Adabas for Linux, UNIX and Windows

Adabas Security Facilities

Version 6.6

October 2017

ADABAS & NATURAL

This document applies to Adabas for Linux, UNIX and Windows Version 6.6 and all subsequent releases.

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

Document ID: ADAOS-SECFAC-66-20190516

Table of Contents

Adabas Security Features

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

The following topics are covered:

- *Adabas Authentication*
- *Authorization for Adabas Utilities*
- *Adabas Password Security (ADASCR)*
- *Ciphering*
- *Security Considerations*
- *SSXLoginModule Configuration Templates*

1 About this Documentation

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

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

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

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

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

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

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

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

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

Data Protection

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

2 Adabas Authentication

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

The Adabas server checks the credentials against security definitions in an external authentication system like LDAP, Active Directory, operating system, or internal repository. If the credentials match, the user is provided access to the database. If the credentials are at variance, authentication fails and database access is denied.

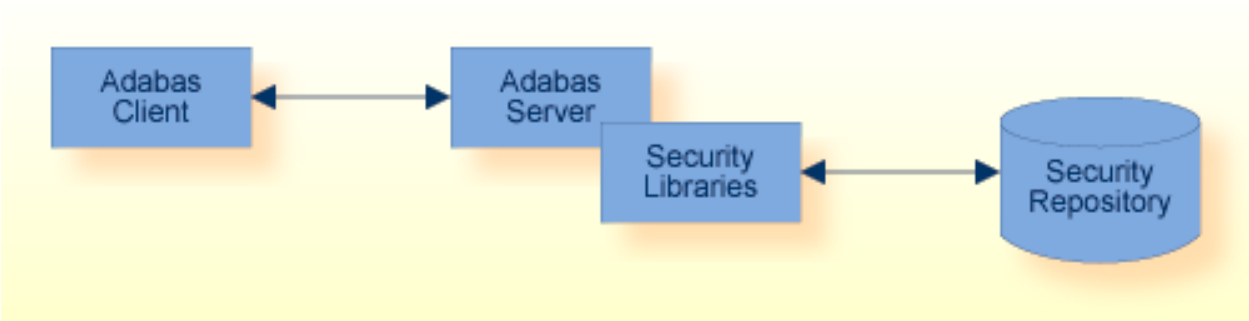
The audit trail logs both successful and failed attempts to access the database.

In the current version:

- The security infrastructure is used to provide access to external security systems.
- Supported credentials are: domain, User ID and password.
- Authentication implies “access or no access” to the database.
- Database utilities do not perform authentication checks.
- The audit trail is written to the ADABAS logging file.

How does Authentication work?

Architecture



Authentication Process

- Application provides credentials: user ID and password.
- Adabas server requests validation of the credentials via the security infrastructure.
- Security infrastructure validates the credentials against the security repository.

Security Infrastructure

The Adabas server uses the predefined login module `SSXLoginModule` from the security infrastructure to authenticate user credentials.

Authentication is done against LDAP, Active Directory, internal repository or operating system.

The `SSXLoginModule` is part of the security libraries delivered with the Software AG Installer. The installation of this component is mandatory for the usage of this feature.

Initial Setup

When Installing Adabas

The Infrastructure Security Libraries are required by the Adabas Server. They contain the necessary functionality to implement the Adabas Authentication feature.

The Infrastructure Security Libraries are installed with the Software AG Installer and are, by default, preselected in the GUI installation.

Enable Authentication Security Facility

1 Use the ADAINI utility

- To define the Extended Operations database log.
- To enable the Extended Operations Analyser.
- To enable the Audit Trail feature.

  Note: This is database-specific and must be performed for each database.

- To configure the `SSXLoginModule` usage.
  - Authorization Type specific options
  - Logging / Diagnostics options

2 Use the *SECURITY* function of the ADADBM utility to enable the Authentication Security Facility.

3 Start the database.

Enable Legacy Applications to use Authentication

The Adabas nucleus user exit 21 is intended to assist when modifying legacy applications to use the authentication feature. For further details, please refer to the section *Modifying Legacy Applications to use Authentication*.

