

# API Guidelines

## App2App SDK Integration Specs

### Description & Use

Rp SDK is an App2App SDK for Samsung Wallet driver's license service online scenarios. This SDK provides an implementation for direct communication between the Samsung Wallet and Partner applications.

### Build the settings

RpSdk requires additional dependencies with:

```
dependencies {  
    ...  
    implementation("rp-sdk-1.0-release.aar")  
    implementation("androidx.core:core-ktx:1.3.2")  
    implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.7.0")  
    implementation("androidx.lifecycle:lifecycle-livedata-core-ktx:2.7.0")  
    implementation("io.reactivex.rxjava2:rxjava:2.2.21")  
    implementation("io.reactivex.rxjava2:rxkotlin:2.4.0")  
    implementation("io.reactivex.rxjava2:rxandroid:2.1.1")  
    implementation("com.squareup.okhttp3:okhttp:4.11.0")  
    implementation("com.google.code.gson:gson:2.10.1")  
    implementation("org.bouncycastle:bcprov-jdk15to18:1.66")  
    implementation("com.nimbusds:nimbus-jose-jwt:9.37.3")  
    ...  
}
```

### AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">  
    ...
```



#### Recommendation

- Blog
- Code Lab
- Foldable and Large Screen Optimization
- Health
- Samsung Wallet
- SDC24

Signature & parameters of the request method.

```
fun request(targetPackageName: String, requestId: String, appLink: String, onResponseListener: OnResponseListener? = null)
```

Parameter name	Description
targetPackageName	The Pakcage name to connect to.
requestId	A Random string to identify each request
appLink	The appLink built by Samsung MCS server guide
onResponseListener	A listener to receive each events or requests

[Sample code]

```
binding.button.setOnClickListener {
    rpClientApis.request("com.samsung.android.spay", UUID.randomUUID().toString(), appLink,
        override fun onGetMdocRequestData(deviceEngagementBytes: ByteArray): ByteArray? {
            Log.i(TAG, "onGetMdocRequestData($deviceEngagementBytes)")
            /**
             * 1. prepare mDoc request data (ISO-18013-5)
             * 2. build sessionEstablishmentBytes (ISO-18013-5)
             * 3. Encrypt it with HKDF (ISO-18013-5, 9.1.1.5 Cryptographic operations)
             */
            return "encryptedSessionEstablishmentBytes"
        }
        override fun onMdocResponse(encryptedResponse: ByteArray) {
            Log.i(TAG, "onMdocResponse($encryptedResponse)")
            /**
             * 1. Decrypt it with HKDF (ISO-18013-5, 9.1.1.5 Cryptographic operations)
             * 2. CBOR decode it
             */
        }
        override fun onMdocResponseFailed(exception: Exception) {
            Log.i(TAG, "onMdocResponseFailed($exception)")
        }
    })
}
```

Error Code Explanation

The below exceptions might occur through the onMdocResponseFailed callback

Exceptions name	Description
-----------------	-------------



Recommendation

Blog

Code Lab

Foldable and Large Screen Optimization

Health

Samsung Wallet

SDC24





[Result]

HTTP status code	Description
200 OK	Success
400 Bad Request	Requests cannot or will not be processed due to something that is perceived to be a client error.
401 Unauthorized	Authorization token is invalid or expired.
500 Internal Server Error	The server encountered an unexpected condition that prevented it from fulfilling the request.
503 Service Unavailable	The server is not ready to handle the request.

Send authentication data

The data is encrypted according to the requested data and then transmitted along with the data card information.

[Request]

Type	Value	Description
Method	POST	
URL	{Partner server URL}/rp/v1.0/{cardId}/{RefId}/auth	
Headers	Authorization <code>String(1024)</code>	(Required) Credential token. The token can have the prefix "Bearer" as an authorization type, e.g., Bearer <i>&lt;credentials&gt;</i> . * Refer to <a href="#">Authorization Token</a> for more details.
PathParameters	cardId <code>String(32)</code>	(Required) Wallet card identifier issued from Partner portal when the partner manager signs up for partner services and registers the wallet card they want to service.
	refId <code>String(32)</code>	(Required) Unique content identifier defined by the content provider
QueryParameter		N/A
Payload	data <code>String(3000)</code>	(Required) JWT data encrypted with the public key information and card type. If decrypted this data is decoded, it has the following format information. { "data": "XXXXXXXXXXXX", "card": { "type": "idcard", "subType": "drivers", "designType": "us-01" } }



Recommendation

Blog

Code Lab

Foldable and Large Screen Optimization

Health

Samsung Wallet

SDC24

