THALES

IBM DB2

INTEGRATION GUIDE

THALES LUNA HSM

THALES DATA PROTECTION ON DEMAND

Document Information

Document Part Number	007-013741-001
Release Date	17 June 2020

Revision History

Revision	Date	Reason
A	1 February 2018	New
B	1 November 2018	Update
C	17 June 2020	Update

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Introduction

This document is intended to guide security administrators through the steps for the IBM DB2 Integration with Thales HSM, and also covers the necessary information to install, configure and integrate IBM DB2 with Thales HSM.

IBM DB2 encrypt the databases and backup images using DB2 native encryption. Native encryption provides transparent and secure key management and requires no changes to your hardware, software, applications, or schemas.

The primary benefit of a PKCS #11 keystore is the protection it provides to encryption keys. This protection is accomplished by imposing a restriction that keys never leave the secure environment of the keystore. Data on disk is encrypted with a data encryption key (DEK) that is stored with the database. The DEK, in turn, is encrypted by a master key (MK), which is stored externally to the database.

The DEK is sent to the PKCS #11 keystore, where it is decrypted by the MK. The only exception to this principle of keys not leaving the keystore is when migrating keys from a local keystore file to a PKCS #11 keystore. In such cases, these keys are marked as external. However, an immediate key rotation following migration will start to make use of internally defined keys.

Using a PKCS #11 keystore is more secure alternative, when you have multiple databases and you do not want to maintain individual keystores.

The following are the benefits of using Thales HSMs to secure the IBM DB2 Master Key:

- Secure generation, storage and protection of the encryption key on FIPS 140-2 level 3 validated hardware.
- Full life cycle management of the keys.
- HSM audit trail.
- Take advantage of cloud services with confidence.
- Significant performance improvements by off-loading cryptographic operations from application servers.

Supported HSM Devices & Services

Below is the list of the supported HSMs:

Thales Luna HSM: Thales Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Thales Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing.

The Thales Luna HSM on premise offerings include the Thales Luna Network HSM, Thales PCIe HSM, and Thales Luna USB HSMs.

Thales DPOD: Thales Data Protection on Demand (DPoD) is a cloud-based platform that provides on-demand HSM and Key Management services through a simple graphical user interface. With DPOD, security is simple, cost effective, and easy to manage because there is no hardware to buy, deploy, and maintain. As an Application Owner, you click and deploy services, generate usage reports and maintain the required services.

Certified Platforms

This integration is certified on the following platforms:

- Red Hat Enterprise Linux
- CentOS
- AIX

> **NOTE:** All platforms which are not listed but supported by Luna/DPoD Client are subjected to support this integration.

Configuring Luna HSM

To configure Luna HSM:

1. Ensure that the HSM is set up, initialized, provisioned and ready for deployment.
2. Create a partition, establish a Network Trust Link (NTL) between the HSM and client, and enable the client to access the partition.
3. Initialize partition and Crypto Officer Role for the partition.
4. Ensure that the partition is successfully registered and configured. The command to see the registered partitions is:

```
# /usr/safenet/lunaclient/bin/lunacm
lunacm (64-bit) v10.2.0-111. Copyright (c) 2020 SafeNet. All rights reserved.
   Available HSMs:


   Slot Id ->              0
   Label ->                ibm_db2
   Serial Number ->        1382576042518
   Model ->                LunaSA 7.4.0
   Firmware Version ->     7.4.0
   Configuration ->        Luna User Partition With SO (PW) Signing With
   Cloning Mode
   Slot Description ->     Net Token Slot
   FM HW Status ->         FM Ready


   Current Slot Id: 0
```

5. For PED-authenticated HSM, enable partition policies 22 and 23 to allow activation and auto-activation.

> **NOTE:** For a detailed description of the steps involved in Luna HSM configuration, refer to *Thales Luna HSM Documentation*.

Thales Luna Network HSM HA Setup

Refer to the Thales Luna Network HSM Product Documentation for steps and details regarding configuring and setting up two or more HSM boxes on host systems. You must enable the HAOnly setting in HA for failover to work.

