

Adabas Encryption for z/OS

Operations

Version 8.5.1

October 2021

This document applies to Adabas Encryption for z/OS Version 8.5.1 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2021 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: EZ-OPERATIONS-851-20210730EN

Table of Contents

| | |
|---|----|
| Operations | v |
| 1 About this Documentation | 1 |
| Document Conventions | 2 |
| Online Information and Support | 2 |
| Data Protection | 3 |
| 2 ADARUN ENCRYPTION Parameter | 5 |
| 3 Creating a New, Encrypted Database | 7 |
| 4 Creating an Encrypted Copy of an Existing Database | 9 |
| 5 Migrating to an Encrypted Database | 11 |
| Method 1: Adabas Down During SAVE & RESTORE DB | 12 |
| Method 2: Adabas Down During RESTONL DB | 13 |
| Method 3: Adabas Down During REGENERATE DB | 14 |
| Method 4: Adabas Down During Shorter REGENERATE DB | 15 |
| 6 Rotating the Dataset Encryption Key(s) | 19 |
| 7 Encrypting Only Parts of ASSO and DATA | 21 |
| 8 Copying Database Container Datasets | 23 |
| 9 Inspecting the Encryption Parameters and Attributes | 25 |
| Is Adabas running with encryption support? | 26 |
| Is my sequential Adabas input/output dataset encrypted? | 26 |
| Is my Adabas container dataset encrypted? | 26 |
| How can I see what data is actually stored on disk in my encrypted dataset? | 27 |

Operations

This chapter describes how to operate Adabas Encryption.

| | |
|---|--|
| ADARUN ENCRYPTION Parameter | |
| Creating a New, Encrypted Database | |
| Creating an Encrypted Copy of an Existing Database | |
| Migrating to an Encrypted Database | |
| Rotating the Dataset Encryption Key(s) | |
| Encrypting Only Parts of ASSO and DATA | |
| Copying Database Container Datasets | |
| Inspecting the Encryption Parameters and Attributes | |

1

About this Documentation

| | |
|--|---|
| ■ Document Conventions | 2 |
| ■ Online Information and Support | 2 |
| ■ Data Protection | 3 |

Document Conventions

| Convention | Description |
|----------------|--|
| Bold | Identifies elements on a screen. |
| Monospace font | Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties. |
| <i>Italic</i> | Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources. |
| Monospace font | Identifies: Text you must type in. Messages displayed by the system. Program code. |
| { } | Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols. |
| | Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol. |
| [] | Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols. |
| ... | Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...). |

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <https://documentation.softwareag.com>.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.aspx and give us a call.

Software AG Tech Community

You can find documentation and other technical information on the Software AG Tech Community website at <https://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have Tech Community credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 ADARUN ENCRYPTION Parameter

With the ADARUN parameter ENCRYPTION you specify that Adabas shall support encrypted database container datasets.

| Parameter | Specify... | Possible Values | Default |
|------------|--|-----------------|---------|
| ENCRYPTION | Whether or not to activate Adabas Encryption | YES NO | NO |

The ENCRYPTION parameter controls whether Adabas supports encrypted database container datasets.

If ENCRYPTION=YES is specified, any container dataset of the database (ASSO, DATA, WORK, CLOG, PLOG, RLOG, DSIM, SORT, or TEMP) may be encrypted.



Note: Adabas Encryption requires its own license module (named AEZLIC) or file (DD-name DDLAEZ), in addition to the standard Adabas license. If the AEZ license is missing or invalid, the Adabas nucleus will refuse to start.

If ENCRYPTION=NO is specified (the default), Adabas does not support encrypted database container datasets. In this case, if a database container dataset is encrypted, Adabas terminates with an error when it attempts to open the dataset.

Setting ENCRYPTION=YES is required if one or more of the database container datasets accessed by the Adabas nucleus or utility are encrypted. (YES may also be specified if none of the container datasets is encrypted.)

In a cluster, ENCRYPTION is a local, fixed parameter. Still, if ASSO, DATA, or any of the xLOG datasets are encrypted, ENCRYPTION must be set to YES for all nuclei in the cluster.

3 Creating a New, Encrypted Database

To create an encrypted database, allocate its container datasets as encrypted, format them and define the database:

1. Run ADAFRM ASSOFRM, DATAFRM and so on, specifying the database container datasets with DISP=(NEW,CATLG) and the encryption key label (DSKEYLBL parameter). Alternatively, the key label can be derived from the RACF profile or the SMS policy for each dataset.
2. Run ADADEF DEFINE on the database container datasets.

