

**ARIA**

## 방법1. 현재(AES)와 같은 방법- procedure로 function 만들어 사용

```

CREATE OR REPLACE FUNCTION SHERPA_CMS.f_s__getdecrypt(input_string VARCHAR2
, key_string VARCHAR2)
RETURN VARCHAR2
AS
    gv_sqlerrm          varchar2(1000) ;
    gv_pgmstep          varchar2(100) ;

    output_string       VARCHAR2 (200);
    encrypted_raw       RAW (2000) := input_string;          -- stores encrypted binary text
    decrypted_raw       RAW (2000);                          -- stores decrypted binary text

    key_bytes_raw       RAW (32);                             -- stores 256-bit encryption key
    encryption_type     PLS_INTEGER :=                        -- total encryption type
        DBMS_CRYPTO.ENCRYPT_AES256
        + DBMS_CRYPTO.CHAIN_CBC
        + DBMS_CRYPTO.PAD_PKCS5;

BEGIN
    --gv_pgmstep := 'f_s__getdecrypt_0001';
    --dbms_output.put_line('step = ' || gv_pgmstep);

    key_bytes_raw := UTL_RAW.CAST_TO_RAW(CONVERT(key_string||'00000000000000000000', 'AL32UTF8', 'UTF8'));

    decrypted_raw := DBMS_CRYPTO.DECRYPT
    (
        src => encrypted_raw,
        typ => encryption_type,
        key => key_bytes_raw
    );
    output_string := UTL_I18N.RAW_TO_CHAR (decrypted_raw, 'AL32UTF8');

    --gv_pgmstep := 'f_s__getdecrypt_0002';
    --dbms_output.put_line('step = ' || gv_pgmstep);

RETURN output_string;

EXCEPTION
WHEN OTHERS THEN

    gv_pgmstep := 'f_s__getdecrypt_Exception_step';
    --dbms_output.put_line('step = ' || gv_pgmstep);

    gv_sqlerrm := substr(sqlerrm, 1, 900);
    -- dbms_output.put_line(gv_pgmstep || ', ' || gv_sqlerrm);

    output_string := 'Invalid Password';

```

AES 암호화 방식을 Oracle에서 지원하기 때문에 가능했던 방식이며,

ARIA 암호화 방식은 **Oracle 12c R2** 이상부터 지원함.

따라서 procedure를 이용한 function 사용시,


수집서버 **Oracle 버전이 12c R2 이상일 경우에만 ARIA 암호화가 가능하게 됨.**

## 방법2. Oracle External Procedure 생성

### 장점>

- ① 재 활용성이 우수
- ② 유지보수 용이
- ③ 11g 버전에서도 사용 가능

### 단점>



```
[oracle@smpark 2944]$ ps -eo "comm,pid,rss" | grep extproc  
extproc          3035 12944  
extproc          3139 12948
```

- ① Session이 종료되지 않으면 extProc는 Oracle에서 메모리를 관리하는 영역이 아니라 O/S영역이기 때문에 한번 호출 될 때마다 해당 Session이 종료되지 않으면 끝까지 살아남게 되어 지속적인 메모리에 남아 있게 되어 O/S의 메모리 부하가 생김
- ② 실행 시키기 위해 listener.ora / tnsname.ora 파일을 변경해야함

## External Procedure 사용 시 O/S 부하 줄이는 법

- ① Session의 수를 제한하여 O/S의 메모리 한계치를 벗어나지 않도록 조정한다.
- ② O/S에서 extProc로 생성된 것 중 오래된 Process를 Kill한다. Process를 Kill하더라도 없으면 재생성 되므로 큰 문제는 발생되지 않는다.
- ③ 어플리케이션에서 불필요하게 External Procedure를 호출하는 Function의 사용을 제거하고, 사용 완료 후 Session를 종료하여 메모리의 부하를 최소화 하게 한다.

## External Procedure 사용 및 빌드 방법(커맨드 창에서)

```
return passwd;
```

- ① 빌드 할 C파일 제작  
:select시 가져올 최종값을 return 하는 것이 원칙
- ② obj파일 so파일 생성  
: /usr/bin/gcc -fPIC -m64 -nostdlib -c [파일명].c -o [파일명].o  
/usr/bin/gcc -fPIC -m64 -nostdlib -shared [파일명].o -o [파일명].so

## External Procedure 사용 및 빌드 방법(listener 설정)

### ③ listener.ora 파일 수정

```
LISTENER =  
  (DESCRIPTION_LIST =  
    (DESCRIPTION =  
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))  
      (ADDRESS = (PROTOCOL = TCP)(HOST = )(PORT = ))  
    )  
  )  
SID_LIST_LISTENER =  
  (SID_LIST =  
    (SID_DESC =  
      (SID_NAME = PLSExtProc)  
      (ORACLE_HOME = /usr/oracle/app/product/11.2.0/dbhome_1)  
      (PROGRAM = extproc)  
      (ENVS="EXTPROC_DLLS=ANY")  
    )  
  )
```

## External Procedure 사용 및 빌드 방법(listener 설정)

- ④ tnsnames.ora 설정 (listener.ora와 SID ,KEY 꼭 맞춰주어야함)

```
EXTPROC_CONNECTION_DATA =  
  (DESCRIPTION =  
    (ADDRESS = (PROTOCOL = IPC)(HOST = ) (KEY = EXTPROC1521))  
    (CONNECT_DATA = (SID = PLSExtProc))  
  )
```

- ⑤ 리스너 재시작



## External Procedure 사용 및 빌드 방법(SQL 창에서)

### ⑥ 라이브러리 생성

```
: create or replace library libaria as '/home/oracle/ARIA/libaria.so';  
/
```

### ⑦ 함수 생성

```
create or replace function
```

```
f_s_getdecrypt(input VARCHAR2) return varchar2
```

```
as external
```

```
language C
```

```
library libaria
```

```
name " f_s_getdecrypt "
```

```
parameters (input STRING);
```

```
/
```

## External Procedure 사용 및 빌드 방법(SQL 창에서)

### ⑧ 생성한 함수 실행

```
SQL> select f_s__getdecrypt('b5b8e6d22e385d9135f7e6567e0577ea') from dual;
```

```
F_S__GETDECRYPT('B5B8E6D22E385D9135F7E6567E0577EA')
```

```
-----  
sh
```