Software AG Security eXtensions Administrator's Guide

Innovation Release

Version 10.0

April 2017

This document applies to Software AG Security eXtensions Version 10.0 and to 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: SSX-AG-100-20170419

Table of Contents

About this Guide

Software AG Security eXtensions provides a common interface for various ways of authenticating a user, for example, using an LDAP server, the local operating system, or a simple text file.

This guide explains the setup of Software AG Security eXtensions, also known as SSX. SSX is a component that is used by a number of Software AG products, and is automatically installed with these products. This guide needs to be seen in context of the documentation of the product that is using SSX. It is intended as an extended documentation focusing on common tools and configuration options.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	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.

Convention	Description
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information

Software AG Documentation Website

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

Software AG Empower Product Support Website

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".

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.

1 SSX Tools

This chapter briefly describes the utilities that are shipped with Software AG Security eXtensions. You will find information on the following:

- ssxenv.sh / ssxenv.bat
- createTechUserCreds
- ssxtxtpasswd
- sagssxauthd2

Preparing the Environment

The SSX tools described in this chapter require the SSX and OpenSSL libraries. Therefore, you have to prepare the environment accordingly using the following scripts:

- ssxenv.bat on Windows
- ssxenv.sh on UNIX; just source the script

Both scripts appropriately set up the command and library search paths for the SSX tools. With a default installation, enter the following commands:

- Windows command prompt (specify the appropriate drive and Software AG directory if you do not use the default settings):

```
C:\SoftwareAG\common\security\ssx_64\bin\ssxenv.bat
```

  Note: If the backwards compatible 32-bit SSX tools are to be used, you have to use ssxenv.bat from the ssx_32\bin directory instead.

- UNIX shell (specify the appropriate Software AG directory if you do not use the default settings):

```
. /opt/softwareag/common/security/ssx/bin/ssxenv.sh
```

  Note: As this script is to be sourced, make sure that there is a space character between the leading dot (.) and the script.

Both scripts load yet another script (tlsenv.bat/tlsenv.sh) to ensure that the necessary OpenSSL libraries are found. With a default installation, this script is available at the following location:

- Windows:

  C:\SoftwareAG\common\security\openssl\extras\tlsenv.bat

- UNIX:

  /opt/softwareag/common/security/openssl/extras/tlsenv.sh

Under certain circumstances, the environment setup described above might be ignored by the operating system, for example, when using elevated privileges. On AIX, it may be necessary in this case to create a symlink from the expected default directory to the actual installation directory of the libraries (root privileges might be needed to do that):

```
mkdir -p /opt/softwareag/common/security/ssx
ln -s /actual/path/to/ssx/lib /opt/softwareag/common/security/ssx/lib
mkdir -p /opt/softwareag/common/security/openssl
ln -s /actual/path/to/openssl/lib /opt/softwareag/common/security/openssl/lib
```

The scripts also create the environment variables `SSXDIR` and `TLSDIR`, which contain the full qualified paths to the ssx and openssl base directories.

Creating Technical User Credential Files

Software AG Security eXtensions provides a tool that you can use to create technical user credential files:

- createTechUserCreds.exe on Windows
- createTechUserCreds on UNIX

At a later stage, you can use the technical user credential files to search for and discover LDAP users securely on LDAP servers that do not support anonymous requests. With a default installation, this tool is available in the following directory:

- Windows:

  C:\SoftwareAG\common\security\ssx_64\bin\

  Note: The tool may also be available in the ssx_32 directory (instead of ssx_64). This is only for backwards compatibility.

- UNIX:

  /opt/softwareag/common/security/ssx/bin/

To start the createTechUserCreds tool, you can use a command prompt. When you start the tool, you enter a user name and a password which are then encrypted and provided in the result text file.

Even though this is optional, you definitely should specify and use a key file to encrypt the technical user's password in the result. See "More About Key Files" on page 13.
If you do not use a key file, the result is still encrypted, but a hardcoded standard key is used in this case. For production environments, this would be considered a security risk!

To create a technical user credential file

1. Set up the environment as described in "Preparing the Environment" on page 8.
2. Start the tool using the following command:

   ```
   createTechUserCreds -f result_file_name -k key_file_name
     -p password user_ID -o
   ```

   When you execute the tool without specifying an argument for the result file name, it still creates a text file with the corresponding technical user credentials. The file is created in the same directory in which you started the tool and has a predefined default name (techuser).