Step 1 is the only point in the lifetime of each dataset where its key label must be provided (via DSKEYLBL, RACF or SMS). The key label is then stored in the catalog entry for the dataset. Afterwards, it need not be specified and cannot be changed anymore.

When creating encrypted database container datasets, specify ADARUN parameter ENCRYPTION=YES in the ADAFRM job steps that format the datasets as well as in all later Adabas nucleus or utility jobs or started tasks that access any of the datasets.

In the following example, two job steps create, format, and define a database with ASSO, DATA, WORK and PLOG datasets that are encrypted using the key referred to by 'ADABAS.KEY.LABEL'. Parameters required for the creation and use of encrypted database container datasets are shown in italics:

```
/** CREATE AND FORMAT NEW DATABASE CONTAINER DATASETS
/**
//FORMAT EXEC PGM=ADARUN
//DDASSOR1 DD DSN=ADABAS.DB215.ASSOR1,DISP=(NEW,CATLG),
//           UNIT=3390,SPACE=(CYL,100),
//           DSKEYLBL='ADABAS.KEY.LABEL'
//DDDATAR1 DD DSN=ADABAS.DB215.DATAR1,DISP=(NEW,CATLG),
//           UNIT=3390,SPACE=(CYL,300),
//           DSKEYLBL='ADABAS.KEY.LABEL'
//DDWORKR1 DD DSN=ADABAS.DB215.WORKR1,DISP=(NEW,CATLG),
//           UNIT=3390,SPACE=(CYL,20),
```

```
//          DSKEYLBL='ADABAS.KEY.LABEL '
//DDPLOGR1 DD DSN=ADABAS.DB215.PLOGR1,DISP=(NEW,CATLG),
//          UNIT=3390,SPACE=(CYL,20),
//          DSKEYLBL='ADABAS.KEY.LABEL '
//DDPLOGR2 DD DSN=ADABAS.DB215.PLOGR2,DISP=(NEW,CATLG),
//          UNIT=3390,SPACE=(CYL,20),
//          DSKEYLBL='ADABAS.KEY.LABEL '
//DDDRUCK DD SYSOUT=*
//DDPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DDCARD DD *
ADARUN PROG=ADAFRM,DBID=215,SVC=249,DEVICE=3390,MODE=MULTI
ADARUN ENCRYPTION=YES
//DDKARTE DD *
ADAFRM ASSOFRM SIZE=100
ADAFRM DATAFRM SIZE=300
ADAFRM WORKFRM SIZE=20
ADAFRM PLOGFRM SIZE=20,NUMBER=1
ADAFRM PLOGFRM SIZE=20,NUMBER=2
//*
//* DEFINE NEW DATABASE IN ENCRYPTED CONTAINER DATASETS
//*
//DEFINE EXEC PGM=ADARUN
//DDASSOR1 DD DSN=ADABAS.DB215.ASSOR1,DISP=OLD
//DDDATAR1 DD DSN=ADABAS.DB215.DATAR1,DISP=OLD
//DDWORKR1 DD DSN=ADABAS.DB215.WORKR1,DISP=OLD
//DDDRUCK DD SYSOUT=*
//DDPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DDCARD DD *
ADARUN PROG=ADADEF,DBID=215,SVC=249,DEVICE=3390,MODE=MULTI
ADARUN ENCRYPTION=YES
//DDKARTE DD *
ADADEF DEFINE DBID=215,DBNAME='MY ENCRYPTED DB'
ADADEF ASSOFSIZE=100,DATASIZE=300,WORKSIZE=20
ADADEF MAXFILES=255,FILE=1,CHECKPOINT
ADADEF MAXISN=1000,DSSIZE=10B,NISIZE=5B,UISIZE=3B
```

See sample job CRE010 in the AEZ_{vr}s.JOBS library.

