SignKorea 마이데이터 서버 모듈 적용을 위한

JAVA 개발자 가이드

Version 1.0



|  |  |  |  |
| --- | --- | --- | --- |
| 작성일시 | 변경내용 | 수정페이지 | 작성자 |
| 2021.05.24 | 최초 작성 | - | 최서현 |

**본 책자에 대하여**

본 책자는 SignKorea의 마이데이터 서버 모듈을 적용하려는 JAVATM

개발자들에게 개발자 가이드를 제공하려는 목적으로 발간되었습니다.

|  |
| --- |
| 이 책자는 저작권법에 의해 보호되는 (주)코스콤 소유의 정보가 담겨 있습니다. 이 책자의 어떠한 부분도 사전에 동의 없이 복사 또는 다른 언어로 번역될 수 없습니다. |

|  |
| --- |
| (주)코스콤 인증센터 SignKorea  경기도 안양시 동안구 호계동 1026 – 7 B동 3층       ￭ Tel : 1577-7337       ￭ Fax : 02-767-7325 |

**1. 개요**

    SignKorea는 본인확인서비스를 위한 요청메시지(UCPIDRequest) 생성 api와 본인확인 서버로부터 받은 본인확인 응답메시지(UCPIDResponse)에서 필요한 정보를 획득할 수 있는 api를 함께 제공한다. 관련 api는 jar 파일 형태로 제공되며, 개발자가 사용하는 개발 환경에서 함수를 호출하여 사용하도록 구성되어있습니다.

* 1. **구성**
  2. **개발환경 설정**
  3. **샘플 코드**
  4. **구성**
* 제공함수 분류

1. RequestUtils - UCPIDRequest 메시지 생성 함수 관련 Util
2. ResponseUtils - UCPIDResponse 객체 생성 관련 Util
3. ClientUtils - Client로부터 넘어온 데이터 처리 관련 Util
4. StatusUtils - UCPIDResponse Status 확인 관련 Util
5. VarUtils - 파일 또는 config 파일 read 관련 Util
6. MydataException - 메시지 생성 및 응답 복호화 관련 exception

* 제공 파일 (jdk 1.6 버전부터 사용 가능)

1. Jar 파일
2. bcprov-jdk1.5on-1.59.jar - bouncyCastle 관련 api
3. bcpkix-jdk1.5on-1.59.jar - bouncyCastle 관련 api
4. local\_policy.jar
5. US\_export\_policy.jar
6. mydata.jar - 코스콤이 제공하는 메시지 관련 api
7. 인증서 파일
8. kmCert.der - 키분배용 인증서
9. kmPri.key - 키분배용 인증서에 상응하는 key 파일
10. signCert.der - 서명용 인증서
11. signPri.key - 서명용 인증서에 상응하는 key 파일
12. class 파일
13. encryptPassword.class - properties에 저장할 인증서 비밀번호 암호화 모듈
    1. **개발환경 설정**

* Bouncycastle

1. ASN.1 및 인증 관련한 다양한 기능을 제공하는 자바 암호화 라이브러리

* Bouncycastle 환경 설정 방법 (리눅스 및 윈도우의 경우 동일)

1. **Java 설치 경로/jre/lib/ext**에 하기의 두 파일 추가
2. bcprov-jdk1.5on-1.59.jar
3. bcpkix-jdk1.5on-1.59.jar
4. **Java 설치 경로/jre/lib/security**에 하기의 두 파일 추가
5. US\_export\_policy.jar
6. local\_policy.jar
7. **Java 설치 경로/jre/lib/security/java.security** 파일에 하기의 provider 설정 코드 추가
8. security.provider.N=org.bouncycastle.jce.provider.BouncyCastleProvider
9. 예시



1. IDE에서 개발 시 build path에 하기의 세 jar 파일 추가 필요.
2. bcprov-jdk1.5on-1.59.jar
3. bcpkix-jdk1.5on-1.59.jar
4. mydata.jar (코스콤에서 제공하는 서버 모듈 api)

* 정상 설치 확인 코드

String providerName = “BC”;

If(Security.getProvider(providerName)==null){

System.out.println(providerName + “ provider not installed”);

