“운전자격확인시스템(RIMS) 구축”

사용자/API 서버 간 통신규약

Ver. 1.0

한국교통안전공단

2024. 06.17

<제목 차례>

1 개요

1.1 목적

- 운전자격확인시스템은 주로 대여사업자와 지자체 담당자를 위한 시스템으로, 대여 전 운전자의 운전자격 확인 기능을 제공한다. 본 문서는 검증 API를 호출함에 있어서의 통신규약을 정의한다.

2 목록

2.1 용어정의

- 본 문서에서 사용하는 용어는 다음과 같다.

용어	설명
RIMS	- 운전자격확인시스템

2.2 기능정의

- 사용자는 인증키와 Secret Key를 시스템(https://rims.kotsa.or.kr)의 OpenAPI 메뉴를 통해 발급한다.
- 사용자는 인증키를 기반으로 인증 토큰을 발급한다.
- 인증 토큰은 발급마다 신규로 생성되며 유효 시간이 초기화된다.
- 사용자는 발급한 토큰을 첨부하여 운전자격확인 API를 호출한다.
- 시스템은 사용자의 호출정보로 도로교통공단 운전자격확인API를 호출하여 결과를 받는다.
- 시스템은 이력을 저장하고 검증 결과를 사용자에게 리턴한다.

3 상세

3.1 송신 데이터의 형식

- 데이터는 암호화 후 Base64로 인코딩하여 “encryptedData” 를 key로 Body에 JSON타입으로 구성한다.
- 데이터 송수신은 HTTP 1.1 방식을 따르며 몇 가지 HTTP Header를 추가로 정의하여 사용한다.

3.2 보안 방법

3.2.1 암호화 알고리즘

- AES (Advanced Encryption Standard) 암호화 알고리즘을 사용하여 암호화하고

데이터는 Base64 인코딩한다.

- 이때, 표준 옵션인 ECB 모드와 PKCS5 패딩으로 진행해야 한다.

3.2.2 암호화 키 정보

- 시스템에서 발급받은 Secret Key 정보를 통해 데이터를 암호화 한다.

3.2.3 암호화 알고리즘 제공 방식

- 각 언어의 표준 라이브러리나 서드파티 라이브러리의 문서를 참조

3.3 운전자격확인 API 사용 시 차량 정보 공통(소화물배송대행서비스사업자 API 제외)

- 자격확인 시 차량번호는 필수 값이며 차량배치 혹은 차량번호가 미정일 경우 “99임9999” 의 임시 차량번호를 입력한다.

- 임시차량번호의 자격확인 결과 값은 정상 차량번호와 동일하지만 정상적인 운전자격확인의 증빙으로 사용할 수 없으며 정상 차량번호로 추후 운전자격확인하여야 한다.

- 전산(국토교통부 자동차관리 전산망)에 등록된 차량의 경우, 차량 확인 코드(‘vhcl_idnty_cd‘)는 1로 반환되며, 이는 운전자격확인 확인을 의미한다.

- 전산에 등록된 차량의 기준은 요청 시 ‘bizinfo‘ 항목이 첨부되었을 경우 해당 사업자로 등록된 차량을 기준으로 하되, 미첨부 시 요청자의 사업자 차량을 기준으로 한다.

- 차량 확인 코드(‘vhcl_idnty_cd‘)가 2(미확인)로 반환된 경우, 해당 차량의 확인 유예 기간이 한 달로 적용된다.

3.4 데이터 Request 목록

3.4.1 request 목록

인터페이스명	Request uri	내용
토큰 발급	/col/oauth2	토큰 발급을 위한 인터페이스
운전자격확인 단건	/licenseVerification	렌터카 사업자용 운전자격확인조회를 위한 인터페이스
운전자격확인 배치	/licenseVerificationBatch	렌터카 사업자용 복수의 운전자에 대한 운전자격확인조회를 위한 인터페이스
사용자검증	/col/isUserInDatabase	DB에 등록된 사용자인지 검증에 대한 인터페이스
소화물배송대행서비스사업자 운전자격확인 단건	/dapi/licenseVerification	소화물배송대행서비스사업자용 운전자격확인조회를 위한 인터페이스
소화물배송대행서비스사업자 운전자격확인 배치	/dapi/licenseVerificationBatch	소화물배송대행서비스사업자용 복수의 운전자에 대한 운전자격확인조회를 위한 인터페이스

3.5 사용자 인증

3.5.1 토큰발급

- OAuth2 인증방식에 따라 제공되는 인증키로 인증 토큰을 발급받아 Token을 통해 운전자격확인 API 호출을 한다.