1 Customize the sample Adabas Nucleus User Exit 21 to meet the site-specific requirements.

2 Build the site-specific nucleus user exit.

3 Modify the environment settings for the nucleus user exit.

4 Start the database with the USEREXITS parameter.

Configuration

This section describes how to configure the authentication and audit trail features.

- Performance Considerations

Audit Trail

The Audit Trail is written to the database log file and thus requires that the Event Analyser (AEO analyser or simply the analyser) be enabled. The analyser is enabled via the `NODE_PARAMETER` in the *ADABAS.INI* configuration file and thus enables the analyser in all databases within the node.

The Audit Trail feature is database-specific and is configured via the *DBnnn.INI* configuration files. The audit trail filter receives security events and writes them to the database log file, where they can be analyzed later. It is possible to either log all access attempts or only security violations.

Configuration File	Section	Subtopic	Item
`ADABAS.INI`	`NODE_PARAMETER`	`ANALYSER`	`ACTION`
		`LOGGING`	`ACTION` `LOG_FILE`
`DB`*`nnn`*`.INI`	`DB_PARAMETER`	`AUDIT_TRAIL`	`ACTION` `FILTER`

For further information on the configuration files and syntax, see the descriptions of *ADABAS.INI* and *DBnnn.INI* in the *Extended Operation* documentation.

Authentication

The authentication checking is database-specific and is configured via the `SSXLoginModule` Options. These options are to be entered in the *DBnnn.INI*.

Configuration File	Section	Subtopic	Item
`DB`*`nnn`*`.INI`	`DB_PARAMETER`	`SSX_CONFIGURATION`	*SSX Configuration Options*

For further information on the configuration files and syntax, see the descriptions of *DBnnn.INI* in the *Extended Operation* documentation.

The `SSXLoginModule` supports multiple authorization types (or methods), for example:

AuthType	Description
TEXT	Authentication is performed using the security definitions, which are located in an Software AG internal user repository.
LDAP	Authentication is performed using the security definitions in LDAP.
ADSI	Authentication is performed using the security definitions in an Active Directory.
OS	Authentication is performed using the security definitions from the operating system.

In the section *SSXLoginModule Configuration Templates* you can find examples and example templates for the different authorization types. These templates are not complete as some of the settings are customer-specific and must be modified were necessary.

Performance Considerations

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

Feature	Option	Explanation
Audit Trail	`LOG_FILE`	A large volume of entries being written to the database log file may be detrimental to performance, when multiple databases are competing for the same resource; e.g. accessing the same database log file.
Audit Trail	`FILTER`	A large number of user-sessions will result in a large number of security entries being written to the database log file. The size of the database log file increases rapidly, with the numbers of user-sessions. Default Value: `FILTER = ALL`. Recommended Value: `FILTER = REJECT`.
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`.

Enable Security Mode

The security mode is enabled via the *SECURITY* function of the ADADBM utility.

The following database Security modes are available:

- Security mode ACTIVE activates the security functionality and only authenticated users get access to the database. The mode ACTIVE can neither be changed nor disabled.
- Security mode WARN simulates the security functionality and, if defined, writes warnings to the database log file in case of a security violation, but does not reject access to the database. The mode WARN can only be changed to ACTIVE.

Important: The Database Security mode can be either set to WARN or ACTIVE. Once enabled, the security mode cannot be disabled.

Tip: It is recommended that you create a backup for the database for recovery purposes, prior to activating the security mode.

Utilities Required for Adabas Authentication

The utilities required to configure and administer the authentication security feature are:

- ADADBM - Enable Security Mode
- ADAREP - Query Database Security Mode
- ADAINI - Configure Security Features

ADADBM - Enable Security Mode

Use the *SECURITY* function of the ADADBM utility to activate the authentication and auditing features.

Note: The SECURITY function requires that the database that is to be secured is offline.

```
adadbm: dbid=nnn
%ADADBM-I-DBOFF, database nnn accessed offline
adadbm: security=active
%ADADBM-I-FUNC, function SECURITY executed
```

ADAREP - Query Database Security Mode

Use the *SUMMARY* function of the ADAREP utility to display the database security mode settings.