Provision DPOD Service

The DPoD service provides your client machine with access to an HSM Application Partition for storing cryptographic objects used by your applications. Application partitions can be assigned to a single client, or multiple clients can be assigned to, and share a single application partition.

To use the DPoD service, you need to provision your application partition by initializing the following roles:

- **Security Officer (SO)** - responsible for setting the partition policies and for creating the Crypto Officer.
- **Crypto Officer (CO)** - responsible for creating, modifying, and deleting crypto objects within the partition. The CO can use the crypto objects and create an optional, limited-capability role called Crypto User that can use the crypto objects but cannot modify them.
- **Crypto User (CU)** - optional role that can use crypto objects while performing cryptographic operations.

Prerequisites

Ensure that you fulfill the following prerequisites.

Verifying GSKit Installation and Configuration on DB Server

To use DB2 native encryption, GSKit must be installed and configured. The DB2 installer installs GSKit locally.

For each instance, the GSKit libraries will be located in the following directory:

`/home/<db2inst>/sqllib/lib32/gskit` or `/home/<db2inst>/sqllib/lib64/gskit`

Where **<db2inst>** is the DB2 owner.

1. Log on to the system as db2 instance owner.
2. Set the environment variable using below command.

```
. /home/<db2_instance>/sqllib/db2profile
```

Migrating Master Key from Local Keystore to Thales HSM

This section describes the procedure to migrate the Master Key from local PKCS#12 keystore to HSM based PKCS#11 keystore. If you are already using the native encryption and generated the Master Key in local keystore and now want to move to more secure HSM keystore for securing the Master Key follow the steps provided below:

Creating and Verifying Software Based PKCS#12 Keystore

To migrate the software keystore to HSM keystore, verify that software keystore is created and native encryption based on software keystore is working.

To Create and Configure Local Software Keystore

1. Log on to the system as db2 instance owner.
2. Change the current directory to the folder where GSKit binaries are installed.

   ```
   $ cd /home/<Instance_owner>/sqllib/gskit/bin
   ```

3. Create the local keystore by executing the `gsk8capicmd` command.

   ```
   $ gsk8capicmd_64 -keydb -create -db "/home/<db2_instance>/sqllib/security/my-keystore.p12" -pw "<keystore password>" -type pkcs12 -stash
   ```

   Change the variable provided in < > with the actual values in your environment. Keystore password provided in the command will be saved in a stash file with the same base name as the keystore file but with the file extension ".sth".

```
bash-3.2$ ls -ltr /home/db2inst1/sqllib/security/
total 10408
-r-s--x--x    1 db2inst1 db2iadm1      27043 May 29 06:43 db2aud
-rw-r--r--    1 db2inst1 db2iadm1       4096 May 29 06:43 db2audit.cfg
-r-s--x--x    1 root     db2iadm1    5207605 May 29 06:43 db2ckpw
-r-x--s--x    1 db2inst1 db2iadm1      53298 May 29 06:43 db2flacc
-r-s--x--x    1 root     db2iadm1      24086 May 29 06:43 db2chpw
lrwxrwxrwx    1 root     system           38 May 29 06:43 db2chkau -> /opt/IBM/db2/V11.5/security64/db2chkau
drwxr-xr-x    2 db2inst1 db2iadm1        256 May 29 06:43 auditdata
-rw-------    1 db2inst1 db2iadm1          0 Jun 04 06:47 my-keystore.p12
-rw-------    1 db2inst1 db2iadm1        193 Jun 04 06:47 my-keystore.sth
bash-3.2$
```

4. To configure DB2 instance to use a keystore for native encryption, set two database manager configuration parameters: **keystore_location** and **keystore_type**.

   ```
   $ db2 update dbm cfg using keystore_location
   /home/<db2_instance>/sqllib/security/my-keystore.p12
   ```

   ```
   $ db2 update dbm cfg using keystore_type pkcs12
   ```

   Change the variable provided in < > with the actual values in your environment.