3.5.2 인증 URL

환경	API명	URL
운영	토큰 발급	https://rims.kotsa.or.kr:8114/col/oauth2

3.5.3 토큰 발급 헤더

필드명	상태	내용	필수여부	비고
host	Req	웹서버 정보	생략가능	
Authorization	Req	유효한 토큰 정보	필수	"Basic " + 인증키

- 토큰 발급 시 HTTP Request (클라이언트 → 인증서버)

```
GET /col/oauth2?grantType=password
Host: rims.kotsa.or.kr
Authorization: "Basic " + Base64 인코딩(인증키)
ex) Basic YTFiO...
```

- 요청을 전송할 때는 위와 같은 형태로 전송하여야 한다.
- 위 표의 빨간색 글꼴이 있는 부분은 파라미터로 포함한다.
- 위 표의 파란색 글꼴이 있는 부분은 인증키를 Base64로 인코딩한다.
- 요청 본문은 사용하지 않는다.

- 토큰 발급 시 인증서버의 HTTP Response (인증서버 → 클라이언트)

```
HTTP 200 OK
Authorization: Bearer 토큰 값
```

- 인증서버는 HTTP 응답메시지를 위의 표와 같이 클라이언트에게 전송한다. 위의 표는 응답코드(HTTP Status code)가 200일 경우의 예이다.

3.6 운전자격확인 단건 조회

3.6.1 API 호출

- 발급받은 Token을 통해 면허소유자를 검증한다.

3.6.2 운전자격확인 단건 조회 URL

환경	API명	URL
운영	운전자격확인	https://rims.kotsa.or.kr:8114/licenseVerification

3.6.3 운전자격확인 헤더

필드명	상태	내용	필수여부	비고
host	Req	웹서버 정보	생략가능	
Authorization	Req	유효한 토큰 정보	필수	"Bearer " + token

- 운전자격확인 시 HTTP Request (클라이언트 → 인증서버)

```
POST /licenseVerification
Host: rims.kotsa.or.kr
Authorization: “Bearer ” + 토큰
ex) Bearer e03ee90d-83b8-4b1d-ae80-84e3d2a27ytr
```

- 요청을 전송할 때는 위와 같은 형태로 전송하여야 한다.

- 운전자격확인 시 인증서버의 HTTP Response (인증서버 → 클라이언트)

```
HTTP 200 OK
```

- 인증서버는 HTTP 응답메시지를 위의 표와 같이 클라이언트에게 전송한다. 위의 표는 응답코드(HTTP Status code)가 200일 경우의 예이다.

- 토큰 검증 실패 시 401 Unauthorized를 반환한다.

3.6.4 운전자격확인 바디

항목명	항목 Key	항목 형식	길이	필수여부	내용
사용자정보	bizinfo	String	32	API 호출 사용자와 일치 할 경우 생략 가능	사용자ID
운전면허번호	f_license_no	String	12	필수	
운전면허자명	f_resident_name	String	40	필수	
면허종별(코드)	f_licn_con_code	String	2	필수	
대여기간시작	f_from_date	String	8	필수	
대여기간종료	f_to_date	String	8	필수	
차량등록번호	vhcl_reg_no	String	16	필수	임시차량의 경우 “99임9999”

- Body 정보는 “3.2 보안방법”과 같이 암호화한다.
- API 호출 인증키의 사용자와 검증요청 차량의 소유자가 일치 하지 않는 플랫폼, 대행사, 조합 등의 경우 사용자정보(bizinfo)를 차량 소유자의 사용자 ID를 입력한다.
- 암호화한 데이터는 “3.1 송신 데이터의 형식”에 맞게 본문을 구성한다.
- 운전면허번호(f_license_no)은 총 12자리이며, 면허증 운전면허번호 앞에 지역(서울, 인천 등) 이름이 쓰여진 경우 “4.5 지역 코드표”를 참고하여 지역 명을 숫자로 치환하여 전달한다.
- 면허종별(f_licn_con_code)은 총 2자리이며, “4.4 면허 종별 코드표”를 참고하여 숫자로 치환하여 전달한다.

3.6.5 단건 조회 예시

```
POST https://rims.kotsa.or.kr:8114/licenseVerification
Authorization: Bearer ...
Content-Type: application/json
Content-Length: ...



암호화 전
```

```
{
  "bizinfo": "asdf12341....",
  "f_license_no": "221212121212",
  "f_resident_name": "홍길동",
  "f_licn_con_code": "12",
  "f_from_date": "20241023",
  "f_to_date": "20241023",
  "vhcl_reg_no": "67라6767"
}
```


암호화 후


```
{
  "encryptedData": "G+joSdT..."
}
```


응답샘플


```
HTTP/1.1 200 OK

{
    "header": {
        "f_rtn_msg": "",
        "f_request_date": "20231010",
        "f_rtn_cd": "0",
        "f_send_cnt": "1"
    },
    "body": {
        "vhcl_idnty_cd": "1",
        "f_rtn_code": "00",
        "f_license_no": "2*1*1*1*2*2"
    }
}
```

3.7 운전자격확인 단건 조회 응답(response) Body 목록

3.7.1 Header

항목명	항목 Key	항목 형식	내용
요청건수	f_send_cnt	String	요청건수(Default1)
요청일시	f_request_date	String	요청일시 (형식 : YYYYMMDD)
요청처리	f_rtn_cd	String	요청처리코드
요청처리실패메세지	f_rtn_msg	String	요청처리실패메세지