Note: The security mode setting is only displayed, when the feature has been activated. It is not displayed, when the feature is not activated.

ADAINI - Configure Security Features

Use the ADAINI utility to set and modify the configuration of the security features. The following examples show how ADAINI can be used to configure the security features.

Example 1: Activate Extended Operations Logging

```
ADABAS.INI
> adaini   add topic=NODE_PARAMETER topic=LOGGING  ↩
item=LOG_FILE=path_and_name_adabas_log_file
> adaini   add topic=NODE_PARAMETER topic=LOGGING  item=ACTION=YES
> adaini   add topic=NODE_PARAMETER topic=ANALYSER item=ACTION=YES
```

Example 2: Activate Audit Trail

```
DB<nnn>.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
```

Example 3: Configure Authorization Type TEXT

```
DB<nnn>.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=path_and_name_ssxuser_file
```

Example 4: Configure Security Infrastructure Logging

```
DBnnn.INI
> adaini dbid=<nnn> add topic=DB_PARAMETER topic=SSX_CONFIGURATION ↩
item=nativeLogFile=path_and_name_of_ssxlog_file
> adaini dbid=<nnn> add topic=DB_PARAMETER topic=SSX_CONFIGURATION item=nativeLogLevel=6
```

Example 5: Display Configuration Settings

```
ADABAS.INI
> adaini   show topic=NODE_PARAMETER

DB<nnn>.INI
> adaini dbid=<nnn> show topic=DB_PARAMETER
```

Application Development

Developing Applications to use Authentication

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:

Steps	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 on the above Adabas client functions can be found in the section *Calling Adabas with Authentication* in the section *Calling Adabas* of the *Command Reference* documentation

Modifying Legacy Applications to use Authentication

Without modification, legacy applications will receive 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 further details, see the *Nucleus User Exit 21* in the section *User Exits and Hyperexits*.

SSXLoginModule Configuration Templates

The following configuration options must be set in the configuration file *DBnnn.INI* in the subtopic `SSX_CONFIGURATION` of the `DB_PARAMETER` section.

```
DBnnn.INI

[DB_PARAMETER]
  [SSX_CONFIGURATION]
    <option>=<value>
  [SSX_CONFIGURATION-END]
[DB_PARAMETER-END]
```

In ***SSXLoginModule Configuration Templates*** you can find additional configuration templates you can use with the different authorization types (OS, TEXT, LDAP and ADSI).

3 Authorization for Adabas Utilities

Adabas uses the concept of Role-Based Access Control (RBAC) to implement Authorization for Adabas Utilities.

You can restrict the scope of your security definitions to a database (mode ADABAS) or to the machine with all installed databases (mode INI).

For more detailed information, please refer to ***Authorization for Adabas Utilities (Mode ADABAS)*** or ***Authorization for Adabas Utilities (Mode INI)***.

Adabas Role-Based Access Control

Authorization provides a means of restricting the usage of Adabas utilities on databases by assigning users a role which represents selective access privileges

The user is identified by the credentials, which were used to access the local machine.

The Adabas utilities check the provided credentials against the security definitions in the security repository. The security repository depends on the mode that is chosen during configuration. For further details, please refer to *Configuration*.

If the credentials have been assigned a role, the access privileges for the requested operation on the database are determined. If the access privileges are sufficient the user is allowed to execute the utility on the database. If the access privileges are insufficient or lacking, authorization fails and the utility usage is denied.

The security repository stores granted privileges, denial of specific operations is not supported.

The audit trail logs both successful and failed attempts to use an Adabas utility.

In the current version:

- Adabas utilities do not perform authentication checks.
- Supported credentials are the local system credentials.
- Authorization implies "usage or no usage" of the utility and the database on which the operation was to be performed.
- The audit trail is written to the log file, which can be configured.
- The authorization feature is only available in the following Adabas utilities: ADABCK, ADADBM, ADAELA,ADAFDU, ADAFRM,ADAOPR, ADAORD, ADARBA,ADAREC, ADAREP, ADASCR and ADAULD.