4 Creating an Encrypted Copy of an Existing Database

To create an encrypted copy of an existing database while keeping the database active, create and format new container datasets for a copy of the database, save the database (online or offline), and restore it into the new container datasets:

1. Run *ADAFRM ASSOFRM*, *DATAFRM* and so on, specifying the database container datasets with *DISP=(NEW,CATLG)* and the encryption key label (*DSKEYLBL* parameter). Alternatively, the key label can be derived from the RACF profile or the SMS policy for each dataset.
2. If an up-to-date copy of the database is desired, run *ADASAV SAVE* (database) online or offline. Otherwise, use the save dataset(s) from a previous database save operation.
3. If an online save was performed in step 2, force a PLOG switch (*FEOFPL*) and wait for the resulting *ADARES PLCOPY* job to finish.
4. Using the save dataset(s), run *ADASAV RESTORE* — or if an online save was taken, *ADASAV RESTONL* with the sequential PLOG dataset(s) spanning the online save operation — to restore the database into the new, encrypted container datasets.

If appropriate for the circumstances, use the *NEWDBID* parameter to set a new DBID for the encrypted database.

See sample job DUP020 in the *AEZ_{vrs}.JOBS* library.

5

Migrating to an Encrypted Database

| | |
|--|----|
| ■ Method 1: Adabas Down During SAVE & RESTORE DB | 12 |
| ■ Method 2: Adabas Down During RESTONL DB | 13 |
| ■ Method 3: Adabas Down During REGENERATE DB | 14 |
| ■ Method 4: Adabas Down During Shorter REGENERATE DB | 15 |

Migrating an existing database to a new, encrypted one necessitates some downtime of Adabas while new copies of the database container datasets are created or brought up to date. Simple migration methods cause downtime proportional to the size of the database, which can be substantial. More complex migration methods shorten the downtime, making it proportional to the amount of update processing in the database during the migration. This section describes four methods, in the order of decreasing downtime and increasing complexity.

Method 1: Adabas Down During SAVE & RESTORE DB

Method 1 is the simplest: you shut down Adabas, save the database and restore it into the new, encrypted container datasets. This is a good approach for databases that are small or that can be taken offline at the right moment for the duration it takes to save and restore them.

To perform Method 1, follow these steps:

1. Run ADAFRM ASSOFRM, DATAFRM and so on, specifying the database container datasets with new (temporary) names, DISP=(NEW,CATLG) and the encryption key label (DSKEYLBL parameter). Alternatively, the key label can be derived from the RACF profile or the SMS policy for each dataset.
2. *Shut down Adabas* on the existing database and wait for the final ADARES PLCOPY job to finish.
3. Run ADASAV SAVE to save the database (offline).
4. Run ADASAV RESTORE to restore the saved database into the new, encrypted container datasets.
5. Delete (or rename) the old database container datasets and rename the new container datasets to the old names.
6. Restart Adabas on the new, encrypted database with the original dataset names.

See sample job MIG030 in the AEZ_{vr}s.JOBS library.

With this method, Adabas is down between steps 2 and 6. The main contributors to the downtime are:

- Step 2 – ADARES PLCOPY
- Step 3 – ADASAV SAVE (database)
- Step 4 – ADASAV RESTORE (database)

Method 2: Adabas Down During RESTONL DB

The preceding section described Method 1, which takes the database down while it is being saved and restored. This section describes Method 2, which reduces the duration of downtime by performing the save online, while Adabas stays active.

Method 2 creates the new instance of the database while the old one is still active. One way to keep them apart is to continue using the regular Adabas SVC for the database in the old container datasets and to use another, temporary SVC for the database in the new datasets while they are brought up to date.



Note: Method 2 becomes more complicated if the database gets updated by utilities (ADAINV, ADALOD, ADAORD, ADARES BACKOUT/REGENERATE, ADASAV RESTORE/RESTONL) after the online database save operation at the start of the migration procedure.

To perform Method 2, follow these steps:

1. Run ADAFRM ASSOFRM, DATAFRM and so on, specifying the database container datasets with new (temporary) names, DISP=(NEW,CATLG) and the encryption key label (DSKEYLBL parameter). Alternatively, the key label can be derived from the RACF profile or the SMS policy for each dataset.
2. Run ADASAV SAVE to save the database (online).
3. *Shut down Adabas* on the existing database and wait for the final ADARES PLCOPY job to finish.
4. Run ADASAV RESTONL to restore the saved database into the new, encrypted container datasets.

Provide the sequential PLOG dataset written by the PLCOPY execution triggered in step 3. If a PLOG switch occurred during the online save in step 2, provide to the restore operation all sequential PLOG datasets spanning the online save (from the SYN1 to the SYN2 checkpoint).

5. Run ADARES REGENERATE on the new datasets to regenerate the last updates to the database from the end of the online save until its shutdown (between steps 2 and 3).