3.7.2 Body

항목명	항목 Key	항목 형식	내용
차량확인코드	vhcl_idnty_cd	String	(1:확인, 2:미확인)
검증결과코드	f_rtn_code	String	검증결과코드 (적격 : 00, 부적격 : 01~)
운전면허번호	f_license_no	String	번호는 1*1*3*5*1*1* 형태로 마스킹 처리

- 응답은 “3.6.5 단건 조회 예시”를 참고한다.
- “3.7.1 Header”의 f_rtn_cd는 “4.3 요청처리 응답 코드표”를 참고한다.
- “3.7.1 Header”의 f_rtn_cd가 0이 아닐 경우 “3.7.2 Body”는 생략된다.

3.8 운전자격확인 배치 조회

3.8.1 API 호출

- 발급받은 Token을 통해 복수의 면허소유자를 검증한다.

3.8.2 운전자격확인 배치 URL

환경	API명	URL
운영	운전자격확인	https://rims.kotsa.or.kr:8114/licenseVerificationBatch

3.8.3 운전자격확인 헤더

필드명	상태	내용	필수여부	비고
host	Req	웹서버 정보	생략가능	
Authorization	Req	유효한 토큰 정보	필수	"Bearer " + token

- 운전자격확인 시 HTTP Request (클라이언트 → 인증서버)

```
POST /licenseVerification
Host: rims.kotsa.or.kr
Authorization: "Bearer " + 토큰
ex) Bearer e03ee90d-83b8-4b1d-ae80-84e3d2a27ytr
```

- 요청을 전송할 때는 위와 같은 형태로 전송하여야 한다.

- 운전자격확인 시 인증서버의 HTTP Response (인증서버 → 클라이언트)

```
HTTP 200 OK
```

- 인증서버는 HTTP 응답메시지를 위의 표와 같이 클라이언트에게 전송한다. 위의 표는 응답코드(HTTP Status code)가 200일 경우의 예이다.

- 토큰 검증 실패 시 401 Unauthorized를 반환한다.

3.8.4 운전자격확인 바디

항목명	항목 Key	항목 형식	길이	필수여부	내용
요청건수	f_send_cnt	String	4	필수	리스트 인덱스와 일치(최대 1,000개)
사용자정보	bizinfo	String	32	API 호출 사용자와 일치 할 경우 생략 가능	사용자ID
요청리스트	requestList	String		필수	복수 요청의 키
운전면허번호	f_license_no	String	12	필수	
운전면허자명	f_resident_name	String	40	필수	
면허종별(코드)	f_licn_con_code	String	2	필수	
대여기간시작	f_from_date	String	8	필수	
대여기간종료	f_to_date	String	8	필수	
차량등록번호	vhcl_reg_no	String	16	필수	임시차량의 경우 “99임9999”

- Body 정보는 “3.2 보안방법”과 같이 암호화한다.

- 암호화한 데이터는 “3.1 송신 데이터의 형식”에 맞게 본문을 구성한다.

- 운전면허번호(f_license_no)은 총 12자리이며, 면허증 운전면허번호 앞에 지역

(서울, 인천 등) 이름이 쓰여진 경우 “4.5 지역 코드표”를 참고하여 지역 명을 숫자로 치환하여 전달한다.

- 면허종별(f_licn_con_code)은 총 2자리이며, “4.4 면허 종별 코드표”를 참고하여 숫자로 치환하여 전달한다.
- 요청건수는 요청리스트의 인덱스와 일치해야한다.

3.8.5 배치 조회 예시

```
POST https://lrims.kotsa.or.kr:8114/licenseVerification
Authorization: Bearer ...
Content-Type: application/json
Content-Length: ...




암호화 전

{
  "f_send_cnt": "2",
  "bizinfo": "asdf12341....",
  "requestList": [{
            "f_license_no": "221212121212",
            "f_resident_name": "홍길동",
            "f_licn_con_code": "12",
            "f_from_date": "20241023",
            "f_to_date": "20241023",
            "vhcl_reg_no": "67라6767"
  },
  {
            "f_license_no": "221212121212",
            "f_resident_name": "홍길동",
            "f_licn_con_code": "12",
            "f_from_date": "20241023",
            "f_to_date": "20241023"
            "vhcl_reg_no": "67라6767"
  }]
}

암호화 후

{
```

```
  "encryptedData": "G+joSdT..."
}
```

응답샘플

```
HTTP/1.1 200 OK

{
        "header" : {
                        "f_send_cnt " : "2",
                        "f_request_date" : "20220922122321",
                        "f_rtn_yn" : "00",
                        "f_rtn_msg" : ""
        }
        "body" : [{
                        "vhcl_idnty_cd": "1",
                        "f_license_no" : "121234561212",
                        "f_rtn_code" : "25"
        },
        {
                        "vhcl_idnty_cd": "1",
                        "f_license_no" : "121234561212",
                        "f_rtn_code" : "00"
        }]
}
```

