EVIDEN

Identity and Access Management

DirX Directory

Recovery

Version 9.1, Edition June 2025

Table of Contents

Preface

This document describes detection and recovery procedures that can be used to recover the DirX Directory Basic Access Method (DBAM) database component of a deployed DirX Directory service from general problems and from the main types of database errors that can occur.

The goal of this document is to provide recommendations, guidelines, and procedures for recovering from DBAM database errors both generally and for a set of specific error cases. The step-by-step procedures collect and summarize information about detecting and recovering from database errors that is described in more detail in other documents in the DirX Directory set.

This document consists of the following chapters:

- Chapter 1 describes the prerequisites to using the recovery procedures described in the document and defines terms used in recovery procedure descriptions
- Chapter 2 describes the steps that should be followed to recover from any type of DBAM database error
- Chapter 3 describes procedures for determining and recovering from specific types of DBAM database errors
- Chapter 4 describes the main methods that can be used to restore a DBAM database when the type of database error cannot be determined

DirX Directory Documentation Set

DirX Directory provides a powerful set of documentation that helps you configure your directory server and its applications.

The DirX Directory document set consists of the following manuals:

- *DirX Directory Introduction*. Use this book to obtain a description of the concepts of DirX Directory.
- *DirX Directory Administration Guide*. Use this book to understand the basic DirX Directory administration tasks and how to perform them with the DirX Directory administration tools.
- *DirX Directory Administration Reference*. Use this book to obtain reference information about DirX Directory administration tools and their command syntax, configuration files, environment variables and file locations of the DirX Directory installation.
- *DirX Directory Syntaxes and Attributes*. Use this book to obtain reference information about DirX Directory syntaxes and attributes.
- *DirX Directory LDAP Extended Operations*. Use this book to obtain reference information about DirX Directory LDAP Extended Operations.
- *DirX Directory External Authentication*. Use this book to obtain reference information about external authentication.
- *DirX Directory Supervisor*. Use this book to obtain reference information about the DirX Directory supervisor.
- *DirX Directory Plugins for Nagios*. Use this book to obtain reference information about DirX Directory plugins for Nagios.
- *DirX Directory Disc Dimensioning Guide*. Use this book to understand how to calculate and organize necessary disc space for initial database configuration and enhancing existing configurations.
- *DirX Directory Guide for CSP Administrators*. Use this book to obtain information about installing, configuring and managing DirX Directory in the context of a Certificate Provisioning Service operating in accordance with regulations like the German “Signaturgesetz”.
- *DirX Directory Release Notes*. Use this book to install DirX Directory and to understand the features and limitations of the current release.

Notation Conventions

Boldface type
In command syntax, bold words and characters represent commands or keywords that must be entered exactly as shown.

In examples, bold words and characters represent user input.

Italic type
In command syntax, italic words and characters represent placeholders for information that you must supply.

[ ]
In command syntax, square braces enclose optional items.

{ }
In command syntax, braces enclose a list from which you must choose one item.

In Tcl syntax, you must actually type in the braces, which will appear in boldface type.

|
In command syntax, the vertical bar separates items in a list of choices.

...
In command syntax, ellipses indicate that the previous item can be repeated.

install_path
The exact name of the root of the directory where DirX Identity programs and files are installed. The default installation directory is *userID_home_directory**/DirX Identity* on UNIX systems and **C:\Program Files\DirX\Identity** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathnames is represented by the notation *install_path*.

1. Overview

This document provides information on how to recover from inconsistencies in the DBAM database component of a DirX Directory service. It describes:

- The general process for recovering from a DBAM database inconsistency
- Methods for restoring the DBAM database to a consistent state that are generic to any type of database inconsistency
- Methods for detecting and recovering from particular types of DBAM database inconsistencies

In descriptions of recovery procedures in this document:

- The general term “database” refers to 1) the target DBAM database being operated on with DirX commands and 2) a target Directory Information Tree (DIT) being supplied or replicated on a DirX DSA.
- The general term “node” refers to a target DirX DSA in a shadowing configuration, either a supplier DSA or a consumer DSA.
- The general term “system” refers to a target DirX Directory service configuration

The next sections describe prerequisites for using the recovery procedures on a DirX Directory service and list the related DirX documentation that should be consulted before following the recovery procedures and used for reference when following these procedures.