Provide the sequential PLOG (or PLOGs) from the SYN2 checkpoint till the session end. There should be only a few updates, so it may be simplest to run this REGENERATE with MODE=SINGLE.

6. Run ADADBS RESETPPT to reset the PPT of the database in the new datasets before they are renamed. This step is optional. It prevents a warning message in step 8 when Adabas restarts on the renamed datasets.
7. Delete (or rename) the old database container datasets and rename the new container datasets to the old names.
8. *Restart Adabas* on the new, encrypted database with the original dataset names.

See sample job MIG040 in the AEZ*vers*.JOBS library.

With this method, Adabas is down between steps 3 and 8. The main contributors to the downtime are:

- Step 3 – ADARES PLCOPY
- Step 4 – ADASAV RESTONL (database)

Method 3: Adabas Down During REGENERATE DB

With Method 1 and Method 2 (described in the preceding sections), the downtime of the database is proportional to its size.

Method 3 (described in this section) reduces the downtime with additional steps for regenerating updates, such that the duration of the downtime becomes proportional to the amount of update processing during the migration process.

Method 3 creates the new instance of the database while the old one is still active. One way to keep them apart is to continue using the regular Adabas SVC for the database in the old container datasets and to use another, temporary SVC for the database in the new datasets while they are brought up to date.



Note: Method 3 becomes more complicated if the database gets updated by utilities (ADAINV, ADALOD, ADAORD, ADARES BACKOUT/REGENERATE, ADASAV RESTORE/RESTONL) after the online database save operation at the start of the migration procedure.

To perform Method 3, follow these steps:

1. Run ADAFRM ASSOFRM, DATAFRM and so on, specifying the database container datasets with new (temporary) names, DISP=(NEW,CATLG) and the encryption key label (DSKEYLBL parameter).

Alternatively, the key label can be derived from the RACF profile or the SMS policy for each dataset.

2. Run ADASAV SAVE to save the database (online).
3. Run ADADBS OPERCOM FEOFPL to force a PLOG switch.

Wait for the associated ADARES PLCOPY job to finish.

4. Run ADASAV RESTONL to restore the saved database into the new, encrypted container datasets.

Provide the sequential PLOG dataset written by the PLCOPY execution triggered in step 3. If a PLOG switch occurred during the online save in step 2, provide to the restore operation all sequential PLOG datasets spanning the online save (from the SYN1 to the SYN2 checkpoint).

5. Start a second Adabas nucleus on the new database container datasets (using another, temporary Adabas SVC).

This nucleus will be used only temporarily and does not need PLOGs. It may be set to UTIONLY=YES.

6. *Shut down Adabas* on the existing database container datasets and wait for the final ADARES PLCOPY job to finish.
7. Run ADARES REGENERATE against the nucleus on the new database container datasets to regenerate the updates to the database from the end of the online save until its shutdown (between steps 2 and 6).
8. Shut down the second Adabas nucleus on the new database container datasets.
9. Run ADADBS RESETPPT to reset the PPT of the database in the new datasets before they are renamed.

This step is optional. It prevents a warning message in step 11 when Adabas restarts on the re-named datasets.

10. Delete (or rename) the old database container datasets and rename the new container datasets to the old names.
11. *Restart Adabas* on the new, encrypted database with the original dataset names.

See sample job MIG050 in the AEZ_{vrs}.JOBS library.

With this method, Adabas is down between steps 6 and 11. The main contributors to the downtime are:

- Step 6 – ADARES PLCOPY
- Step 7 – ADARES REGENERATE (database) for all updates between steps 3 and 6, which includes the time needed for the ADASAV RESTONL (database)

Method 4: Adabas Down During Shorter REGENERATE DB

With Method 3 (described in the previous section), the database is down while all the updates made after the end of the online save are regenerated. Method 4 (described in this section) can reduce this downtime further by keeping the database active while the updates made during the database restore operation are being regenerated. Thereafter, the database is shut down to regenerate the final updates from the end of the database restore.

Method 4 creates the new instance of the database while the old one is still active. One way to keep them apart is to continue using the regular Adabas SVC for the database in the old container datasets and to use another, temporary SVC for the database in the new datasets while they are brought up to date.



Note: Method 4 becomes more complicated if the database gets updated by utilities (ADAINV, ADALOD, ADAORD, ADARES BACKOUT/REGENERATE, ADASAV RESTORE/RESTONL) after the online database save operation at the start of the migration procedure.