3.9 운전자격확인 배치 조회 응답(response) Body 목록

3.9.1 Header

항목명	항목 Key	항목 형식	내용
요청건수	f_send_cnt	String	요청건수(Default1)
요청일시	f_request_date	String	요청일시 (형식 : YYYYMMDD)
요청처리	f_rtn_cd	String	요청처리코드
요청처리실패메세지	f_rtn_msg	String	요청처리실패메세지

3.9.2 Body

항목명	항목 Key	항목 형식	내용
차량확인코드	vhcl_idnty_cd	String	(1:확인, 2:미확인)
검증결과코드	f_rtn_code	String	검증결과코드 (적격 : 00, 부적격 : 01~)
운전면허번호	f_license_no	String	번호는 1*1*3*5*1*1* 형태로 마스킹 처리

- 응답은 “3.8.5 배치 조회 예시”를 참고한다.
- “3.9.1 Header”의 f_rtn_cd는 “4.3 요청처리 응답 코드표”를 참고한다.
- “3.9.1 Header”의 f_rtn_cd가 0이 아닐 경우 “3.9.2 Body”는 생략된다.

3.10 사용자검증

3.10.1 API 호출

- 사용자ID와 사업자등록번호로 시스템에 등록된 사용자인지 검증한다.
- 플랫폼, 대행사, 조합 등 API인증키의 사용자와 차량 소유주가 일치하지 않을 경우 사용자정보(bizinfo) 값의 정합성 체크를 위한 API

3.10.2 사용자검증 URL

환경	API명	URL
운영	사용자검증	https://rims.kotsa.or.kr:8114/col/isUserInDatabase

3.10.3 사용자검증 헤더

필드명	상태	내용	필수여부	비고
host	Req	웹서버 정보	생략가능	
Authorization	Req	유효한 토큰 정보	필수	“Bearer ” + token

- 사용자검증 시 HTTP Request (클라이언트 → 인증서버)

```
GET /col/isUserInDatabase?user_id=test&brno=1234567890
Host: rims.kotsa.or.kr
Authorization: “Bearer ” + 토큰
ex) Bearer e03ee90d-83b8-4b1d-ae80-84e3d2a27ytr
```

- 요청을 전송할 때는 위와 같은 형태로 전송하여야 한다.
- 위 표의 빨간색 글꼴이 있는 부분은 파라미터로 포함한다.
- 위 표의 파란색 글꼴이 있는 부분은 사용자 ID, 사업자등록번호 순으로 포함한다.
- 사용자검증 시 인증서버의 HTTP Response (인증서버 → 클라이언트)

```
HTTP 200 OK
```

- 인증서버는 HTTP 응답메시지를 위의 표와 같이 클라이언트에게 전송한다. 위의 표는 응답코드(HTTP Status code)가 200일 경우의 예이다.
- 토큰 검증 실패 시 401 Unauthorized를 반환한다.

3.10.4 사용자검증 응답 예시

```
응답샘플

HTTP/1.1 200 OK

등록
{
    "aprv_stts_cd": "2",
    "acnt_stts_cd": "1",
    "user_id": "test",
    "registration_status": "registered",
    "brno": "1234567890"
}

미등록
{
```

```
    "user_id": test",
    "registration_status": "unregistered",
    "brno": "1234567890"
}
```


3.11 사용자검증 응답(response) Body 목록

항목명	항목 Key	항목 형식	내용
승인상태코드	aprv_stts_cd	String	승인 상태 코드표 참조
계정상태코드	acnt_stts_cd	String	계정 상태 코드표 참조
사용자ID	user_id	String	요청 사용자ID
등록여부	registration_status	String	unregistered(미등록), registered(등록)
사업자등록번호	brno	String	요청 사업자등록번호

- 응답은 “3.10.4 사용자검증 응답 예시”를 참고한다.
- 계정상태코드는 “4.6 계정 상태 코드표”를 참고한다.
- 승인상태코드는 “4.7 승인 상태 코드표”를 참고한다.

3.12 소화물배송대행서비스사업자 운전자격확인 단건 조회

3.12.1 API 호출

- 발급받은 Token을 통해 면허소유자를 검증한다.

3.12.2 운전자격확인 단건 조회 URL

환경	API명	URL
운영	소화물배송대행서비스 사업자 운전자격확인 단건 조회	https://rims.kotsa.or.kr:8114/dapi/licenseVerification

3.12.3 운전자격확인 헤더

필드명	상태	내용	필수여부	비고
host	Req	웹서버 정보	생략가능	
Authorization	Req	유효한 토큰 정보	필수	"Bearer " + token

- 운전자격확인 시 HTTP Request (클라이언트 → 인증서버)

```
POST /dapi/licenseVerification
Host: rims.kotsa.or.kr
Authorization: “Bearer ” + 토큰
ex) Bearer e03ee90d-83b8-4b1d-ae80-84e3d2a27ytr
```

- 요청을 전송할 때는 위와 같은 형태로 전송하여야 한다.

