Adabas for Linux and Cloud

Adabas Security Features

Version 7.5.0

October 2026

ADABAS & NATURAL

This document applies to Adabas for Linux and Cloud Version 7.5.0 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 © 1987-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH 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 GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://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 GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://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 GmbH.

Document ID: ADAOS-SECFAC-750-20261004

Table of Contents

Adabas Security Features v
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 About this Documentation 5
3 Adabas Role-Based Security (ADARBA) 7
Adabas Role-Based Security Model 8
Security Definitions 10
Authentication 11
Authorization 13
Audit Trail 18
Configuration 22
Administration 24
Performance Considerations 26
Application Development 27
Getting Started 29
Infrastructure Security Library 41
4 Adabas Password Security (ADASCR) 55
Introduction 56
File Protection Levels 56
User Passwords 57
Security by Value Criteria 57
Adabas Security Processing 58
5 Adabas Encryption for Linux 61
Prerequisite 62
Encryption of Data-At-Rest 62
Key Management System 63
Administration 70
Database Access 75
6 Adabas OIDC Authentication 77
Overview 78
Authentication Flow 79
Configuration 82
Getting Started 83
Error Handling 85
7 Ciphering 87
8 Security Considerations 89
Using the Linux Group Concept 90
Securing Administration of Adabas Role-Based Security 91
Preventing Loss of Administration Privileges 92
Securing PUBLIC Access Privileges 93
Securing the Audit Trail Log File 93

9 SSL Trusted Relationship for Natural 95
Restrictions 96
Adabas Operation 96
Client Configuration 97
Adabas Configuration 97
10 GDPR Compliance 99
Adabas Role-Based Security 100
Adabas Audit Trail 101
Adabas Command Log 101
Adabas Log File 102

Adabas Security Features

- *About this Documentation*
- *Adabas Role-Based Security (ADARBA)*
- *Adabas Password Security (ADASCR)*
- *Adabas Encryption for Linux*
- *Adabas OIDC Authentication*
- *For maintenance only: Ciphering*
- *Security Considerations*
- *SSL Trusted Relationship for Natural*
- *GDPR Compliance*

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 *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	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

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

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 About this Documentation

This documentation describes the security facilities provided by Adabas and its subsystems.

Adabas Role-based Security (ADARBA)

Adabas Role-based Security (ADARBA) provides facilities for controlling access to database resources; for example:

- Restricting the execution of an Adabas utility;
- Restricting access to data via Adabas commands.

Adabas Password Security (ADASCR)

Adabas Password Security (ADASCR) provides facilities for controlling access and update to Adabas files; for example:

- Restricting user read and/or update requests on the basis of a whole file;
- Restricting user access to individual records within a file.

Adabas Encryption for Linux

Adabas Encryption for Linux prevents the unauthorized analysis of Adabas data-at-rest. It provides functionality to encrypt ASSO and DATA container files as well as PLOG and backup copies.

Adabas OIDC Authentication

Adabas OIDC Authentication provides Single Sign-On (SSO) capabilities for Adabas command-line utilities using the industry-standard OpenID Connect protocol.

It enables:

- Authentication with central Identity Providers (Keycloak, Google, Microsoft Entra ID).
- Device Authorization Flow for terminal/CLI environments.
- Token-based authentication with automatic caching and refresh.
- Integration with Adabas Role-Based Security for authorization.

Ciphering

Ciphering prevents the unauthorized analysis of Adabas container files.

Notes:

1. Ciphering is no longer recommended for safe implementations.
2. Ciphering does not fulfill modern encryption standards and security regulations.
3. Ciphering is maintained for historic and maintenance reasons only.

Adabas Encryption for Linux is highly recommended as a state-of-the-art replacement.

SSL Trusted Relationship for Natural

SSL Trusted Relationship for Natural enables Natural to make use of the Role Based Access Control without providing user credentials or using the Adabas user exit 21.

3 Adabas Role-Based Security (ADARBA)

- Adabas Role-Based Security Model 8
- Security Definitions 10
- Authentication 11
- Authorization 13
- Audit Trail 18
- Configuration 22
- Administration 24
- Performance Considerations 26
- Application Development 27
- Getting Started 29
- Infrastructure Security Library 41

Important: Before securing databases with Adabas Role-based Security, please familiarize yourself with the concepts and implementation of this feature. Once enabled, security cannot be disabled.

Adabas Role-based Security implements Role-Based Access Control (RBAC) and restricts access based on the roles assigned a user and the permissions that are assigned to the role.

With Adabas Role-based Security, you can control what end-users can do at both broad and granular levels. You can designate the roles assigned a user and align the roles and access privileges with your users' tasks. This means a user can be assigned a minimum of privileges, only the privileges essential to perform a specific task.

This feature includes the functionality:

Authentication
This provides a means of validating credentials against an authority.

Authorization for Direct Call Interface
This provides a means of restricting the usage of Adabas commands, which access a file, by assigning users a role which represents selective access privileges.

Authorization for Adabas Utilities
This provides a means of restricting the execution of Adabas utilities by assigning users a role which has the execute privilege.

Audit Trail
This provides a means of tracking access attempts and security violations.

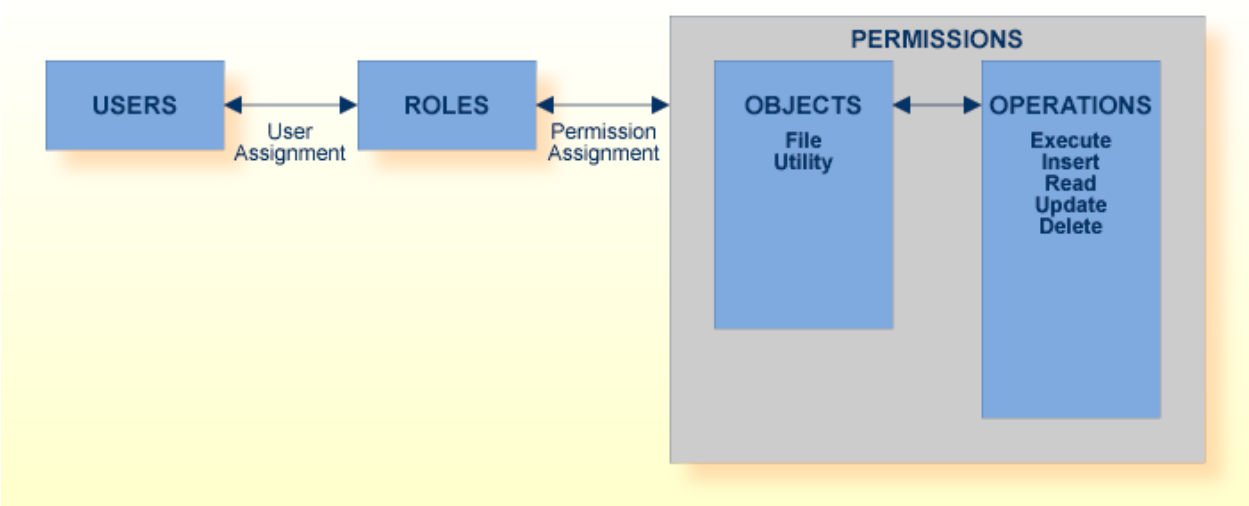
The utility ADARBA provides the functionality required to administer security definitions.

Adabas Role-Based Security Model

The RBAC reference model used by Adabas is based on the Core RBAC (RBAC0) of the ANSI Specification (ANSI INCITS 359-2004).

Core RBAC defines a minimum Role-Based Access Control system. This includes user-role assignment and permission-role assignment relations, considered fundamental in any RBAC system.

In the standard, the core RBAC model includes a set of sessions, where each session is a mapping between a user and an activated subset of roles that are assigned to the user. Sessions are currently not supported and are thus not part of the Adabas role-based security model.



The Adabas RBAC reference model enables the database or security administrator to implement the following:

- The assignment of USERS to ROLES,
- The assignment of ROLES to PERMISSIONS,
- where a PERMISSION consists of a set of OPERATIONS which can be performed on OBJECTS.

A *user* (USER) is defined as a human being. Although the concept of a user can be extended to include applications, the definition is limited to a person here for simplicity reasons.

A *role* (ROLE) is a job function within the context of an organization, with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role.

Permissions (PERMISSIONS) are an approval to perform an operation on one or more RBAC protected objects.

An *operation* (OPERATION) is an action or function, such as the execution of an Adabas utility, Adabas command, etc.

An *object* (OBJECT) is a resource that is subject to access control, such as an Adabas database, file, etc.

Security Definitions

This section describes the location and content of the security definitions.

The security definitions are stored in the RBAC system file of a security-enabled database. The contents of the system file constitute the security repository.

The security repository stores granted privileges, the denial of specific operations is not supported. It is accessible when the database is online or offline.

Initial Security Definitions

The initial security definitions implement unrestricted access to all operations and can be adapted and extended as required.

The unrestricted access to operations is achieved via the **PUBLIC** user and **PUBLIC** role security definitions.

PUBLIC User

The **PUBLIC** user is a pre-defined security definition. It is a convenience definition that is used when the provided credentials cannot be validated. This user has the permissions that can be derived from the assigned roles.

The **PUBLIC** user may be modified as follows:

- The **PUBLIC** user may be dropped;
- The roles assigned to the **PUBLIC** user may be modified or removed as required. Additional roles may be assigned.

By default, the **PUBLIC** user is assigned the **PUBLIC** role.

PUBLIC Role

The **PUBLIC** role is a predefined security definition. All users have the permissions which are assigned to the **PUBLIC** role.

The **PUBLIC** role may be modified as follows:

- The **PUBLIC** role is a system definition and as such *should not be dropped*.
- The permissions assigned to the **PUBLIC** role may be modified or removed as required. Additional permissions may be assigned.

By default, the **PUBLIC** role has “legacy privileges”; for example, the privilege to execute to all Adabas utilities.

Authentication

Authentication provides a means of identifying a user, by having the user provide a valid user name and a valid password before access is granted.

The credentials are checked against an authentication authority, for example against an external authentication system such as LDAP, Active Directory, operating system, or an internal user repository. If the credentials match, the user is provided access to the database. If the credentials are at variance, authentication fails and access to the database is denied.

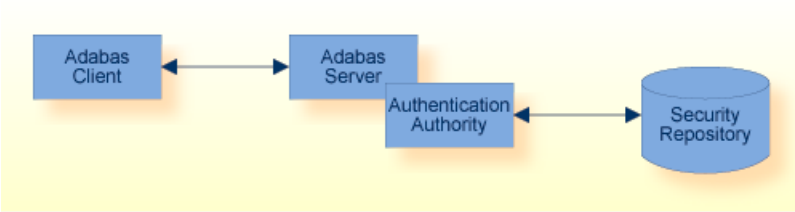
The results of the authentication check are protocolled in the audit trail log.

Refer to the *Configuration of the Infrastructure Security Libraries* section for more information about the possible authentication authorities.

- Architecture
- Credentials

- Default Authentication State

Architecture



Adabas uses the Infrastructure Security Libraries to access the authentication authority for validation of credentials. These libraries enable the authentication of credentials against LDAP, Active Directory, and internal repository or the operating system.

The installation of the Infrastructure Security Libraries is mandatory for usage of this feature.

Refer to the section on *Configuration* for more information on how to configure the authentication feature.

Credentials

The credentials used by Adabas are user identification and password. The user identification consists of the user account name. In the case of Windows, the user identification also contains the domain of the user.

Adabas Direct Call Interface

Credentials can be provided by one of the following means:

- The application provides the credentials;
- The credentials are provided by the nucleus user exit 21.

The application should always provide the credentials to ensure that the application user can be identified. How applications credentials are supplied is described in the *Developing Applications* section of *Application Development*.

The nucleus user exit 21 can be used to transition existing applications to a secure database. Further information on this topic is provided in the *Modifying Legacy Applications* section.

The authentication authority that is to be used is defined in a database configuration file.

Adabas Utilities

Currently, the supported credentials are the local system credentials, which have been authenticated by the operating system.

- Linux: User ID
- Windows: Domain and User ID

Important: The Adabas utilities currently do not perform authentication checks. The local system credentials are used to perform the authorization check. This is subject to change in a future release.

Default Authentication State

By default, authentication is not enabled.

The configuration and usage of the authentication feature are described in *Configuration > Authentication* below.

Authorization

Authorization provides the means of restricting operations which are to be performed on a resource or, in other words, the execution of

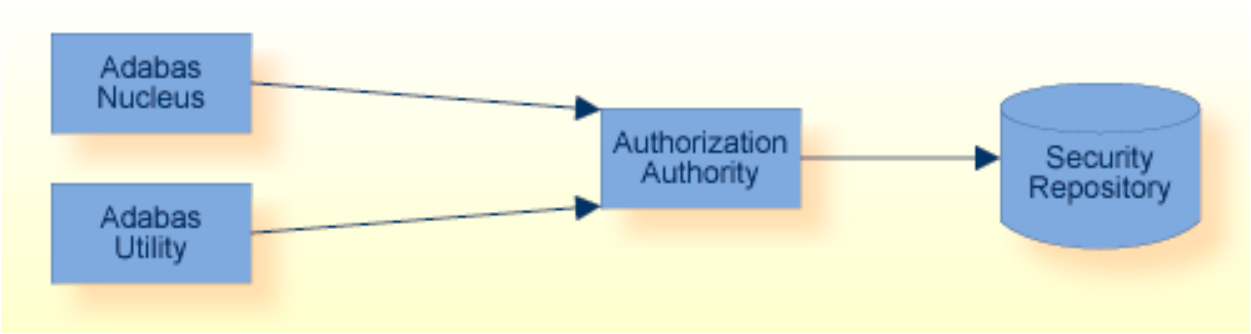
- An Adabas command on a file, or
- An Adabas utility on a database.

An access request can be issued by either the Adabas nucleus or an Adabas utility.

- The Adabas nucleus issues an access request prior to processing an Adabas command.
- An Adabas utility issues an access request prior to executing the requested utility.

The access request is validated by the authorization authority. It is validated against the security repository stored in the database. The credentials used to process the access request were validated previously by the authentication authority.

The results of the authorization check are protocolled in the audit trail log.



- Adabas Direct Call Interface
- Adabas Utilities
- Protection on Utility Control Parameter Level
- Default Authorization State

Adabas Direct Call Interface

Authorization for Adabas Direct Call Interface allows you to secure specific database files against unauthorized access via Adabas commands.

This feature uses role-based security definitions to restrict and monitor the usage of the Adabas commands, which can be performed on a specific file in the database.

If a matching access privilege is found, the user is authorized to execute the requested operation, otherwise the request is rejected.

Attempts to execute a controlled operation, here an Adabas command, are documented in an audit trail. The configuration of the audit trail log is described in *Configuration > Audit Trail* below.

The following table is a list of controlled access operations on files:

Operation	Description	Commands
READ	Read data stored in file	L1, L2, L3, L4, L5, L6, L9, S1, S2, S4, S8, S9
INSERT	Insert data into the file	N1, N2
UPDATE	Update data that is stored in a file	A1
DELETE	Delete data that is stored in a file	E1
ANY	This is a convenience notation. It includes the READ, INSERT, UPDATE and DELETE operations.	All of the commands above

Note: In the current version, the execution of Adabas commands, which are not listed above, is not restricted. This will change in a future release.

The operation names above are the external representation of the security definitions.

When using the LIST function of the ADARBA utility, the internal representation of the security definition is shown.