  - Data Model
  - Architecture
  - Authentication
  - Authorization Process

Data Model

Adabas Role-Based Access Control restricts database access to users based on their roles

Permissions to perform specific Adabas utilities are granted to roles, and users are authorized to execute Adabas utilities through their assigned roles.

Adabas RBAC knows the object types Users, Roles, Actions and Resources. Assignments and Privileges describe the relations between these entities.

User
The user name.

Role
The role name groups access rights.

Action
The Adabas utility.

Resource
The database ID. The current database is denoted as *DBID.CURRENT*.

Assignment

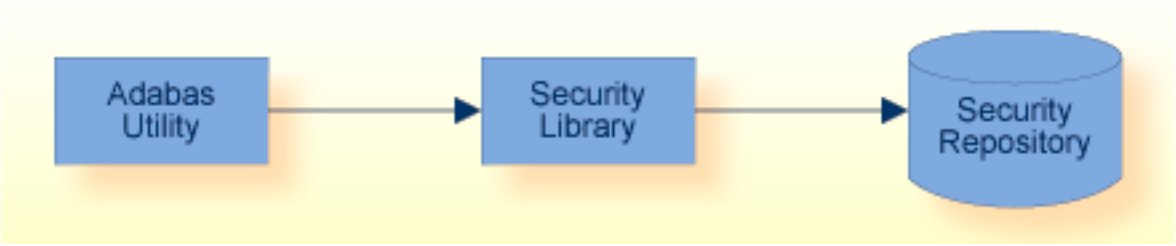
- A user can have multiple roles.
- A role can be assigned to multiple users.

Privilege

- A role can have multiple permissions.
- A specific permission can be assigned to multiple roles.

Architecture

The security library validates an Adabas utility's access request against the access privileges defined in the security repository.



Authentication

Authentication for Adabas Utilities uses the login credentials to authenticate a user:

- Unix: User ID
- Windows: Domain and user ID

Authorization Process

Authorization for Adabas Utilities uses RBAC security definitions to authorize a requested operation on a specified database.

The security library validates the access request against the access privileges defined in the security repository.

If a matching access privilege is found, the Adabas Utility is authorized to perform the requested operation, otherwise the request is rejected.

Security Infrastructure

The installation of the required components and configuration files is mandatory.

Initial Setup

The security infrastructure is required by the Adabas utilities.

An initial security configuration is created during the installation; e.g.

- The Audit Trail is pre-configured.
- A minimal set of not restrictive role-based security definitions is installed.
- Authorization is not active.

Configuration Files

This section describes how to configure the authorization feature.

The configuration of Authorization for Adabas Utilities is stored in the following files:

Configuration File	Description
`adaauth.ini`	Configuration of Authorization for Adabas Utilities
`adaaudit.ini`	Configuration of Audit Trail
`adarbac.ini`	Role-Based Access Control Definitions (MODE=INI)

These files configure the local machine and apply to all databases, to all product installations and product versions that are greater than or equal to Adabas Version 6.5 on the machine.

The configuration files are ASCII files and can be edited with a standard text editor. Access to these file should be restricted. Please refer to the section ***Security Considerations***, which describes how to secure ("harden") the database.

The location of the configuration files is platform-dependent and is described in the section *Configuration of Authorization for Adabas Utilities* of the *Extended Operation* documentation.

Authorization Configuration

The *adaauth.ini* configuration file contains information, which applies to the machine and to all databases, to all product installations and product versions that are greater than or equal to Version 6.5 on the machine.

Section	Item	Description
`AUTHZ`	`AUDIT_FILE`	Location of Audit Trail configuration file
	`RBAC_FILE`	Location of security definitions (MODE=INI)
	`ACTION`	Enable the Authorization for Adabas Utilities feature
	`MODE`	Define the source of the security definitions

Note: Currently, this file enables or disables the Authorization for Adabas Utilities feature. The default setting is "disabled" (`ACTION = NO`). This setting may be removed or changed in later product versions.