1.1. Prerequisites

Using the recovery procedures described in this document imposes the following prerequisites on the target DirX Directory service:

- Backup archives of the database must be created frequently with the **dirxbackup** command.We recommend performing daily backups.See the *DirX Directory Administration Reference* for usage information on the **dirxbackup** tool.
- The consistency of each newly created backup archive must be verified with the **dbamverify** command.We recommend using all the tool’s verification options (**-AXDST**).See the *DirX Directory Administration Reference* for usage information on the **dbamverify** tool.
- The shadowing scenario for the DirX directory system in use must consist of a supplier DSA and at least one consumer DSA that replicates the entire database; this type of consumer configuration is called a “full shadow”.Having one or more full shadows provides a more recent database snapshot than the content in the last saved backup should a recovery operation become necessary.See the *DirX Directory Administration Guide* for details on shadowing scenarios and the *DirX Directory Administration Reference* for usage information on performing shadowing operations with the **dirxadm** command.
- The recommendations for maintaining the database (for example, periodically creating

an LDIF dump file of the database and checking it for errors) that are given in the chapter "Recommendations for an operation concept" in the release notes for the DirX Directory version in use must be followed.

If the directory service in use does not meet one of these conditions, another recovery method can be used that does not depend on it.However, we highly recommend implementing these prerequisites at a minimum to provide a directory service that can be restored in case of a database error.

1.2. Related Documentation

The following documents provide additional details about the concepts and procedures referenced in this document.We recommend that you become familiar with the information in these documents before proceeding with the tasks described in this document:

DirX Directory Administration Reference:

- **dirxadm** command and the following operations:
- **audit** object
- **db check** operation
- **sob** object
- **lob** object
- **sys start** and **sys stop** operations
- **util** object (**dirxadm** utilities like **ldif_dump**, **export_dbconfig**, **import_dbconfig**)
- **dbamboot** command
- **dbamverify** command
- **dirxbackup** command
- **dirxdumplog** command
- **dirxload** command
- **dirxmodify** command
- The environment variable **$DIRX_INST_PATH**

DirX Directory Administration Guide

- Using the DirX Directory Administration Tools
- Performing a Total Update by Media
- Creating a Shadow DSA
- Creating a Synchronous Shadow DSA

the **dbamdump** command is used in several recovery procedures described in this document. This command is an internal tool used primarily by DirX Directory support and is not described in the DirX

> documentation.

2. General Database Error Recovery Process

The general process to recover from a database inconsistency is:

- Determine the type of database inconsistency and its severity
- Collect all logs and data related to the inconsistency
- Report the database inconsistency to DirX Directory support, supplying the collected logs and data
- Choose a recovery strategy according to the type and severity of the inconsistency
- Preserve all data related to the inconsistency

The next sections describe these steps in more detail.

2.1. Determining the Type of Database Inconsistency

When a database inconsistency is detected, the first step is to determine its type:

- If the database inconsistency is detected by the **dbamverify** command, its type can usually be determined by examining the **dbamverify** command-line output and the generated log files.The chapter “Error-Specific Database Recovery Procedures” describes how to analyze the **dbamverify** output for the main types of database inconsistencies that can occur and how to recover from these errors.
- If the inconsistency is detected via DSA malfunction, it can be difficult to determine the type of database inconsistency that has caused the DSA problem.The chapter “General Methods for Database Recovery” describes the main types of recovery procedures you can use when you can’t determine the type of database inconsistency.

2.2. Determining the Database Inconsistency Severity

The second step should be to determine the severity of the inconsistency:

- Check whether a consistent database exists in the system, either a supplier for a consumer inconsistency or a consumer for a supplier inconsistency.
- Collect information about the available backups: check whether there is a recent LDIF dump or a verified backup archive available and when it was created.This step is important because the availability of a consistent database and verified backups will determine the extent of the data loss during the recovery.

2.3. Collecting Data for Analysis

The files listed below should be collected and provided with all database inconsistency reports to DirX Directory support:

- **dbamverify**
- **$DIRX_INST_PATH/tools/log/LOG*_dbamverify_PID.sequence_number_ of the erroneous *dbamverify** execution, decoded with **dirxdumplog**
- The information written into the **stdout** and **stderr** output of **dbamverify**
- DSA
- **DSA_EXC*_DSA_PID.sequence_number_ (on Linux) and *DSA_LOG*_DSA_PID_** .*_sequence_number_ written between the last error-free backup and the detection of the inconsistency
- DSA_LOG should be decoded using the **dirxdumplog** tool
- *$DIRX_INST_PATH/server/log/fatalDSA_*_PID_**_helper_number_
- **$DIRX_INST_PATH/server/log/schema*_DSA_PID_**.txt*
- Watchdog
- **SRV_EXC*_Watchdog_PID.sequence_number_ (on Linux) and *SRV_LOG*_Watchdog_PID_**.*_sequence_number_
- SRV_LOG should be decoded using the **dirxdumplog** tool
- *$DIRX_INST_PATH/server/log/fatalSRV_*_PID___helper_number_
- Audit
- DSA audit created between the last error-free backup and the detection of the inconsistency
- If DSA audits are not available, then LDAP audit logs of all the LDAP servers from the same time period
- All audit files should be decoded with **dirxauddecode -v -v**

Additional error-specific log files and other output may also need to be provided.See the chapter “Error-specific Database Recovery Procedures” for details on the files to be provided for the different error types.

2.4. Reporting the Inconsistency to DirX Directory Support

As database inconsistencies are usually caused by a software problem in the DSA or some external circumstances, we recommend opening a ticket as soon as possible to report a database inconsistency.Repairing a database inconsistency with one of the methods described here without fixing the root cause of the problem means that sooner or later, the database may become inconsistent again.

Provide at least the following information in all tickets opened for database inconsistencies:

- The DirX Directory version (the output of the **dirxdsa -V** command)
- The platform in use:
- Windows or Linux
- Virtual or physical machine
- Virtualization platform
- The output of the **dirxadm sob show all** and **dirxadm lob show all** operations executed on the supplier node
- The node(s) on which the inconsistency was observed and the node(s) that are error free
- The type and age of the available backups
- When the **dirxbackup** archive files were verified and the **dbamverify** options that were used for verification
- The log and data files you collected in the previous step

2.5. Choosing a Database Recovery Strategy

After sending a report to DirX Directory support about the problem, the next step is to determine the recovery procedure to use.

If you've determined the type of inconsistency and there is a specific recovery procedure for it (see the chapter "Error-specific Database Recovery Procedures"), follow that procedure.

If you have not been able to determine the type of inconsistency, use the following guidelines to determine the best recovery strategy for your scenario:

- *Is the problem causing a severe system outage*?A solution to a database inconsistency is highly dependent on what type of problem it is.Even if all nodes in a system report an inconsistent state, a database problem can sometimes be solved without having to restore the database from a backup.Thus, if the problem does not cause a severe system outage, consider contacting DirX Directory support and waiting for a possible solution that does not involve data loss.
- *Is there a consistent database in the system*?A consistent database on one of the nodes in the system allows you to use the "Restore by Initiating a Total Update" method described in the chapter "General Methods Database Recovery" and minimize or avoid data loss.
- *Is there a recent LDIF dump or a verified backup archive*?When the system has no consistent database to use for recovery, a backup must be used instead.If an LDIF dump exists, you can use the "Restore Using an LDIF Dump" method described in the chapter "General Methods fo Database Recovery" to recover the database.If a verified **dirxbackup** archive file exists, you can use the "Restore Using a Binary Backup" method described in the chapter "General Methods for Database Recovery" to recover the database.We generally recommend using the "Restore Using an LDIF Dump" method because it completely rebuilds the internal structure of the database.

If the system does not have a node with a consistent database and there is no LDIF dump or verified backup archive of the database to use for recovery, the database may be lost.

2.6. Preserving Relevant Data

The section “Collecting Data for Analysis” describes the log files that are usually necessary for investigating a database inconsistency.However, there are more log files that may be necessary in a later phase of the investigation.To make sure that all the data is available for finding the inconsistency, we recommend backing up all the log files, audit files, files in the **tmp** directory, and database backups.The backup should contain at least the files modified between the creation of the last error-free database and the detection of the inconsistency.

In some cases, the root cause of the inconsistency originates from an earlier problem or operation, and it may become necessary to obtain log files that were created before the last consistent backup; for example, the log of the purge operations of the last month in case of a tree inconsistency.Thus, if there is an automatic cleanup mechanism activated, it should be paused for the duration of the analysis to preserve all the log files that may become necessary during the analysis.

3. Error-specific Database Recovery Procedures

The previous chapter described the general process to follow to recover a DBAM database. This chapter describes specific recovery processes for the main types of inconsistencies that can occur in a DBAM database. Each section in this chapter describes one of these inconsistencies, including what it means, how to detect it, additional data that should be collected for analysis, and how to recover from it.