} // 비정상

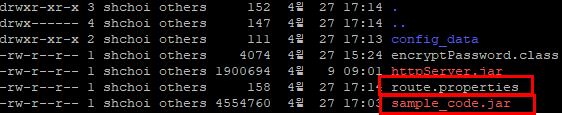
else{

System.out.println(providerName + “ is installed.”); // 정상

}

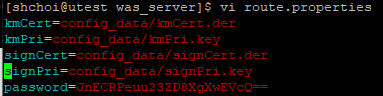
* Properties 파일 설정

1. 명칭 : **route.properties**
2. 위치 : 원하는 위치에 설정 후, 코드 상에서 해당 경로 설정 (샘플코드 참조)



1. 내용

* 반드시 변수의 명칭을 kmCert, kmPri, signCert, signPri, password로 지정



* 인증서 및 키 파일 경로의 경우 **절대경로 혹은 상대 경로 모두 가능**

(위 예시의 경우, route.properties와 동일한 위치에 config\_data 폴더에 키쌍 저장)

* password의 경우, 제공하는 class 파일인 **encryptPassword.class**를 통해 암호화 하여 properties 파일에 저장
* encryptPassword 클래스 파일의 경우, ”**AES**”알고리즘을 사용해 패스워드 암호화



* 1. **요청 메시지 생성 및 응답 메시지에서의 값 획득 함수**

**[RequestUtils]**

**getUCPIDRequest** - 본인확인에 맞는 규격의 Request 메시지 생성

**get\_UCPIDNonce\_from\_signedPersonInfoReq** – 이용약관 서명값으로부터 ucpidNonce값 획득

**[ResponseUtils]**

**getStatusCode** - 본인확인서버로부터 받은 메시지의 상태값 획득

**getInstance** - UCPIDResponse byte 값을 UCPIDResponse 객체로 변환

**[ClientUtils]**

**getSigningTime** - 클라이언트 모듈의 서명한 시간 획득

**get\_consentNonce\_from\_signedConsentInfo**

- 클라이언트 모듈의 서명데이터에서 consentNonce 획득

**get\_ucpidNonce\_from\_signedConsentInfo**

- 클라이언트 모듈의 서명데이터에서 ucpidNonce 획득

**get\_certSerialNumber\_from\_cms**

- 클라이언트 모듈의 서명데이터에서 인증서 일렬번호 획득

**get\_certificate\_from\_cms**

- 클라이언트 모듈의 서명데이터에서 인증서 byte 값 획득

**[UCPIDResponse]**

**getInstance**  - UCPIDResponse 객체 생성

**getVersion** - UCPIDResponse의 Version 획득

**getUcpidNonce** - UCPIDResponse의 ucpidNonce 획득

**getCpRequestNumber** - UCPIDResponse의 cpRequestNumber 획득

**getCertDn**  - UCPIDResponse에서 이용자 인증서 고유명 획득

**getCpCode**  - UCPIDResponse에서 ISP의 기관코드 획득

**getDi** - UCPIDResponse에서 이용자의 중복확인 정보 획득

**getRealNumber** - UCPIDResponse에서 이용자의 실명 획득

**getGender** - UCPIDResponse에서 이용자의 성별 획득

**getNationalInfo** - UCPIDResponse에서 내외국인 정보 획득

**getBirthDate** - UCPIDResponse에서 이용자의 생년월일 정보 획득

**getCiUpdate** - UCPIDResponse에서 ciupdate 정보 획득

**getCi, getCi2** - UCPIDResponse에서 ci, ci2 정보 획득

**[StatusUtils]**

**isOk** - 본인확인서버로부터 받은 메시지의 상태값 정상여부 확인

**[VarUtils]**

**getResourceFromproperty** - properties 파일로부터 해당 변수 값 획득

**getFromFile** - 특정 위치의 파일을 byte 값으로 획득

**get\_data\_for\_for\_CMSVerify** - 검증 모듈(라이브러리)에 넣을 인코딩된 서명데이터 획득

**base64\_url\_decoding** - base64\_url\_safe로 encoding된 데이터 decoding한 값 획득

**[MydataException]**

**getErrCode** - 발생한 에러와 관련한 에러코드 획득

**RequestUtils - getUCPIDRequest**

|  |  |
| --- | --- |
| **◦ 기 능** | 마이데이터 사업자로부터 받은 값들을 사용해 UCPIDRequest 규격 메시지 생성 |