For further information on the syntax of this file, see the descriptions of *adaauth.ini* in the *Extended Operation* documentation.

Audit Trail Configuration

The *adaaudit.ini* configuration file defines the layout and location of the Audit Trail.

Section	Item	Description
AUDIT	LOG_FILE	Location of Audit Trail log file
	FORMAT	Determines the layout of the log file Entry ■ TEXT (Default) ■ CSV
	SEPARATOR	The separator used in CSV layout

Requirements on the Audit Trail log file:

- The Audit Entries are appended to the log file.
- Thus, users of Adabas utilities require WRITE access to the log file.
- The size of the log file should be monitored and backed-up or moved as need be.

Note: It is mandatory that all users, who are authorized to execute an Adabas utility, have READ/WRITE access permissions to both the LOG_FILE and the directory in which it is located.

For further information on the syntax of this file, see the descriptions of *adaaudit.ini* in the *Extended Operation* documentation.

Performance Considerations

This feature has minimal impact on the overall system performance.

Authorization for Adabas Utilities (Mode ADABAS)

If *MODE=ADABAS* is defined in the *adaauth.ini* configuration file, the accessed Adabas database is the source of the security definitions.

- Administration

Security Definitions

The Adabas database contains a specific system file, the RBAC system file that stores the security definitions. When it is created, a basic set of initial definitions is loaded. These security definitions can be adapted and extended to the application's needs.

The advantage of this mode is that each authorization request is decided locally, hence performance impact is minimalized.

This approach supports authorization for utilities that access offline and online databases.

Initial State

In mode ADABAS, the initial security definitions implement unrestricted access to all Adabas utilities.

Initially, the RBAC system file defines user *PUBLIC* and role *PUBLIC* with role *PUBLIC* assigned to user *PUBLIC*. The role *PUBLIC* has the privilege to execute all Adabas utilities.

Any user who is not yet known to the security repository is treated as user *PUBLIC* to ensure unrestricted access as in previous Adabas versions.

Administration

The utilities required to configure and administer the Authorization feature are:

- ADADBM - Create the RBAC system file
- ADAREP - Query the RBAC system file
- ADARBA - Administer the RBAC system file

ADADBM - Create the RBAC System File

Use the RBAC_FILE function of ADADBM to create the RBAC system file.

Note: The RBAC_FILE function requires that the database is offline.

ADAREP - Query the RBAC System File

Use the SUMMARY function of ADAREP to display the database system files.

Note: The RBAC system file is only displayed if the RBAC file is defined. It is not displayed if the file is not loaded.

ADARBA - Administer the RBAC System File

Use the functions of ADARBA to create, read, update and delete security definitions.

