How to create a session token

Learn how to create a session token so your apps can talk to the Compliance platform

Before you start

- You should be familiar with JSON Web Tokens (JWTs) and public-key cryptography.
- You should have exchanged encryption keys with BetSecure.

Load the exchanged keys

Grab the keys you exchanged with the BetSecure API and load them into your app. To produce a session token, you'll need your private signing key and BetSecure's public encryption key.

Java

LoadProduceKeys.java

```java
import java.nio.file.*;
import org.jose4j.jwk.*;

// Load the keys for producing session tokens
JsonWebKeySet produceKeys =
    new JsonWebKeySet(Files.readString(Paths.get("path/to/produce-keys.json")));

// Find the signing key
SimpleJwkFilter signFilter = new SimpleJwkFilter();
signFilter.setKeyOperations(new String[] {KeyOperations.SIGN}, false);
PublicJsonWebKey sign =
    (PublicJsonWebKey)
        signFilter.filter(produceKeys.getJsonWebKeys()).stream()
            .findFirst()
            .orElseThrow(() -> new Exception("Missing sign key"));

// Find the encryption key
SimpleJwkFilter encryptFilter = new SimpleJwkFilter();
encryptFilter.setKeyOperations(new String[] {KeyOperations.ENCRYPT}, false);
PublicJsonWebKey encrypt =
    (PublicJsonWebKey)
        encryptFilter.filter(produceKeys.getJsonWebKeys()).stream()
            .findFirst()
            .orElseThrow(() -> new Exception("Missing encrypt key"));
```

DANGER

Don't use this example in your app. You should never store your private keys un-encrypted on disk or in a database. You should always use a secure key management system to store your private keys.

Create the session token

Use the keys you loaded to create a signed and encrypted JSON Web Token (JWT) containing the license to use for authorizing patron actions. You can also add the patron ID if you have an authenticated patron session.

DANGER

You should use a well-vetted library to handle as much of the token creation process as possible, and follow JSON Web Token Best Current Practices.

TIP

See Recommended libraries for a list of libraries that can help you create session tokens.

Java

CreateSessionToken.java

```java
import java.util.*;
import org.jose4j.jwe.*;
import org.jose4j.jwk.*;
import org.jose4j.jws.*;
import org.jose4j.jwt.*;

// These claims are publicly visible
JwtClaims nonSensitiveClaims = new JwtClaims();
// A unique identifier for the token
nonSensitiveClaims.setJwtId(UUID.randomUUID().toString());
// Your app's session ID
nonSensitiveClaims.setClaim("nsid", "<YOUR_SESSION_ID>");
// When the token was issued
```

```java
nonSensitiveClaims.setIssuedAtToNow();
// When the token expires
nonSensitiveClaims.setExpirationTimeMinutesInTheFuture(60); // e.g. 1 hour

// These claims are only visible to the Compliance platform
JwtClaims sensitiveClaims = new JwtClaims();
// Copy non-sensitive claims
nonSensitiveClaims.getClaimsMap().forEach(sensitiveClaims::setClaim);
// Your unique operator ID
sensitiveClaims.setIssuer("<YOUR_OPERATOR_ID>");
// Target Compliance environment
sensitiveClaims.setAudience("compliance.integration.betsecure.io");
// The ID of the license to use
sensitiveClaims.setClaim("nlic", new String[] {"<LICENSE_ID_A>"});

// Add the patron ID if your app's session is authenticated
if (patronID != null) {
  sensitiveClaims.setSubject(patronID);
}

// Sign the inner JWT
JsonWebSignature jws = new JsonWebSignature();
// Token type for session tokens
jws.setHeader("typ", "x.session+jwt");
jws.setAlgorithmHeaderValue(AlgorithmIdentifiers.EDDSA);
jws.setKey(sign.getPrivateKey());
jws.setKeyIdHeaderValue(sign.getKeyId());
jws.setPayload(sensitiveClaims.toJson());
String signed = jws.getCompactSerialization();

// Encrypt the outer JWT
JsonWebEncryption jwe = new JsonWebEncryption();
jwe.setHeader("typ", jws.getHeader("typ"));
jwe.setAlgorithmHeaderValue(KeyManagementAlgorithmIdentifiers.ECDH_ES_A128KW);
jwe.setEncryptionMethodHeaderParameter(ContentEncryptionAlgorithmIdentifiers.AES_128_G
jwe.setKey(encrypt.getPublicKey());
jwe.setKeyIdHeaderValue(encrypt.getKeyId());
// Replicate non-sensitive claims as JWE header parameters
// See https://datatracker.ietf.org/doc/html/rfc7519#section-5.3
for (String claimName : nonSensitiveClaims.getClaimNames()) {
  jwe.setHeader(claimName, nonSensitiveClaims.getClaimValue(claimName));
}
jwe.setContentTypeHeaderValue("JWT");
jwe.setPayload(signed);
String encrypted = jwe.getCompactSerialization();
```

INFO

For an explanation of the role of session tokens in the Compliance platform, see Session tokens. If you'd like a detailed breakdown of the claims and headers in the session token, see the Session token structure reference.

Next steps

Now that you've got a session token to work with, you might find these guides useful:

- Web SDK guides
- How to consume a transaction token

See also

- Session token structure reference
- Key groups
- Session tokens
- Encryption and signing algorithms
- Sessions