| **◦ 원 형** | public byte[] getUCPIDRequest(String ucpIdNonce,String cpCode, String cpRequestNumber, byte[] certificate, byte[] signed\_personInfoReq, int flag) |
| **◦ 인자값** | 1) ucpIdNonce   * 재전송공격 방지정보. * 통합인증요청(통합인증-002 API)시 마이데이터 사업자가 정보제공자에게 제공.   2) cpCode  - 기관 코드  - 통합인증요청(통합인증-002 API)시 마이데이터 사업자가 정보제공자에게 제공.  3) cpRequestNumber   * 본인확인 요청의 추적성을 위한 트랜잭션 ID(tx\_id) * 통합인증요청(통합인증-002 API)시 마이데이터 사업자가 정보제공자에게 제공.   4) certificate: 키분배용 공동인증서(이하 명칭 “kmCert”으로 정의)  5) signed\_personInfoReq   * 본인확인 이용동의 전자서명 값 * 통합인증요청(통합인증-002 API)시 마이데이터 사업자가 정보제공자에게 제공. |
| **◦ 반환값** | 6) flag: 1일경우 signedAttribute 필드 존재, 0 일경우 해당 필드 존재 X    성공 시 UCPIDRequest byte array, 실패시 Exception 발생 |
| **◦ 비 고** | * Certificate 파라미터 값에서 AKI 값 추출 후 ISPReqInfo 생성. |

**RequestUtils- get\_UCPIDNonce\_from\_signedPersonInfoReq**

|  |  |
| --- | --- |
| **◦ 기 능** | 본인확인 서버로부터 받은 응답으로부터 StatusCode 값 획득 |
| **◦ 원 형** | public String get\_UCPIDNonce\_from\_signedPersonInfoReq(byte[] signed\_personInfoReq) |
| **◦ 인자값** | Signed\_PersonInfoReq : 마이데이터사업자로부터 전달 받은 본인확인약관 서명값 |
| **◦ 반환값** | 성공 시 UCPIDNonce 값 리턴 |
| **◦ 비 고** |  |

**ResponseUtils- getStatusCode**

|  |  |
| --- | --- |
| **◦ 기 능** | 본인확인 서버로부터 받은 응답으로부터 StatusCode 값 획득 |
| **◦ 원 형** | public String getStatusCode(byte[] bUCPIDResponse) |
| **◦ 인자값** | bUCPIDResponse: 인증기관으로부터 받은 본인확인결과 응답(UCPIDResponse) 에 대한 byte array |
| **◦ 반환값** | 성공 시 UCPIDResponse 객체, 실패시 Exception 발생 |
| **◦ 비 고** |  |

**ResponseUtils- getInstance**

|  |  |
| --- | --- |
| **◦ 기 능** | 인증기관으로부터 받은 본인확인결과 byte array를 UCPIDResponse 객체로 변환 |
| **◦ 원 형** | public UCPIDResponse getInstance(byte[] bUCPIDResponse) |
| **◦ 인자값** | bUCPIDResponse: 인증기관으로부터 받은 본인확인결과 응답(UCPIDResponse) 에 대한 byte array |
| **◦ 반환값** | 성공 시 UCPIDResponse 객체, 실패시 Exception 발생 |
| **◦ 비 고** |  |

**ClientUtils- getSigningTime**

|  |  |
| --- | --- |
| **◦ 기 능** | 마이데이터사업자로부터 받은 서명데이터에서 서명 시간 획득 |
| **◦ 원 형** | public Date getSigningTime(byte[] signed\_personInfoReq) |
| **◦ 인자값** | signed\_personInfoReq: 마이데이터사업자로부터 받은 서명데이터 |
| **◦ 반환값** | 성공 시 Date 객체, 실패시 null값 반환 |
| **◦ 비 고** |  |

**ClientUtils- get\_ucpidNonce\_from\_sigendPersonInfoReq**

|  |  |
| --- | --- |
| **◦ 기 능** | 마이데이터사업자로부터 받은 서명데이터에서 ucpidNonce값 획득 |
| **◦ 원 형** | public String get\_ucpidNonce\_from\_signedPersonInfoReq(byte[] signed\_personInfoReq) |
| **◦ 인자값** | signed\_personInfoReq: 마이데이터사업자로부터 받은 본인확인요청 서명데이터 |
| **◦ 반환값** | 성공 시 ucpidNonce 반환, 실패시 null값 반환 |
| **◦ 비 고** |  |