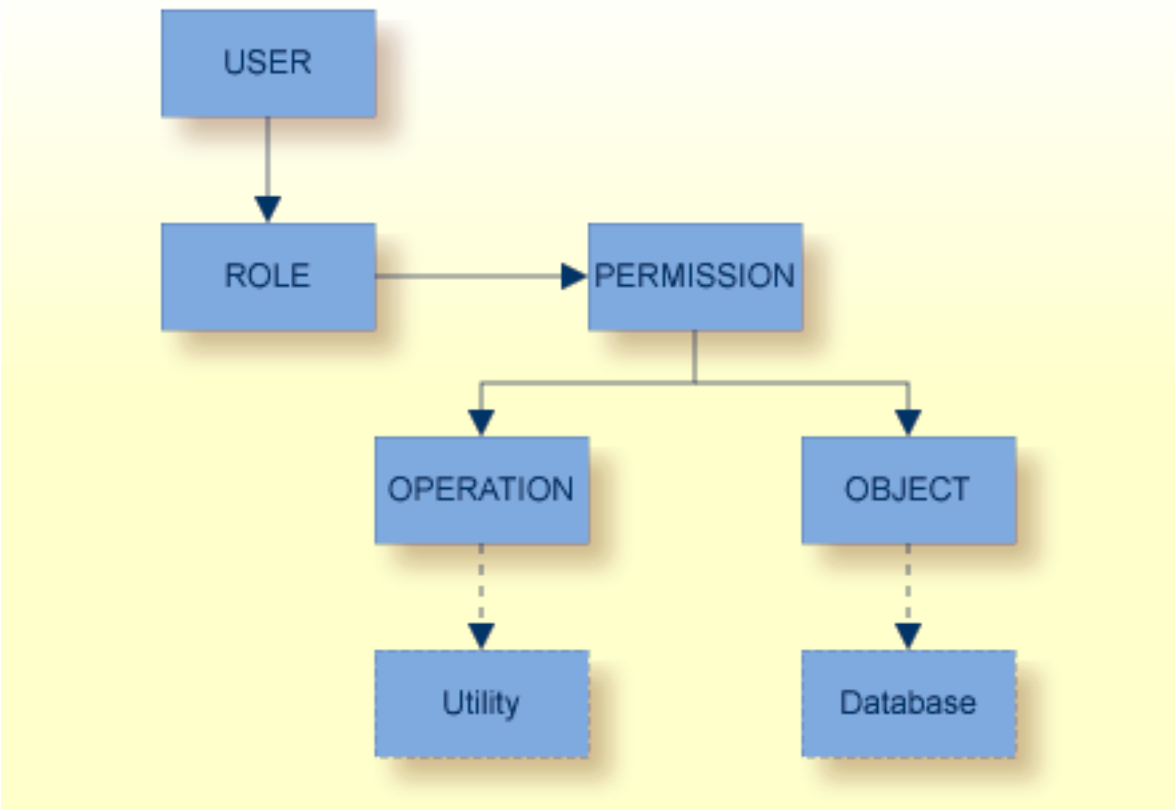
Note: ADARBA requires that the database is online.

Authorization for Adabas Utilities (Mode INI)

In Authorization mode INI, the security definitions apply to the machine and to all databases, and to all product installations and product versions that are greater than or equal to Version 6.5 on the machine.

Security Definitions

The *adarbac.ini* configuration file defines roles and assigns permissions to execute sets of operations on one or more objects. Below is an overview of the relationships between the entries in the file:



Section	Item	Description
USER_ROLE	user_name	User/Role assignment
ROLE_PERMISSION	role_name	Role/Permission assignment
PERMISSIONS	permission_name	Operation/Object assignment
OBJECTS	object_name	Assignment of database ID
OPERATIONS	operation_name	Assignment of utility

For further information on the syntax of this file, see the descriptions of *adarbac.ini* in the *Extended Operation* documentation.

Getting Started with Authorization for Adabas Utilities

This *Getting Started* documentation covers the following topics:

- Prerequisites
- Configuration
- Authorization

Prerequisites

Authorization for Adabas Utilities requires the security infrastructure. An initial security configuration is set up during installation, the Audit Trail is pre-configured, and Authorization is not active.

Adabas authorization uses your login credentials. This is the user ID on Unix platforms, and the domain and user ID on Windows platforms.

The Adabas utility ADARBA is used to maintain your RBAC security definitions, to create or drop user and role definitions, and to grant or revoke assignments and privileges.

Configuration

This section describes how to configure Authorization for Adabas Utilities to use the RBAC security definitions that are stored in the Adabas RBAC system file.

To configure Authorization for Adabas Utilities

1 Set the following in the security configuration file *adaauth.ini*, section [AUTHZ]:

- ACTION=YES - to enable Adabas Authorization
- MODE=ADABAS - to use the authorization mode ADABAS

The setting RBAC_FILE is not used for this configuration.

The setting AUDIT_FILE can be left unchanged for the purpose of this Getting Started.

2 Create a demo database:

```
crdemodb 100
```

3 Query the database information:

```
adarep dbid=100 summary
```

At this point, you are authorized to execute the Adabas utility ADAREP, even though MODE=ADABAS is set, because an RBAC system file has not yet been defined.

4 Create the RBAC system file and load the initial security definitions:

```
adadbm db=100 rbac_file=<any_file_number>
```

At this point, you are authorized to execute the Adabas utility ADAREP because the RBAC system file has been defined and the initial definitions have been loaded.

