부하가 증가할 수 있는 단점이 존재할 수 있으므로 질의 시점 설정에 주의를 요합니다.

만약 어떠한 이유로든 AppGuard 인증서버가 응답이 없을 시 또는 HTTP:200 이외의 상태 코드를 응답 시 해당 세션에 대한 인증 확인 여부를 "Valid"로 간주해야 합니다.



## 2.3 커스터마이징 유의사항

만약 AppGuard 서버인증 시스템을 위에서 제시된 2-1(동기화), 2-2(비동기화) 방식 이외의 방법으로 커스터 마이징(Customizing)하여 적용하는 경우 다음과 같은 서버인증 시스템 특성 및 해당 시스템이 방어하고자 하는 MOD 공격의 특성을 고려하여 주시기 바랍니다.

MOD(Modified)의 경우 일반적으로 앱에 적용된 AppGuard 를 제거하고 APK/IPA 를 재구성하여 배포하는 방식을 취합니다. 즉, 해당 공격이 완료된 환경에서는 AppGuard 가 동작하지 않거나 일부 복호화 기능만 동작할 가능성이 있습니다. 따라서 MOD 의 경우 앱에서 서버인증 적용 시 수신될 것이라고 예상되는 서버인증 시도 콜백(Callback)이 전달되지 않을 가능성이 높습니다.

따라서 클라이언트 또는 서버에서 특정 시간 또는 조건을 기반으로 강제적으로 서버 인증 유무를 확인하는 방법 또는 절차가 도입되어야 합니다.

첫째, 만약 앱 자체에서 특정 시간 조건(예를 들어 3 분) 동안 AppGuard 서버인증 콜백이 미수신 여부를 검사하여 앱 서버로 전달하는 방식으로 구현하는 경우 공격자에 의하여 해당 로직 또한 변조될 가능성이 존재함을 고려해야 합니다.

둘째, 만약 서버 자체에서 특정 시간 조건(예를 들어 3 분) 동안 앱 클라이언트로부터 실패 또는 성공을 포함한 서버인증 완료 패킷이 전달되지 않았을 강제로 인증서버로 인증 확인 요청 수행하는 방식으로 구현되었을 경우 서버인증 실패를 최종 확인 받기 위하여 6 분의 시간(강제 검증 시간이 3 분이라고 가정하였을 경우) 걸리는 인증 확인 지연 문제가 발생할 수 있습니다. 그 이유는 MOD 가 AppGuard 를 제거한 상태라고 가정하기 때문에 인증서버가 인증 유효 확인 프로세스를 시작하는 시점은 앱 서버에서 강제 발생 시킨 최초 인증 질의 시점을 기준으로 판단할 수 밖에 없기 때문임을 고려해야 합니다.

### 3. 서버인증 세션 식별 아이디 생성 메커니즘

서버인증의 인증 확인의 기준 값은 세션 아이디이며 해당 아이디는 사용자 또는 장치를 식별할 수 있는 유일(Unique) 식별자를 포함하여야 합니다. 따라서 사용자를 식별할 수 없는 임시 인증 토큰을 유일 식별자로 사용할 수 없으며, 만약 사용자 유일 식별자가 아닌 세션 유일 식별자를 사용 시 추후 인증 관련 장애/공격 유형 분석 서비스를 제공할 수 없습니다.

#### 3.1. 아이디(Unique ID) 생성 규칙

서버 인증은 UniqueClientID 값을 기반으로 동작하며, 해당 키는 SHA1 형태의 UniqueKey 값과 해당 키 생성 시의 Timestamp 값 그리고 앱 에 발급된 AppGuard License 키 값의 조합으로 구성됩니다.

아래 해당 ID 값의 포맷을 보입니다.

항목	데이터 타입	설명
UniqueID	char(40)	SHA1 또는 최대 40 자로 구성된 클라이언트 유일 식별 값
TimeStamp	char(13)	13 자로 구성된 TimeStamp 값 해당 값은 반드시 UTC(Coordinated Universal Time) 기준 값의 TimeStamp 값을 사용해야 합니다.
LicenseKey	string	해당 앱에 발부된 AppGuard License Key 값

각 데이터 항목은 ":" 으로 구분되어 조합됩니다. 다음은 포맷에 맞게 생성된 UniqueClientID 의 예를 보입니다.

```
4977d446325510b58008f6a0f48d8f5fed0f456f:1469094337856:Sw==
```

항목 UniqueID 의 값은 최대 40 자로 구성된 클라이언트 식별 유일 값이라는 것 이외의 특별한 제약 사항은 없습니다. 필요에 따른 완전한 난수값으로 처리하는 것이 가능합니다.  
하지만 기본적으로 사용자 추적 등의 식별을 위하여 sha1\_func(내부 사용자 식별값)과 같이 sha1 형태의 사용자 식별 유일 값을 사용하는 것을 권장합니다.