To customize the parameters for invoking this tool, you can use a set of predefined optional arguments. The available arguments and their descriptions are as follows:

Argument	Description
`-f`	Provide a name for the result text file which contains the technical user credentials. If you do not use this argument, the tool creates a default result file.
`-k`	Provide an alternative key file to encrypt the result text file that contains the technical user credentials. If you do not use this argument, the tool uses a default key. Relying on the default key is considered insecure.
`-p`	Provide the password for the *`user_ID`* on the command line. If you do not use this argument, the tool interactively asks for the password. Using this argument is considered insecure.
`user_ID`	Provide the full DN of the technical user on UNIX or the usual domain\user name tuple on Windows. This depends, however, to which kind of LDAP server the connection will be made.
`-o`	Overwrite existing technical user credentials without asking.

3. Press ENTER.

   If `-p` is not provided, the tool will ask you to provide the password. If `-o` is not provided, the tools will ask for a confirmation to overwrite the existing file.

Examples

The following examples provide information about more typical use cases of the tool:

```
createTechUserCreds.exe -f techUser.txt -k techuser.key DOM\admin

createTechUserCreds -f techUser.txt -k techuser.key cn=admin,dc=domain,dc=com
```

The tool creates a text file which contains the encrypted technical user credentials and stores it in the same directory in which you started it.

As a next step, you can provide the file to the configuration option `techLdapUserCredFile` (see the corresponding product documentation for more information). Do not forget to also provide the `techLdapUserKeyFile` option. See also "More About Key Files" on page 13.

Re-Encrypting Technical User Credentials Files

You can re-encrypt technical user credentials files that were previously encrypted

- with the default key, or
- with a custom key file.

This is described in the topics below.

Re-Encryption Using the Default Key

You can re-encrypt a technical user credential file that was previously encrypted with the default key.

The tool first decrypts the old technical user credentials using its default key. It then encrypts the credentials again, but this time using the provided key file. The result is stored in the new technical user credentials file.

Important: Do not use the same file name for the old and the new file.

To re-encrypt a technical user credentials file that was encrypted with the default key

1. Set up the environment as described in "Preparing the Environment" on page 8.
2. Start the tool using the following command:

```
createTechUserCreds -f result_file_name -r new_key_file_name
  -c old_techuser_file_name
```

When you execute the tool without specifying an argument for the result file name, it still creates a text file with the corresponding technical user credentials. The file is created in the same directory in which you started the tool and has a predefined default name (techuser).

To customize the parameters for invoking this tool, you can use a set of predefined optional arguments. The available arguments and their descriptions are as follows:

Argument	Description
`-f`	Provide a name for the new result text file which contains the re-encrypted technical user credentials. If you do not use this argument, the tool creates a default result file.
`-r`	Provide a key file to re-encrypt the result text file that contains the technical user credentials.

Argument	Description
`-c`	Provide the name of the text file containing already encrypted technical user credentials. These credentials were encrypted with the program's default key, that is, no key file was provided for the encryption.

3. Press ENTER.

Re-Encryption Using a Custom Key File

You can re-encrypt a technical user credential file that was previously encrypted with a custom key file.

The tool first decrypts the old technical user credentials using the old key file (`-k`). It then encrypts the credentials again, but this time using the provided new key file (`-r`). The result is stored in the new technical user credentials file.

Important: Do not use the same file name for the old and the new file.

To re-encrypt a technical user credentials file that was encrypted with a custom key file

1. Set up the environment as described in "Preparing the Environment" on page 8.
2. Start the tool using the following command:

```
createTechUserCreds -f result_file_name -r new_key_file_name
  -c old_techuser_file_name -k new_key_file_name
```

When you execute the tool without specifying an argument for the result file name, it still creates a text file with the corresponding technical user credentials. The file is created in the same directory in which you started the tool and has a predefined default name (techuser).

To customize the parameters for invoking this tool, you can use a set of predefined optional arguments. The available arguments and their descriptions are as follows:

Argument	Description
`-f`	Provide a name for the new result text file which contains the re-encrypted technical user credentials. If you do not use this argument, the tool creates a default result file.
`-r`	Provide a key file to re-encrypt the result text file that contains the technical user credentials.
`-c`	Provide the name of the text file containing already encrypted technical user credentials.

Argument	Description
`-k`	Provide the key file necessary to decrypt the text file containing already encrypted technical user credentials.

3. Press ENTER.

More About Key Files

A key file contains the key that is used to encrypt or decrypt the password of the technical user. The algorithm used for the encipherment is AES-128. Therefore, the key must be 32 bytes long. It actually consists of a 16 byte key and a 16 byte so-called initial vector, or IV.