```
POST {Partner server URL}/rp/v1.0/{cardId}/{RefId}/auth
Content-Type: application/json

{
  "data":
    "eyJjdHkiOiJBVVRIliwidmVyljoiMilsInBhcnRuZXJJZCI6InRlc3QiLCJ1dGMI0jE3MTYyMDYzNjAxMTAsImFsZyI6IiJTMjU2In
0.ZXIKbGJtTWIPaUpCTVRJNFJwTk5JaXdpWVd4bkIqb2IVbE5CTFU5QlJWQXRNaUySW4wLIZ5aFAxS0FnMVJHbzBDN2
NIX2pYdGtfODdQbnhrRmpfWkpPcnNSUUs4MnN0OWVxTjEyVzVMOEJaX1d5NGVzMzE3VDNad0pncmpwZWdZOEK3aVl
CWWRIOGJ5LXFimjBLU3RUc3JsSzIIPSIFnN1FaM2xZaUxscXITb0VlbERvd0FPaTRMRy1JUkZWdVlrbXRiNTg3UTd1ZW
uQ1IWWGZWaIVEcG01YXBFBzDV3SzM1UGZ3d0dkREM2TmowZ1AwbTZ3Nk1kdI9mdDBvZWc2MWZjaGdBYnY0emxMZj
U2cVYzM0t6ZjdjbWVpbkJRNNpMSGUtYmFWYXhVZk5Ld2htZWVjUzFTV3laRm1NVIJ6MEFsMnBxa0dQLVJkT1Iza3VzaV
o0VjFIldy1aQ2lyVWVwYVdZRU9nUEdrVW1mbTFuOWJWJT1ZmZ1NUV1F0SE5pVTFJYVRHTG1DWlpVQS5PMzZrd1g4W
mJnQ21wd3o2LI9KZEhFVXNnbm13b1drdDRMcU4xMUNCaUNTSnUtbWpYV2ZrckxoS0ZVenBsS085ckdXbUdPZ0pqUkF1
NTFsOTRYc2VlVWdfWU9NS2RGR1VOMWJhMHB3Y0tFNGtJMEt2dkFOWHprODN0azBjQzROT2F6VzlmOVNTT0RhMU
9IMEFoavFzQzdDeVFqNndNLWFIVk8waEJwSEJkMEdURUh1Z3Exc21vVmxRbjBISnJqWHM4X3FwcnpLekwtaDFPcFk1a
Es1ZUg5Q3NiSms0aEhCNGNmWUIKRUJFZ09BcGZxcGFuMGFSVGFMODhhdXlqSGZHdGRMa0tLWDV0Q0RTajlxSE5T
T0FhWTJVVWIZrR0hxU0wzNGJabTU5aEZMNvdHa0IJcE9BMHIWUE9tQzNWTfIKV2JsMm85LkFoeDBVYTUVGeTZudkxKV
XVkeTAzSHc.E07YYI7IoR3885VYKsS5_q1lcpX750uU2Ge5suJSedx3Dr_U0x4tSe9_0NxM46dyWnFUXrUAGfjDnjHIBc707
Li9VI3XtyiHwnwEiFydgV1QB9oddkYyZuahXQmJHVUqNcdt6DF2CAqzF5QgMVqfMGSE_t7IPU8vQFXE34DO-
sKzj8ftdusS2EcdANBqOKCHih3m39NouBPFhcX68PIPCW50diXlupxwEGniU2t3Co24YllaKLGac669aCcXDQr34utVUqhTJt
_FTXkahAlzoA34_Hj_s82FivIXh1ITD74UOjZSe7IBWya_kVysoZavnmzTz2tH9CbwyCvx8wA"
}
```

[Response]

Type	Value	Description
HTTP Status code	200 OK	
	400 Bad Request	

[Result]

HTTP status code	Description
200 OK	Success
400 Bad Request	Requests cannot or will not be processed due to something that is perceived to be a client error.
401 Unauthorized	Authorization token is invalid or expired.
500 Internal Server Error	The server encountered an unexpected condition that prevented it from fulfilling the request.
503 Service Unavailable	The server is not ready to handle the request.

Code explanation based on the sample code



Recommendation

Blog

Code Lab

Foldable and Large Screen Optimization

Health

Samsung Wallet

SDC24