3.1. Attribute Index Inconsistency

Attribute index inconsistency means that there is an error in the stored attribute indexes, either a structural inconsistency in the stored indexes, or misinformation stored in the attribute index; for example, an entry is present in the index but shouldn't be there based on its data.This type of inconsistency can lead to incorrect search results or to rejected modifications in the case of a structural inconsistency.

3.1.1. How to Detect It

The **dbamverify** command writes the result of attribute index checks into the **aidxcheck*_dbamverify_PID_**.txt* file in the **$DIRX_INST_PATH/tools/log** directory.At the end of this file, there is a table with a line "attribute index(es) inconsistent".If this line starts with a number other than zero, there is an attribute index inconsistency in the database.

3.1.2. Additional Data to be Collected for Analysis

The **dbamverify** command writes detailed logs about attribute index checks into the **aidxcheck*_dbamverify_PID_**.txt* file it creates in the **$DIRX_INST_PATH/tools/log** directory.Be sure to provide this file for analysis in addition to the files described in the section "Collecting Data for Analysis" in the chapter "General Database Error Recovery Process".

3.1.3. Specific Recovery Steps

Attribute indexes can be easily repaired or rebuilt using **dirxadm**. Most attribute index problems can be resolved by running the **db check** operation with the **-repair** option or by rebuilding the attribute index if the repair operation doesn't fix the problem. To perform this recovery procedure:

- Run the following **dirxadm** operation to repair the index(es):

  db check -bs ATTRIBUTE [**-attribute** *attribute_type*] **-repair**

  If the problem affects only one attribute index or just a few of them, specify the attribute type to avoid checking all the attribute indices.

- The **db check** operation produces the same **aidxcheck*_PID_**.txt* file as **dbamverify**, but in the **$DIRX_INST_PATH/server/log** directory.After the **db check** operation has finished,

check this generated **aidxcheck** file.At the end of the file, there are two lines: “attribute index(es) inconsistent” and “attribute index(es) repaired”.If the numbers at the beginning of the lines are equal, the inconsistency was repaired, and you can proceed to the next step in the general database recovery process.

- If the numbers are different, run the following **dirxadm** operation to rebuild the index:

  db attrconfig *attribute_type_abbreviation* **-index BUILD** *index_type(s)*

  If *index_type(s)* is not specified, the command re-creates an index as it was originally configured.

- After the index is rebuilt, run the **dirxadm db check** command again without the **-repair** option to check whether the inconsistencies have been resolved.If the “attribute index(es) inconsistent” line at the end of the **aidxcheck** file shows zero inconsistencies, the database recovery was successful, and you can proceed to the next step in the general recovery process.
- If the inconsistency cannot be resolved with the **dirxadm** repair or rebuild operations, use one of the general recovery methods described in the next chapter to restore the database.

3.2. Subordinate Bit String Inconsistency

Subordinate bit string inconsistency means that there is an error in the stored subordinate bit strings, either a structural inconsistency in the stored bit strings or misinformation stored in it; for example, a subordinate entry is present in the index, but it shouldn’t be there based on its data.This type of inconsistency can lead to incorrect search results and rejected modifications.

3.2.1. How to Detect It

The **dbamverify** command writes the result of subordinate bit string check into the **subcheck**dbamverify_PID***.txt* file in the **$DIRX_INST_PATH/tools/log** directory.At the end of the **subcheck** file, there is a line like “Subordinate bit string check finished base=*base n* errors”.If *n* is non-zero, there is a subordinate bit string inconsistency in the database.

3.2.2. Additional Data to be Collected for Analysis

The **dbamverify** command writes detailed logs about subordinate bit string checks into the **subcheck**dbamverify_PID***.txt* file in the **$DIRX_INST_PATH/tools/log** directory.Be sure to provide this file for analysis in addition to the files described in the section “Collecting Data for Analysis” in the chapter “General Database Error Recovery Process”.

3.2.3. Specific Recovery Steps

Subordinate indexes can be repaired using **dirxadm**. Some problems can be resolved by running the **db check** operation with the **-repair** option. To perform this procedure:

- Run the following **dirxadm** operation:

db check -bs SUBORDINATE [**-base** *distinguished_name*] **-repair**

If the inconsistency affects a specific branch of the tree or just a few branches, use the **-base** option to specify the base to avoid checking all the subordinate bit strings.