To perform Method 4, follow these steps:

1. Run ADAFRM ASSOFRM, DATAFRM and so on, specifying the database container datasets with new (temporary) names, DISP=(NEW,CATLG) and the encryption key label (DSKEYLBL parameter).

Alternatively, the key label can be derived from the RACF profile or the SMS policy for each dataset.

2. Run ADASAV SAVE to save the database (online).
3. Run ADADBS OPERCOM FEOFPL to force a PLOG switch.

Wait for the associated ADARES PLCOPY job to finish.

4. Run ADASAV RESTONL to restore the saved database into the new, encrypted container datasets.

Provide the sequential PLOG dataset written by the PLCOPY execution triggered in step 3. If a PLOG switch occurred during the online save in step 2, provide to the restore operation all sequential PLOG datasets spanning the online save (from the SYN1 to the SYN2 checkpoint).

5. Start a second Adabas nucleus on the new database container datasets (using another, temporary Adabas SVC).

This nucleus will be used only temporarily and does not need PLOGs. It may be set to UTIONLY=YES.

6. Run ADADBS OPERCOM FEOFPL to force another PLOG switch.

Wait for the associated ADARES PLCOPY job to finish.

7. Run ADARES REGENERATE with NOAUTOBACKOUT against the nucleus on the new database container datasets to regenerate the updates to the database from the end of the online save until the second PLOG switch (between steps 2 and 6).

Provide all sequential PLOG datasets from the one containing the SYN2 checkpoint (end of step 2) to the last one created so far (in step 6).

8. *Shut down Adabas* on the existing database container datasets and wait for the final ADARES PLCOPY job to finish.

9. Run ADARES REGENERATE (without NOAUTOBACKOUT!) against the nucleus on the new database container datasets to regenerate the updates to the database from the end of the PLOG switch forced in step 6 until its shutdown (i.e., between steps 6 and 8).

Provide the sequential PLOG dataset(s) following the last PLOG in the previous REGENERATE (in step 7).

10. Shut down the second Adabas nucleus on the new database container datasets.
11. Run ADADBS RESETPPT to reset the PPT of the database in the new datasets before they are renamed.

This step is optional. It prevents a warning message in step 13 when Adabas restarts on the re-named datasets.

12. Delete (or rename) the old database container datasets and rename the new container datasets to the old names.

13. *Restart Adabas* on the new, encrypted database with the original dataset names.

See sample job MIG060 in the AEZ_{VRS}.JOBS library.

With this method, Adabas is down between steps 8 and 13. The main contributors to the downtime are:

- Step 8 – ADARES PLCOPY
- Step 9 – ADARES REGENERATE (database) for all updates between steps 6 and 8, which includes only the time needed for the first REGENERATE (database)

If the final ADARES REGENERATE (in step 9) still keeps Adabas down longer than desired, it is possible to repeat steps 6 and 7 (FEOFPL and REGENERATE) until the duration of the final REGENERATE after the shutdown of Adabas is short enough.

6 Rotating the Dataset Encryption Key(s)

The lifetime of an encryption key for data at rest may be shorter than the lifetime of the data it protects. When an encryption key is to be retired but the data it protects must stay (readable and usable), the data must be re-encrypted with a new key. The process is the same as for migrating unencrypted data to encrypted data for the first time.

When new encryption keys are to be established for Adabas container datasets, use the same process as for encrypting the datasets for the first time:

- For ASSO and DATA, use one of the methods for migrating to an encrypted database described in the previous section [Migrating to an Encrypted Database](#).
- For the other database container datasets (WORK, PLOG, and so on), use the appropriate method described in *Encryption of Container Datasets*.

Specify new key labels for the new datasets.

7

Encrypting Only Parts of ASSO and DATA

As with all datasets on DASD, z/OS establishes the encryption-related attributes of Adabas database container datasets when the datasets are created. The Adabas component responsible for I/O operations (ADAIOR) recognizes encrypted datasets and adjusts its processing logic to deal with them properly. The Adabas nucleus and utilities are for the most part unaware of the encryption of database container datasets.

It is technically possible to encrypt some of the ASSO and DATA container datasets and to leave others unencrypted — for example, to encrypt ASSOR1 and DATAR1 but not ASSOR2 and DATAR2. ADAIOR performs the necessary encryption and decryption operations and makes the presence of the encryption transparent to the nucleus and utilities.