To store this key in the key file it has to be converted to 64 hexadecimal printable characters in the first line, without any spaces or other characters. The following is an example key file (do not use in production):

```
000102030405060708090a0b0c0d0e0f00112233445566778899aabbccddeeff
```

Unlike the example above, the key should consist of purely random data. To generate such a random key, you can make use of the openssl tool that is shipped with SSX. Just set up the environment as described in "Preparing the Environment" on page 8.
You can then issue an OpenSSL call such as the following to generate a key file:

```
openssl rand -hex 32 > techuser.key
```

But this is just an example. You can also just type 64 random digits and the characters `a` through `f` in a text editor and save that line as a text file.

It is recommended to store the key file in the same place as the technical user credentials file, for example, in ${SSXDIR}/etc/. Ensure that the files can only be read by the user running the Software AG products.

Creating Internal User Repository Files

You can create and/or modify internal user repository files that contain user names and their respective encrypted passwords.

Software AG Security eXtensions provides a tool that you can use to create internal user repository files:

- ssxtxtpasswd.exe on Windows
- ssxtxtpasswd on UNIX

At a later stage, you can use the internal user repository file to authenticate users independently from your system. With a default installation, the tool is available in the following directory:

- Windows:

C:\SoftwareAG\common\security\ssx_64\bin

> **Note:** The tool may also be available in the ssx_32 directory (instead of ssx_64). This is only for backwards compatibility.

- UNIX:

  /opt/softwareag/common/security/ssx/bin/

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.

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:

```
! " # $ % & ' ( ) * + , - . / : ; < = > ? [ \ ] ^ _ ` { | } ~
```

The user-defined repository files must comply with the following format:

```
*
* Default test repository for INTERNAL or TEXT based authentication
*
version:3.0
*
*
user:user_id:hashed_password
*
```

To create and/or modify an internal user repository file

1. Set up the environment as described in "Preparing the Environment" on page 8.
2. Start the tool using the following command:

   ```
   ssxtxtpasswd [-f result_file_name] [-c] [-p password]
     [-d | -e] user_ID
   ```

   To customize the parameters for invoking this tool, you can use a set of predefined optional arguments. The available arguments and their descriptions are as follows:

Argument	Description
`-f`	Provide a name for the result text file which contains the user credentials. If you do not use this argument, the tool creates a default result file called ssx_user.
`-c`	Using this parameter, you create a text repository file with a specified name (`-f` argument). If you do not use the `-c`

Argument	Description
	argument and the specified text file does not exist, an error is returned. If you specify `-c` and the file already exists, the `-c` argument is ignored and the tool does not create a new file. When you execute the tool without specifying an argument for the result file name (`-f` argument), it still creates a text file with the corresponding internal user repository information. The file is created in the same folder in which you started the tool and has a predefined default name (ssx_user).
`-p`	Provide a password directly on the command line. Thus, the tool does not invoke a non-echo input of the password in the next steps. Providing a password as a command line argument is considered insecure.
`-d`	Remove credentials data for a particular user from the text repository file. When you use the `-d` argument, the tool ignores the presence of the `-c` argument.
`-e`	Just check if a particular user exists in the text repository file.
`user_ID`	Provide a user name which you want to add, delete, check or replace in the text file.

3. Press ENTER.

   If `-p` is not provided, the tool will ask you to provide the password.

Examples

The following examples provide information about more typical use cases of the tool:

```
ssxtxtpasswd.exe -c -f internalUser.txt -p pass myUser
ssxtxtpasswd.exe -f internalUser.txt -p newpass myUser
ssxtxtpasswd.exe -d -f internalUser.txt myUser
```

The tool creates a text file which contains the encrypted internal user repository credentials and stores it in the same directory in which you started it.

As a next step, you can provide the file to the configuration option `internalRepository` (see the corresponding product documentation for more information).

Using the Pluggable Authentication Module (PAM) on UNIX

The Pluggable Authentication Module (PAM) is a standardized architecture to let third parties carry out authentication requests from applications. PAM allows you to perform OS authentication on UNIX.

PAM Authentication

To perform OS authentication using PAM, the sagssxauthd2 module tries to load the client-side PAM library, usually named libpam.so, and a password encryption library, usually named libcrypt.so (libsec.so/libsec.sl on HP-UX), and is using the PAM service `ssxsrv`.

If the PAM library is successfully loaded, the sagssxauthd2 module attempts to perform a PAM authentication first. If the authentication is successful, the module reports success and stops further processing.

If the PAM library could not be loaded or the PAM authentication fails, the module tries to perform a UNIX user authentication using the system's password database(s) and the password encryption library. If the library could not be loaded, an error is returned. If it is successfully loaded, the sagssxauthd2 module calls operating system functions which look for the user's password in the local shadow password user database or the traditional password database.