- 운전자격확인 시 인증서버의 HTTP Response (인증서버 → 클라이언트)

```
HTTP 200 OK
```

- 인증서버는 HTTP 상태코드를 위의 표와 같이 클라이언트에게 전송한다. 위의 표는 응답코드(HTTP Status code)가 200일 경우의 예이다.

- 토큰 검증 실패 시 401 Unauthorized를 반환한다.

3.12.4 운전자격확인 바디

항목명	항목 Key	항목 형식	길이	필수여부	내용
사용자정보	bizinfo	String	32	선택	사용자ID
운전면허번호	f_license_no	String	12	필수	
운전면허자명	f_resident_name	String	40	필수	
면허종별(코드)	f_licn_con_code	String	2	필수	
검증기간시작	f_from_date	String	8	필수	
검증기간종료	f_to_date	String	8	필수	
차량등록번호	vhcl_reg_no	String	16	선택	

- Body 정보는 “3.2 보안방법”과 같이 암호화한다.

- 암호화한 데이터는 “3.1 송신 데이터의 형식”에 맞게 본문을 구성한다.

- 운전면허번호(f_license_no)은 총 12자리이며, 면허증 운전면허번호 앞에 지역(서울, 인천 등) 이름이 쓰여진 경우 “4.5 지역 코드표”를 참고하여 지역 명을 숫자로 치환하여 전달한다.

- 면허종별(f_licn_con_code)은 총 2자리이며, “4.4 면허 종별 코드표”를 참고하여 숫자로 치환하여 전달한다.

3.12.5 단건 조회 예시

```
POST https://rims.kotsa.or.kr:8114/dapi/licenseVerification
Authorization: Bearer ...
Content-Type: application/json
Content-Length: ...
```

암호화 전

```
{
  "bizinfo": "asdf12341....",
  "f_license_no": "221212121212",
  "f_resident_name": "홍길동",
  "f_licn_con_code": "12",
  "f_from_date": "20241023",
  "f_to_date": "20241023",
  "vhcl_reg_no": "67라6767"
}
```

암호화 후

```
{
  "encryptedData": "G+joSdT..."
}
```

응답샘플

```
HTTP/1.1 200 OK

{
    "header": {
        "f_rtn_msg": "",
        "f_request_date": "20231010",
        "f_rtn_cd": "0",
        "f_send_cnt": "1"
    },
```

```
    "body": {
        "f_rtn_code": "00",
        "f_license_no": "2*1*1*1*2*2"
    }
}
```

3.13 소화물배송대행서비스사업자 운전자격확인 단건 조회 응답 (response) Body 목록

3.13.1 Header

항목명	항목 Key	항목 형식	내용
요청건수	f_send_cnt	String	요청건수(Default1)
요청일시	f_request_date	String	요청일시 (형식 : YYYYMMDD)
요청처리	f_rtn_cd	String	요청처리코드
요청처리실패메세지	f_rtn_msg	String	요청처리실패메세지

3.13.2 Body

항목명	항목 Key	항목 형식	내용
검증결과코드	f_rtn_code	String	검증결과코드 (적격 : 00, 부적격 : 01~)
운전면허번호	f_license_no	String	번호는 1*1*3*5*1*1* 형태로 마스킹 처리

- 응답은 “3.12.5 단건 조회 예시”를 참고한다.
- “3.13.1 Header”의 f_rtn_cd는 “4.3 요청처리 응답 코드표”를 참고한다.
- “3.13.1 Header”의 f_rtn_cd가 0이 아닐 경우 “3.13.2 Body”는 생략된다.

3.14 소화물배송대행서비스사업자 운전자격확인 배치 조회

3.14.1 API 호출

- 발급받은 Token을 통해 복수의 면허소유자를 검증한다.

3.14.2 운전자격확인 배치 URL

환경	API명	URL
운영	소화물배송대행서비스 사업자 운전자격확인 배치 조회	https://rims.kotsa.or.kr:8114/dapi/licenseVerificationBatch

3.14.3 운전자격확인 헤더

필드명	상태	내용	필수여부	비고
host	Req	웹서버 정보	생략가능	
Authorization	Req	유효한 토큰 정보	필수	"Bearer " + token

- 운전자격확인 시 HTTP Request (클라이언트 → 인증서버)

```
POST /dapi/licenseVerificationBatch
Host: rims.kotsa.or.kr
Authorization: "Bearer " + 토큰
ex) Bearer e03ee90d-83b8-4b1d-ae80-84e3d2a27ytr
```

- 요청을 전송할 때는 위와 같은 형태로 전송하여야 한다.