항목 TimeStamp 의 값은 동일한 UniqueID 를 사용하는 장비 또는 사용자의 정확한 세션 시간을 구분하기 위한 용도로 13 자로 구성되며, UTC(Coordinated Universal Time) 기준 값의 TimeStamp 값이어야 합니다.

해당 값은 인증 서버의 일시 장애로 클라이언트의 인증 여부가 불확실한 경우 해당 인증을 임의로 허용(강제 인증)할지 판단하는 중요한 기준 값으로 사용될 수 있으며 앱 서버와 클라이언트간의 로그인 세션 생성 시간 값 또는 UniqueClientID 를 생성 시점의 시간 값을 사용하는 것을 권장 합니다.

항목 LicenseKey 의 값은 해당 앱에서 발부된 AppGuard License Key 값으로 (AppGuard Manager Server 에서 확인 가능) 앱 서버는 구동 시 AppGuard License Key 값을 보유해야 합니다.  
해당 UniqueClientID 값을 생성 시 License Key 값을 첨부하여 생성해야만 합니다.

앱 서버는 위 방식으로 생성한 인증용 아이디를 앱 클라이언트에 전달해 줍니다.

### 3.2. 아이디 자동 발급 방법

기본적으로 AppGuard 서버인증 시스템은 사용자/계정 또는 디바이스를 식별할 수 있는 유일 식별자를 포함한 인증용 아이디를 기반으로 동작하도록 설계되어 있습니다. 하지만 부득이 하게 앱 클라이언트와 앱 서버의 설계상 인증용 아이디 전달 메커니즘을 포함 시킬 수 없는 경우를 대비하여 'Null 키' 모드라는 AppGuard 자체의 키 발급 시스템을 제공합니다.

Null 키 모드는 단순의 기존 인증용 아이디 지정 API 의 인자로 Null 을 전달하는 것을 자동 발급 모드로 동작합니다.

[Android]

```
AppGuardClient.setUniqueClientID(null, 180);
```

[iOS]

```
APPGUARD_SET_CERTIFICATION_ID(null, 180);
```

위와 같이 해당 메서드를 호출하면 AppGuard 자체적으로 사용자 식별이 되지 않는 세션 유일 식별자를 포함한 인증용 아이디를 생성하여 발급되며 서버인증이 성공하였을 경우 SecurityEventType.Event.S2AUTH\_CALLBACK 이벤트 수신 시 해당 생성된 아이디가 함께 콜백의 데이터로 클라이언트에 전달됩니다.

Null 키 모드를 사용한 적용 방법 선택 시 주의 사항은 다음 사항을 유의하여야 합니다.

앱 서버에서 인증용 키를 생성할 경우 앱 서버 입장에서 인증 서버에 질의할 대상의 아이디를 정확하게 식별하는 것이 가능하지만, Null 키 모드를 사용 시 공격자에 의하여 앱 서버가 인증 확인에 사용할 아이디를 위/변조하여 전달하게 공격하는 것이 가능합니다. 즉, 공격자는 이미 인증 성공으로 기록된 인증용 아이디를 재 사용하여 앱 서버가 해당 과거의 인증용 아이디를 재 사용하도록 조작할 수 있습니다.

AppGuard 서버인증 시스템을 위 문제를 보완하기 위하여 앱 서버에서 인증 요청을 몇 번을 호출하였는지 정보를 응답 패킷에 포함하여 앱 서버에 전달합니다.

따라서 앱 서버는 인증 성공으로 확인된 응답 패킷 내의 JSON 형태의 필드 중 "validCount" 값을 사용하여 중복 호출 유무를 확인해야 합니다.

AppGuard에서는 일반적으로 해당 "validCount" 값이 3을 초과할 경우 "Invalid" 상태로 간주하여 인증 실패로 처리하기를 권장합니다. (일시적인 인증서버와 앱 서버간의 네트워크 장애로 인하여 중복 질의가 이론상 가능하다고 가정하기 때문에 질의 횟수를 3번 이하로 하는 것을 추천합니다.)