```
// Verify JWS using Samsung Public key
public RequestBody getRequestBody(final KeyRing keyRing) {
    final SignedJWT signedJWT = JWTUtils.verify(keyRing.getTargetPublicKey(), encryptedData,
    try {
        final String strBody = JWTUtils.getDecryptedPayloadFrom(keyRing.getSourcePrivateKey(), J
        return objectMapper.readValue(strBody, RequestBody.class); // Convert to data format req
    } catch (ParseException | JsonProcessingException e) {
        log.error("getRequestBody : {}", e.getMessage());
        throw new CustomException(HttpStatus.INTERNAL_SERVER_ERROR, "data body parse error")
    }
}
```

## 2. Decrypt the JWE using the JWS

```
JWEObject.parse(signedJWT.getPayload().toString())
```

```
public static String getDecryptedPayloadFrom(final Key privateKey, final JWEObject data) {
    try {
        data.decrypt(new RSADecrypter((PrivateKey) privateKey)); // Decryption JWE using Pa
        return data.getPayload().toString();
    } catch (JOSEException e) {
        log.error("JOSEException message : {}", e.getMessage());
        throw new CustomException(HttpStatus.INTERNAL_SERVER_ERROR, "getDecryptedPayloadFrom
    }
}
```

## 3. Convert to the format send by the client

```
public RequestBody getRequestBody(final KeyRing keyRing) {
    final SignedJWT signedJWT = JWTUtils.verify(keyRing.getTargetPublicKey(), encryptedData,
    try {
        final String strBody = JWTUtils.getDecryptedPayloadFrom(keyRing.getSourcePrivateKey(
        return objectMapper.readValue(strBody, RequestBody.class); // Convert to data format
    } catch (ParseException | JsonProcessingException e) {
        log.error("getRequestBody : {}", e.getMessage());
        throw new CustomException(HttpStatus.INTERNAL_SERVER_ERROR, "data body parse error")
    }
}
```

## Generate MdocEstablishment

### 1. Generate RSA Key per refId



## Recommendation

Blog

Code Lab

Foldable and Large Screen Optimization

Health

Samsung Wallet

SDC24



```

public class TransactionContext {
    private final KeyPair keyPair;    // RSA Key
    private final byte[] clientEngagement; // Body data received through key API, base64URL

    @EqualsAndHashCode.Exclude
    private int encryptMessageCounter = 0; // Count value when encrypted
    @EqualsAndHashCode.Exclude
    private int decryptMessageCounter = 0; // Count value when decrypted
}

private Cache<String, TransactionContext> contextCache; // RSA key management per refid with

// Generate and store RSA Key per refId only once upon first request
public TransactionContext setTransactionContext(final String key, final String base64Encoded
    log.info("base64EncodedClientPublicKey : {}", base64EncodedClientEngagement);
    this.contextCache.put(key, new TransactionContext(KeyUtils.generateKeyPair(), Base64Util
    return this.getTransactionContextBy(key);
}

// Part of retrieving RAS Key based on refId
public TransactionContext getTransactionContextBy(final String key) {
    return Optional.ofNullable(this.contextCache.getIfPresent(key)).orElseThrow(() -> {
        log.info("{} is empty", key);
        return new CustomException(HttpStatus.BAD_REQUEST, "No key matching the refId");
    });
}

```

## 2. Create request field values

```

@Override
public Mono<List<String>> createRequest(final PartnerInputDTO inputDTO) {
    final String mockData = "{ \"docType\": \"org.iso.18013.5.1.mDL\", \"nameSpaces\": { \"o
    return Mono.just(Collections.singletonList(mockData));
}

```

## 3. Generate Establishment

```

@AllArgsConstructor
public class Establishment {
    private final TransactionContext context; // Info of client public key , partner private
    private final List<String> strReqs; // Data field information required for authenticatio
    private final KeyRing keyRing; // RSA Key information for JWT(JWS + JWE) encryption and
}

```

×



## Recommendation

Blog

Code Lab

Foldable and Large Screen Optimization

Health

Samsung Wallet

SDC24