- The **db check** operation produces the same **subcheck** file as **dbamverify**, but in the **$DIRX_INST_PATH/server/log** directory.Examine this generated **subcheck** file.At the end of the file, there is a line like “Subordinate bit string check finished base=*base n* errors”.If *n* is zero in this line, the repair operation was successful, and you can proceed to the next step in the general recovery process.
- If *n* is non-zero, the inconsistency cannot be resolved with the **db check** operation.Use one of the general recovery methods described in the next chapter to restore the database.

3.3. Tree Inconsistency

Tree inconsistency means that one of the structural building blocks of the DIT is damaged.It can result in several kind of problems, like rejected operations, failed purge operation, etc.

3.3.1. How to Detect It

The **dbamverify** command writes tree verification problems into its **stderr** output.If this output contains one or more references to pseudo or tree blocks, there is a tree inconsistency in the database.

3.3.2. Additional Data to Collect for Analysis

A tree inconsistency is related to the database objects that define the structure of the tree, so a tree dump must be generated on 1) the last error-free backup and 2) on the backup or the database in which the inconsistency was detected.Provide these tree dumps for analysis in addition to the files described in the section “Collecting Data for Analysis” in the chapter “General Database Recovery Process”.

Use the the following **dbamdump** command to generate the tree dump:

dbamdump -Tfl

The **dbamdump** command writes the tree dump to **stdout**, so we recommend redirecting **stdout** to a file.

Note: the **dbamdump** command is an internal tool used primarily by DirX Directory support.It is not described in the DirX documentation.

3.3.3. Specific Recovery Steps

There are no specific recovery steps for a tree inconsistency.Use one of the recovery methods described in the chapter “General Methods for Database Recovery” to restore the database.Note that the backup archive file is the dump of the database blocks and can

thus be in a state that is just a few modifications away from the same inconsistency.So, for a tree inconsistency, we recommend rebuilding the database from an LDIF dump file.If the total update by media feature is enabled for the system, you can use the LDIF file with the "Restore by Initiating a Total Update" method.Otherwise, use the "Restore Using an LDIF Dump" method.

3.4. Real Object Inconsistency

Real object inconsistency means that the data stored in the real object blocks or in the follow blocks of the database is inconsistent.Any data stored in the database can be corrupt according to the rules checked during the verification, including the entries of the database, the database schema, configuration subentries, etc.This type of inconsistency has several sub types.Most of them are briefly described in the heading of the **entrycheck** log files.

3.4.1. How to Detect It

The **dbamverify** command writes the result of data consistency checks into the **entrycheck*dbamverify_PID***.txt* log file in the **$DIRX_INST_PATH/tools/log** directory.If this file contains a line like "*Timestamp* Error in REAL block/DSE-ID *DSE_ID*", the entry with the specified *DSE_ID* has an inconsistency.

3.4.2. Additional Data to be Collected for Analysis

The **dbamverify** command writes detailed logs about data consistency checks into the **entrycheck*dbamverifyPID***.txt* file in the **$DIRX_INST_PATH/tools/log** directory.Be sure to provide this file for analysis in addition to the files described in the section "Collecting Data for Analysis" in the chapter "General Database Recovery Procedures".

A dump of the erroneous entries should also be created and attached to the ticket. If there are a lot of errors of the same kind, the block dump of at least three entries and their follow blocks should be collected. Use the following **dbamdump** command to create a complete block dump of the DSE:

dbamdump -B -b *DSE_ID* **-t 128 -z** *real_object_block_size*

where *real_object_block_size* is the number that corresponds to the real object block size in use: **0**=1kB, **1**=4kB, **2**=16kB, **3**=64kB. The **dbamdump** command writes the DSE dump to **stdout**, so we recommend redirecting **stdout** to a file.

Note: the **dbamdump** command is an internal tool used primarily by DirX Directory support. It is not described in the DirX documentation.

3.4.3. Specific Recovery Steps

To recover from real object inconsistencies, follow this procedure:

- Examine the error message itself. If it contains a line starting with "Recommendation", follow the steps described in this field.

- If the FailedCheck field of the error message is “Normalisation-RDN”, use the **db check -rob -repair** operation to fix the inconsistency.
- After performing any recovery step(s) indicated in the error message, check the database consistency again with either the **dbamverify -D** command or the **dirxadm db check -rob** command. If the error was resolved by the specific recovery step, the database is consistent, and you can proceed to the next step in the general recovery process.
- If the problem cannot be resolved by the previous steps, use one of the general recovery methods described in the next chapter to recover the database.