5 Start the database, and then display the initial security definitions (the administration of RBAC definitions is only supported if the database is online):

```
adarba db=100 list users
```

shows the inital user *PUBLIC*.

```
adarba db=100 list roles
```

shows the inital role *PUBLIC*.

```
adarba db=100 list assignments
```

shows the inital assignment *PUBLIC*,*PUBLIC*.

By default, the user *PUBLIC* is granted the role *PUBLIC*, and therefore has all of the privileges that are granted to the role *PUBLIC*.

Authorization

With the initial set of RBAC security data, your login user ID is authorized to execute the security-enabled Adabas utilities because, by default, a user who is not known to the RBAC system is authorized as the user *PUBLIC* with the role *PUBLIC*.

In the next steps, you will restrict the privileges for your current login ID

To restrict the privileges for your current login ID

1 Create a new user:

```
adarba db=100 create user=my-login-user
```

where *my-login-user* is your current user ID.

2 Create a new role:

```
adarba db=100 create role=newrole
```

3 Grant the new role to the new user:

```
adarba db=100 grant assignment role=newrole user= my-login-user
```

4 Check the assignments:

```
adarba db=100 list assignments
```

now shows

- PUBLIC,PUBLIC (the initial assignment)
- PUBLIC,my-login-user (default assignment for a new user)
- newrole,my-login-user (the new assignment)

5 Grant a restricted privilege to the new role:

```
adarba db=100 grant privilege action=ada.uti.rba  role=newrole
```

6 Check the privileges:

```
adarba db=100 list privileges
```

shows

- all default privileges
- ada.uti.rba,DBID.CURRENT,newrole

Because the user *my-login-user* has the roles *PUBLIC* and *newrole*, you are still authorized to do everything.

7 Restrict the access privilege of the user *my-login-user* to ADARBA:

```
adarba db=100 revoke assignment role=PUBLIC user= my-login-user
```

8 Now try to execute an Adabas utility other than ADARBA:

```
adarep db=200 sum
```

This will return ‘Security violation. Permission denied.’

If you want to adapt or restore your access privileges, you can still use ADARBA to define them according to your needs, e.g.

```
adarba db=100 grant privilege action=ada.uti.rep  role=newrole
```

or

```
adarba db=100 grant assignment role=PUBLIC user= my-login-user
```

4 Adabas Password Security (ADASCR)

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.

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.

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.

5 Ciphering

Important: The implementation of the ciphering feature is different to that which is available in Adabas for mainframes. The cipher code in Adabas for UNIX and Windows is static and is not provided in Additions 4.

The purpose of the ciphering in Adabas for UNIX 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.

6 Security Considerations

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

Using the UNIX Group Concept

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

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

Example:

Assume you have two UNIX 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 Configuration Files

This section describes how to secure the configuration files used to configure authorization for Adabas utilities.

File	Description
`adaauth.ini`	Configuration of Authorization for Adabas Utilities
`adaaudit.ini`	Configuration of Audit Trail
`adarbac.ini`	Role-Based Access Control Definitions

To secure the configuration files, please ensure the following:

- **READ-ACCESS**
  All users, which execute an Adabas utility, must be able to read these files.
- **WRITE-ACCESS**
  Only the administrator of a file should have write access to the file.

The location of the configuration files is platform-dependent and is described in the section *Configuration of Authorization for Adabas Utilities* of the *Extended Operation* documentation.

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.

The location of the audit trail log file is set via the `LOG_FILE` option in the *adaaudit.ini* configuration file and is described in the section *Configuration of Authorization for Adabas Utilities* of the *Extended Operation* documentation.

7 SSXLoginModule Configuration Templates

The following sections contain configuration templates for the `SSXLoginModule` organized by authorization type.

Authorization Type OS

The security definitions for authorization 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 Unix 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: UNIX 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.

##unixAddMachineName

# 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]
```

Authorization Type TEXT

The security definitions for authorization 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]
```

Further examples:

Creating Internal User Repository Files

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==
```

Authorization 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]
```