5. Restart the database using the command below.

   ```
   $ db2stop
   ```

   ```
   $ db2start
   ```

To Verify Encryption Using Software Keystore

6. In native encryption, when **ENCRYPT** parameter is specified, by default the database manager creates a new master key for the database and adds that master key to the keystore. Create an encrypted database with the default settings and specify the **ENCRYPT** option on the **CREATE DATABASE** command.

   ```
   $ db2 create db <database_name> encrypt
   ```

   Change the variable provided in < > with actual values in your environment.

7. To verify that the database has been successfully encrypted by DB2 native encryption, ensure that the value of the '**Encrypted database**' db configuration parameter value is **YES** in the output of the following command.

   ```
   $ db2 get db cfg for <database_name>
   ```

   Change the variable provided in < > with the actual values in your environment.

```
SMTP Server                                 (SMTP_SERVER) =
SQL conditional compilation flags           (SQL_CCFLAGS) =
Section actuals setting                  (SECTION_ACTUALS) = NONE
Connect procedure                           (CONNECT_PROC) =
Adjust temporal SYSTEM_TIME period (SYSTIME_PERIOD_ADJ) = NO
Log DDL Statements                        (LOG_DDL_STMTS) = NO
Log Application Information               (LOG_APPL_INFO) = NO
Default data capture on new Schemas     (DFT_SCHEMAS_DCC) = NO
Strict I/O for EXTBL_LOCATION           (EXTBL_STRICT_IO) = NO
Allowed paths for external tables        (EXTBL_LOCATION) = /home/db2inst1
Default table organization                (DFT_TABLE_ORG) = ROW
Default string units                       (STRING_UNITS) = SYSTEM
National character string mapping        (NCHAR_MAPPING) = CHAR_CU32
Database is in write suspend state                       = NO
Extended row size support               (EXTENDED_ROW_SZ) = ENABLE
Encryption Library for Backup                   (ENCRLIB) = libdb2encr.a
Encryption Options for Backup                 (ENCROPTS) = CIPHER=AES:MODE=CBC:KEY LENGTH=256

WLM Collection Interval (minutes)       (WLM_COLLECT_INT) = 0
Target agent load per CPU core      (WLM_AGENT_LOAD_TRGT) = AUTOMATIC(12)
WLM admission control enabled        (WLM_ADMISSION_CTRL) = NO
Allocated share of CPU resources          (WLM_CPU_SHARES) = 1000
CPU share behavior (hard/soft)       (WLM_CPU_SHARE_MODE) = HARD
Maximum allowable CPU utilization (%)      (WLM_CPU_LIMIT) = 0
Encrypted database                                         = YES
Procedural language stack trace           (PL_STACK_TRACE) = NONE
HADR SSL certificate label                (HADR_SSL_LABEL) =
```

8. Connect to the database to use the native encryption.

   ```
   $ db2 connect to <database_name>
   ```

   Change the variable provided in < > with the actual values in your environment.

9. Type `db2` and press **Enter** to launch interactive mode.

10. Create **EMPLOYEE_SALARY** table in the database:

    ```
    db2 => CREATE TABLE EMPLOYEE_SALARY (DEPTNO CHAR(3) NOT NULL,DEPTNAME
    VARCHAR(36) NOT NULL,EMPNO CHAR(6) NOT NULL,SALARY DECIMAL(9,2) NOT NULL WITH
    DEFAULT)
    ```

11. Enter the following in the **EMPLOYEE_SALARY** table:

    ```
    db2 => INSERT INTO EMPLOYEE_SALARY VALUES (001,'IT',001,10000)
    db2 => INSERT INTO EMPLOYEE_SALARY VALUES (001,'IT',002,15000)
    db2 => INSERT INTO EMPLOYEE_SALARY VALUES (001,'IT',003,20000)
    ```

12. Display the contents of the **EMPLOYEE_SALARY** table with the following command:

    ```
    db2 => SELECT * FROM EMPLOYEE_SALARY
    ```

13. Verify the access to keystore by moving or renaming the keystore to ensure that it is not available.

    ```
    $ mv my-keystore.p12 my-keystore.p24
    ```

14. Connect to the database. You will see an error message as keystore is not available.

    ```
    db2 => connect to <database_name>
    SQL1728N  The command or operation failed because the keystore could not be
    ```

```
accessed. Reason code "2".
```

Change the variable provided in < > with actual values in your environment.