Operation	Description	LIST Layout
READ	Read data stored in a file	ada.dml.read
INSERT	Insert data into a file	ada.dml.insert
UPDATE	Update data that is stored in a file	ada.dml.update
DELETE	Delete data that is stored in a file	ada.dml.delete

Adabas Utilities

Authorization for Adabas Utilities allows you to secure a specific database against unauthorized access via Adabas utilities.

This feature uses role-based security definitions to restrict and monitor the usage of the Adabas utilities, which can be performed on a specific database.

If a matching access privilege is found, the user is authorized to execute the requested operation, otherwise the request is rejected. The authorization of a request implies the privilege to execute the Adabas utility.

All attempts to execute an Adabas utility are documented in an audit trail.

The following table is a list of possible access operations on Adabas utilities; all values are case-sensitive:

Operation Name	Utility	Description
ada.uti.bck	ADABCK	Backup and restore database or files
ada.uti.clp	ADACLP	Command log report
ada.uti.cmp	ADACMP	Compression of data
ada.uti.cvt	ADACVT	Convert a database from a previous version
ada.uti.dbm	ADADBM	Database modification
ada.uti.dcu	ADADCU	Decompression of data
ada.uti.dev	ADADEV	Disk space management
ada.uti.ela	ADAELA	Event analytics administration
ada.uti.elp	ADAELP	Event log report
ada.uti.err	ADAERR	Error file report
ada.uti.fdu	ADAFDU	File definition
ada.uti.fin	ADAFIN	File information report
ada.uti.inv	ADAINV	Creating, removing and verifying inverted lists
ada.uti.mon	ADAMON	Monitoring the database nucleus

Operation Name	Utility	Description
ada.uti.mup	ADAMUP	Mass add and delete
ada.uti.nuc	ADANUC	Starting the database, defining nucleus parameters
ada.uti.opr	ADAOPR	Operator utility
ada.uti.ord	ADAORD	Reorder the database or export/import files
ada.uti.plp	ADAPLP	Protection log printout
ada.uti.pri	ADAPRI	Print Adabas blocks
ada.uti.rba	ADARBA	RBAC administration
ada.uti.rec	ADAREC	Recovery of database or files
ada.uti.rep	ADAREP	Database report
ada.uti.scr	ADASCR	Manage and enable security functionality
ada.uti.uld	ADAULD	File unloading
ada.uti.vfy	ADAVFY	Database consistency check

Note: The operation names listed above are subject to change.

Protection on Utility Control Parameter Level

This part of authorization allows you to secure specific functions in the Adabas utilities ADADBM and ADAOPR. This is a fine- grained extension of the utility authorization. The following table shows the specific operations on Adabas utilities; all values are case-sensitive:

Operation Name	Utility Operation
ada.uti.dbm.add_container	ADADBM add_container
ada.uti.dbm.add_fields	ADADBM add_fields
ada.uti.dbm.allocate	ADADBM allocate
ada.uti.dbm.bt	ADADBM bt
ada.uti.dbm.change	ADADBM change
ada.uti.dbm.change_fields	ADADBM change_fields
ada.uti.dbm.deallocate	ADADBM deallocate
ada.uti.dbm.define_refint	ADADBM define_refint
ada.uti.dbm.del_cp	ADADBM del_cp
ada.uti.dbm.delete	ADADBM delete
ada.uti.dbm.delete_database	ADADBM delete_database
ada.uti.dbm.drop_fields	ADADBM drop_fields
ada.uti.dbm.drop_lobfile	ADADBM drop_lobfile
ada.uti.dbm.drop_refint	ADADBM drop_refint
ada.uti.dbm.extend_container	ADADBM extend_container

Operation Name	Utility Operation
ada.uti.dbm.newwork	ADADBM newwork
ada.uti.dbm.pgm_refresh	ADADBM pgm_refresh
ada.uti.dbm.recordspanning	ADADBM recordspanning
ada.uti.dbm.reduce_container	ADADBM reduce_container
ada.uti.dbm.refresh	ADADBM refresh
ada.uti.dbm.remove_container	ADADBM remove_container
ada.uti.dbm.remove_replication	ADADBM remove_replication
ada.uti.dbm.rename	ADADBM rename
ada.uti.dbm.renumber	ADADBM renumber
ada.uti.dbm.replication_files	ADADBM replication_files
ada.uti.dbm.reset	ADADBM reset
ada.uti.dbm.reset_replication_target	ADADBM reset_replication_target
ada.uti.dbm.reuse	ADADBM reuse
ada.uti.dbm.security	ADADBM security
ada.uti.dbm.syfmax	ADADBM syfmax
ada.uti.opr.abort	ADAOPR abort
ada.uti.opr.add_replication	ADAOPR add_replication
ada.uti.opr.bfio_parallel_limit	ADAOPR bfio_parallel_limit
ada.uti.opr.cancel	ADAOPR cancel
ada.uti.opr.change_replication	ADAOPR change_replication
ada.uti.opr.cluster_log_level	ADAOPR cluster_log_level
ada.uti.opr.delete_replication	ADAOPR delete_replication
ada.uti.opr.delui	ADAOPR delui
ada.uti.opr.ext_backup	ADAOPR ext_backup
ada.uti.opr.feof	ADAOPR feof
ada.uti.opr.free_clq	ADAOPR free_clq
ada.uti.opr.lock	ADAOPR lock
ada.uti.opr.logging	ADAOPR logging
ada.uti.opr.nishnq	ADAOPR nishnq
ada.uti.opr.options	ADAOPR options
ada.uti.opr.response_check	ADAOPR response_check
ada.uti.opr.shutdown	ADAOPR shutdown
ada.uti.opr.stop	ADAOPR stop
ada.uti.opr.stopi	ADAOPR stopi
ada.uti.opr.tnaa	ADAOPR tnaa
ada.uti.opr.tnae	ADAOPR tnae

Operation Name	Utility Operation
ada.uti.opr.tnax	ADAOPR tnax
ada.uti.opr.tt	ADAOPR tt
ada.uti.opr.unlock	ADAOPR unlock
ada.uti.opr.wcharset	ADAOPR wcharset
ada.uti.opr.write_limit	ADAOPR write_limit

Default Authorization State

By default, authorization is not enabled.

The configuration and usage of the authorization feature are described in *Configuration > Authorization* below.

Audit Trail

An audit trail entry is written to the audit trail log file for each authentication and authorization attempt. The audit trail logs both successful and failed attempts to access the database.

The content of each audit trail entry provides the following information on the attempted access:

- Timestamp of access attempt;
- Information about the application or utility, which attempted access;
- User and session identification of access attempt;
- Information about the attempted operation;
- Information about the object of the operation.

The following requirements apply to the audit trail log file:

- The audit entries are appended to the log file;
- The size of the log file should be monitored and the log file should be backed-up or moved as required.

- Audit Trail Authentication
- Authorization for Adabas Direct Call Interface
- Authorization for Adabas Utilities

- Default Audit Trail State

Audit Trail Authentication

The location and layout of the audit trail log file is described in the *Authorization for Adabas Direct Call Interface* section.

Authorization for Adabas Direct Call Interface

Location

The audit trail log is database-specific and is located in the database directory. The file name is *NUCADT.log*.

Layout of Audit Trail Entry

The layout of the audit trail log file is CSV format. The values of the audit trail entry are separated by comma “,”.

The following table contains a description of the audit trail entries for authentication and authorization, the values provided in each audit trail entry are dependent on the type of entry.

Column	Description	Value
Timestamp	Timestamp	
Security Mode	Security Mode	A[ctive] W[arn]
Result	Processing result - access is allowed or not	YES NO
DBID	Database ID	
DBName	Database name	
Session ID	Client information, as specified in the Adabas User Queue	
ET User	ET User	
Security User	User credential provided for authentication	
RBAC User	User credential used to determine access privileges	
RBAC Role	Role with which access privilege was given	
Operation	Adabas command category	INSERT READ UPDATE DELETE

Column	Description	Value
Command	Adabas command code	
File Number	Adabas file number	
File Name	Adabas file name	
ISN	ISN accessed in Adabas file	
Authority	Authentication or Authorization authority	SSX RBAC
Response Code	Adabas response code	
Subcode	Adabas subcode	
SSX Response	SSX response	
SSX Message	SSX message	

An ISN can only be displayed for commands that require an ISN input. These commands are `A1`, `E1`, `HI`, `L1`, `L2`, `L3`, `L4`, `L5`, `L6`, `N2` and `RI` and in case of multi fetch `BT` and `ET`, as well. For all other commands and in case of a failed authentication or authorization, the value 0 will be displayed.

Authorization for Adabas Utilities

The location of the audit trail file is defined in the RBAC security definition file, which is stored in the RBAC system file of the database.

Refer to the *Location of Configuration and Logging Files* section for more information about the location, configuration and content of the audit trail for Authorization for Adabas Utilities.

- Layout of Audit Trail Entry
- Logging of Administrative Operations
- Example

Layout of Audit Trail Entry

Column	Description	Prefix	Values
Timestamp	Timestamp		yyyy-mm-ddThh:mm:ssZ
Message Type	Message indicator	%AUTHORIZATION-	I[nformation] E[rror]
Hostname	Host name of machine	HOST=	<hostname>
Opsys Version	Name and version of operating system	OS=	<operating_system>
Credentials	Name of user account	USER=	[<domain>/]<user>
Operation	Name of attempted operation	OPERATION=	<operation>
DBID	Database ID	DBID=	<number>
Result	Result of authorization	RESULT=	OK

Column	Description	Prefix	Values
			FAIL

Logging of Administrative Operations

The administrative changes made in ADARBA are also logged in the audit trail.

Operation	Syntax
initialize	OPERATION=ada.uti.rba.initialize,<file>
create_user	OPERATION=ada.uti.rba,create_user,<user>
drop_user	OPERATION=ada.uti.rba,drop_user,<user>
create_role	OPERATION=ada.uti.rba,create_role,<role>
drop_role	OPERATION=ada.uti.rba,drop_role,<role>
create_operation	OPERATION=ada.uti.rba,create_operation,<operation>
drop_operation	OPERATION=ada.uti.rba,drop_operation,<operation>
create_object	OPERATION=ada.uti.rba,create_object,<object>
drop_object	OPERATION=ada.uti.rba,drop_object,<object>
grant_role	OPERATION=ada.uti.rba,grant_role,<role>,<user>
revoke_role	OPERATION=ada.uti.rba,revoke_role,<role>,<user>
grant_permission	OPERATION=ada.uti.rba,grant_permission,<operation>,<object>,<role>

Example

The following example shows an audit trail entry for a failing ADARBA create_user=newuser command.

```
2025-01-01T00:00:00Z %AUTHORIZATION-E HOST=redada6c OS="Linux x86_64" USER=user ↩
OPERATION=ada.uti.rba.create_user.newuser DBID=189 RESULT=FAIL
```

Default Audit Trail State

The audit trail is automatically enabled when either authentication or authorization is enabled.

Refer to the *Location of Configuration and Logging Files* section for more information on the configuration of the contents of the audit trail log file.

The configuration and usage of the authorization feature are described below.

Configuration

This section describes the configuration of the Adabas Role-based Security feature.

- Initial Security Configuration
- Authentication Configuration
- Authorization Configuration
- Audit Trail Configuration

Initial Security Configuration

An initial security configuration is created during installation.

- Authentication is not enabled;
- Authorization is not enabled;
- Audit Trail is not enabled.

Authentication Configuration

Important: Database security cannot be disabled once it has been activated.

Enable Authentication

Use the SECURITY function of the ADADBM utility to enable the authentication. The security mode can either be set to WARN or ACTIVE.

The security mode ACTIVE implies that only authenticated users are allowed access to the database. Security violations, such as authentication errors, are protocolled as “Error” in the audit trail.

The security mode WARN is intended for transitioning applications to use a secure database. It implies that all users are allowed access to the database. Security violations, like authentication or authorization errors, are protocolled as “Warning” in the audit trail. In case of a security violation, access to the operation is not rejected.

Once enabled, the security mode can only be changed from mode WARN to ACTIVE.

When you introduce this feature, we strongly recommend that you initially start with security mode WARN.

Refer to the *Nucleus user exit 21* section for more information on how to provide authentication credentials for legacy applications.

Configure Authentication Authority

The following types of authentication authority are supported:

- Authentication Type OS (Operating System)
- Authentication Type TEXT (Internal User Repository)
- Authentication Type LDAP
- Authentication Type ADSI

These settings are described in the *Configuration of the Infrastructure Security Library* section, where you can find example templates for the different authentication types. These templates are not complete as some of the settings are customer-specific and must be modified where necessary.

These configuration settings are stored in the [SSX_CONFIGURATION] section of the *DBnnn.INI* file. Use the administration command ADAINI to set and modify these settings.

Authorization Configuration

Enable Authorization

Use the INITIALIZE function with the RBAC_FILE parameter of the ADARBA utility to load the RBAC system file.

Define Security Definitions

Use the ADARBA utility to manage customer-specific security definitions.

Please refer to the examples in section *Getting Started* for further information.

Note: The RBAC System File is closely related to the database. It has to be backed up and restored together with the complete database and cannot be restored individually.

Audit Trail Configuration

Authentication and Authorization for Direct Call Interface

The audit trail log is written to the file *NUCADT.log*, which is located in the database directory.

The configuration settings are stored in the [AUDIT_TRAIL] section of the *DBnnn.INI* file. Use the administration command ADAINI to set and modify these settings.

The following AUDIT_TRAIL settings can be used to configure the audit trail:

ACTION
Enable or disable Audit Trail logging for authentication and authorization.

FILTER
Log ALL audit trail entries or just the entries for REJECTED authentications and authorizations.

MAXFILESIZE
Create a new audit file with the next sequential number if the current NUCADT log file exceeds the value given to MAXFILESIZE. The default value of MAXFILESIZE is 100 MB.

TIMEZONE
Choose the timestamp layout in Universal Coordinated Time (UTC) or local time.

The possible options are UTC and LOCAL. If you do not specify TIMEZONE or you provide an invalid option, the system uses the default setting `TIMEZONE=UTC`.

Refer to the *Configuration Files* section in *Adabas Extended Operation* for more information on AUDIT_TRAIL settings.

Authorization for Utilities

The configuration settings for the audit trail log file are stored in the database.

The following options are available:

- The item FORMAT defines the layout of an audit log entry.
- The item SEPARATOR defines the character to be used to separate values in CSV format.
- The item LOG_FILE defines the location and file name of the audit log.

For more information, refer to the *Configuring Adabas Role-based Security* section.

Administration

The utilities required to configure and administer the authorization feature are described below.

- ADADBM
- ADAINI
- ADAREP

- ADARBA

ADADBM

Use the SECURITY function to enable Adabas RBAC Security;

Note: These functions require that the database is offline.

ADAINI

- Use the administration command ADAINI to set and modify the configuration of security features;
- Configure the authentication authority;
- Configure the audit trail;
- Display the configuration settings.

ADAREP

- Use the SUMMARY function to display the database system files and whether security is enabled;
- The SECURITY setting is displayed if security is enabled;
- The RBAC system file information is displayed if the RBAC system file is defined.

ADARBA

- Use the CREATE, DROP, GRANT, REVOKE functions to create, modify and delete security definitions;
- Use the LIST functions to display the security definitions.
- Use the RBAC_FILE function to create the RBAC system file.