**ClientUtils- get\_consentNonce\_from\_sigendConsentInfo**

|  |  |
| --- | --- |
| **◦ 기 능** | 마이데이터사업자로부터 받은 서명데이터에서 consentNonce값 획득 |
| **◦ 원 형** | public String get\_consentNonce\_from\_signedConsentInfo(byte[] signed\_consentInfo) |
| **◦ 인자값** | signed\_consentInfo: 마이데이터사업자로부터 받은 전송요구내역 서명데이터 |
| **◦ 반환값** | 성공 시 consentNonce 반환, 실패시 null값 반환 |
| **◦ 비 고** |  |

**ClientUtils- get\_certSerialNumber\_from\_cms**

|  |  |
| --- | --- |
| **◦ 기 능** | 마이데이터사업자로부터 받은 서명데이터(CMS)에서 인증서 일렬 번호 획득 |
| **◦ 원 형** | Public int get\_certSerialNumber\_from\_cms(byte[] cms\_data) |
| **◦ 인자값** | cms\_data: 마이데이터사업자로부터 받은 서명데이터(cms 서명 규격) |
| **◦ 반환값** | 성공 시 인증서 일렬번호 반환, 실패시 null값 반환 |
| **◦ 비 고** |  |

**ClientUtils- get\_certificate\_from\_cms**

|  |  |
| --- | --- |
| **◦ 기 능** | 마이데이터사업자로부터 받은 서명데이터(CMS)에서 인증서 byte값 획득 |
| **◦ 원 형** | Public byte[] get\_certificate\_from\_cms(byte[] cms\_data) |
| **◦ 인자값** | cms\_data: 마이데이터사업자로부터 받은 서명데이터(cms 서명 규격) |
| **◦ 반환값** | 성공 시 인증서 byte 값 반환, 실패시 null값 반환 |
| **◦ 비 고** |  |

**UCPIDResponse - getVersion**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의version 값을 반환 |
| **◦ 원 형** | public int getVersion() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 version 값 반환, statusCode값이 OK가 아닐 경우, 0 값 반환. |
| **◦ 비 고** | UCPIDResponse의 version의 경우 현재 2 |

**UCPIDResponse - getUcpidNonce**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의ucpidNonce 값을 반환 |
| **◦ 원 형** | public String getUcpidNonce() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 ucpidNonce 값 반환, statusCode값이 OK가 아닐 경우, null 값 반환. |
| **◦ 비 고** | * UCPID 요청 및 응답을 재연공격으로부터 안전하게 전송하기 위해 활용하는 값 * UCPIDResponse의 ucpidNonce의 경우, 통합인증요청(통합인증-002 API)시 마이데이터 사업자가 정보제공자에게 제공하는 값과 동일 |

**UCPIDResponse - getCpRequestNumber**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의cpRequestNumber 값을 반환 |
| **◦ 원 형** | public String getCpRequestNumber() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 cpRequestNumber 값 반환, statusCode값이 OK가 아닐 경우, null 값 반환. |
| **◦ 비 고** | * cpRequestNumber:ISP가 부여하는 세션정보로 트랜잭션 Id에 해당 * UCPIDResponse의 cpRequestNumber의 경우, 통합인증요청(통합인증-002 API)시 마이데이터 사업자가 정보제공자에게 제공하는 값과 동일 |

**UCPIDResponse - getCertDn**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의certDn 값을 반환 |
| **◦ 원 형** | public String getCertDn() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 certDn 값 반환, statusCode값이 OK가 아닐 경우, null 값 반환. |
| **◦ 비 고** | certDn의 경우 마이데이터 앱 이용자의 공인인증서 Dn에 해당. |

**UCPIDResponse - getCpCode**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의cpCode 값을 반환 |
| **◦ 원 형** | public String getCpCode() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 cpCode 값 반환, statusCode값이 OK가 아닐 경우, null 값 반환. |
| **◦ 비 고** | * cpCode의 경우, 기관 코드에 해당 * **인증기관 및 중계기관 이용여부에 따라, cpCode를 넣는 방식 상이**  1. 중계기관 미이용하는 경우   (모든 인증기관)   * cpCode: 정보제공자 cpCode * cpRequestNumber: 트랜잭션 ID(tx\_id)  1. 중계기관 사용하는 경우   (코스콤/한국정보인증)   * cpCode: 정보제공자 cpCode * cpRequestNumber: 트랜잭션 ID(tx\_id)   (금융결제원)   * cpCode: 중계기관 cpCode * cpRequestNumber: 트랜잭션 ID + 정보제공자 cpCode |
|  |  |
|  |  |

