**Addressing the race condition while exchanging the AES-256 bit Encryption key**

**Issue**

Today when concurrent threads make an FSAPI request with the **same expired key**, there is a likelihood that these threads may get **different AES keys,** leading to inconsistencies in the AES keys persisted.

**Approach**

Inputs: p\_in\_partner\_id

Outputs: p\_out\_current\_aes256\_key, p\_out\_previous\_aes256\_key

Logic:

1. Check if the current key has expired (if this throws ORA-01403: no data found exception, it implies that the p\_in\_partner\_id is bogus and hence throw an error)

select (key\_expiry\_in\_days - (sysdate - current\_key\_creation\_date)), current\_key, previous\_key into l\_days\_left, l\_current\_key, l\_previous\_key from apix.keymanagement where partner\_id = **p\_in\_partner\_id**;

If l\_days\_left > 0, then the key hasn’t expired yet. Hence just return current\_key as p\_out\_current\_aes256\_key and previous\_key as p\_out\_previous\_aes256\_key with no change (the BAU flow)

1. If l\_days\_left <=0, then the key has expired. So proceed as follows:
   1. Generate new l\_aes256\_new\_key using a java SP or dbms\_crypto package
   2. Attempt to update the table with the l\_aes256\_new\_key

update apix.keymanagement

set current\_key = l\_aes256\_new\_key,

previous\_key\_creation\_date = current\_key\_creation\_date,

previous\_key = l\_current\_key, current\_key\_creation\_date = sysdate

where current\_key = l\_current\_key and partner\_id = **p\_in\_partner\_id**;

if sql%rowcount = 0 then

-- This implies that no update happened, probably due to race

So we simply select current\_key from table (for l\_current\_key matching

the previous\_key and p\_in\_partner\_id) and return it as

p\_out\_current\_aes256\_key.

select current\_key, previous\_key into p\_out\_current\_aes256\_key,

p\_out\_previous\_aes256\_key from

apix.keymanagement where previous\_key = l\_current\_key and

partner\_id = **p\_in\_partner\_id**;

else

-- Successful. New key persisted.

Return the l\_aes256\_new\_key as p\_out\_current\_aes256\_key and

l\_current\_key as the p\_out\_previous\_aes256\_key

end if;

**Generating the AES256 key using dbms\_crypto**

declare

l\_num\_key\_bytes NUMBER := 256/8; -- key length 256 bits (32 bytes)

l\_key\_bytes\_raw RAW (32); -- stores 256-bit encryption key

l\_encryption\_type PLS\_INTEGER := -- Instantiate an encryption type

DBMS\_CRYPTO.ENCRYPT\_AES256

+ DBMS\_CRYPTO.CHAIN\_CBC

+ DBMS\_CRYPTO.PAD\_PKCS5;

begin

l\_key\_bytes\_raw := dbms\_crypto.randombytes(l\_num\_key\_bytes);

end;

Note: Although l\_key\_bytes\_raw is RAW, it can be inserted into the VARCHAR(100)columns - CURRENT\_KEY and PREVIOUS\_KEY of apix.keymanagement