## 4. 서버인증 상세 적용

첫번째 방법은 앱 서버가 Rest API 형태와 같이 현재 클라이언트의 세션을 식별 관리할 수 있는 객체(또는 세션 객체)를 가지고 있지 않는 형태의 서버 구조에 적합하도록 설계된 방식이며, 또한 앱 서버 측에서 서버 인증 확인을 시도할 정확한 시점(Callback 을 통한)을 알 수 있다는 장점이 있습니다

### 4.1. 앱 서버에서 인증용 아이디 생성하기

위 3.1. 아이디(Unique ID) 생성 규칙을 참고하여 주시기 바랍니다.

### 4.2. 앱 클라이언트에서 서버인증 요청하기

앱 클라이언트에 Server-Side Authentication 을 적용하기 위해서는 제공된 별도의 SDK 를 사용해야만 합니다. (appguard-sdk.jar)

해당 SDK 에는 AppGuardClient 클래스를 제공하며, 서버인증의 위한 setUniqueClientID 라는 static 메서드를 제공합니다.

해당 메서드의 원형은 다음과 같습니다.

```
public class
AppGuardClient
implements BaseEventInvoker
```

```
com.inca.security.Interface.BaseEventInvoker
↳com.inca.security.AppGuard.AppGuardClient
```

#### Public Methods

static void	setUniqueClientID(String clientID, int retryTimeout) 서버 인증에 사용한 UniqueClientID 를 설정합니다.
-------------	--

```
public void setUniqueClientID(String clientID, int retryTimeout)
```

서버인증에 사용될 사용자 식별 값을 설정합니다.

만약 clientID 값을 null 로 입력할 경우 AppGuard 시스템에 의하여 랜덤으로 UniqueClientID 가 자동으로 생성됩니다. 해당 생성된 ID 값은 SecurityEventType.Event.S2AUTH\_CALLBACK 으로 받을 수 있습니다.

(주의 1: 해당 메시드는 2017/4/26 일자 AppGuard 모듈부터 static 에서 non-static 으로 변경되었습니다.)

(주의 2: 18005 모듈부터 nullKey 모드를 지원합니다.)

아래 해당 ID 값의 포맷을 보입니다.

항목	데이터 타입	설명
UniqueID	char(40)	SHA1 또는 40 자로 구성된 클라이언트 유일 식별 값
TimeStamp	char(13)	14 자로 구성된 해당 UniqueClientID 생성 시 TimeStamp 또는 클라이언트와 서버의 세션이 생성된 시간의 TimeStamp
LicenseKey	string	해당 앱에 발부된 AppGuard License Key 값

#### Parameters

*clientID* UniqueClientID 키 포맷을 준수하는 앱 서버에서 생성한 사용자 식별 값, null 값을 입력할 경우 AppGuard 에 의하여 자동으로 랜덤 ID 를 생성합니다.

*retryTimeout* 서버인증 재시도(retry)를 종료할 최대 시간값(초단위, 기본값 SecurityEventType.S2Auth.S2AUTH\_OPTION\_DEFAULT\_TIMEOUT(= 180))

서버 인증 재시도 Timeout 값의 설정 가능 값의 범위는 0 ~ 180 (즉, 최대값 180 초=3 분)이며, 0 으로 설정 시 재시도를 하지 않으며, 180 으로 설정 시 일반적으로 3 분안에 약 15 번의 인증 재시도가 수행됩니다.

*retryTimeout* 의 기본 설정값을 180 로 설정하는 것을 권장합니다.

다음은 해당 API 호출에 대한 예제를 보입니다.

```
...
AppGuardClient.setUniqueClientID(uniqueID, 180);
...
```

만약 AppGuard 에 의하여 자동으로 Unique Client ID 값이 생성되는 nullKey 모드를 사용하고자 할 경우의 예를 보입니다.

```
...
AppGuardClient.setUniqueClientID(null, 180);
...
```

해당 API 메서드 호출을 위한 특별한 제약 사항은 없으며, 앱 서버에서 UniqueClientID 를 수신하는 경우 해당 메서드를 호출하면 됩니다. 단, 서버 인증 전체 프로세스의 타임아웃 값이 3 분이기 때문에 가급적 빠른 시간 안에 호출되는 것을 권장합니다.