Note: These functions require that the database is online.

Performance Considerations

Authentication

This feature could have a detrimental effect on the overall system performance for the following reasons:

- Due to the effort required to authenticate the credentials;
- Due to the large number of entries being written to the audit trail log file.

Authorization for Direct Call Interface

This feature could have a detrimental effect on the overall system performance for the following reasons:

- Due to the effort required to authenticate the credentials;
- Due to the effort required to determine the user's access privileges;
- Due to the large number of entries being written to the audit trail log file.

Authorization for Adabas Utilities

This feature only has minimal impact on the overall system performance.

Audit Trail

Due to the high number of synchronized file I/Os, this feature could have a detrimental influence on the overall system performance.

The impact can be minimized by using the AUDIT_TRAIL parameter setting FILTER=REJECTED for authentication and authorization for Direct Call Interface requests.

The following configuration options have a detrimental influence on performance, and should be used with care:

Feature	Option	Explanation
Audit Trail	Filter	A large number of user sessions or Adabas commands will result in a large number of security entries being written to the audit trail log file. The size of the audit trail log file increases rapidly with the numbers of user sessions or Adabas commands. Default value: FILTER = ALL Recommended value: FILTER = REJECTED

Feature	Option	Explanation
SSX Logging/Diagnostics	nativeloglevel	Multiple user sessions attempting to write diagnostic information concurrently to the security infrastructure log file. Default value: None Recommended value: 0 or None

Application Development

This section covers the following topics:

- Developing Applications
- Modifying Legacy Applications
- Error Handling
- Messages and Codes

Developing Applications

The application is responsible for setting the user credentials prior to opening a database session.

The following Adabas client functions are provided to manage client sessions and set credentials:

Step	Function	Description
1	lnk_set_adabas_id()	Set the session identification.
2	lnk_set_uid_pw()	Set the authentication credentials for a specific database.

Details about the above Adabas client functions can be found in the *Calling Adabas with Authentication* section in *Command Reference* > *Calling Adabas*.

It is recommended to use OPTIONS=OPEN_REQUIRED.

Modifying Legacy Applications

Without modification, legacy applications will receive the nucleus response 200 “security violation”, when accessing secured databases.

The Adabas nucleus user exit 21 can be used to set authentication credentials via the Adabas server API functions. The routine is called when the processing of a session begins.

This routine should be used as briefly as possible. It is intended for use during the transition period, until all applications use and support the Adabas security authentication feature.

For more information, see the nucleus user exit 21 in the *User Exits and Hyperexits* section.

Error Handling

Security Mode Active

If a security violation occurs during authentication processing, Adabas issues a response code, backs out the user's current transaction, and closes the user session.

- Response code 9, subcode SE
- Response code 200, subcode 31

If a security violation occurs during authorization processing, Adabas issues a response code:

Response code 200, subcode 175

Security Mode WARN

The error handling for security mode WARN is intended for transitioning to a secure database. The error handling is described below. It differs to that of mode ACTIVE.

- The current transaction is not backed out;
- The response code returned indicates “success” (response 0);
- A “Warning” entry is entered in the audit trail log file.

Messages and Codes

The following Adabas nucleus response codes indicate that a security issue has occurred.

Authentication

Response Code	Explanation	Action
9/SE	The credentials have been modified since the start of the session. Hence the credentials are invalid, and the user session has been terminated.	Ensure that the credentials are not set or modified during a user session. To proceed processing, set the credentials and open a new user session.
200/31	The credentials could not be validated by the authentication authority.	Ensure that the credentials have been provided and that they are valid.

Authorization

Response Code	Explanation	Action
200/175	The user identified by the provided credentials does not have the privileges to perform the requested operation.	Ensure that the credentials have the required privileges.

Getting Started

This section covers the following topics:

- Prerequisites
- Getting Started with Configuration
- Getting Started with Authentication
- Getting Started with Authorization for Direct Call Interface
- Getting Started with Authorization for Adabas Utilities

Prerequisites

In order to execute the steps described in the following section, you will require a database.

The following prerequisites are mandatory:

- The Infrastructure Security Libraries are installed.
- The security feature is enabled.

Authentication has the following prerequisites:

- The security mode is enabled.
- The user credentials that are to be validated by the authentication authority.

Authorization has the following prerequisites:

- The RBAC system file is loaded.
- Security definitions will be required.

You, or an administrator, will require the necessary access privileges:

- To modify the security configuration files.
- To create and configure a database.
- To start and stop a database.

Important: When starting with the Adabas RBAC security feature, it is recommended to create a database and to initially use the SECURITY mode WARN. Once the Adabas role-based security feature has been enabled, it cannot be disabled.

Getting Started with Configuration

This section describes how to configure Adabas role-based security (ADARBA).

Authentication

The following steps describe how to enable authentication.

1. Ensure the database is offline.
2. Configure the authentication authority for the database.
3. Enable authentication by setting the SECURITY mode.
4. Start the database.

Note: Authentication is not required and does not have to be enabled for Authorization for Adabas Utilities.

The authentication examples below use the internal user repository as the authentication authority.

Example: Create Credential in an Internal User Repository

This example shows how to create a user, password credential in an internal user repository.

```
> ssxtxtpasswd -f SAGInternalUserRepository.txt -c -p mypsw myuid

Hash: ↩
bOEOAPEEEJBKv+4zOELiYcFqY7qFhlLZz1ha7Ztf7j/drJHGy2MLOLXEu/kX7TD52Aj7XfwiZ+vpIl9DqRbVkA==
User entry for "myuid" successfully added
```

The contents of the file *SAGInternalUserRepository.txt* are shown below:

```
*
*
* SAG Internal User Repository
*
version:3.0
*
user:myuid:$6a$bOEOAPEEEJBKv+4zOELiYcFqY7qFhlLZz1ha7Ztf7j/drJHGy2MLOLXEu/kX7TD52Aj7XfwiZ+vpIl9DqRbVk
```

Refer to the *Configuring the Infrastructure Security Libraries* section for further information about how to create and administer an internal user repository.

Example: Configure the Authentication Authority

This example shows the configuration of authentication with a text file. This file contains authenticated credentials and passwords. Refer to the section describing the ***Configuration*** for more information on the creation and administration of an authentication text file.

```
DBnnn.INI

> adaini dbid=nnn add topic=DB_PARAMETER topic=SSX_CONFIGURATION item=authType=TEXT
> adaini dbid=nnn add topic=DB_PARAMETER topic=SSX_CONFIGURATION ↩
item=internalRepository=SAGInternalUserRepository.txt
```

Refer to the *Validating the Configuration* section for an example on how to display the SSX_CONFIGURATION settings.

Example: Enable Security Mode

This example shows how to enable the security mode, which enables authentication. When starting with the Adabas RBAC security feature, it is recommended to initially use the SECURITY mode WARN.

Use the SECURITY function of the ADADBM utility to activate the authentication feature.

```
> adadbm dbid=nnn security=warn
%ADADBM-I-STARTED,      15-AUG-2018 11:13:39, Version 6.7.0.0
%ADADBM-I-DBOFF, database 224 accessed offline
%ADADBM-I-FUNC, function SECURITY executed
```

Authorization

The following steps describe how to enable authorization.

1. Start the database and ensure the database is online.
2. Load the RBAC system file. See the following example.

Example: Load the RBAC System File

Ensure the database is online. Use the INITIALIZE function of the ADARBA utility to load the RBAC system file.

Note: The RBAC_FILE function of the ADADBM utility, which was previously used to load the RBAC system file, has been removed.

```
> adarba dbid=190 initialize rbac_file=250
%ADARBA-I-STARTED,      05-JUN-2023 10:35:21, Version 7.2.0.0 (Linux 64Bit)
%ADARBA-I-INP, dbid=190
%ADARBA-I-DBON, database 190 accessed online
%ADARBA-I-INP, initialize,rbac_file=250
%ADARBA-I-LOADED, file 250, "RBAC-FILE" loaded
%ADARBA-I-STATCMD, 0 error(s) occurred
Restore files dumped on 27-JUL-2018 09:00:19
%ADARBA-I-TERMINATED,   05-JUN-2023 10:35:22, elapsed time: 00:00:01
```

Refer to the *Validating the Configuration* section for an example of how to display the RBAC system file information.

Audit Trail

Authentication and Authorization Adabas Direct Call Interface

The audit trail log file for authentication and authorization for Adabas direct call interface is located in the database directory. The file name is *NUCADT.LOG*.

The following steps describe how to enable the audit trail for authentication and authorization for Adabas direct call interface.

1. Configure the audit trail for authentication and authorization for Adabas direct call interface.
2. Restart the database, so that the modifications are active.

The following example shows the configuration of the audit trail for authentication and authorization for Adabas direct call interface.

```
DBnnn.INI

> adaini dbid=nnn add topic=DB_PARAMETER topic=AUDIT_TRAIL item=FILTER=ALL
> adaini dbid=nnn add topic=DB_PARAMETER topic=AUDIT_TRAIL item=ACTION=YES
```

Refer to the *Validating the Configuration* section for an example of how to display the audit trail settings.

Authorization Adabas Utilities

The modification of the audit trail configuration file settings is described in the *Configuration for Authorization for Adabas Utilities* section.

Example: Display Audit Definitions

The example shows the initial values for audit of utility authorization. Start the database and display the audit definitions.

```
> adarba db=nnn audit,display

%ADARBA-I-STARTED,      26-MAR-2024 14:58:52, Version 7.3.0.0 (Linux 64Bit)
%ADARBA-I-INP, dbid=nnn
%ADARBA-I-DBON, database nnn accessed online
%ADARBA-I-INP, audit,display

AUDIT=YES
FORMAT=TEXT
SEPARATOR=;
LOGFILE=${ADADATADIR}/dbnnn/adaaudit.log

%ADARBA-I-STATCMD, 0 error(s) occurred
%ADARBA-I-TERMINATED,   26-MAR-2024 14:58:52, elapsed time: 00:00:00
```

Validating the Configuration

The following examples show how you can validate the configuration of authentication and authorization.

Example: Database Report - Security Configuration

Use the SUMMARY function of the ADAREP utility to display the SECURITY mode setting and the RBAC system file information.

When enabled, the SECURITY mode will be displayed with a value of either WARN or ACTIVE. When not enabled, the SECURITY mode is not displayed.

When loaded, the file number of RBAC system file will be shown.

```
> adarep dbid=nnn summary

%ADAREP-I-STARTED,      15-AUG-2018 15:37:23, Version 6.7.0.0

Summary of Database nnn          15-AUG-2018 15:37:23


DATABASE NAME                    EXAMPLE-DB
DATABASE ID                        224
MAXIMUM FILE NUMBER LOADED         255
SYSTEM FILES                       255 (CHK),   251 (SEC),   254 (USR)
                                   250 (RBAC)
ACTUAL FILES LOADED                  8
CURRENT PLOG NUMBER                  4
CURRENT CLOG NUMBER                  0
SECURITY                           WARN


Container  Device      Extents in Blocks    Number of      Block    Total Size
  File     Type         from          to     Blocks        Size     (Megabytes)
```

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

 ASSO1     file            1       2,000        2,000      2,048           3.91
 ASSO2     file        2,001      10,000        8,000      2,048          15.63

 DATA1     file            1       2,000        2,000      4,096           7.81
 DATA2     file        2,001      10,000        8,000      4,096          31.25

 WORK1     file            1       2,000        2,000      4,096           7.81

-------------------------------------------------------------------------------
                                                                          66.41
                                                                          =====

%ADAREP-I-IOCNT, 17 IOs on dataset ASSO
%ADAREP-I-TERMINATED,   15-AUG-2018 15:37:23, elapsed time: 00:00:00
```

Example: Display Authentication Configuration

Use the SHOW function of the administration command ADAINI to display the authentication configuration.

Show the configuration of the authentication authority.

```
> adaini dbid=nnn show topic=DB_PARAMETER topic=SSX_CONFIGURATION

authType=TEXT
internalRepository=path_and_name_ssxuser_file
```

Display Audit Trail Configuration

Use the SHOW function of the administration command ADAINI to display the audit trail configuration.

Show the audit trail configuration for authentication and authorization for Adabas direct call interface.

```
> adaini dbid=nnn show topic=DB_PARAMETER topic=AUDIT_TRAIL

ACTION=YES
FILTER=ALL
```

Example: Security Infrastructure Initialized

The security infrastructure is initialized during the start of the ADANUC utility. This infrastructure is used for authentication and indicates that the authentication feature is enabled and that the SSX_CONFIGURATION parameters have been processed.

```
> adanuc dbid=nnn

%ADANUC-I-STARTED,      15-AUG-2018 16:21:59, Version 6.7.0.0

%ADANUC-I-SSXINI, SSX Security Infrastructure initialized
%ADANUC-I-CREATED, dataset NUCTMP1, file D:\ ADADATADIR\dbnnn\NUCTMP1.nnn created
%ADANUC-I-PLOGCRE, plog NUCPLG, file 'D:\ADADATADIR\dbnnn\NUCPLG.0001' created
%ADANUC-I-DBSTART, Database nnn, session 5 started, 15-AUG-2018 16:22:02
```

Example: Display Security Definitions

The examples below show the initial values of the security definitions stored in the security repository. Start the database and display the security definitions.

```
> adarba dbid=nnn list,user
```

This shows the user *PUBLIC*.

```
> adarba dbid=nnn list,role
```

This shows the role *PUBLIC*.

```
> adarba dbid=nnn list,assignment,user
```

This shows the assignment *PUBLIC,PUBLIC*. That is the role *PUBLIC* has been assigned to the user *PUBLIC*.

Getting Started with Authentication

The usage of authentication requires that the credentials be set by the application, or be provided by the Adabas nucleus user exit 21. Refer to the ***Application Development*** section for further information on the setting credentials.

Example: Use ADATST to validate Credentials

The example below show how ADATST can be used to verify the authentication of credentials.

```
> adatst

%ADATST-I-STARTED,      16-AUG-2018 12:11:22, Version 6.7.0.0

> adatst: ;-------- Database ID
> adatst: dbid=<dbid>

> adatst: ;-------- Credentials (without trailing whitespace)
> adatst: sec_uid=<userid>
> adatst: sec_pwd=<password>
```

```
> adatst: ;-------- Session Open
> adatst: cc=op
> adatst: rb=.
> adatst: go

  Command : OP     Test-Nr : 1 Started
%ADATST-I-NORMAL, normal successful completion
  Command : OP     Test-Nr : 1 Completed

> adatst: ;-------- Session Close
> adatst: cc=cl
> adatst: go

  Command : CL     Test-Nr : 2 Started
%ADATST-I-NORMAL, normal successful completion
  Command : CL     Test-Nr : 2 Completed

> adatst: quit

%ADATST-I-TERMINATED,   16-AUG-2018 12:11:22, elapsed time: 00:00:00
```

A nucleus response code 200 with subcode 31 indicates a security violation; the authentication check in the external security system failed.

Possible causes of a security violation are:

- The credentials are invalid.
- The configuration of the authentication authority is invalid.
- The authentication authority is not accessible.

A nucleus response code 9 with subcode SE indicates that the credentials have been modified; for example by altering the credentials during the open session.

Example: Use ADATST to validate the usage of Nucleus user exit 21

The example below shows how ADATST can be used to verify that valid credentials are provided by nucleus user exit 21. The user exit is called to provide credentials if the calling application has not provided credentials.