Conceivable motivations for encrypting only parts of ASSO and DATA might be:

- To encrypt only sensitive data
- To save part of the CPU overhead inherent with encryption

If only parts of ASSO and DATA are encrypted, sensitive files would be put on encrypted containers (for both ASSO and DATA!) and the other files, on unencrypted containers (to keep enough free space for the sensitive files).

Such a setup is possible but *not recommended*. The fact that the Adabas nucleus and utilities are not involved in the encryption logic generates practical issues:

- The DBA must direct the placement of all files manually (for example, using the xxRABN or xxxxVOLUME parameters in ADALOD, ADAORD or ADASAV), distinguishing the sensitive files from the others.
- The DBA must ensure manually that the Adabas nucleus and utilities do not perform secondary file extent allocations automatically, which might not adhere to the intentions of the DBA.

For many databases, organizing the placement of files in the database by targeting the desired database container datasets is probably impractical.

8

Copying Database Container Datasets

Non-Adabas programs are not as versatile in copying encrypted Adabas database container datasets as they are with unencrypted container datasets.

Encrypted database container datasets cannot be copied using non-Adabas programs that read the datasets record by record, such as:

- IEBGENER, ICEGENER
- IDCAMS REPRO
- ISPF 3.3
- SORT MERGE with FIELDS=COPY
- Any program that uses BSAM or QSAM to read the dataset

Even if the user running the program has the authorization to use the dataset's encryption key label, BSAM and QSAM cannot decrypt Adabas container datasets. Trying to open an encrypted Adabas container dataset with BSAM or QSAM results in system abend 219 with return code 99.

In contrast, encrypted Adabas database container datasets can still be copied using non-Adabas programs that read and write the datasets track by track, such as:

- DFSMSdss COPY
- FlashCopy®, Concurrent Copy®, and so on.

These programs copy the data as is, without decrypting and re-encrypting it. They do not need the authorization to access the encryption key label either.

For instance, a procedure like the following also works if the database container datasets are encrypted:

1. ADADBS TRANSACTIONS SUSPEND,TTSYN=...,TRESUME=...
2. Copy the database container datasets using FlashCopy, Concurrent Copy, or a similar program

3. ADADBS TRANSACTIONS RESUME

9 Inspecting the Encryption Parameters and Attributes


| | |
|---|----|
| ■ Is Adabas running with encryption support? | 26 |
| ■ Is my sequential Adabas input/output dataset encrypted? | 26 |
| ■ Is my Adabas container dataset encrypted? | 26 |
| ■ How can I see what data is actually stored on disk in my encrypted dataset? | 27 |

This section shows where to see how Adabas Encryption is working.

Is Adabas running with encryption support?

- Look into the DDPRINT protocol dataset of the Adabas nucleus or utility in question and check whether it contains ADARUN parameter ENCRYPTION=YES.
- Alternatively, for the Adabas nucleus, navigate to the *Display Installed Products* screen in the Adabas Online System (via option 'A' from the main menu and then option 'I' from the Session Monitoring menu) and check the 'Encryption' setting on the right-hand side.

Is my sequential Adabas input/output dataset encrypted?

-  **Note:** Just browsing through a sequential dataset does *not* show whether it is encrypted. If it is encrypted, DFSMS will decrypt it on the fly and show the plaintext data.
- Look into the catalog entry of the dataset using TSO or IDCAMS LISTCAT ENTRIES('dataset-name') ALL and check the 'ENCRYPTIONDATA' section.

Is my Adabas container dataset encrypted?

- Look into the DDPRINT protocol dataset of the Adabas nucleus or a utility that uses the container dataset and check the ADAI64 message for the dataset. If the dataset is encrypted, this message includes a third line saying that the 'File dd-name' or 'Dataset dataset-name' is encrypted.
- If you try to browse an encrypted Adabas container dataset like a sequential dataset, DFSMS will raise system abend 213-99. This can be taken as an indication that the dataset is encrypted.
- As a more regular way, look into the catalog entry of the dataset using TSO or IDCAMS LISTCAT ENTRIES('dataset-name') ALL and check the 'ENCRYPTIONDATA' section.

How can I see what data is actually stored on disk in my encrypted dataset?

- Determine the cylinder/track location of the dataset on disk and use the DFSMS*dss* PRINT DATASET or PRINT TRACKS functions to print its unaltered contents. For an encrypted dataset, expect to see random data without any discernible structure.