### 4.3. 앱 클라이언트에서 서버인증 콜백 적용하기

AppGuard 는 Event 콜백 수신을 위한 두 가지 방식을 제공합니다. 첫번째는 AppGuardEventListener 를 상속하여 이벤트는 수신하는 방식이며, 두번째는 setCallbackHandler() 메소드를 사용하여 이벤트 수신 핸들러는 등록하는 방식입니다.

다음은 AppGuardEventListener 를 통한 Event 콜백 수신에 대한 구현 예제를 보입니다.

```
class MyEventListener implements AppGuardEventListener {
    @Override
    public void onEvent(int code, byte[] data) {
        switch ( code ) {
            case SecurityEventType.Event.S2AUTH_CALLBACK:
                Object[] result = SecurityEventParser.getInstance().parse(
                    SecurityEventType.EVENT,
                    SecurityEventType.Event.S2AUTH_CALLBACK,
                    data);

                int stateValue = (Integer)result[0];
                if ( stateValue == SecurityEventType.S2Auth.S2AUTH_RESULT_SUCCESS ) {
                    // 서버 인증이 성공하였습니다.
                    // 성공 시 한번만 콜백이 호출됩니다.
                    // Null Key Mode 일 경우 Unique Client ID 를 추출합니다.
                    String uniqueClientID = (String)result[1];
                    // App 서버에 인증 검증을 요청합니다.
                }
                else if ( stateValue == SecurityEventType.S2Auth.S2AUTH_RESULT_FAIL ) {
                    // 서버 인증이 실패했습니다.
                    // retry time (180 초) 만큼 서버 인증을 재시도 하였으나 실패한 경우입니다.
                    // 이 경우도 한번만 콜백이 호출 됩니다.
                    // App 을 종료 하시거나, 혹은 정책에 따라서 다음 작업을 진행하시기 바랍니다.
                }
                else if ( stateValue == SecurityEventType.S2Auth.S2AUTH_RESULT_RETRY ) {
                    // 서버 인증을 시도하는 중입니다.
                    // 네트워크 문제 혹은 인증 각 단계별 원인에 의해 인증이 이루어지지 않은 경우
                    // retrytime(180 초) 동안 인증을 재 시도 합니다.
                }
            }
        }
    }
}
```

```

        //. 각 재시도 시 마다 콜백이 호출되며, 여러번의 콜백이 날아 올 수 있습니다.
        //. 본 콜백은 무시하셔도 됩니다.

    }
  }
}

```

다음은 위 예제에서 사용된 SecurityEventType.S2Auth 클래스의 명세를 보입니다.

```
public class
```

### SecurityEventType.S2Auth

```
com.inca.security.Common.SecurityEventType.S2Auth
```

#### Class Overview

서버인증 이벤트 식별값의 열거자 클래스입니다.

#### Fields

```
public static final int S2AUTH_RESULT_SUCCESS
```

보안모듈이 서버인증을 위해 인증서버에 시도한 인증 요청이 성공했음을 의미하는 이벤트입니다.

Constant Value : 1 (0x00000001)

```
public static final int S2AUTH_RESULT_RETRY
```

보안모듈이 서버인증을 위해 인증서버에 시도한 인증 요청이 진행중임을 의미하는 이벤트입니다.  
 해당 이벤트는 네트워크가 불안정하거나 인증서버의 장애 발생시 발생  
 할 수 있습니다.

Constant Value : 2 (0x00000002)

```
public static final int S2AUTH_RESULT_FAIL
```

보안모듈이 서버인증을 위해 인증서버에 시도한 인증 요청이 실패했음을 의미하는 이벤트입니다.

Constant Value : 3 (0x00000003)



#### 4.4. 앱 서버에서 인증 결과 확인하기

Event 콜백을 통하여 클라이언트가 SecurityEventType.S2Auth.S2AUTH\_RESULT\_SUCCESS 이벤트를 수신하였을 경우 앱 서버를 통하여 인증 확인 작업을 수행하여야 합니다.

앱 서버는 특정 클라이언트의 세션을 식별하는 인증용 아이디를 사용하여 특정 세션에 대한 인증 유무를 인증 서버에 반드시 질의하여야 합니다.

인증 서버는 클라이언트 상태에 따라 "not yet", "invalid", "valid" 상태를 응답합니다. 만약 인증 서버의 문제로 인하여 응답하지 않거나 응답 패킷이 이상이 있을 시 앱 서버 내부적으로 "valid" 상태로 간주해야 합니다.