```

public static String encryptedStringJWE(final Key publicKey, final String data) { // Please
    final JWEObject jwe = new JWEObject(
        new JWEHeader.Builder(JWEAlgorithm.RSA_OAEP_256, EncryptionMethod.A128GCM).build(), new
    try {
        jwe.encrypt(new RSAEncrypter((RSAPublicKey) publicKey));
        return jwe.serialize();
    } catch (JOSEException e) {
        log.error("encryptedStringJWE Exception message : {}", e.getMessage());
        throw new CustomException(HttpStatus.INTERNAL_SERVER_ERROR, "encryptedStringJWE erro
    }
}

```

## 2. Generate JWS by JWE

```

public static String generateSignedStringJWS(final Key privateKey, final Key publicKey, fina
    try {
        final JWSObject jwsObj = new JWSObject(getDefaultJWSHeader(), new Payload(payload));
        JWSSigner signer = new RSASSASigner(new RSAKey.Builder((RSAPublicKey) publicKey).private
        jwsObj.sign(signer);
        return jwsObj.serialize();
    } catch (JOSEException e) {
        log.error("encryptedStringJWS Exception message : {}", e.getMessage());
        throw new CustomException(HttpStatus.INTERNAL_SERVER_ERROR, "generateSignedStringJWS err
    }
}

```

## 3. Generate JWT(JWS + JWE)

```

public PartnerOutputDTO toPartnerOutputDTO() {
    final CBORObject generate = this.generate();
    final String establishment = Base64.getUrlEncoder().encodeToString(generate.EncodeToByte
    final String strJWE = JWTUtils.encryptedStringJWE(keyRing.getTargetPublicKey(), establis
    final JWSHeader jwsHeader = JWTUtils.getDefaultJWSHeader(keyRing.getVersion(), keyRing.g
    return new PartnerOutputDTO(JWTUtils.generateSignedStringJWS(jwsHeader, keyRing.getSourc
}

```

## Authentication processing for values in data fields requested for authentication

### 1. Retrieve transactionContext value stored in cache using refId value

```

@Override
public Mono<TransactionContext> getContext(final PartnerInputDTO inputDTO) {
    return Mono.just(this.transactionContextManager.getTransactionContextBy(inputDTO.getRefId
}

```



## Recommendation

Blog

Code Lab

Foldable and Large Screen Optimization

Health

Samsung Wallet

SDC24






4. Get the field values requested for authentication from the data in mDocResponse.



```
public String getData() {
    // SessionData = {
    //     ? "data" : bstr ; Encrypted mdoc response or mdoc request
    //     ? "status" : uint ; Status code
    // }
    final CBORObject response = CBORObject.DecodeFromBytes(data);
    checkType(response, CBORType.Map);
    final CBORObject data = response.get(DATA);
    checkType(data, CBORType.ByteString);


    return CBORObject.DecodeFromBytes(isEncryptedMode ? CipherUtils.decrypt(this.context, da
}
}
```

5. Create a Session value using the transactionContext value managed by refId and then decrypt it.



```
private static byte[] processCipher(final CipherMode cipherMode, final TransactionContext co
try {
    Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
    final int counter = CipherMode.ENCRYPT == cipherMode ? context.getEncryptMessageCount() : 0;
    GCMParameterSpec parameterSpec = new GCMParameterSpec(128, getSessionKeyIv(cipherMode));
    cipher.init(cipherMode.cipherMode , getSecretKeySpec(context, cipherMode.info), parameterSpec);
    return cipher.doFinal(bytes);
} catch (InvalidAlgorithmParameterException | NoSuchPaddingException | IllegalBlockSizeException |
    NoSuchAlgorithmException | BadPaddingException | InvalidKeyException e) {
    log.error("error type : {}, message :{}", e.getClass(), e.getMessage());
    throw new CustomException(HttpStatus.INTERNAL_SERVER_ERROR, "processCipher error");
}
}
```

6. Examining data received from the client.



```
@Override
public Mono<Void> authentication(final String response) {
    log.info("response info : {}", response);
    return Mono.empty();
}
```

Quick Link	▼
Family Site	▼
Legal	▼
Social Communications	▼