Move the keystore back to the keystore location and it will be accessible again. It shows the native encryption is working with software based keystore and Master Key is secured in software keystore.

Migrating from a local keystore to a PKCS #11 keystore

It is assumed that the Master Key is generated in local keystore and database is already encrypted by Master key in local keystore.

To Migrate Master Key to HSM Keystore

1. Create a PKCS#11 configuration file named **luna.cfg**. To secure the master keys in a PKCS #11 keystore with DB2 native encryption, a configuration file must contains details of the HSM.

   On the DB2 server, create the PKCS #11 keystore configuration file **luna.cfg** with the following details:

   ```
   ----------------------------------------------------------------------

   VERSION=1

   PRODUCT_NAME=Luna

   ALLOW_KEY_INSERT_WITHOUT_KEYSTORE_BACKUP=true

   LIBRARY=<LunaClient installation dir>/lib/libCryptoki2_64.so

   SLOT_LABEL=<Partition_label>

   NEW_OBJECT_TYPE=PRIVATE

   KEYSTORE_STASH=/home/<db2_instance>/sqllib/security/pkcs11_pw.sth

   ----------------------------------------------------------------------
   ```

   Change the variable provided in < > with the actual values in your environment.

   Where **SLOT_LABEL** identifies the slot in the HSM by a label. The label is a name that is defined by the application and is assigned during token initialization.

   KEYSTORE_STASH is the absolute path and name of the stash file that holds the PKCS #11 keystore password. The instance uses the stash file to authenticate to the PKCS #11 keystore.

   Ensure that the **ALLOW_KEY_INSERT_WITHOUT_KEYSTORE_BACKUP** parameter is set to **TRUE** in the PKCS #11 keystore configuration file.

2. Create a stash file.

   ```
   $ db2credman -stash -password <partition password> -to
   /home/<db2_instance>/sqllib/security/pkcs11_pw.sth
   ```

   Change the variable provided in < > with the actual values in your environment.

   > **NOTE:** It should be noted that storing the PKCS#11 keystore password in a stash file is optional. If not specified, the password needs to be provided manually.

3. Migrate the Master key from the local keystore to the PKCS #11 keystore by issuing the `db2p12top11` command.

```
$ db2p12top11 -to /home/<db2_instance>/sqllib/security/luna.cfg -pin <partition password>
```

Change the variable provided in < > with the actual values in your environment.

```
bash-3.2$ db2p12top11 -to /home/db2inst1/sqllib/security/luna.cfg -pin userpin1
Migrating keys from <> local keystore
to PKCS#11 HSM using vendor library </usr/safenet/lunaclient/lib/libCryptoki2_64.so>
defined in configuration file </home/db2inst1/sqllib/security/luna.cfg>.

Migrating key: <DB2_SYSGEN_db2inst1_MIGRATE_2020-06-04-07.02.38_823A7A08> ... Successful.

Out of 1 key(s): 1 key(s) inserted successfully, 0 failed.
```

Executing the above command will migrate the master key present in the local keystore to the HSM partition.

```
lunacm:>partition contents

        The 'Crypto Officer' is currently logged in. Looking for objects
        accessible to the 'Crypto Officer'.

        Object list:

        Label:          DB2_SYSGEN_db2inst1_MIGRATE_2020-06-04-07.02.38_823A7A08
        Handle:         194
        Object Type:    Symmetric Key
        Object UID:     ab000000460000010940f0900


        Number of objects:  1


Command Result : No Error
```

4. Set the **ALLOW_KEY_INSERT_WITHOUT_KEYSTORE_BACKUP** parameter to **FALSE** in the PKCS #11 keystore configuration file (**luna.cfg**).
5. Configure the DB2 instance to use the PKCS #11 keystore by specifying **keystore_location** and **keystore_type**.