**UCPIDResponse - getDi**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의di 값을 반환 |
| **◦ 원 형** | public String getDi() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 di 값 반환, statusCode값이 OK가 아닐 경우, null 값 반환. |
| **◦ 비 고** | Di : 이용자의 중복가입 확인정보 |

**UCPIDResponse - getRealName**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의realName 값을 반환 |
| **◦ 원 형** | public String getRealName() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 realName 값 반환, statusCode값이 OK가 아닐 경우, null 값 반환. |
| **◦ 비 고** | realName: 이용자의 실명 |

**UCPIDResponse - getGender**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의Gender 값을 반환 |
| **◦ 원 형** | public int getGender() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 gender 값 반환, |
| **◦ 비 고** | * gender: 이용자의 성별 * 0 : 여성 / 1: 남성 * 111 : UCPIDResponse 메시지에서 gender 값이 존재하지 않을 경우 |

**UCPIDResponse - getNationalInfo**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의nationalInfo 값을 반환 |
| **◦ 원 형** | public int getNationalInfo() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 nationalInfo 값 반환, |
| **◦ 비 고** | * nationalInfo: 이용자의 내외국인 정보 * 0 : 내국인 / 1: 외국인 * 111 : UCPIDResponse 메시지에서 nationalInfo값이 존재하지않을 경우 |

**UCPIDResponse - getBirthDate**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의birthDate 값을 반환 |
| **◦ 원 형** | public String getBrithDate() |
| **◦ 인자값** | void |
| **◦ 반환값** | statusCode값이 “OK”일 경우 UCPIDResponse의 birthDate 값 반환, statusCode값이 OK가 아닐 경우, null 값 반환. |
| **◦ 비 고** | * birthDate: 이용자의 생년월일 정보 * 형식 : YYYYMMDD |

**UCPIDResponse - getCiUpdate**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의ciupdate 값을 반환 |
| **◦ 원 형** | public int getCiUpdate() |
| **◦ 인자값** | void |
| **◦ 반환값** | 하기의 조건에 모두 만족할 경우, ciupdate값 반환. 조건에 만족하지 않을 경우,null 값 반환.   1. statusCode값이 “OK”일 경우 2. version이 2일 경우 |
| **◦ 비 고** |  |

**UCPIDResponse - getCi**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의ci 값을 반환 |
| **◦ 원 형** | public String getCi() |
| **◦ 인자값** | void |
| **◦ 반환값** | 하기의 조건에 모두 만족할 경우, ci값 반환. 조건에 만족하지 않을 경우, null값 반환.   1. statusCode값이 “OK”일 경우 2. version이 2일 경우 |
| **◦ 비 고** |  |

**UCPIDResponse - getCi2**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의ci2 값을 반환 |
| **◦ 원 형** | public String getCi2() |
| **◦ 인자값** | void |
| **◦ 반환값** | 하기의 조건에 모두 만족할 경우, ci2값 반환. 조건에 만족하지 않을 경우, null값 반환.   1. statusCode값이 “OK”일 경우 2. version이 2일 경우 |
| **◦ 비 고** |  |

**StatusUtils - isOk**

|  |  |
| --- | --- |
| **◦ 기 능** | UCPIDResponse의status 값이 “Ok”인지 확인 |
| **◦ 원 형** | public boolean isOk(String status) |
| **◦ 인자값** | UCPIDResponse의 Status 값 |
| **◦ 반환값** | UCPIDResponse의 Status가 Ok일 경우 true, 그 외의 상태는 false를 반환 |
| **◦ 비 고** | Status : 본인확인 서버로부터 보낸 요청의 응답에 대한 상태값을 의미 |

**VarUtils - getResourceFromProperty**

|  |  |
| --- | --- |
| **◦ 기 능** | route.properties 파일로부터 변수의 value를 획득 |
| **◦ 원 형** | protected String getResourceFromProperty(String fileName) |
| **◦ 인자값** | route.properties 파일 내의 정의된 변수 명 |
| **◦ 반환값** | 성공일 경우, 해당 변수의 value, 실패일 경우 null 값 반환 |
| **◦ 비 고** |  |

**VarUtils - getFromFile**

|  |  |
| --- | --- |
| **◦ 기 능** | 특정 위치의 파일을 byte array 형태로 획득 |
| **◦ 원 형** | protected String getFromFile(String path) |
| **◦ 인자값** | path : 읽어드리고자 할 file 경로 |
| **◦ 반환값** | 성공일 경우, 해당 파일의 byte 값, 실패일 경우 Exception 발생 |
| **◦ 비 고** |  |