```
> adatst

%ADATST-I-STARTED,      16-AUG-2018 12:11:22, Version 6.7.0.0

> adatst: ;-------- Database ID
> adatst: dbid=<dbid>

> adatst: ;-------- Session Open
> adatst: cc=op
> adatst: rb=.
> adatst: go
```

```
  Command : OP     Test-Nr : 1 Started
%ADATST-I-NORMAL, normal successful completion
  Command : OP     Test-Nr : 1 Completed

> adatst: ;-------- Session Close
> adatst: cc=cl
> adatst: go

  Command : CL     Test-Nr : 2 Started
%ADATST-I-NORMAL, normal successful completion
  Command : CL     Test-Nr : 2 Completed

> adatst: quit

%ADATST-I-TERMINATED,   16-AUG-2018 12:11:22, elapsed time: 00:00:00
```

A nucleus response code 200 with subcode 31 indicates a security violation; the authentication check in the external security system failed.

Possible causes of a security violation are:

- Nucleus user exit 21 is not enabled.
- The provided credentials are invalid.
- The configuration of the authentication authority is invalid.
- The authentication authority is not accessible.

Getting Started with Authorization for Direct Call Interface

This section describes how to secure access to application data, for example allowing all users READ access to the EMPLOYEES file, and at the same time restricting the privilege to modify data.

In this example, you will create and modify the security definitions. In the end, only users assigned the HR_department role are able to modify data in the EMPLOYEES file, but all users are allowed to READ the file.

To achieve this, you need to:

- Create a HR_department role;
- Grant the role permission to modify the EMPLOYEES file;
- Create an HR_userid user;
- Grant the user the HR_department role;
- Grant the *PUBLIC* role permission to READ the EMPLOYEES file.

Example: Secure access to the EMPLOYEES File

The examples below show the security definitions required to restrict access to the EMPLOYEES file.

```
> adarba
> adarba: dbid=nnn

> adarba: create,role=HR_department
> adarba: grant,operation=ANY,object=11,to,role=HR_department

> adarba: create,user=HR_userid
> adarba: grant,role=HR_department,to,user=HR_userid

> adarba: grant,operation=READ,object=11,to,role=PUBLIC
```

The following security definitions are created or modified:

- Created:
  - Role: *HR_department*
  - User: *HR_userid*
  - Permission to Operation and Object: *ANY, File 11 (EMPLOYEES-NAT)*
- Modified:
  - Role: *PUBLIC*
  - Permission to Operation and Object: *READ, File 11 (EMPLOYEES-NAT)*

Note: The role name and the user name are site-specific.

The user name used for authorization is the same as the one used for authentication.

The user name must be defined for both authorization and authentication.

List Security Definitions

The example below shows the security definitions which secure access to the EMPLOYEES file.

```
> adarba
> adarba: dbid=nnn


> adarba: ; ---------- USERS
> adarba: list,user
PUBLIC
HR_userid
```

```
> adarba: ; ---------- ROLES
> adarba: list,role
PUBLIC
HR_department


> adarba: ; ---------- USER-ROLE ASSIGNMENTS
> adarba: list,assignment,user
PUBLIC,PUBLIC
HR_department,HR_userid


> adarba: ; ---------- PERMISSIONS
> adarba: list,assignment,permission
ada.uti....,DBID.CURRENT,PUBLIC
ada.uti....,DBID.CURRENT,PUBLIC
ada.uti....,DBID.CURRENT,PUBLIC

ada.dml.delete,FILE.00000011,HR_department
ada.dml.insert,FILE.00000011,HR_department
ada.dml.read,FILE.00000011,HR_department
ada.dml.update,FILE.00000011,HR_department

ada.dml.read,FILE.00000011,PUBLIC
```

Getting Started with Authorization for Adabas Utilities

This section describes how to secure access to an Adabas utility, for example the ADADBM utility.

In this example, you will create and modify the security definitions. By default, all users with the *PUBLIC* role have access to and can execute the ADADBM utility. In the end, only users assigned the database administrator role are able to execute the utility.

To achieve this, you need to:

- Create a *database administrator* role;
- Grant the role permission to execute the utility;
- Create a user that is to be assigned the *database administrator* role;
- Grant the user the *database administrator* role;
- Revoke the permission to execute the utility from the role *PUBLIC*.

Example: Secure access to the ADADBM utility

```
> adarba
> adarba: dbid=nnn

> adarba: create,role=database_administrator
> adarba: grant,operation=ada.uti.dbm,to,role=database_administrator

> adarba: create,user=dba_userid
> adarba: grant,role=database_administrator,to,user=dba_userid

> adarba: revoke,operation=ada.uti.dbm,from,role=PUBLIC
```

In this example, the following security definitions are created or modified:

- Created:
  - Role: *database_administrator*
  - User credentials: *dba_userid*
  - Permission to operation: *ada.uti.dbm*
- Modified:
  - Role: *PUBLIC*
  - Permission to operation: *ada.uti.dbm*

Note: The role name and the user credentials are site-specific. The user credentials used for authorization are local system credentials.

Example: List Security Definitions

The example below shows the security definitions that secure access to the ADADBM utility.

```
> adarba
> adarba: dbid=nnn


> adarba: ; ---------- USERS
> adarba: list,user
PUBLIC
dba_userid


> adarba: ; ---------- ROLES
> adarba: list,role
PUBLIC
database_administrator


> adarba: ; ---------- USER-ROLE ASSIGNMENTS
```

```
> adarba: list,assignment,user
PUBLIC,PUBLIC
database_administrator,dba_userid


> adarba: ; ---------- PERMISSIONS
> adarba: list,assignment,permission
ada.uti....,DBID.CURRENT,PUBLIC
ada.uti....,DBID.CURRENT,PUBLIC
ada.uti....,DBID.CURRENT,PUBLIC

ada.uti.dbm,DBID.CURRENT,database_administrator
```

Infrastructure Security Library

The infrastructure security libraries are used to access the authentication authority. The following sections contain configuration templates for the `SSX_CONFIGURATION` parameters, organized by type.

- Authentication Type OS (Operating System)
- Authentication Type TEXT (Internal User Repository)
- Authentication Type LDAP
- Authorization Type ADSI
- Creating Internal User Repository Files

Authentication Type OS (Operating System)

The security definitions for authentication type OS are managed by the local operating system.

```
[SSX_CONFIGURATION]

# This is a sample properties file for the case
# when authType is OS and the user database is
# the local operating system -
# On Linux Systems it is using PAM authentication
# On Windows a local LogonUser()

# Specifies the authentication type.
# Is Required: Yes
# Valid values: {"OS", "TEXT", "LDAP", "ADSI"}
# Default Value: None

  authType=OS

# Specifies the explicit path of the privileged daemon process.
# Specify this parameter -
# if the sagssxauthd2 executable file is not in the current directory.
```

```
# Valid value is the valid path to the sagssxauthd2 module.
# Default Value: None
# Note: Linux only.

##authDaemonPath

# Specify a default group name here to be returned
# with any of the group results that are returned by the repository manager.
# A valid value is any valid group name.
# Default Value: None
# Optional.

##defaultGroup

# If this parameter is specified, its value is used at authentication time
# when domain name is not specified by the user.
# If a domain name is specified, the value of this parameter is not used.
# A valid value is any valid domain name.
# Default Value: None
# Optional.

##defaultDomain

# Specifies how to access data.
# Valid values are:
# o true  - Access is under the account of the running process.
# o false - Access is under the impersonated user ID of the logged on user.
# Default Value: FALSE
# Note: Windows only.
# Optional.

##noImpersonation

# Specifies the local machine name (on which the user is authenticated).
# The machine name is added before users and groups;
# for example,machine_name\user.
# Valid values are:
# o true - If set to TRUE (and there is no domain field), you are authenticated ↩
against the local machine only.
# o false - You are authenticated on the domain that you logged on.
# Default Value: FALSE
# Optional.

##linuxAddMachineName

# Specifies the log level.
# Is Required: No
# Valid values:
#    0 - No logging
#    Min: 1
#    Max: 6
# Default Value: None
```

```
##nativeLogLevel=0

# Specifies the log file.
# Is Required: No
# Valid values:
#    fully qualified file name
# Default Value: None

##nativeLogFile=SAGSSXCLIENTA_SSX.LOG

[SSX_CONFIGURATION-END]
```

Authentication Type TEXT (Internal User Repository)

The security definitions for authentication type TEXT are stored in a text file. The definitions can either be database-specific or be shared by multiple databases.

```
[SSX_CONFIGURATION]

# This is a sample properties file for the case
# when authType is TEXT and the user database is
# an SAG Internal User Repository
# created by the ssxtxtpasswd utility

# Specifies the authentication type.
# Is Required: Yes
# Valid values: {"OS", "TEXT", "LDAP", "ADSI"}
# Default Value: None

  authType=TEXT

# Specifies the internal repository file
# which has been created with ssxtxtpasswd utility
# Is Required: No
# Valid values:
#   fully qualified file name
# Default Value: None

  internalRepository=<fullpath>/<filename>.<ext>

# Specifies the log level.
# Is Required: No
# Valid values:
#    0 - No logging
#    Min: 1
#    Max: 6
# Default Value: None

##nativeLogLevel=0
```

```
#  Specifies the log file.
#  Is Required: No
#  Valid values:
#   fully qualified file name
#  No default value

##nativeLogFile=SAGSSXCLIENTA_SSX.LOG

[SSX_CONFIGURATION-END]
```

Authentication Type LDAP

```
[SSX_CONFIGURATION]

# This is a sample properties file for the case
# when authType is LDAP and the user database is OpenLDAP

# Specifies the authentication type.
# Is Required: Yes
# Valid values: {"OS", "TEXT", "LDAP", "ADSI"}
# Default Value: None

  authType=LDAP

# Specifies which server type will be used.
# Is Required: No
# Valid values: {"ActiveDirectory", "SunOneDirectory", "OpenLdap"}
# Default value: "OpenLdap"

  serverType=OpenLDAP

# Property name that denotes a user entry.
# Is Required: No
# Valid values: (attribute name according to LDAP conventions)
# Default Value: None

  userIdField=cn

# Enumeration of LDAP objectclasses that the user entries use in
# the target LDAP server.
# Is Required: No
# Valid values: (Comma separated list of objectclass names,
#  according to LDAP conventions)
# Default value - depending on serverType:
#  OpenLdap:
#  "top,person"
#  SunOneDirectory:
#  "top,person,organizationalperson, inetorgperson"
#  ActiveDirectory:
#  "top,person,organizationalPerson,user"
```

```
  personObjClass=inetOrgPerson

# Enumeration of LDAP objectclasses that the group entries use in
# the target LDAP server.
# Is Required: No
# Valid values: (Comma separated list of objectclass names,
#  according to LDAP conventions)
# Default value - depending on serverType:
#    OpenLdap:
#    "top,groupOfUniqueNames"
#    SunOneDirectory:
#    "top,groupofuniquenames"
#    ActiveDirectory:
#    "top,group"

  groupObjClass=groupOfUniqueNames

# Property name that denotes a group entry.
# Is Required: No
# Valid values: (attribute name according to LDAP conventions)
# Default value: cn

  groupIdField=cn

# Property name of a user entry that points to the group that
# the user is member of.
# Is Required: No
# Valid values: (attribute name according to LDAP conventions)
# Default value:
#  depending on serverType:
#  OpenLdap:
#  "ou"
#  SunOneDirectory:
#  NULL
#  ActiveDirectory:
#  "memberOf"

  personGrpAttr=ou

# Property name of a group entry that points to users (members)
# Is Required: No
# Valid values: (attribute name according to LDAP conventions)
# Default value:
#  depending on serverType:
#  OpenLdap:
#  "uniqueMember"
#  SunOneDirectory:
#  "uniqueMember"
#  ActiveDirectory:
#  "member"

  groupPrsAttr=uniqueMember
```

```
# Seconds how long auth. user remains in cache.
# Is Required: No
# Valid values:
#  0 - No cache
#  Min: 1, Max: No limit
# Default value: 180

  cacheTime=12

# Specify the max. number of cached users that have been successfully
# authenticated. When the cache overflows, the oldest entry is removed.
# Is Required: No
# Valid values:
#  0 - No cache
#  Min: 1, Max: No limit
# Default value: 300

  cacheSize=4

# Time (in seconds) how long to ignore any further authentication
# requests for a particular User-Id.
# Is Required: No
# Valid values:
#  Min: 1, Max: No limit
# Default value: 100

  denyTime=4

# Number of invalid logon attempts.
# Is Required: No
# Valid values:
#  Min: 1, Max: No limit
# Default value: 3

  denyCount=3

# Specifies an output file for logging.
# Is Required: No
# Valid values: (Valid log file path)
# Default Value: None

  logCallback=true

# Specifies the log level.
# Is Required: No
# Valid values:
#    0 - No logging
#    Min: 1
#    Max: 6
# Default Value: None
```

```
##nativeLogLevel=0

#  Specifies the log file.
#  Is Required: No
#  Valid values:
#   fully qualified file name
#  No default value

##nativeLogFile=SAGSSXCLIENTA_SSX.LOG

# Default group to be automatically included for all requests
# that return any groups
# Is Required: No

##defaultGroup=DefGroup

# BaseBindDN where to find the users.
# Is Required: Yes
# and should contain the most detailed DN to find the users

#  personBindDn=ou=User,o=Org,dc=mycom,dc=com

# BaseBindDN where to find the groups.
# Is Required: Yes
# and should contain the most detailed DN to find the groups

##groupBindDn=ou=Groups,o=Org,dc=mycom,dc=com

# Attribute name of the password.
# Required when changeing the password
# Is Required: Not always
# Default value:
#  depending on serverType:
#  OpenLdap:
#  "userPassword"
#  SunOneDirectory:
#  "userPassword"
#  ActiveDirectory:
#  "unicodePwd"

##passwdField=userPassword

# Allow to pass a complete BaseBindDN
# via the domain parameter.
# Is Required: No
# Valid values: 0, 1

##allowdomainasbasebinddn=0

# Allow to specify which fields to search for as properties
# of a user entry
# Is Required: No
```

```
# Valid values: string, for example: "cn,sn,description"

##personPropAttr

# Allow to specify which fields to search for as properties
# of a group entry
# Is Required: No
# Valid values: string, for example: "cn,description"

##groupPropAttr

# Allow to use the special secure authentication using SASL,
# providing the directory supports this mechanism.
# Is Required: No
# Valid values: 0, 1 (default: 0)

##ldapSaslBind

# Allow to switch from a non-secure connection to a TLS connection,
# providing the directory supports this mechanism.
# of a group entry
# Is Required: No
# Valid values: 0, 1 (default: 0)

##ldapStartTls

# By default, the first "dc=" occurrence within the distinguished name
# name string denotes the domain name.
# If additional abbreviations want to be defined, one can use
# the following 2 parameter.
# Example:  Short="RnD;Admins;board"
#       with ↩
Long="ou=Rnd,ou=user,dc=mycom,dc=com;ou=Administrators,dc=mycom,dc=com;ou=VIP,dc=mycom,dc-com"

##ldapDomainShort
##ldapDomainLong

# If NOT the automatic domain name should be used to compose
# the canonical user id (SSXGetCanonicalUserId_A/W),
# specify this part of the ID here.