- 운전자격확인 시 인증서버의 HTTP Response (인증서버 → 클라이언트)

```
HTTP 200 OK
```

- 인증서버는 HTTP 응답메시지를 위의 표와 같이 클라이언트에게 전송한다. 위의 표는 응답코드(HTTP Status code)가 200일 경우의 예이다.

- 토큰 검증 실패 시 401 Unauthorized를 반환한다.

3.14.4 운전자격확인 바디

항목명	항목 Key	항목 형식	길이	필수여부	내용
요청건수	f_send_cnt	String	4	필수	리스트 인덱스와 일치(최대 1,000개)
사용자정보	bizinfo	String	32	선택	사용자ID
요청리스트	requestList	String		필수	복수 요청의 키
운전면허번호	f_license_no	String	12	필수	
운전면허자명	f_resident_name	String	40	필수	
면허종별(코드)	f_licn_con_code	String	2	필수	
대여기간시작	f_from_date	String	8	필수	
대여기간종료	f_to_date	String	8	필수	
차량등록번호	vhcl_reg_no	String	16	선택	

- Body 정보는 “3.2 보안방법”과 같이 암호화한다.
- 암호화한 데이터는 “3.1 송신 데이터의 형식”에 맞게 본문을 구성한다.
- 운전면허번호(f_license_no)은 총 12자리이며, 면허증 운전면허번호 앞에 지역(서울, 인천 등) 이름이 쓰여진 경우 “4.5 지역 코드표”를 참고하여 지역 명을 숫자로 치환하여 전달한다.
- 면허종별(f_licn_con_code)은 총 2자리이며, “4.4 면허 종별 코드표”를 참고하여 숫자로 치환하여 전달한다.
- 요청건수는 요청리스트의 인덱스와 일치해야한다.

3.14.5 배치 조회 예시

```
POST https://lrims.kotsa.or.kr:8114/dapi/licenseVerificationBatch
Authorization: Bearer ...
Content-Type: application/json
Content-Length: ...




암호화 전

{
  "f_send_cnt": "2",
  "bizinfo": "asdf12341....",
  "requestList": [{
            "f_license_no": "221212121212",
            "f_resident_name": "홍길동",
            "f_licn_con_code": "12",
            "f_from_date": "20241023",
            "f_to_date": "20241023",
            "vhcl_reg_no": "67라6767"
  },
  {
            "f_license_no": "221212121212",
            "f_resident_name": "홍길동",
            "f_licn_con_code": "12",
            "f_from_date": "20241023",
            "f_to_date": "20241023"
            "vhcl_reg_no": "67라6767"
  }]
}
```

암호화 후


```
{
  "encryptedData": "G+joSdT..."
}
```


응답샘플

HTTP/1.1 200 OK


```
{
        "header" : {
                        "f_send_cnt " : "2",
                        "f_request_date" : "20220922122321",
                        "f_rtn_yn" : "00",
                        "f_rtn_msg" : ""
        }
        "body" : [{
                        "vhcl_idnty_cd": "1",
                        "f_license_no" : "121234561212",
                        "f_rtn_code" : "25"
        },
        {
                        "vhcl_idnty_cd": "1",
                        "f_license_no" : "121234561212",
                        "f_rtn_code" : "00"
        }]
}
```


3.15 소화물배송대행서비스사업자 운전자격확인 배치 조회 응답 (response) Body 목록

3.15.1 Header

항목명	항목 Key	항목 형식	내용
요청건수	f_send_cnt	String	요청건수(Default1)
요청일시	f_request_date	String	요청일시 (형식 : YYYYMMDD)
요청처리	f_rtn_cd	String	요청처리코드
요청처리실패메세지	f_rtn_msg	String	요청처리실패메세지

3.15.2 Body

항목명	항목 Key	항목 형식	내용
차량확인코드	vhcl_idnty_cd	String	(1:확인, 2:미확인)
검증결과코드	f_rtn_code	String	검증결과코드 (적격 : 00, 부적격 : 01~)
운전면허번호	f_license_no	String	번호는 1*1*3*5*1*1* 형태로 마스킹 처리

- 응답은 “3.14.5 배치 조회 예시”를 참고한다.
- “3.15.1 Header”의 f_rtn_cd는 “4.3 요청처리 응답 코드표”를 참고한다.
- “3.15.1 Header”의 f_rtn_cd가 0이 아닐 경우 “3.15.2 Body”는 생략된다.

3.16 공통 API 오류 응답 예시

```
응답샘플

HTTP/1.1 200 OK

요청한 API 가 제공되지 않을 경우
{
    "respCode": -100,
    "errorMsg": "API를 찾을 수 없습니다."
}

bizinfo에 첨부된 아이디가 유효하지 않을 경우
{
    "respCode": -130,
    "errorMsg": "bizinfo에 첨부된 아이디가 유효하지 않습니다."
}
이하 생략
```

…

- 응답은 “4.1 공통 API 오류 응답 코드표”를 참고한다.