   ```
   $ db2 update dbm cfg using keystore_location
   /home/<db2_instance>/sqllib/security/luna.cfg
   ```

   ```
   $ db2 update dbm cfg using keystore_type pkcs11
   ```

   Change the variable provided in < > with actual values in your environment.
6. Restart the database for changes take effect.

   ```
   $ db2stop
   ```

   ```
   $ db2start
   ```

To Verify Encryption Using HSM Keystore

7. Connect to the database.

   ```
   $ db2 connect to <database_name>
   ```

Change the variable provided in < > with the actual values in your environment.

8. Type `db2` and press enter to launch interactive mode.
9. Display the contents of the **EMPLOYEE_SALARY** table with the following command:

   ```
   db2 => SELECT * FROM EMPLOYEE_SALARY
   ```

   The command should display the encrypted Table in clear text.
10. Rename or move the local keystore to ensure that it is not available and the database is being able to access the HSM master key.

    ```
    $ mv my-keystore.p12 my-keystore.p24
    ```
11. Verify that the database is accessing the new master key from HSM by running the below command.

    ```
    $ db2 get dbm cfg
    ```

    Ensure that the `KEYSTORE_TYPE` and `KEYSTORE_LOCATION` have the following values:

    ```
    Keystore type              (KEYSTORE_TYPE) = PKCS11
    Keystore location          (KEYSTORE_LOCATION)
    =/home/<db2_instance>/sqllib/security/luna.cfg
    ```

    ```
    Communication buffer exit library list (COMM_EXIT_LIST) =
    Current effective arch level         (CUR_EFF_ARCH_LVL) = V:11 R:5 M:0 F:0 I:0 SB:0
    Current effective code level         (CUR_EFF_CODE_LVL) = V:11 R:5 M:0 F:0 I:0 SB:0

    Keystore type                           (KEYSTORE_TYPE) = PKCS11
    Keystore location                    (KEYSTORE_LOCATION) = /home/db2inst1/sqllib/security/luna.cfg
    ```
12. Stop the NTLS service on HSM or break the NTLS connection.
13. Connect to the database using the below command.

    ```
    $ db2 connect to <database_name>
    ```

    Change the variable provided in < > with the actual values in your environment.

    You will see the following error as the NTLS is stopped:

    ```
    SQL1783N  The command or operation failed because an error was encountered
    accessing the PKCS #11 key manager. Reason code "13".
    ```
14. Start the NTLS service and restart database using the following commands.

    ```
    $ db2stop
    $ db2start
    ```

Generating Master Encryption Key directly onto the HSM

It is assumed that no local keystore has been created yet. The database has been installed and configured without any encryption in place and neither any keystore is created nor master key is generated. The steps below will create the HSM keystore and directly generate the master key on HSM partition registered.

To Create and Configure PKCS#11 HSM Keystore

1. Log on to the system as db2 instance owner.

2. To secure the master keys in a PKCS #11 keystore with DB2 native encryption, a configuration file must contains details of the HSM.

   On the DB2 server, create the PKCS #11 keystore configuration file **luna.cfg** with the following details:

```
-----------------------------------------------------------------------
VERSION=1
PRODUCT_NAME=Luna
ALLOW_KEY_INSERT_WITHOUT_KEYSTORE_BACKUP=true
LIBRARY=<LunaClient installation dir>/lib/libCryptoki2_64.so
SLOT_LABEL=<Partition_label>
NEW_OBJECT_TYPE=PRIVATE
KEYSTORE_STASH=/home/<db2_instance>/sqllib/security/pkcs11_pw.sth
-----------------------------------------------------------------------
```

   Change the variable provided in < > with the actual values in your environment.

   Where **SLOT_LABEL** identifies the slot in the HSM by a label. The label is a name that is defined by the application and is assigned during token initialization.

   KEYSTORE_STASH is the absolute path and name of the stash file that holds the PKCS #11 keystore password. The instance uses the stash file to authenticate to the PKCS #11 keystore.