다음은 "valid" 상태로 간주해야하는 상황의 목록을 보입니다.

1. 인증서버가 응답 패킷에 정상적으로 "valid" 상태를 지정하여 응답하는 경우
2. 인증서버가 응답하지 않는 경우
3. 인증서버가 HTTP: 200 코드 이외의 상태를 응답하는 경우

상태 질의에는 특별한 제약 사항이 있으며, Null 키 모드를 사용 시 유의 사항은 "3.3. 아이디 자동 발급 방법" 에 기재된 내용을 참고하여 주시기 바랍니다.

다음은 인증 서버에 질의를 위한 WEB API 명세를 보입니다.

상세 내용			
리소스 경로	<a href="https://auth.appguard.co.kr/queryStatus">https://auth.appguard.co.kr/queryStatus</a>		
컨텐츠 타입	Json		
요청 메소드	POST		
요청 파라미터			
Param Name	Type	필수	내용

<b>data</b>	String	○	인증처리를 할 수 있는 ":"(콜론)을 구분자로한 정해진 형식의 문자열 -형식 사용자 Session 이 구분 가능한 유일 ID(40 자리) + ":" + timestamp(13 자리) + ":" + Appguard 라이선스키 예) a1b2c3d4e5f6g7h8i9j0:1469087298707:ABC
<b>응답 내용</b>			
<b>내용</b>	<pre>{   "status" : 1,   "message" : "not yet",   "validCount" : 3   "token" : "16 자리의 보안 토큰 값"   "returnTime" : "13 자리의 long type Timestamp" }</pre> <p>※ <b>status, message</b> 내용 <b>1 : not yet / 2 : invalid / 3 : valid</b></p>		

위 응답 내용 중 주요한 필드 값은 "message or status"입니다. "message or status"는 사용자의 서버 인증 유무를 확인하는 값이며, 만약 **message 값이 "valid" 이면(또는 status 값이 3)**, 인증이 완료된 클라이언트 세션이며 이 경우 정상적인 작업을 진행하시면 됩니다.

아래 내용은 앱 서버에서 AppGuard 인증서버로 질의를 하기 위하여 nodeJS 로 제작한 샘플 코드입니다.

```
/*
 * Function Name : checkAuth
 * Parameter : string UUID
 * Desc :
 * - AppGuard Client 에서 서버 인증 작업이 정상적으로 진행 되었는지를 AppGuard Auth Server 에 질의
 * - https://auth.appguard.co.kr/queryStatus 에 POST 로 data=인증용 아이디 형태로 질의
 */

function checkAuth(UDID){
  request.post({
    headers: {'content-type' : 'application/x-www-form-urlencoded'},
    url: 'https://auth.appguard.co.kr/queryStatus',
```

```
body: "data=" + UDID
},function(err, response, body){
  if(err){
    // AppGuard 인증 서버가 응답이 없는 경우
    // 이때는 valid 처리를 한다. (서버인증에 성공한 것으로 간주 한다.)
  }else{
    var json = JSON.parse(body);
    if(json.status === "1") {
      // not yet 인 경우는 서버인증이 아직 진행 중인 상태이긴 하나 .
      // 클라이언트 콜백을 통해 서버 인증 재 검증이 진행되는 경우는
      // 앱 클라이언트를 종료 처리하거나 혹은 고객사 정책에 따라 결정
    }
    else if(json.status === "2") {
      // invalid 인 경우는 서버인증에 실패 한 경우 이므로, 정책에 따라 처리한다.
      // 앱 클라이언트를 종료 처리하거나 혹은 고객사 정책에 따라 결정
    }
    else if(json.status === "3") {
      // valid 인 경우는 서버인증이 성공 한 경우.
      // 예정된 정상적인 작업을 진행 한다.
    }
  }
});
}
```

앱 서버에서 AppGuard 인증 서버로 질의를 한 경우 status 값이 1 (not yet)으로 리턴되는 값은 아직 서버 인증을 진행 중이라는 의미입니다.



■ (주)잉카인터넷

메일: [appguard@inca.co.kr](mailto:appguard@inca.co.kr)

서울시 구로구 구로 3 동 235-2 에이스 하이엔드타워 12 층 1201 호 (우 152-848)

Copyright ©INCA Internet Corp. All rights reserved.