##canonicalDomainName

# Three algorithms are supported to find the groups of a user:
# "ru", recurse up: take the group pointer from the user entry
#                   and continue to search up for all groups
#                   found
# "rd", recurse down: search for all groups that have the
#                     user as member (no recursion)
# "cp", computed property: use a special field in the user
#                          entry to find all groups
#                          --> computedGroupProp retired
```

```
# Default: "ru"

##resolveGroups

# If resolveGroup is set to "cp", this parameter must provide
# the field name to look for in the user entry that denotes
# the user groups
# Default: None

##computedGroupProp=

# If the LDAP connection is protected by SSL/TLS, this
# parameter must be set.
# Valid Values: 0, 1
# Default: 0

##ldapSSLConnection=1

[SSX_CONFIGURATION-END]
```

Authorization Type ADSI

```
[SSX_CONFIGURATION]

# This is a sample properties file for the case
# when authType is ADSI and the user database is Active Directory

# Specifies the authentication type.
# Is Required: Yes
# Valid values: {"OS", "TEXT", "LDAP", "ADSI"}
# Default Value: None

  authType=ADSI

# Specifies the name of the AD Forest.
# Is Required: No, but should be specified
# Example: "dc=mycom,dc=com"
# (with a possible domain called "dc=eur,dc=mycom,dc=com")
# Default Value: None

##adsiForestDn

# Seconds how long auth. user remains in cache.
# Is Required: No
# Valid values:
#  0 - No cache
#  Min: 1, Max: No limit
# Default value: 180

  cacheTime=12
```

```
# Specify the max. number of cached users that have been successfully
# authenticated. When the cache overflows, the oldest entry is removed.
# Is Required: No
# Valid values:
#  0 - No cache
#  Min: 1, Max: No limit
# Default value: 300

  cacheSize=4

# Time (in seconds) how long to ignore any further authentication
# requests for a particular User-Id.

# Is Required: No
# Valid values:
#  Min: 1, Max: No limit
# Default value: 100

  denyTime=4

# Number of invalid logon attempts.
# Is Required: No
# Valid values:
#  Min: 1, Max: No limit
# Default value: 3

  denyCount=3

# Specifies an output file for logging.
# Is Required: No
# Valid values: (Valid log file path)
# Default Value: None
#   nativeLogFile=SIN_SSX.log

  logCallback=true

# Specifies the log level.
# Is Required: No
# Valid values:
#    0 - No logging
#    Min: 1
#    Max: 6
# Default Value: None

##nativeLogLevel=0

#  Specifies the log file.
#  Is Required: No
#  Valid values:
#   fully qualified file name
#  No default value
```

```
##nativeLogFile=SAGSSXCLIENTA_SSX.LOG

# In case the scope for the node to access users needs to be limited,
# one can specify a particular subtree:
# Example: "ou=user,ou=Rnd,dc=mycom,dc=com"

##adsiPersonBindDn

# In case the scope for the node to access groups needs to be limited,
# one can specify a particular subtree:
# Example: "ou=groups,ou=Rnd,dc=mycom,dc=com"

##adsiGroupBindDn

# By default, the first "dc=" occurrence within the distinguished name
# name string denotes the domain name.
# If additional abbreviations want to be defined, one can use
# the following 2 parameter.
# Example:  Short="RnD;Admins;board"
#       with   ↩
Dn="ou=Rnd,ou=user,dc=mycom,dc=com;ou=Administrators,dc=mycom,dc=com;ou=VIP,dc=mycom,dc-com"

##adsiDomainShort
##adsiDomainDn

# If NOT the automatic domain name should be used to compose
# the canonical user id (SSXGetCanonicalUserId_A/W),
# specify this part of the ID here.

##canonicalDomainName

# Three algorithms are supported to find the groups of a user:
# "ru", recurse up: take the group pointer from the user entry
#                   and continue to search up for all groups
#                   found
# "rd", recurse down: search for all groups that have the
#                     user as member (no recursion)
# "cp", computed property: use a special field in the user
#                          entry to find all groups
#                          --> computedGroupProp retired
# Default: "ru"

##resolveGroups

# If resolveGroup is set to "cp", this parameter must provide
# the field name to look for in the user entry that denotes
# the user groups
# Default: None

##computedGroupProp=

[SSX_CONFIGURATION-END]
```

Creating Internal User Repository Files

The following section describes the creation and administration of the internal user repository file, which is used with Authorization Type TEXT.

You can create and/or modify internal user repository files with the `ssxtxtpasswd` tool.

To start the `ssxtxtpasswd` tool, you use a command prompt. When you start the tool, you enter a user name and a password which are then encrypted (SHA512 and Base64) and provided in the result text file. The tool adds new or replaces existing user credentials in the text file.

Note: When you enter a user name, you can use only digits, Latin letters, and the following characters: ! ( ) - . ? [ ] _ ~ . When you enter a password, you can use only digits, Latin letters, and the following characters: !"#$%&'()*+,-./:;<=>?[\]^_`{|}~.

Example: Usage of ssxtxtpasswd tool

```
Tool to create or update an entry in the SSX text file based
user repository.

Usage: ssxtxtpasswd [-f filename] [-c] [-p password] [-d | -e] userId

Use "-c" to create a new file.
      Usually, the file should exist and user entries are
      replaced/added.

Use "-p" to provide the password on the command line instead
      via an extra prompt.

Use "-d" to remove the specified user entry from the text file.

Use "-e" to check, whether the userId is already stored in the text file.

Note: The password usually will be read via a non-echo command input.
      When no filename is specified, a default of "ssx_user" is assumed.
```

Example: Add User and Password

```
ssxtxtpasswd -f SAGInternalUserRepository.txt -c -p mypsw myuid

Hash: ↩
bOEOAPEEEJBKv+4zOELiYcFqY7qFhlLZz1ha7Ztf7j/drJHGy2MLOLXEu/kX7TD52Aj7XfwiZ+vpIl9DqRbVkA==
User entry for "myuid" successfully added
```

Contents of SAGInternalUserRepository.txt

```
*
*
* SAG Internal User Repository
*
version:3.0
*
user:myuid:$6a$bOEOAPEEEJBKv+4zOELiYcFqY7qFhlLZz1ha7Ztf7j/drJHGy2MLOLXEu/kX7TD52Aj7XfwiZ+vpIl9DqRbVkA==
```

4 Adabas Password Security (ADASCR)

- Introduction 56
- File Protection Levels 56
- User Passwords 57
- Security by Value Criteria 57
- Adabas Security Processing 58

Introduction

The Adabas database security utility ADASCR provides facilities for controlling access and update to Adabas files.

Adabas supports two classes of data access/update security: the first restricts user read and/or update requests on the basis of a whole file; the second restricts user access to individual records within a file.

Protection at File Level

Adabas files are security-protected if a file protection level greater than zero is assigned to the file. File protection levels are assigned separately for access (i.e. read) and update.

A user can access/update a security-protected file by entering a password with a permission level that is equal to or greater than the file protection level. Protection levels and password permission levels are assigned with the security utility ADASCR.

The file numbers that can be protected are limited by the block size of the ASSO1 container file. If the block size is 2KB, only files in the range 1 - 2047 can be protected (for 3KB the limit is 3071).

Protection at Record Level - Security by Value

Security by Value extends the Adabas File Protection Level security by enabling a user to further define separate data access and update restrictions according to the content of one or more fields in a data record.

Security by Value criteria are defined by using the ADASCR security utility. Each password may include value criteria for up to 99 separate Adabas files.

Record level security can only be used with security-protected files.

File Protection Levels

An Adabas file can be security-protected by assigning an access protection level and/or an update protection level greater than zero to the file.

File protection levels range from 0 to 15, with 15 being the maximum level of protection. A protection level of 0 means that all users can access/update the file. A value of 15 prevents all users from accessing/updating the file.

All Adabas files that have no protection levels assigned have a default protection level of zero. All users can access and/or update such files.

File	Protection Level Number	
	Access	Update
10	7	11
11	2	2
12	4	4

User Passwords

Separate access and update levels for each Adabas file can be assigned for each password. Password permission levels can be assigned in the range from 0 to 14.

Password	FILE 10 - ACC/UPD	FILE 11 - ACC/UPD	FILE 12 - ACC/UPD
PASSWRD1	4/0	4/0	4/0
PASSWRD2	2/2	2/2	2/2
PASSWRD3	14/0	0/0	14/0
PASSWRD4	14/14	14/14	14/14
PASSWRD5	7/7	0/0	7/0

If the password access/update permission level is equal to or greater than the file access/update protection level, the password can be successfully used to access/update the file in question.

Using the examples for file protection levels and for password permission levels shown above,

- PASSWRD1 can be used to access file 11 or file 12.
- PASSWRD2 can be used to access/update file 11.
- PASSWRD3 and PASSWRD5 can each be used to access file 10 or file 12.
- PASSWRD4 can be used to access/update file 10, file 11 or file 12.

Security by Value Criteria

Separate access and update criteria for each Adabas file can be assigned for each password. Each value criterion consists of an Adabas search buffer and an associated value buffer.

The search and value buffers are constructed in the same manner as for regular Adabas search commands, including the use of non-descriptor fields and multiple value fields. However, soft-coupling and sub-, super-, hyper- and phonetic descriptor fields are not supported in Security by Value search criteria.

For further information on the syntax and construction of search buffers and value buffers, see the Command Reference Manual, Calling Adabas, Search and Value Buffers.

Value checking is performed only if data storage is either read or updated by the Adabas command.

The following table illustrates which criteria are tested for the various Adabas commands:

Adabas Command	Security by Value Check Performed	
	Test Criterion	Test Data
A1	Update	Before Image
E1	Update	Before Image
L1 - L6	Access	Before Image
L9	*(Access to index only, no value check performed)*	
N1, N2	Update	After Image
S1(*)	Access	Before Image

Note: * An S1 command with a valid format buffer is handled in the same manner as if an S1 command followed by an L1 command with a given User ISN had been issued.

Security by Value criteria are ignored if the associated security level of the requested file is zero.

Adabas Security Processing

Using the password entered by a user and the file-protection information defined for the file, Adabas checks whether the user is authorized to access/update a given Adabas file. If the file is not security-protected, Adabas ignores any password that is entered. The following describes Adabas security processing for read and update commands.

Adabas determines whether the file to be accessed or updated is security-protected. If there is no security protection, security processing stops.

Security Response Codes

If the file to be processed is security-protected, Adabas checks whether the password supplied is defined in the password table.

If the password is undefined or if no password has been supplied, response code 201 is returned.

If the password is defined but not valid for the file to be processed, response code 202 is returned.

If the password is valid for the file to be processed, Adabas checks the permission level associated with this password against the file's access or update protection level. Response code 200 is returned if the password's permission level is less than the file's protection level.

If the permission level is sufficient, the password is further checked for Security by Value criteria for the current file. If a search criterion has been defined for the supplied password, this is tested against either the before or after image of the data, depending on the Adabas command issued. If the Security by Value check is unsuccessful, response code 200 is returned, otherwise the user's request is finally granted and the Adabas command is processed.

If an ET logic user receives the response codes 200 - 202, processing continues as if the user had exceeded the transaction time limit (see the TT parameter of ADANUC in the Utilities Manual for further information).

All Security by Value security violations cause response code 200 to be returned.

The following is a summary of the response codes that may be returned by Adabas security processing:

RESPONSE 200

Explanation	Security violation detected.
User Action	Supply the correct password.

RESPONSE 201

Explanation	The password specified was not found.
User Action	Supply the correct password.

RESPONSE 202

Explanation	An attempt was made to use a file for which the user is not authorized.
User Action	Supply the correct password.

5 Adabas Encryption for Linux

- Prerequisite 62
- Encryption of Data-At-Rest 62
- Key Management System 63
- Administration 70
- Database Access 75

Adabas Encryption for Linux provides functionality to protect data-at-rest from unauthorized access. With Adabas Encryption for Linux, data-at-rest refers to container files, backup copies, PLOG files and other files created by Adabas, containing sensitive data container files and also PLOG and backup copies. Access to the information in these containers is controlled by means of encryption.

Prerequisite

A license for Adabas Encryption for Linux (AEL) is required.

Encryption of Data-At-Rest

Encryption Object

In the context of Adabas Encryption for Linux, data-at-rest refers to the following containers and files:

- ASSO
- DATA
- WORK
- TEMP
- SORT
- PLOG
- backup copies
- CMPDTA, DCUDTA, MUPDTA, ULDDTA (Compressed data)
- CMPDVT, MUPDVT, ULDDVT (Descriptor Value Table)
- MUPLOG and MUPLOB (Compressed data and LOB data)
- ORDEXP (Export copy)

Encryption Algorithm

Encryption is the process of converting plain data into cipher text which cannot be deciphered without access to the encryption key.

Decryption is the process of converting encrypted data back to its original form. Decryption of encrypted data requires the key. Without the key the original information cannot be recovered.

Adabas Encryption for Linux supports Advanced Encryption Standard (AES) in XTS mode, a symmetric-key algorithm, where the same key is used for both processes, encrypting and decrypting the data.

XTS-AES with a key length of 128 and 256 bits are supported.

Encryption Process

When formatting a database, the corresponding containers are encrypted according to the given parameters. All blocks written to disk are encrypted as well. Keys for encryption are managed by Adabas using the given Key Management System.

Note: Encryption cannot be disabled for existing databases. Encryption can only be undone by dumping and restoring a backup with the option ENCRYPTION=NO.

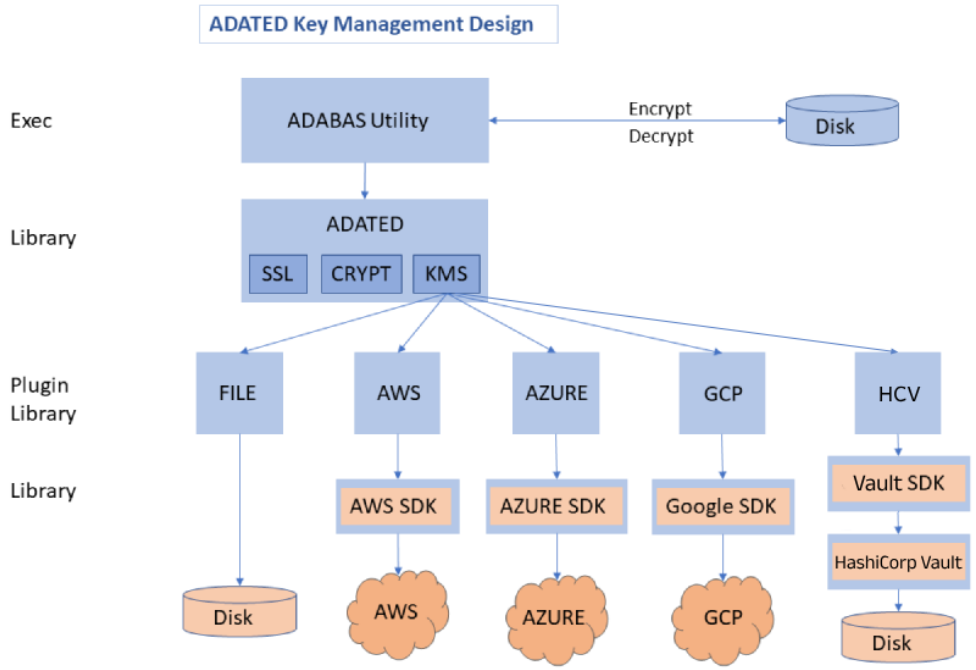
Key Management System

A key management system (KMS), also known as a cryptographic key management system (CKMS) or enterprise key management system (EKMS), is an integrated approach for generating, distributing, and managing cryptographic keys for devices and applications.

Adabas currently supports the following types of key management systems:

- File-based management system on the local disk
- Microsoft Azure Key Management System
- Amazon AWS Key Management Service
- Google GCP Key Management Service
- HashiCorp Vault Key Management System

This section describes how Adabas uses the KMS.



Adabas provides the ADATED library for the cryptographic functionality. The KMS function of ADATED uses plugin libraries, also provided by Adabas, for each supported key management system.

Adabas uses two types of encryption keys, the key encryption key (KEK) and the encrypted data encryption key (DEK). The KEK encrypts the DEK, which in turn encrypts the data in the database. The advantage of using two keys is that you can regularly change the key encryption key, so only the data encryption key is decrypted and newly encrypted. The data itself does not have to be decrypted and encrypted again.

Important: If the KEK is lost or corrupted, the data in the database is also lost regardless of which KMS is used.

- File-Based Key Management System
- Amazon AWS Key Management System
- Microsoft Azure Key Management System
- Google GCP Key Management Service

- HashiCorp Vault Key Management System

File-Based Key Management System

The file-based key management system uses a local disk file to store the key encryption keys. Database operations are performed only with the KMS file.

Select the file-based KMS by configuring the KMSTARGET=FILE parameter for either the ADAFRM or ADABCK utility when creating a database. The selected KMS type is stored in the database.

The file to store the keys is specified by the ADAKMSFILE environment variable. You can specify a fully qualified path name or directory name. The default location is $ADADATADIR/etc and the default file name is adatedkmsfile.db.

Important: If the Adabas file-based key management system is used, all Adabas utilities require access to the file $ADADATADIR/etc/adatedkmsfile.db. Otherwise an encrypted database becomes unusable.

The KMS file is stored encrypted in the file system. Nevertheless, access to it must be restricted with operating system features. File permissions must allow access only to the user/group that will run Adabas utilities.

Caution: It is recommended to implement a dedicated backup strategy to have the file-based KMS available in case of a disaster recovery. Without the corresponding KEK, it is not possible to restore Adabas backup copies or use encrypted data. To avoid this effort and possible risks, cloud-based key management services should be preferred.

Amazon AWS Key Management System

Amazon Web Services (AWS) provides a built-in key management service which you can use when Adabas is on an AWS virtual machine. The AWS KMS allows for the creation and storage of the key encryption key. The encrypted DEK is passed to the AWS KMS for decryption.

Select the AWS KMS by configuring the KMSTARGET=AWS parameter for either the ADAFRM or ADABCK utility when creating a database. The selected KMS type is stored in the database.

Within AWS, you must configure your virtual machine to be able to access the AWS Key Management System. The Adabas AWS plugin uses a number of AWS APIs and KMS resources, which are managed through AWS Identity and Access Management (IAM). To use the APIs and access the specific KMS resources you must create an IAM role with a permission policy. The IAM role must be assigned to an EC2 instance (Amazon Elastic Compute Cloud) so that the EC2 instance can assume that role during its lifetime. The IAM role can be used for multiple EC2 instances.

For more information on how to create an IAM role and assign it to an EC2 instance, see the AWS Identity and Access Management and Amazon Elastic Compute Cloud documentation.

Microsoft Azure Key Management System

Azure provides a key management service for generating, storing, and accessing secrets, such as cryptographic keys. You can use this service when running Adabas in the Azure cloud. The cryptographic keys are stored and accessed in an Azure Key Vault. Adabas can use the Azure Key Management Service to store the keys used in an encrypted database.

To use the Azure Key Vault for Key Management purposes, specify KMSTARGET=Azure for either the ADAFRM or ADABCK utility when creating a database. The selected KMS type is stored in the database.

You must reference the key vault with the environment variable AZURE_KEY_VAULT.