4. General Methods for Database Recovery

There are three general methods for restoring a database when the type of database inconsistency cannot be determined or the recovery steps followed for a specific type of inconsistency do not solve it:

- Recovery from an LDIF dump file
- Recovery from a backup archive
- Recovery from a total update

The following table compares the pros and cons of these methods and the level of data loss they can cause.

Method	Restore using an LDIF dump	Restore using a backup archive	Restore by initiating a total update
Pros	Data defragmentation occurs automatically LDIF is interchangeable with other vendors	Fast	No backup needed Restored data are "up-to-date"
Cons	Loading process slower than with backup archive	Must be verified to ensure usability Not interchangeable with other vendors	More complex to execute Increased network traffic during total update
Data loss	Modifications since the LDIF dump was created	Modifications since the backup was created	No data loss in a synchronous agreement and maybe a few modifications in an asynchronous agreement

These procedures cannot account for all details of all customer installations. If your directory service configuration is more complex than a supplier DSA and one or more full shadow DSAs, we recommend contacting a DirX consultant to create a customized recovery plan that is specific to your setup.

The next sections describe how to perform each procedure.

4.1. Restore Using an LDIF Dump

To restore a database from an LDIF dump file:

- Stop the DirX service.

- Reset the database with the **dbamboot** command.
- Load the data from the LDIF file with the **dirxload** command.
- Start the DirX service.
- Reload any saved LDIF change files with the **dirxmodify** command.
- Synchronize the databases of the supplier and consumer(s) nodes:
- If the recovered database is held by the supplier node, initiate a total update by running the **dirxadm** commands **sob terminate** and **sob establish** on the supplier node for all SOB agreements.
- If the recovered database is held by a consumer node, initiate a total update by running the **dirxadm** commands **sob terminate** and **sob establish** on the supplier node for the SOB agreement(s) between the supplier and the recovered consumer.

4.2. Restore Using a Backup Archive

Note: Backup archives created with **dirxbackup** are simply a dump of the blocks stored in the database.The current state of the database cannot be determined, so it's possible that the inconsistency will recur after a short period of time in a database recovered using a backup archive.As a result, we recommend using an LDIF dump file if it's available.

To restore the database from a **dirxbackup** archive:

- Stop the DirX service.
- Restore the backup archive and any delta backup archives with **dirxbackup**.
- Start the DirX service.
- Synchronize the databases of the supplier and consumer(s) nodes:
- If the recovered database is held by the supplier node, initiate a total update by running the **dirxadm** commands **sob terminate** and **sob establish** on the supplier node for all SOB agreements.
- If the recovered database is held by a consumer node, initiate a total update by running the **dirxadm** commands **sob terminate** and **sob establish** on the supplier node for the SOB agreement(s) between the supplier and the recovered consumer.

4.3. Restore by Initiating a Total Update

If there is a consistent database in the system, an inconsistent database can be recovered by replicating the data from the consistent database using a total update.The procedures for recovering a consumer inconsistency and a supplier inconsistency are a bit different; each one is described below.

Note that this recovery method cannot be used when there is no consistent database in the system.In this case, use the "Restore Using an LDIF Dump" or the "Restore Using a Binary Backup" methods instead.

4.3.1. Recovering an Inconsistent Consumer

When a consumer database becomes inconsistent, perform the following steps to recover it:

- Terminate the agreement between the supplier and the inconsistent consumer by running the **dirxadm sob terminate** command.
- Create a backup of the inconsistent consumer.
- Stop the inconsistent consumer.
- Reset the database on the inconsistent consumer with the **dbamboot** tool.
- Start the inconsistent consumer.
- If the SOB agreements in the system are configured to use the “total update by media” feature:
- Establish the agreement using the **dirxadm sob establish** command.
- Run the **dirxadm sob show** command on the inconsistent consumer and check whether the SOB agreement configuration was properly replicated.
- Create a backup (preferably in LDIF format) on the supplier.
- Transfer the backup to the inconsistent consumer.
- Load the backup created in the previous step on the inconsistent consumer.
- Enable the agreement using the **dirxadm sob enable** command.
- If the SOB agreements in the system are configured to use the “total update by DISP” method:
- Establish the agreement again using the **dirxadm sob establish** command.
- If the Context Prefix (CP) of the SOB agreement is not root, you may want to transfer the indexing configuration from the supplier, too. To do this:
- Run the **dirxadm export_dbconfig** command on the supplier.
- Transfer the file **DirXDBconfig.out** to the inconsistent consumer. It’s generated into the **$DIRX_INST_PATH/tmp** directory and should be moved into the same directory on the inconsistent consumer.
- Run the **dirxadm import_dbconfig** command on the inconsistent consumer.