**VarUtils – get\_data\_for\_CMSVerify**

|  |  |
| --- | --- |
| **◦ 기 능** | Base64-url-safe encoding된 데이터를 CMS검증모듈에서 사용할 수 있는  형태로 변경 |
| **◦ 원 형** | public String get\_data\_for\_CMSVerify(String encoded\_data) |
| **◦ 인자값** | encoded\_data   * base64-url-safe 형식으로 encoding된 데이터. |
| **◦ 반환값** | * 규격상의 signedPeronInfoReq의 인코딩된 값   성공일 경우, CMS 검증에 사용 가능한 인코딩 형태 String 값 반환 |
| **◦ 비 고** |  |

**VarUtils – base64\_url\_decoding**

|  |  |
| --- | --- |
| **◦ 기 능** | Base64-url-safe로 encoding된 데이터를 decoding하여 원본 데이터 획득 |
| **◦ 원 형** | public byte[] base64\_url\_decoding(String encoded\_data) |
| **◦ 인자값** | encoded\_data   * base64-url-safe 형식으로 encoding된 데이터. * 규격상의 signedPeronInfoReq의 인코딩된 값 |
| **◦ 반환값** | 성공일 경우, 해당 파일의 byte 값, 실패일 경우 Exception 발생 |
| **◦ 비 고** |  |

**MydataException – getErrCode**

|  |  |
| --- | --- |
| **◦ 기 능** | 메시지 생성 및 복호화 과정에서 발생하는 예외와 관련한 에러코드 획득 |
| **◦ 원 형** | public int getErrCode() |
| **◦ 인자값** |  |
| **◦ 반환값** | 발생한 Exception과 관련한 오류코드(integer) 반환 |
| **◦ 비 고** | 101: 함수의 파라미터 값이 존재하지않을 경우  102: byte 데이터를 ASN.1 sequence 객체로의 변환이 실패했을 경우  103: 특정 함수로부터 얻은 값이 존재하지 않을 경우  104: properties 파일에서 해당 변수의 값이 존재하지 않을 경우  105: 주어진 데이터에 대한 ASN.1 구조가 유효하지않을 경우  106: 특정 값을 획득하는 것에 대한 조건이 맞지 않을 경우 |

**1.4 샘플 코드**

**package** SK.Utils;

**import** java.io.DataInputStream;

**import** java.io.DataOutputStream;

**import** java.io.IOException;

**import** java.net.Socket;

**import** SK.Utility.UCPIDResponse;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

String person\_encoded = "마이데이터 사업자로부터 넘어온 signed\_personInfoReq url-safe encoding 된 값";

String consent\_encode = "마이데이터 사업자로부터 넘어온 signed\_consentInfo url-safe encoding 된 값";

//1. url-safe decoding

**byte**[] signed\_personInfoReq = VarUtils.*base64\_url\_decoding*(person\_encoded);

**byte**[] signed\_consentInfo = VarUtils.*base64\_url\_decoding*(consent\_encode);

//2. 검증 데몬(혹은 검증라이브러리)에 넣기 위한 포맷의 서명 데이터 획득

String personInfoForVerify = VarUtils.*get\_data\_for\_CMSVerify*(person\_encoded);

String consentInfoForVerify = VarUtils.*get\_data\_for\_CMSVerify*(consent\_encode);

//3. 각 서명 데이터에서 인증서 byte 값 획득

**byte**[] personCert = ClientUtils.*get\_certificate\_from\_cms*(signed\_personInfoReq);

**byte**[] consentCert = ClientUtils.*get\_certificate\_from\_cms*(signed\_consentInfo);

//4. 각 서명 데이터에서 nonce값 획득

String ucpidNonceInCms = ClientUtils.*get\_ucpidNonce\_from\_signedPersonInfoReq*(signed\_personInfoReq);

String consentNonceInCms = ClientUtils.*get\_consentNonce\_from\_signdConsentInfo*(signed\_consentInfo);

String ucpidNonceInApi = "마이데이터 사업자로부터 api-002를 통해 전달 받은 ucpidNonce 값";

String consentNonceInApi = "마이데이터 사업자로부터 api-002를 통해 전달 받은 consentNonce 값";

**if**(ucpidNonceInCms.equals(ucpidNonceInApi) && consentNonceInCms.equals(consentNonceInApi)){//nonce 값이 동일할 경우 (재전송공격 방지)

**if**(isSameCertificate(personCert,consentCert)){ // byte 배열이 동일한 지 확인하는 함수 정보제공자가 개발 필요

/\*

\* 위의 personInfoForVerify, consentInfoForVerify 값을 검증데몬 혹은 검증라이브러리에 파라미터값으로 넣어 서명 검증 및 인증서 유효성 검증 실시.