4 첨부표

4.1 공통 API 오류 응답 코드표

코드명	코드 값	내용
SERVER_CONNECT_ERROR	-90	연계 서버와 연결 실패
NOT_FOUND_API	-100	요청한 API 가 제공되지 않을 경우
RIMS_DATABASE_ERROR	-110	운전자격확인시스템 데이터베이스 오류
BIZINFO_ID_INVALID	-130	bizinfo에 첨부된 아이디가 유효하지 않을 경우
CLIENT_AUTHENTICATION_FAILURE	-140	클라이언트 인증 실패
TOKEN_GENERATION_ERROR	-150	토큰 생성 중 오류 발생
INVALID_PARAMETER	-160	유효하지 않은 파라미터
MISSING_PARAMETER	-170	파라미터 누락
INVALID_ENCRYPTED_DATA	-180	규약 위반: encryptedData 필드가 존재하지 않거나 문자열이 아닌 경우
DECRYPTION_FAILURE	-190	복호화 실패: 데이터 복호화에 실패한 경우
INVALID_ARRAY_SIZE_MESSAGE	-200	배열 크기와 요청 건수 불일치
KOROAD_API_CALL_FAILURE	-210	도로교통공단 API 호출 실패

4.2 면허정보 응답 코드표

코드명	코드 값	내용
검증결과코드(코드 값과 동일)	00	정상
검증결과코드(코드 값과 동일)	01	면허번호 없음
검증결과코드(코드 값과 동일)	02	재발급된 면허
검증결과코드(코드 값과 동일)	03	분실 된 면허
검증결과코드(코드 값과 동일)	04	사망 취소된 면허
검증결과코드(코드 값과 동일)	11	취소 된 면허
검증결과코드(코드 값과 동일)	12	정지 된 면허
검증결과코드(코드 값과 동일)	13	기간 중 취소 면허
검증결과코드(코드 값과 동일)	14	기간 중 정지 면허
검증결과코드(코드 값과 동일)	21	정보불일치(이름)
검증결과코드(코드 값과 동일)	22	정보불일치(생년월일)
검증결과코드(코드 값과 동일)	23	정보불일치(암호일련번호)
검증결과코드(코드 값과 동일)	24	정보불일치(종별)
검증결과코드(코드 값과 동일)	25	필수값 누락(대여기간)
검증결과코드(코드 값과 동일)	31	암호화 안 된 면허

4.3 요청처리 응답 코드표

코드명	코드 값	내용
요청처리(코드 값과 동일)	0	처리 완료
요청처리(코드 값과 동일)	1	인증 정보 없음 (인증 토큰 없음)
요청처리(코드 값과 동일)	2	잘못된 인증 정보 (인증 토큰 값 오류)
요청처리(코드 값과 동일)	3	인증 실패
요청처리(코드 값과 동일)	4	만료된 토큰 정보
요청처리(코드 값과 동일)	10	잘못된 경로를 통한 접근 (등록된 IP와 다름)
요청처리(코드 값과 동일)	20	복호화 키 정보 없음 (사용자 정보 중 복호화에 사용될 비밀 값이 없음)
요청처리(코드 값과 동일)	21	메시지 복호화 실패
요청처리(코드 값과 동일)	22	메시지 암호화 실패
요청처리(코드 값과 동일)	40	수수료 결제 정보 없음
요청처리(코드 값과 동일)	41	수수료 결제 중 오류
요청처리(코드 값과 동일)	97	자동 검증 시스템 작업 장애
요청처리(코드 값과 동일)	98	경찰청 운전 면허 조회 장애
요청처리(코드 값과 동일)	99	자동 검증 시스템 장애

4.4 면허 종별 코드표

코드	코드명
11	1종대형
12	1종보통
13	1종소형
14	대형견인차
15	구난차
16	소형견인차
32	2종보통
33	2종소형
38	2종원자

4.5 지역 코드표

코드	코드명
11	서울
12	부산
13	경기
13	경기남부
14	강원
15	충북
16	충남
17	전북
18	전남
19	경북
20	경남
21	제주
22	대구
23	인천
24	광주
25	대전
26	울산
28	경기북부