```
export AZURE_KEY_VAULT=<key-vault-name>
```

Authentication

To access the key vault, you must be an authenticated user with access rights. It is recommended to use a managed identity if Adabas is run on the Azure Cloud.

To create a managed identity:

- Go to the Azure Portal and navigate to the virtual machine or container Adabas is running on.
- Select **Identity**.
- In the tab **System assigned**, set the **Status** to **On**. This creates a service principal that is tied to the resource.

To authenticate multiple resources, you can use a User assigned managed identity. If you require a different form of authentication, visit the Azure resources.

Key Vault Access

The created security principal needs to be given access to the Azure Key Vault resource. You can do this on the Azure Portal.

- Navigate to the used Key Vault and select **Access Policies**.
- Click **Create**.
- From the **Permissions**, check **Select all** for **Key Management Operations** and **Cryptographic Operations**.
- Choose the created managed identity for authentication with the Key Vault.
- Create the access policy.

Note: As the Key Vault holds the key encryption key of your database, the access rights to this resource should be kept to a minimum.

For more information about the Azure Key Vault, managed identities, or authentication with Azure, you can visit **Microsoft Learn**.

Google GCP Key Management Service

Google Cloud Portal (GCP) provides a key management service for generating, storing, and accessing secrets, such as cryptographic keys. This key management service can be used when running Adabas in the Google Cloud. The cryptographic keys are stored and accessed in a Google key ring. Adabas can use the Google Key Management Service to store the keys used in an encrypted database.

To use the key ring for key management purposes, specify the `KMSTARGET=GCP` parameter for either the ADAFRM or ADABCK utility when creating a database. The selected KMS type is stored in the database.

The key ring resource must be located in the same region as your virtual machine. You must reference the key ring with: `export GCP_KEY_RING=`*`key-ring-name`*

Within the GCP, you must configure your VM service account to be able to access the key ring resource. Ensure that the VM service account has access to the following permissions:

cloudkms.cryptoKeyVersions.create
cloudkms.cryptoKeyVersions.destroy
cloudkms.cryptoKeyVersions.get
cloudkms.cryptoKeyVersions.list
cloudkms.cryptoKeyVersions.useToDecrypt
cloudkms.cryptoKeyVersions.useToEncrypt
cloudkms.cryptoKeys.create
cloudkms.cryptoKeys.get
cloudkms.cryptoKeys.list

Note: As the key ring holds the encryption key of your database, the access rights to this resource should be kept to a minimum.

For more information about the hierarchy of resources and roles within Cloud KMS, see the Google Cloud Key Management Service documentation.

HashiCorp Vault Key Management System

HashiCorp Vault provides a key management service for generating, storing, and accessing secrets, such as cryptographic keys. Adabas uses Vault to securely manage cryptographic keys, particularly for encrypting databases.

Select the file-based KMS by configuring the `KMSTARGET=HCV` parameter for either the ADAFRM or ADABCK utility when creating a database. The selected KMS type is stored in the database.

Using the Vault Transit Engine with Adabas

To use Vault for key management, the following environment variables must be set:

```
export HCV_VAULT_ADDR=<vault-address>
export HCV_AUTH_METHOD=<auth-method>
export HCV_BASE_PATH=<base-path> # Required: Set the base path for transit and metadata
export HCV_EXPIRATION_DAYS=<expiration-days> # Optional: Set the number of days after which the key expires (Default: 7 days)
```

- The `HCV_BASE_PATH` is not optional and must be set. This path defines the base path for transit and metadata in Vault.
- The `HCV_EXPIRATION_DAYS` environment variable is optional. If not set, the default is 7 days. You can specify a different number of days for the key expiration. If set to 0, the key will be immediately deleted.

Automatic Path Mounting by Adabas

After configuring the `HCV_BASE_PATH`, Adabas automatically mounts the necessary paths in Vault for managing cryptographic keys. These paths are used for encryption operations in the Adabas database, ensuring seamless integration between Adabas and Vault.

Soft Deletion of Keys by Adabas

Adabas implements a Soft Deletion feature, allowing keys marked for deletion to be recovered within a specified time frame. This helps prevent accidental data loss and provides an additional layer of security.

1. Mark for Deletion: A key is marked for deletion but not removed immediately.
2. Recovery: The deleted key can be restored within the specified recovery period.
3. Permanent Deletion: After the recovery period expires, the key is automatically permanently deleted.

Note: Adabas will show the keys, and once the key is marked for deletion and the recovery period has expired, these keys will be automatically deleted from Vault when displayed by Adabas.

Authentication and Access Control

To use Vault, Adabas must authenticate either via AppRole or UserPass. The relevant environment variables must be set:

AppRole Authentication

For AppRole authentication, the following environment variables must be set:

- `HCV_ROLE_ID`: The Role ID for AppRole authentication.
- `HCV_SECRET_ID`: The Secret ID for AppRole authentication.
- `HCV_APPROLE_PATH`: The path where the AppRole is configured (e.g., approle).

Example:

```
export HCV_ROLE_ID=<role-id>
export HCV_SECRET_ID=<secret-id>
export HCV_APPROLE_PATH=<approle-path>
```

UserPass Authentication

For UserPass authentication, the following environment variables must be set:

`HCV_USERNAME`: The username for authentication.
`HCV_PASSWORD`: The password for the user.
`HCV_USERPASS_PATH`: The path where the UserPass authentication is configured (e.g., userpass).

Example:

```
export HCV_USERNAME=<username>
export HCV_PASSWORD=<password>
export HCV_USERPASS_PATH=<userpass-path>
```

Vault Policies

It is highly recommended to grant AppRole or UserPass users only the minimum necessary rights. Use appropriate Vault Policies to restrict access to sensitive data to the minimum required. The Principle of Least Privilege should be strictly followed to ensure that only authorized entities can access keys and other resources.

Important Note on Access Control

Vault uses policies to control which users and applications can access specific resources. Policies allow you to define precisely which permissions are required for key operations such as create, read, update, delete, and list.

Example of a Vault Policy

Here is an example of a Vault policy that restricts access to the necessary paths:

```
# Required permissions for AppRole or UserPass authentication
path "sys/mounts/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}

path "<HCV_BASE_PATH>/*" {  # Replace <HCV_BASE_PATH> with your specific base path
  capabilities = ["create", "read", "update", "delete", "list"]
}
```

Security Notes

It is recommended to restrict access to Vault and the associated key resources to authorized users and machines only. Apply the principle of least privilege and ensure that only users and applications that genuinely need access to the keys have it. A regular review of access rights should also be conducted to ensure no unnecessary permissions exist.

For detailed information on the Vault API, Vault policies and other Vault configuration options, please refer to the official **HashiCorp Vault Documentation**.

Administration

Database Creation

Adabas automatically creates a new KEK with a unique name (KEKNAME) within the KMS. The KMS API provides a function to create an encrypted DEK. Adabas stores the encrypted DEK along with the unique KEKNAME of the corresponding KEK and other encryption related settings in the ASSO1 container file.

Encrypted databases are created with Adabas utilities ADAFRM and ADABCK.

- ADAFRM

  The control parameters ENCRYPTION and KMSTARGET specify the encryption algorithm and the key management system to be used. The container files are encrypted according to the parameter values.

```
ADAFRM ENCRYPTION=AES_256_XTS | AES_128_XTS
        KMSTARGET=FILE (default) | AWS | Azure | GCP | HCV
```

Example 1:

The following example shows how to create an encrypted database with the encryption algorithm AES_256_XTS. The Adabas file-based key management system is used (default: KMSTARGET=FILE)

```
adafrm dbid=100 datasize=100 assosize=100 worksize=100 encryption=aes_256_xts
```

- ADABCK RESTORE/OVERLAY

The control parameters ENCRYPTION and KMSTARGET specify the encryption algorithm and the key management system to be used. The container files are encrypted according to the parameter values.

As the target database is created and containers are encrypted, it must not already exist. For existing databases, these ADABCK control parameters ENCRYPTION and KMSTARGET will be rejected. An existing database will always retain its encryption settings, even in the case of "no encryption".

```
ADABCK ENCRYPTION=AES_256_XTS | AES_128_XTS
        KMSTARGET=FILE (default) | AWS | Azure | GCP | HCV
```

Example 1:

The following example shows how to restore an encrypted database with the encryption algorithm AES_256_XTS from an non-encrypted Adabas backup copy. The Adabas file-based key management system is used (default: KMSTARGET=FILE). In this example, BCK001 is a backup copy of a non-encrypted database.

```
adabck dbid=100 restore=* encryption=aes_256_xts
```

Example 2:

The following example shows how to overlay an encrypted database with the encryption algorithm AES_128_XTS from an non-encrypted Adabas backup copy. The Adabas file-based key management system is used. In this example, BCK001 is a backup copy of a non-encrypted database.

```
adabck dbid=100 overlay=* encryption=aes_128_xts kmstarget=file
```

Note: The control parameter ENCRYPTION is rejected if the database to be created already exists.

Creating a Non-encrypted Database from an Encrypted Database

ADABCK, can be used to restore or overlay a non-encrypted database from the backup of an encrypted database.

ADABCK

```
ADABCK  ENCRYPTION=NO
```

Example:

The following examples show how to restore or overlay a non-encrypted database from an Adabas backup copy of an encrypted database. In these examples, BCK001 is a backup copy of an encrypted database.

```
adabck dbid=100 overlay=* encryption=no
```

```
adabck dbid=100 restore=* encryption=no
```

Database Report

The encryption settings include the encryption algorithm, the key management system and name of the key encryption key.

The ADAREP control parameter SUMMARY shows the encryption settings of an encrypted database.

The ADABCK control parameter SUMMARY shows the encryption settings of an Adabas backup copy of an encrypted database.

- **ADAREP**

  ```
  ADAREP SUMMARY
  ```

  Example:

  The following example shows the encryption settings of an encrypted database.

```
adarep dbid=100 summary
%ADAREP-I-STARTED,      02-MAY-2024 18:10:26, Version 7.3.0.0 (Linux 64Bit)
%ADAREP-I-DBOFF, database 100 accessed offline
Summary of Database 100           2-MAY-2024 18:10:26

DATABASE NAME                    Database-100
DATABASE ID                        100
MAXIMUM FILE NUMBER LOADED           3
SYSTEM FILES                         1 (CHK),     2 (SEC),     3 (USR)
ACTUAL FILES LOADED                  3
CURRENT PLOG NUMBER                  0
CURRENT CLOG NUMBER                  0
ENCRYPTION                       AES_256_XTS
KMSTARGET                        FILE
KEKNAME                          ↩
1714666215B382A0FD08997726468168646 4A029C3FA4497EE1EF150C0217295


Container  Device      Extents in Blocks    Number of      Block    Total Size
  File     Type         from          to     Blocks        Size     (Megabytes)
-------------------------------------------------------------------------------

 ASS01     file            1      25,600       25,600      4,096         100.00

 DATA1     file            1       3,200        3,200     32,768         100.00

 WORK1     file            1       6,400        6,400     16,384         100.00

-------------------------------------------------------------------------------
                                                                        300.00
                                                                        ======

%ADAREP-I-IOCNT, 7 IOs on dataset ASS0
%ADAREP-I-TERMINATED,   02-MAY-2024 18:10:26, elapsed time: 00:00:00
```

ADABCK

```
ADABCK SUMMARY
```

Example:

The following example shows the encryption settings of an encrypted database within a backup copy (BCK001).