4.3.2. Recovering an Inconsistent Supplier

When a supplier database becomes inconsistent, perform the following steps to recover it:

- Switch to a consistent consumer using the **dirxadm sob switch** command.
- Terminate the SOB agreement between the old and the new supplier.
- Create a backup of the old supplier.
- Stop the old supplier.
- Reset the database on the old supplier with the **dbamboot** tool.
- If the SOB agreements in the system are configured to use the “total update by media”

feature:

- Establish the agreement using the **dirxadm sob establish** command.
- Run the **dirxadm sob show** command on the old supplier and check whether the SOB agreement configuration was replicated.
- Create a backup (preferably in LDIF format) on the new supplier.
- Transfer the backup to the old supplier.
- Load the backup created in the previous step on the old supplier.
- Enable the agreement using the **dirxadm sob enable** command.
- If the SOB agreements in the system are configured to use the “total update by DISP” method:
- Establish the agreement using the **dirxadm sob establish** command.
- If the Context Prefix (CP) of the SOB agreement is not root, you may want to transfer the indexing configuration from the new supplier, too. To do this:
- Run the **dirxadm export_dbconfig** command on the new supplier.
- Transfer the file **DirXDBconfig.out** to the old supplier. It’s generated into the **$DIRX_INST_PATH/tmp** directory and should be moved into the same directory on the old supplier.
- Run the **dirxadm import_dbconfig** command on the old supplier.

DirX Product Suite

The DirX product suite provides the basis for fully integrated identity and access management; it includes the following products, which can be ordered separately.

DirX Identity

DirX Identity provides a comprehensive, process-driven, customizable, cloud-enabled, scalable, and highly available identity management solution for businesses and organizations. It provides overarching, risk-based identity and access governance functionality seamlessly integrated with automated provisioning. Functionality includes lifecycle management for users and roles, cross-platform and rule-based real-time provisioning, web-based self-service functions for users, delegated administration, request workflows, access certification, password management, metadirectory as well as auditing and reporting functionality.

DirX Directory

DirX Directory provides a standards-compliant, high-performance, highly available, highly reliable, highly scalable, and secure LDAP and X.500 Directory Server and LDAP Proxy with very high linear scalability. DirX Directory can serve as an identity store for employees, customers, partners, subscribers, and other IoT entities. It can also serve as a provisioning, access management and metadirectory repository, to provide a single point of access to the information within disparate and heterogeneous directories available in an enterprise network or cloud environment for user management and provisioning.

DirX Access

DirX Access is a comprehensive, cloud-ready, scalable, and highly available access management solution providing policy- and risk-based authentication, authorization based on XACML and federation for Web applications and services. DirX Access delivers single sign-on, versatile authentication including FIDO, identity federation based on SAML, OAuth and OpenID Connect, just-in-time provisioning, entitlement management and policy enforcement for applications and services in the cloud or on-premises.

DirX Audit

DirX Audit provides auditors, security compliance officers and audit administrators with analytical insight and transparency for identity and access. Based on historical identity data and recorded events from the identity and access management processes, DirX Audit allows answering the “what, when, where, who and why” questions of user access and entitlements. DirX Audit features historical views and reports on identity data, a graphical dashboard with drill-down into individual events, an analysis view for filtering, evaluating, correlating, and reviewing of identity-related events and job management for report generation.

For more information: support.dirx.solutions/about

Legal remarks

On the account of certain regional limitations of sales rights and service availability, we cannot guarantee that all products included in this document are available through the Eviden sales organization worldwide. Availability and packaging may vary by country and is subject to change without prior notice. Some/All of the features and products described herein may not be available locally. The information in this document contains general technical descriptions of specifications and options as well as standard and optional features which do not always have to be present in individual cases. Eviden reserves the right to modify the design, packaging, specifications and options described herein without prior notice. Please contact your local Eviden sales representative for the most current information. Note: Any technical data contained in this document may vary within defined tolerances. Original images always lose a certain amount of detail when reproduced.