4.6 계정 상태 코드표

코드	코드명
1	정상
2	잠김
3	휴면
4	정지
5	삭제
6	탈퇴

4.7 승인 상태 코드표

코드	코드명
1	요청
2	승인
3	반려

5 개발 예시

- 아래 예시는 Java로 작성한 참고용 예시이며 환경과 사용하는 라이브러리에 따라 코드는 달라질 수 있음.

```
public class ExService {

    private final WebClient webClient;

    public ExService(WebClient webClient) {
        this.webClient = webClient;
    }

    private static final int AES_KEY_LENGTH = 256;
    private static final String INSTANCE_TYPE = "AES/ECB/PKCS5Padding";
    private static final String CLIENT_SEC = "1234b0842f2848c090a0835c6e7ac599";

    // 토큰 발생
    public void getToken() throws UnsupportedEncodingException {
        String certKey = "사용자 인증키";
        String uri = "http://서버도메인/col/oauth2?grantType=password";
        String authHeader = "Basic " +
Base64.getEncoder().encodeToString(certKey.getBytes("UTF-8"));
        webClient.get()
                .uri(uri)
                .header(HttpHeaders.AUTHORIZATION, authHeader)
                .exchange()
                .flatMap(response -> {
                    HttpHeaders headers = response.headers().asHttpHeaders();
                    List<String> authHeaderValues =
headers.get(HttpHeaders.AUTHORIZATION);
                    String authorizationHeaderValue = authHeaderValues.get(0);
                    log.info("토큰 : {}", authorizationHeaderValue);
                    return Mono.empty();
                })
                .doOnError(error -> {
                    log.error("에러 : {}", error.getMessage());
                }).subscribe();
    }

    public static String encrypt(String plaintext, String secretKey) throws Exception {
        SecretKeySpec keySpec = generateKey(secretKey);
        Cipher cipher = Cipher.getInstance(INSTANCE_TYPE);
        cipher.init(Cipher.ENCRYPT_MODE, keySpec);
        byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes(StandardCharsets.UTF_8));
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }
```

```
    private static SecretKeySpec generateKey(String secretKey) {
        byte[] keyData = Arrays.copyOf(secretKey.getBytes(StandardCharsets.UTF_8),
AES_KEY_LENGTH / 8);
        return new SecretKeySpec(keyData, "AES");
    }

    public static String single = "{\n" +
            "   \"f_license_no\": \"123456789012\",\n" +
            "   \"f_resident_name\": \"홍길동\",\n" +
            "   \"f_licn_con_code\": \"12\",\n" +
            "   \"f_from_date\": \"20231106\",\n" +
            "   \"f_to_date\": \"20231106\",\n" +
            "   \"vhcl_reg_no\": \"99임9999\"\n" +
            "}";

    public static String batch = "{\n" +
            "   \"f_send_cnt\": \"2\",\n" +
            "   \"bizinfo\": \"asdf12341....\",\n" +
            "   \"requestList\": [\n" +
            "     {\n" +
            "        \"f_license_no\": \"123456789012\",\n" +
            "        \"f_resident_name\": \"홍길동\",\n" +
            "        \"f_licn_con_code\": \"12\",\n" +
            "        \"f_from_date\": \"20231106\",\n" +
            "        \"f_to_date\": \"20231106\",\n" +
            "        \"vhcl_reg_no\": \"99임9999\"\n" +
            "     },\n" +
            "     {\n" +
            "        \"f_license_no\": \"123456789012\",\n" +
            "        \"f_resident_name\": \"홍길동\",\n" +
            "        \"f_licn_con_code\": \"12\",\n" +
            "        \"f_from_date\": \"20231106\",\n" +
            "        \"f_to_date\": \"20231106\",\n" +
            "        \"vhcl_reg_no\": \"99임9999\"\n" +
            "     }\n" +
            "   ]\n" +
            "}";

    // 운전자격검증 API 호출(소화물배송대행서비스사업자 API와 동일)
    public void callApi() {
        String singleUri = "http://서버도메인/licenseVerification"; // 단건 조회
        String batchUri = "http://서버도메인/licenseVerificationBatch"; // 배치 조회
        String token = "Bearer 026f1744-1234-4bcb-b0a6-990f9da9a745"; // 발급한 토큰
        JSONObject body = new JSONObject();
        try {
```

```
            String encryptedStr = encrypt(single, CLIENT_SEC); // 목적에 따라 단건, 또는
배치 uri 선택 및 본문을 구성한다.
            body.put("encryptedData", encryptedStr);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        webClient.post()
                .uri(singleUri)
                .header(HttpHeaders.AUTHORIZATION, token)
                .contentType(MediaType.APPLICATION_JSON)
                .bodyValue(body)
                .retrieve()
                .bodyToMono(String.class)
                .flatMap(response -> {
                    log.info("호출 결과 확인 : {}", response);
                    return Mono.empty();
                })
                .onErrorResume(e -> {
                    log.error("에러", e);
                    return Mono.error(RuntimeException::new);
                }).subscribe();
    }

    // 사용자 검증 API 호출
    public void checkBusinessRegistration() {
        String token = "Bearer 026f1744-e3e0-4bcb-b0a6-990f9da9a745"; // 발급한 토큰
        String userCheckUri =
"http://서버도메인/col/isUserInDatabase?user_id=사용자ID&brno=사업자등록번호";
        webClient.get()
                .uri(userCheckUri)
                .header(HttpHeaders.AUTHORIZATION, token)
                .retrieve()
                .bodyToMono(String.class)
                .flatMap(response -> {
                    log.info("호출 결과 확인 : {}", response);
                    return Mono.empty();
                })
                .onErrorResume(e -> {
                    log.error("에러", e);
                    return Mono.error(RuntimeException::new);
                }).subscribe();
    }
```