```
adabck summary
2024-05-02 18:25:55 %ADABCK-I-STARTED,     02-MAY-2024 18:25:55, Version 7.3.0.0 ↩
(Linux 64Bit)
Database dumped on  2-MAY-2024 18:25:50

Database 100, Database-100

Summary of Database 100           2-MAY-2024 18:25:55

DATABASE NAME                    Database-100
DATABASE ID                        100
MAXIMUM FILE NUMBER LOADED           3
SYSTEM FILES                         1 (CHK),     2 (SEC),     3 (USR)
ACTUAL FILES LOADED                  3
CURRENT PLOG NUMBER                  0
CURRENT CLOG NUMBER                  0
ENCRYPTION                       AES_256_XTS
KMSTARGET                        FILE
KEKNAME                          ↩
1714666215B382A0FD089977264681686464A029C3FA4497EE1EF150C0217295


Container  Device      Extents in Blocks    Number of      Block    Total Size
  File     Type         from          to     Blocks        Size     (Megabytes)
--------------------------------------------------------------------------------

 ASS01     file            1      25,600       25,600      4,096         100.00

 DATA1     file            1       3,200        3,200     32,768         100.00

 WORK1     file            1       6,400        6,400     16,384         100.00

--------------------------------------------------------------------------------
                                                                         300.00
                                                                         ======



2024-05-02 18:25:55 %ADABCK-I-IOCNT, 1 IOs on dataset BCK001
2024-05-02 18:25:55 %ADABCK-I-TERMINATED,   02-MAY-2024 18:25:55, elapsed time: ↩
00:00:00
```

Both reports show the current encryption algorithm, the key management system and the name of the key encryption key.

Database Backup and Restore

The Adabas backup copy of an encrypted database contains the encryption settings such as the encryption algorithm and key management system of the encrypted database. This data itself is not encrypted, while all database blocks within the backup copy are encrypted with the encryption settings of the database. The KMS containing the KEK of the database must be available to restore this backup.

Restore as new database

To restore the Adabas backup copy of an encrypted database as a new database you can

- omit the control parameter ENCRYPTION. The encryption settings are taken from the Adabas backup copy.
- specify the control parameter ENCRYPTION. The encryption settings are taken from the parameter.

To restore an Adabas backup copy of an encrypted database as a new non-encrypted database, set the control parameter ENCRYPTION=NO.

Restore to an existing database

To restore an Adabas backup copy to an existing database, whether encrypted or not encrypted, omit the control parameter ENCRYPTION. The data is restored according to the target database's encryption settings.

Database Access

Access to an encrypted database is transparent to the Adabas utilities and to the Adabas direct call interface. The data are encrypted on write access, and decrypted on read access, according to the database's encryption settings.

Adabas Utilities

For all ADABAS utilities when reading encrypted data, access is transparent. The goal in implementation was not to change utility syntax. This could not always be achieved and will be explained in more detailed in the corresponding utility pages. The current syntax changes include:

- ADAFRM can accept a dbid when using tempsize or sortsize.
- ADACMP can accept a dbid when compressing output from ADADCU.

6 Adabas OIDC Authentication

- Overview 78
- Authentication Flow 79
- Configuration 82
- Getting Started 83
- Error Handling 85

Adabas OIDC Authentication provides Single Sign-On (SSO) capabilities for Adabas command-line utilities using industry standard OpenID Connect protocol. It enables users to authenticate with a central Identity Provider rather than relying on operating system credentials.

It implements the OAuth 2.0 Device Authorization Grant to provide secure authentication for command-line utilities running in terminal environments without graphical user interfaces.

This feature includes the following functionality:

Device Authorization Flow
: Users authenticate via a browser on any device by visiting a URL and entering a short code displayed in the terminal.

Token-Based Authentication
: Cryptographically signed JWT tokens prove user identity without transmitting passwords.

Token Caching and Refresh
: Tokens are securely cached locally and automatically refreshed, minimizing re-authentication prompts.

Multi-IdP Support
: Compatible with any OpenID Connect compliant Identity Provider including Keycloak, Google Cloud Platform, and Microsoft Entra ID (Azure AD).

Overview

- What is OIDC Authentication?
- Why OIDC for Adabas Utilities?

What is OIDC Authentication?

OpenID Connect (OIDC) is an authentication layer built on top of OAuth 2.0. It provides a standardized way to verify user identity and obtain basic profile information. Unlike traditional username/password authentication, OIDC delegates authentication to a trusted Identity Provider (IdP) that issues cryptographically signed tokens.

Why OIDC for Adabas Utilities?

Traditional Adabas utilities identify users via operating system calls. While this works well in conventional on-premises deployments, modern environments present challenges. For example, in container-based environments, utilities may run under a generic service account.

OIDC provides a standardized, secure, and auditable way to establish user identity regardless of the execution environment.

It offers the following benefits:

Centralized identity management
User accounts are managed in one place.

Enhanced security
Strong authentication methods (MFA, biometrics) are enforced by the IdP.

Auditable access
All authentication events are logged at the IdP for compliance.

Seamless user experience
Authenticate once, access all connected databases.

Revocation capability
Disabling a user at the IdP immediately revokes access everywhere.

Standards based
OpenID Connect is an industry standard supported by all major IdPs.

Authentication Flow

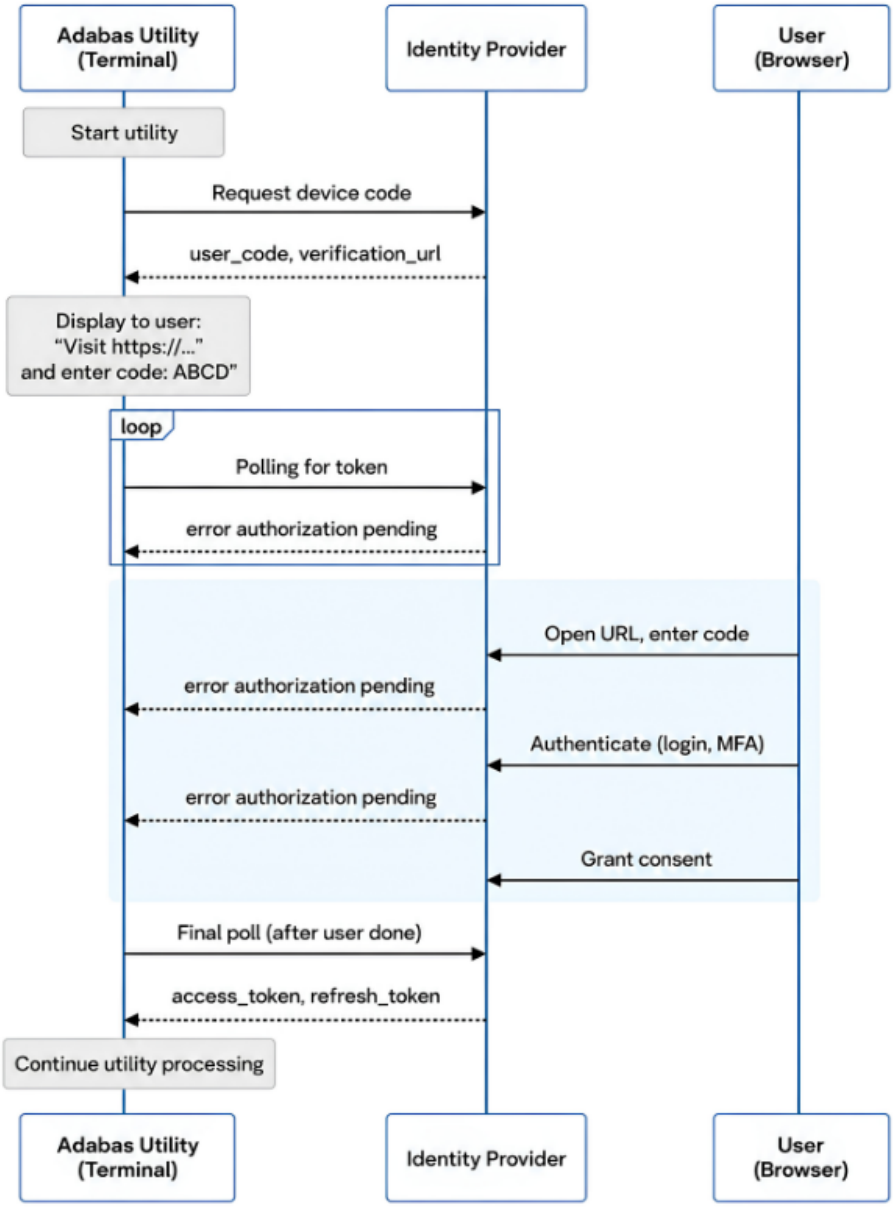
- Device Authorization Grant
- User Experience
- Token Types
- Token Caching

Device Authorization Grant

The Device Authorization Grant is specifically designed for devices with limited input capabilities, like running CLI tools in terminals or SSH sessions where opening a browser window is not possible.

Adabas utilities read the database's OIDC configuration when accessing a database with OIDC enabled. The utility connects to the configured Identity Provider and requests a user code. The utility displays a verification URL and user code to the user. The user opens the URL in a browser on any device, enters the user code, and authenticates with the IdP.

The utility polls the IdP's token endpoint until the user completes authentication. Upon successful authentication and consent grant, the IdP issues ID, access, and refresh tokens. The utility continues processing with the principal claim configured and returned by the IdP.



User Experience

When authentication is required, the utility displays instructions:

```
================================================================================
ADABAS AUTHENTICATION REQUIRED
================================================================================

To sign in, open a browser and visit:
https://login.example.com/device?user_code=ABCD-EFGH

Or visit: https://login.example.com/device
And enter the code: ABCD-EFGH

Waiting for authentication (expires in 600 seconds)...
================================================================================
```

If the libqrencode library is installed, a QR code is also displayed for easy scanning with a mobile device.

Token Types

Token	Purpose	Typical Lifetime
Access Token	Grants access to protected resources; contains user claims	5–60 minutes
Refresh Token	Obtains new access tokens without re-authentication	Hours to days
ID Token	Contains identity claims about the authenticated user	Same as access token

Token Caching

Tokens are cached locally to avoid re-authentication on every utility invocation:

- Linux: `$HOME/.adabas/oidc/tokens.json`
- Windows: `%USERPROFILE%\.adabas\oidc\tokens.json`

The token directory path and file are created with restricted permissions (directory mode 0700 and file 0600 on Linux) to prevent unauthorized access.

When a utility starts:

1. It checks for a cached token.
2. If valid, the cached token is used.
3. If expired, it attempts to refresh using the refresh token.
4. If all attempts have failed, the device authorization flow is initiated.

Configuration

OIDC authentication is configured per database in the DBnnn.INI file. The configuration is stored in the [OIDC] section within the [DB_PARAMETER] section.

- Configuration Parameters
- Example Configuration

Configuration Parameters

Parameter	Required	Default	Description
ENABLED	No	NO	Enable or disable OIDC authentication (YES or NO)
ISSUER	Yes	—	Identity Provider issuer URL (used for endpoint discovery)
CLIENT_ID	Yes	—	OAuth 2.0 client identifier registered with the IdP
CLIENT_SECRET	No	—	OAuth 2.0 client secret (required by some IdPs like Google)
SCOPES	No	openid profile email	Space-separated OAuth scopes to request
PRINCIPAL_CLAIM	No	preferred_username	JWT claim to use as the user principal (sub, preferred_username, or email)

Example Configuration

A OIDC configuration for Microsoft Entra ID could look like this:

```
[DB_PARAMETER]

   [OIDC]
      ENABLED = YES
      ISSUER = https://login.microsoftonline.com/your-tenant-id/v2.0
      CLIENT_ID = your-application-client-id
      SCOPES = "openid profile email offline_access"
      PRINCIPAL_CLAIM = preferred_username
   [OIDC-END]

[DB_PARAMETER-END]
```

Getting Started

- Prerequisites
- Step 1: Register Application with Identity Provider
- Step 2: Configure the Database
- Step 3: Validate Configuration
- Step 4: Test Authentication

Prerequisites

Before enabling OIDC authentication, ensure:

Identity Provider Setup
: You have registered an OAuth 2.0 application with your IdP configured for the Device Authorization Grant flow.

Network Connectivity
: The machine running Adabas utilities can reach the IdP's endpoints via HTTPS.

Dependencies
: The libcurl library is installed (required for HTTP communication).

Step 1: Register Application with Identity Provider

Register your application with your chosen Identity Provider and obtain:

- *Client ID:* The application identifier
- *Client Secret:* (If required by your IdP)
- *Issuer URL:* The IdP's base URL

Configure the application to support the Device Authorization Grant (also called "device code flow" or "TV and Limited Input Device" application type).

Step 2: Configure the Database

Use the ADAINI utility to add the OIDC configuration:

```
# Enable OIDC authentication
adaini dbid=nnn add topic=DB_PARAMETER topic=OIDC item=ENABLED=YES

 # Set the IdP issuer URL
 adaini dbid=nnn add topic=DB_PARAMETER topic=OIDC ↩
item=ISSUER=https://your-idp.example.com

 # Set the client ID
 adaini dbid=nnn add topic=DB_PARAMETER topic=OIDC item=CLIENT_ID=your-client-id

 # (Optional) Set client secret if required
 adaini dbid=nnn add topic=DB_PARAMETER topic=OIDC ↩
item=CLIENT_SECRET=your-client-secret

# (Optional) Set requested scopes
adaini dbid=nnn add topic=DB_PARAMETER topic=OIDC item=SCOPES="openid profile email"

# (Optional) Set principal claim
adaini dbid=nnn add topic=DB_PARAMETER topic=OIDC ↩
item=PRINCIPAL_CLAIM=preferred_username
```

Step 3: Validate Configuration

Display the current OIDC configuration:

```
adaini dbid=nnn show topic=DB_PARAMETER topic=OIDC

ENABLED=YES
ISSUER=https://your-idp.example.com
CLIENT_ID=your-client-id
SCOPES=openid profile email
PRINCIPAL_CLAIM=preferred_username
```

Step 4: Test Authentication

Run for example the Adabas utility ADAREP that requires authentication. If OIDC is enabled and no valid cached token exists, you will see the authentication prompt:

```
adarep dbid=nnn summary

==============================================================================
ADABAS AUTHENTICATION REQUIRED
==============================================================================

To sign in, open a browser and visit:
https://your-idp.example.com/device?user_code=ABCD-EFGH

Or visit: https://your-idp.example.com/device
And enter the code: ABCD-EFGH
```

```
Waiting for authentication (expires in 600 seconds)...
==============================================================================
```

Complete authentication in your browser, and the utility will continue execution.

Error Handling

On any OIDC related error the utility will report a OIDCFAIL error message containing further information.

Troubleshooting

- Device authorization timed out
- Network/HTTP error
- Configuration error
- Token file path permissions too open
- Interactive authentication required but no terminal available
- User denied authorization

Device authorization timed out

- Authentication not done within given time frame: Check the configured expiration time in the Identity Provider settings
- Wrong user code: Ensure to enter the code as shown
- IdP unreachable: Ensure the user's browser can reach the Identity Provider

Network/HTTP error

- Firewall blocking: Ensure outbound HTTPS (port 443) is allowed to the IdP
- Proxy configutation: Set HTTP_PROXY / HTTPS_PROXY environment variables if required
- SSL certificate issues: Ensure the system trusts the IdP's SSL certificate

Configuration error

- Missing required parameters: Verify ISSUER and CLIENT_ID are configured
- Invalid issuer URL: Ensure that the issuer URL does not have a trailing slash
- Typo in configuration: Double check all parameter values

Token file path permissions too open

The token directory path and/or the token file permissions are too open. Ensure that the file path has restrictive permissions.