   Ensure that the **ALLOW_KEY_INSERT_WITHOUT_KEYSTORE_BACKUP** parameter is set to **TRUE** in the PKCS #11 keystore configuration file.

3. Create a stash file.

```
$ db2credman -stash -password <partition password> -to
/home/<db2_instance>/sqllib/security/pkcs11_pw.sth
```

   Change the variable provided in < > with the actual values in your environment.

   NOTE: It should be noted that storing the PKCS#11 keystore password in a stash file is optional. If not specified password needs to be provided manually when prompt.

4. For a PKCS #11 keystore, set **keystore_location** to the absolute path of the PKCS #11 keystore configuration file **luna.cfg** and set **keystore_type** to "**PKCS11**".

```
$ db2 update dbm cfg using keystore_location
/home/<db2_instance>/sqllib/security/luna.cfg
$ db2 update dbm cfg using keystore_type pkcs11
```

   Change the variable provided in < > with actual values in your environment.

5. Restart the database to changes take the effect.

```
$ db2stop
$ db2start
```

To Create a New Encrypted Database

6. Specify the **encrypt** option on the **create db** command.

```
$ db2 create db <database_name> encrypt
```

Change the variable provided in < > with actual values in your environment.

This will create an encrypted database with the default settings. The command will create a new encrypted database and a master key in the HSM partition.

To Encrypt an Existing Database

7. Create a backup image of the database before encrypting an existing database.

```
$ db2 deactivate db <database_name>
$ db2 backup db <database_name> to <backup_directory>
```

Change the variable provided in < > with the actual values in your environment.

8. Drop the original copy of the database that needs to be encrypted.

```
$ db2 drop db <database_name>
```

Change the variable provided in < > with the actual values in your environment.

9. Change the current directory to the directory where backup of the database is saved.
10. Restore the unencrypted backup image with parameter **ENCRYPT** to create a new encrypted database using the Master Keystored in HSM partition.

```
$ db2 restore database <backup_database_name> into <new_dayabase_name> encrypt master key label <HSM_Master_Key_Label>
```

Change the variable provided in < > with the actual values in your environment. Where `HSM_Master_Key_Label` is the key label created on HSM partition.

To Verify Encryption Using HSM Keystore

11. Connect to the database.

```
$ db2 connect to <database_name>
```

Change the variable provided in < > with the actual values in your environment.

12. Type `db2` and press **Enter** to launch interactive mode.
13. Create a **STUDENT_MARKS** table in the database.

```
db2 => CREATE TABLE STUDENT_MARKS (CLASS_NO CHAR(3) NOT NULL,DEPTNAME VARCHAR(36) NOT NULL,STUDNO CHAR(6) NOT NULL,MARKS CHAR(6) NOT NULL WITH DEFAULT)
```

14. Enter the following values in the **STUDENT_MARKS** table.

```
db2 => INSERT INTO STUDENT_MARKS VALUES (10,'SCIENCE',001,95)
db2 => INSERT INTO STUDENT_MARKS VALUES (10,'COMMERCE',002,90)
db2 => INSERT INTO STUDENT_MARKS VALUES (10,'ARTS',003,85)
```

15. Display the contents of the **STUDENT_MARKS** table with the following command:

```
db2 => SELECT * FROM STUDENT_MARKS
```

16. Stop the NTLS service on HSM or break the NTLS connection.
17. If the HSM is not available, database will fail to connect and you will see the following error:

```
$ db2 connect to <database_name>
```

```
SQL1783N The command or operation failed because an error was encountered
accessing the PKCS #11 key manager. Reason code "13".
```

Change the variable provided in < > with the actual values in your environment.

18. Start the NTLS service and restart database, again to work everything.

```
$ db2stop
$ db2start
```

All databases encrypted using HSM keystore will only accessible when HSM is available and Master Key found in the registered HSM partition. This completes the IBM DB2 integration with Thales HSMs.

Contacting Customer Support

If you encounter a problem while installing, registering, or operating this product, refer to the documentation. If you cannot resolve the issue, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support.DIS@thalesgroup.com.