\*/

**if**(isVerifyingOk()){ //검증이 정상적으로 완료되었을 경우

//properties 파일 위치 설정. (절대경로 혹은 상대 경로 모두 가능)

VarUtils.*setPropertiesPath*("config\_data/route.properties");

String ucpIdNonce = "정보제공자가 직접 생성한 ucpIdNonce 값";

String cpCode = "정보제공자 기관코드 ";

String cpRequestNumber = "마이데이터사업자로부터 받은 tx\_id 값";

String ca\_code = "마이데이터사업자로부터 받은 인증기관 기관코드";

String signCertPath = VarUtils.*getResourceFromProperty*("signCert");

**byte**[] certificate = VarUtils.*getFromFile*(signCertPath);

**if**("SignKorea".equals(ca\_code)){

String serverIp = "211.175.81.101"; //koscom ucpid test server ip

**int** serverPort = 8098; //koscom ucpid test server port

/\*\*

\* signedAttirbute 존재 > flag : 1

\* signedAttribute 존재 x > flag : 0

\*/

**byte**[] UCPIDRequest = RequestUtils.*getUCPIDRequest*(ucpIdNonce, cpCode, cpRequestNumber, certificate, signed\_personInfoReq, 1);

System.***out***.println("request to ucpid server is started.");

**byte**[] bUCPIDResponse = *request2UCPID*(UCPIDRequest, serverIp, serverPort);

System.***out***.println("request to ucpid server is successfully done.");

/\*\*

\* get ucpidResponse's status

\* status가 OK일 경우에만 ucpidResponse의 모든 데이터가 정상적으로 존재

\*/

String status = ResponseUtils.*getStatusCode*(bUCPIDResponse);

**if**(StatusUtils.*isOk*(status)){

UCPIDResponse ucpidResponse = ResponseUtils.*getInstance*(bUCPIDResponse);

**int** version = ucpidResponse.getVersion();

String UCPIDNonce = ucpidResponse.getUcpidNonce();

String CpRequestNumber = ucpidResponse.getCpRequestNumber();

String certDn = ucpidResponse.getCertDn();

String CpCode = ucpidResponse.getCpCode();

String di = ucpidResponse.getDi();

String realName = ucpidResponse.getRealName();

**int** gender = ucpidResponse.getGender();

**int** nationalInfo = ucpidResponse.getNationalInfo();

String birthDate = ucpidResponse.getBirthDate();

**int** ciupdate = ucpidResponse.getCiUpdate();

String ci = ucpidResponse.getCi();

String ci2 = ucpidResponse.getCi2();

} **else** {

System.***out***.println("your UCPIDResponse is invalid. your error code is \"" + status + "\"");

}

}

}

}

}

}

**public** **static** **byte**[] request2UCPID(**byte**[] UCPIDRequest,String serverIp, **int** serverPort){

Socket socket = **null**;

DataInputStream reader = **null**;

DataOutputStream writer = **null**;

**byte**[] UCPIDResponse = **null**;

**try**{

socket = **new** Socket(serverIp, serverPort);

**if**(socket != **null**){

**if**(socket.isConnected()){

writer = **new** DataOutputStream(socket.getOutputStream());

**if**(writer != **null**){

writer.writeInt(UCPIDRequest.length);

writer.write(UCPIDRequest,0,UCPIDRequest.length);

writer.flush();

}

reader = **new** DataInputStream(socket.getInputStream());

**if**(reader!=**null**){

**int** readInt = reader.readInt();

UCPIDResponse = **new** **byte**[readInt];

reader.readFully(UCPIDResponse);

}

}

}

}**catch**(IOException e){

e.printStackTrace();

}**finally**{

**try** {

**if**(socket!=**null**) socket.close();

**if**(writer!=**null**) writer.close();

**if**(reader!=**null**) reader.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**return** UCPIDResponse;

}

}