```
chmod 700 ~/.adabas
chmod 700 ~/.adabas/oidc
chmod 600 ~/.adabas/oidc/tokens.json
```

Interactive authentication required but no terminal available

Authentication is required but the utility output is not connected to an interactive terminal. This occurs when the utility output is redirected to a file, piped to another command, or run in a non-interactive context (e.g., cron job, batch script, or background process).

Run the utility from an interactive command line first to complete the authentication process. This ensures the user has a valid token and/or refresh token cached locally. After successful authentication, repeat the non-interactive operation—it will use the cached token without requiring terminal interaction.

User denied authorization

- User explicitly denied consent: When prompted in the browser, the user clicked "Deny" or "Cancel" instead of granting consent. Re-initiate authentication and grant consent when prompted.
- Wrong user authenticated: Verify that the correct user ID is used for authentication at the IdP URL. If another user received the authentication request (e.g., the URL was shared or opened on a different device), they may have denied the request. Ensure only the intended user completes the authentication flow.
- Consent already denied: Some IdPs remember denied consent. Check the IdP's user settings to revoke previous denial decisions, or contact your IdP administrator.

7 Ciphering

Important: The implementation of the ciphering feature is different to that available in Adabas for mainframes. The cipher code in Adabas for Linux and Windows is static and is not provided in Additions 4. **Adabas Encryption for Linux** is highly recommended as a state-of-the-art replacement.

Notes:

1. Ciphering is no longer recommended for safe implementations.
2. Ciphering does not fulfill modern encryption standards and security regulations.
3. Ciphering is maintained for historic and maintenance reasons only.

The purpose of the ciphering in Adabas for Linux and Windows is to prevent unauthorized analysis of Adabas container files; e.g. via file dumps, editors, etc.

Unlike ciphering on the mainframe, it does not prohibit unauthorized access to the data; as both database utilities and database applications can access the data without the cipher code.

Adabas can cipher the data that it stores in container files. This, however, only applies to the data records that are stored in the Data storage, but not the values stored in the inverted lists on the Associator.

Ciphering prevents the unauthorized analysis of Adabas container files. If ciphering is enabled (see below), data records are ciphered when they are stored in a database by either the Adabas nucleus or by the mass update utility ADAMUP. The data records are then deciphered when they are requested by a user or application. This means that the ciphering is completely transparent to the user or application.

Ciphering can be enabled for individual Adabas files. This is done when defining the file with ADAFDU by setting the `CIPHER/NOCIPHER` option. The ciphering process uses internal parameters in order to achieve a maximum level of security. In some systems, identical fields and records

present a possible security risk: if an unauthorized user can decipher one, the other can also be deciphered. The Adabas ciphering process, however, treats identical fields and records as follows:

- Two identical fields within one record will be ciphered differently;
- Two identical records within one Adabas file will be ciphered differently;
- Two Adabas files with identical contents will be ciphered differently.

The following example demonstrates this on the basis of two fields in a record which both contain the value `TEST' (representations are hexadecimal):

```
Record 1  Unciphered=0x54455354  Ciphered=0xDD022537
Record 2  Unciphered=0x54455354  Ciphered=0x55EF0A51
```

Note: The ciphered values shown above are just examples, and do not represent the actual ciphering mechanisms used.

The Adabas ciphering mechanism is characterized by the following features and restrictions:

- System files (checkpoint and security) cannot be ciphered.
- ADAM key files cannot be ciphered.
- The output files produced by the utilities ADACMP (compression) and ADAULD (unload) are not ciphered.
- The data saved on files produced by the backup utility ADABCK, and the EXPORT files produced by the export utility ADAORD are ciphered.
- The restart and recovery records that are written to the WORK and PLOG files are ciphered.
- The output produced by the FILE function of the report utility ADAREP contains information about file ciphering.

8 Security Considerations

- Using the Linux Group Concept 90
- Securing Administration of Adabas Role-Based Security 91
- Preventing Loss of Administration Privileges 92
- Securing PUBLIC Access Privileges 93
- Securing the Audit Trail Log File 93

This section describes means or actions that can or should be taken to secure ("harden") the database.

Using the Linux Group Concept

If the Adabas users belong to different Linux groups, you can restrict the Adabas access to databases assigned to this group.

Note: This feature is only available for Linux, not for Windows platforms.

Example:

Assume you have two Linux groups called Production and Test. There are users belonging to the group Production, who should have access only to the production databases, and there are users belonging to the group Test, who should have access only to the test databases. Assume you have the following users for starting the database:

- dbaprod belongs to the group Production and should start the production databases
- dbatest belongs to the group Test and should start the test databases

The following is necessary to restrict the Adabas access to users of the group to which the databases belong:

- You must use two different NET_WORK_IDs, even if you are not using Net-Work. Because Adabas does not know if a Net-Work server will be started later, Adabas creates a shared memory section common with Net-Work, a GDT (global database table). The permission for GDT access is restricted to the group to which the Adabas nucleus belongs. Therefore, starting a nucleus fails if the same GDT is accessed as used by another nucleus belonging to a different group.

  You can use different GDTs if you start the nucleus with a different NET_WORK_ID because a separate GDT is created for each NET_WORK_ID. NET_WORK_ID is an environment variable, which must be set when the Adabas nucleus is started - two NET_WORK_IDs are considered to be equal if the first character is equal. If the environment variable NET_WORK_ID is not set, an empty NET_WORK_ID is used.

  In this example, you could start the production databases after setting NET_WORK_ID to P, and the test databases after setting NET_WORK_ID to T.
- The nucleus must be started with the parameter ADABAS_ACCESS=GROUP. Assume that you start in this example the Production databases with ADABAS_ACCESS=GROUP, but the Test databases with ADABAS=ACCESS=ALL (or without the parameter ADABAS_ACCESS). Then only the Production users can access the production databases, but all users can access the test databases.

Note: If you are using Net-Work, it is also necessary to start different Net-Work servers for different groups. You must take care to ensure that it is not possible for users to access databases via Net-Work for which they have no permissions.

Securing Administration of Adabas Role-Based Security

This section describes how to secure access to the ADARBA utility, which provides the functionality to administer the security definitions.

By default, all users with the **PUBLIC** role have access to and can execute the ADARBA utility.

The example below shows how access to ADARBA can be restricted by:

- Creating a *security administrator* role;
- Granting the role permission to execute the utility;
- Creating a user credential that is to be assigned the *security administrator* role;
- Granting the user *security administrator* role; and
- Revoking the permission to execute the utility from the role PUBLIC.

```
> adarba
> adarba: dbid=nnn

> adarba: create,role= security_administrator
> adarba: grant,operation=ada.uti.rba,to,role= security_administrator

> adarba: create,user=dbasec_userid
> adarba: grant,role= security_administrator,to,user=dbasec_userid

> adarba: revoke,operation=ada.uti.rba,from,role=PUBLIC
```

In this example, the following security definitions are created or modified:

- Created:
  - Role: *security_administrator*
  - User Credentials: *dbasec_userid*
  - Permission to Operation: **ada.uti.rba**
- Modified:
  - Role: **PUBLIC**
  - Permission to Operation: **ada.uti.rba**

Notes:

1. The role name and the user credentials are site-specific.
2. The credentials used for authorization must also be defined for authentication.

Preventing Loss of Administration Privileges

Lock Out Scenario

A 'lock out' scenario is possible. You must take appropriate action to prevent the 'loss of administration permissions'.

There is no recovery from a loss of administration privilege or a lock out scenario. This is a scenario in which no individual has access to the administration utilities, and the administration privilege for the security definitions has been lost.

The situation can be avoided by ensuring access to the following administration utilities.

- ADABCK
- ADADBM
- ADARBA

Recommended procedures to prevent a lock out scenario are to:

- Create more than one administration role;
- Assign access privileges to these administration roles;
- Assign the administration roles to more than one user credential.

In order to prevent an accidental loss of privileges, it is important to understand the processing logic behind the DROP function in ADARBA.

Drop Role Function

The DROP ROLE function in ADARBA deletes the following security definitions:

- All references to the ROLE definition are deleted:
  - The permissions granted to the ROLE are revoked;
  - The ROLE assignments to all USERS are revoked.
- The ROLE definition is deleted.

The function performs a cascading delete without issuing a warning message.

A loss of privileges scenario can occur if the administrator role is dropped, and no other role has access to the administration utilities.

Securing PUBLIC Access Privileges

You should review and restrict access privileges assigned to the role **PUBLIC**. This includes the roles assigned to the **PUBLIC** user, and the privileges assigned to the **PUBLIC** role.

PUBLIC access is provided when the RBAC system file has not been loaded, even though security is enabled.

You should take the following actions to secure access privileges assigned to **PUBLIC**:

- Ensure that the RBAC system file is loaded;
- Review the roles that are assigned to the **PUBLIC** user;
- Review the privileges that are granted to the **PUBLIC** role.

Securing the Audit Trail Log File

This section describes how to secure the audit trail log file used by the Authorization for Adabas utilities.

File	Description
`adaaudit.log`	Audit Trail log file for Authorization for Utilities

To secure the audit trail log file, please ensure the following:

READ/WRITE-ACCESS

All users, which execute an Adabas utility, must be able to write to the audit trail log file.

9 SSL Trusted Relationship for Natural

- Restrictions 96
- Adabas Operation 96
- Client Configuration 97
- Adabas Configuration 97

The main goal is to use the Adabas RBAC facility from Natural as client. Natural is at the moment not able to provide the user and password through the Adabas Client (using the `lnk_set_uid_pw()` function). The *SSL Trusted Relationship* should overcome this restriction. Neither Natural itself, nor any customer application should need to be changed.

The idea behind the *SSL Trusted Relationship* is that a client provides certificates over an SSL connection which are validated in Adabas. If the client provides certificates and they are successfully validated by Adabas, Adabas would assume that the client is allowed to access Adabas without presenting a password. The user id for the RBAC facility in Adabas will be taken from the so called *Adabas Session* or *Adabas ID*. The client (Natural) must make sure that the session (especially the user id) will be set up correctly.

Restrictions

We restrict the mechanism of trusted relationship on Natural as client. Only if these three conditions are fulfilled, Adabas trusts the incoming request:

1. Client uses ADATCPS (SSL) for the communication with Adabas
2. Client provides valid certificates that can be validated by Adabas
3. The client is Natural

The client is responsible to provide the user that will be authorized with the RBAC facility of Adabas. The user is taken from the Adabas Session (Adabas ID).

Adabas Operation

Adabas will check the validity of certificates at the connection level. As soon as a client connects via ADATCPS, Adabas will check whether the client provides client certificates. If so, they are validated in any case, regardless the setting of the SSLVERIFY nucleus parameter. A failed verification might not be an error, for example, if the SSLVERIFY is not set, clients need not to provide certificates, but credentials. If the SSLVERIFY option is set and the client certificate is invalid, then the connection will be rejected in any case. Adabas can be accessed in the traditional way in parallel. If using a secured database (RBAC), the clients must provide the user id and password (via the Adabas Client Interface) or use the user exit 21.

Client Configuration

On the client side ADATCP must be configured to use SSL and SSL certificates. This is done in normal cases in the $ACLDIR/config/dbmapping.txt file.

Example:

```
<dbid> = ↩
adatcps://<host>:<sslport>?cert_file=<path>/client_cert.pem&key_file=<path>/client_key.pem
```

Adabas Configuration

The Adabas nucleus must be configured to use SSL and SSL certificates. In any case, Adabas will not start without server certificates when SSL should be used. That is the minimum requirement.

To enable Adabas to accept trusted Natural users, the nucleus must be configured with the parameter:

```
SSLTRUST = NATURAL
```

Example:

```
[NUCPARMS]
ADATCP
PORTNUMBER = 0
SSLCAFILE = /atc/certs/ca_chain.pem
SSLCERTFILE = /etc/certs/adabas_cert.pem
SSLKEYFILE = /etc/certs/adabas_key.pem
SSLPORTNUMBER = 56220
SSLTRUST = NATURAL
SSLVERRIFY = 0
...
[NUCPARMS-END]
```

Setting the `PORTNUMBER` parameter to ‘0’ disables the classic TCP/IP port and only SSL can be used.

10 GDPR Compliance

- Adabas Role-Based Security 100
- Adabas Audit Trail 101
- Adabas Command Log 101
- Adabas Log File 102

Adabas on Linux and Windows stores the following personal data in log and audit files:

- Hostname of the machine
- User Identification (Local system credentials)

Therefore, so that your applications can be GDPR compliant, it is important for the Adabas administrator to be aware of the following files where personal data may be stored.

- Adabas Role-based Security
- Adabas Audit Trail
- Adabas Command Log
- Adabas Log file

Basic Actions to be taken

It is the responsibility of the database administrator to configure and delete files containing personal data. Adabas does not provide functionality to automatically delete personal data.

To ensure that personal data is not stored, the following is recommended:

- Disable logging features (when possible)
- Define a schedule and procedure to delete log files

The location and configuration of the database features storing personal data are described below.

Adabas Role-Based Security

When the Adabas Role-based Security feature is enabled, the following personal data is stored in the RBAC system file as a security definition: User Identification.

This information is required and is used to determine the access privileges of a user.

Use the utility ADARBA to manage the security definitions.

Adabas Audit Trail

When the Adabas audit trail feature is enabled, the following personal data is logged:

- User Identification
- Hostname of the Adabas client
- Process Identification
- Timestamp of Access Request
- ET data
- Name of the access operation and the object of the operation

The location and configuration of the audit trail is feature-specific. For more information, see the ***Configuration*** section in *Adabas Security Features* > *Adabas Role-Based Security (ADARBA)*.

The Audit Trail for Authorization for the Adabas Direct Call Interface is automatically enabled with the feature and cannot be disabled.

Adabas Command Log

When the Adabas Command Log feature is enabled, the following personal data is logged:

- User Identification
- Hostname of the Adabas client
- Process Identification
- Timestamp
- Adabas Command

When enabled, a command log file is created in the database directory. The file is identified by the value of the environment variable `NUCCLG`. The file can have multiple extends.

The Adabas Command Log feature can be enabled or disabled via the `LOGGING` control parameter of either utility:

- ADANUC
- ADAOPR

Please refer to either the *ADANUC* or *ADAOPR* section of the *Adabas Utilities*, for further information on the `LOGGING` control parameter.

Adabas Log File

When the Adabas log feature is enabled, the following personal data is logged:

- User Identification
- Process Identification
- Timestamp
- Name of the Adabas Utility with the requested function

The configuration file *ADABAS.INI* contains the location and configuration of the Adabas log file.

By default, the file name is *$ADADATADIR\etc\ADABAS.LOG*. The logging functionality is configured using the topic NODE_PARAMETER and its subtopics:

Subtopic	Description
ANALYSER	Enable the Adabas Extended Operations (AEO) feature
LOGGING	The Adabas log file is configured with the items: ■ ACTION - enables logging and ■ LOG_PATH - defines the file location
ARCHIVE_LOGFILE	Enables the archiving of the Adabas log file
ALERT	The Adabas log file entry can be processed by an alert routine. This is enabled and configured with the items: ■ ACTION - enables alert processing and ■ ACTION_ROUTINE - defines the alert routine

For more information on the location and configuration of the Adabas log file, see the *ADABAS.INI* section in *Adabas Extended Operations* > *Configuration Files*.