- If a password entry is found for the user, the module encrypts the given password in the same way the existing password has been stored by the UNIX system and compares the result.
- If both passwords match, the module reports success.
- Otherwise, an authentication failure is reported.

Conditions for Using PAM

The PAM authentication and querying the UNIX system's password database(s) require specific privileges from the calling process. Therefore, the `sagssxauthd2` module must be owned by the root user. It must reside on a file system which is not mounted with the `nosuid` option and the `setuid` file attribute must be enabled (the file access rights should look like `-rwsr-xr-x ... root ... sagssxauthd2`). The module is typically installed into the directory: /opt/softwareag/common/security/ssx/auth.

A PAM service `ssxsrv` has to be defined. In a Linux system, it is usually made available by copying the default configuration from /opt/softwareag/common/security/ssx/etc/ssxsrv.pamd into the respective system's directory as /etc/pam.d/ssxsrv.

The service also needs to be configured according to your system's usage of PAM. Depending on your needs and your system's configuration, you may follow the example of the `common-auth`, `sshd` or `su` PAM services.

If any of the conditions above is not met, an error can occur. In this case, it is important to double-check the status of the sagssxauthd2 module. You may also want to contact Software AG support for further assistance; in this case, you should also create an SSX trace to be sent to support.

Another source of failure is using an unsupported password hash algorithm for comparing the passwords returned by the operating system. Software AG Security eXtensions currently supports the following hash algorithms:

- MD5 (1)
- Long Blowfish ($2a$)
- BCrypt ($2y$)
- Short Blowfish (2)
- SHA-256 (5)
- SHA-512 (6)
- DES

Note: On HP-UX, the sagssxauthd2 module also uses the crypt2_passwd_match() and bigcrypt() functions to perform the password comparison.

On AIX, the Loadable Password Algorithms (LPA) are supported as of AIX version 6.1.

Troubleshooting sagssxauthd2

When you install Software AG Security eXtensions on a network file system (NFS) which is mapped to the local one, the local policies may not allow access rights, such as root or `setuid` to the remote installation. As a result, the sagssxauthd2 module does not work properly despite the properly configured root ownership and `setuid` file attribute.

To resolve the issues with the remote sagssxauthd2 module

1. Copy the sagssxauthd2 module to a local file system.
2. Set its root ownership and `setuid` file attribute.
3. To use the sagssxauthd2 module on the remote installation, replace the remote file in the corresponding directory with symbolic links that point to the locally copied module.

Directory Structure and Noteworthy Files

The default SSX installation directory is at C:\SoftwareAG\common\security\ssx_64 (Windows) or at /opt/softwareag/common/security/ssx (UNIX). The environment variable `SSXDIR` should point to the actual SSX installation directory.

The default OpenSSL installation directory is at C:\SoftwareAG\common\security\openssl (Windows) or at /opt/softwareag/common/security/openssl (UNIX). The environment variable `TLSDIR` should point to the actual OpenSSL installation directory.

The following table briefly explains the directories and most important files that can be found below the installation directories.

Directory	Contents
ssx/auth	Only available on UNIX systems. This directory belongs to the root user and contains executables and scripts that require elevated privileges. To handle this directory and its content properly, the Software AG Installer and the Software AG Update Manager will need a `sudo` password. ■ sagssxauthd2. This executable must be owned by the root user and have its s-bit set. It is a daemon which handles local authentication requests when `authType` is `OS`; it is started automatically in this case. For more information, see "Using the Pluggable Authentication Module (PAM) on UNIX" on page 16. ■ set_daemon_privs.sh. This Bourne shell script is called during the installation to set up ownership and permissions of `sagssxauthd2`. This script can also be used again after the installation to set up the ownership and permissions. If it is not started by the root user, it uses `sudo` to gain the necessary privileges for its operation.
ssx/bin	Windows: Executables and libraries required to set up and operate SSX. UNIX: Executables and Bourne shell scripts required to set up SSX. Important files are: ■ createTechUserCreds.exe (Windows) or createTechUserCreds (UNIX). See "Creating Technical User Credential Files" on page 9 and "Re-Encrypting Technical User Credentials Files" on page 11. ■ ssxtxtpasswd.exe (Windows) or ssxtxtpasswd (UNIX). See "Creating Internal User Repository Files" on page 13.
openssl/bin	Windows: Executables and libraries required to set up and operate OpenSSL and SSX. UNIX: Optional executables for OpenSSL. Important files are:

Directory	Contents
	■ libeay32.dll and ssleay32.dll (Windows). The OpenSSL libraries used by SSX.
ssx/etc	Important files are: ■ alt_keyfile.txt. This is a sample key file. **Note:** This key file is not suitable for production purposes. You need to create and use a key file like this when creating technical user credential files. See "More About Key Files" on page 13. ■ ssx_user. This is the default internal user repository. It contains the default administrator account, which might be used during installation. **Note:** If using the authentication type `INTERNAL`, it is strongly recommended that you create and configure a different internal user repository, remove the administrator account, or at least change the administrator's password. See "Creating Internal User Repository Files" on page 13. ■ ssxsrv.pamd. Only available on Linux. This is the default configuration of the PAM service. See "Conditions for Using PAM" on page 16.
openssl/lib	UNIX: Libraries required to operate OpenSSL and SSX. ■ libcrypto.so.1.0.0, libssl.so.1.0.0 and respective symlinks. The OpenSSL libraries used by SSX.

2 Configuring Software AG Security eXtensions

Software AG Security eXtensions are to be used through other Software AG products. It is therefore configured by means of the product that is using it (see the corresponding product documentation for more information). However, the products may expose the configuration parameters in one way or another. This chapter therefore provides common descriptions of these parameters.

Parameters for Common Configuration

The following parameters apply to all authentication configurations.

Parameter	Description
`defaultDomain`	Optional. A default domain name. When querying for all users, the `defaultDomain` parameter is added in front of any user ID returned. No default value.
`nativeLogFile`	Optional. The output file for logging. No default value.
`nativeLogLevel`	Optional. The value of the logging level. Valid values are the Integer values from 0 (no logging) up to 6 (debug level logging). The default value is 0.
`cacheTime`	Optional. How long the authenticated user stays in cache. The value is in seconds. A valid value is any Integer value. The default value is 180.
`denyTime`	Optional. How long the authenticated requests of a particular user are ignored. The value is in seconds.

Parameter	Description
	A valid value is any Integer value. The default value is 100.
`denyCount`	Optional. The number of invalid login attempts. A valid value is any Integer value. The default value is 3.
`cacheSize`	Optional. The maximum number of successfully authenticated users that are stored in the cache. When the cache overflows, the oldest entry is removed. A valid value is any Integer value. The default value is 300.

Parameters for Internal Repository Configuration

This section describes the `INTERNAL` repository type which is based on a text file. The current default repository type (`OS`) requires specific root privileges on UNIX. To avoid the necessity of having specific privileges, you can use the internal repository for new installations that use Software AG Security eXtensions on UNIX.

The internal repository text file is an alternative to the `OS` and `LDAP` repositories. It is recommended to use an internal repository only during the initial setup of all required components or until you configure a real repository.

Parameter	Description
`authType`	The user repository type. The required value is `INTERNAL` or `TEXT`. No default value.
`internalRepository`	The path of the internal text repository file. For more information, see "Creating Internal User Repository Files" on page 13.

Parameters for Operating System Configuration

The following parameters are used for authentication against the local operating system.

Parameter	Description
`authType`	The user repository type. The required value is `OS`. No default value.
`authDaemonPath`	The explicit path of the privileged authentication daemon. Specify this parameter if the sagssxauthd2 module is not in the current directory. A valid value is the canonical path to the sagssxauthd2 module. See also "Using the Pluggable Authentication Module (PAM) on UNIX" on page 16. No default value. **Note:** UNIX only.
`defaultGroup`	Optional. A default group name to be returned with any of the group results that are returned by the repository manager. A valid value is any valid group name. No default value.
`noImpersonation`	Optional. Boolean. How to access data. Valid values: ■ true - Access is under the account of the running process. ■ false - Default value. The data access is under the impersonated user ID of the logged on user. **Note:** Windows only.
`unixAddMachineName`	Optional. Boolean.

Parameter	Description
	Prepend 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: ■ true - If set to true (and there is no domain field), you are authenticated against the local machine only. ■ false - Default value. You are authenticated on the domain that you logged on. **Note:** UNIX only.

Parameters for ADSI Configuration

The following parameters are used for authentication against a Microsoft Active Directory Server. They are applicable only on Windows.

Parameter	Description
`authType`	The user database type. The required value is `ADSI`. No default value.
`defaultGroup`	Optional. A default group name to be returned with any of the group results that are returned by the repository manager. A valid value is any valid group of users. No default value.
`serverHost`	Optional. The name of the server. A valid value is any valid server name and any valid IP address. No default value.

Parameter	Description
`adsiPersonBindDn`	Optional. The Personal Bind Distinguished Name (DN) for LDAP required for accessing a user entry. Use it only when all the user entries that are accessed are under the same node. Do not use it in cases of normal authentication. Valid values (example): `ou=users,ou=germany,dc=eur,dc=sa,dc=com` No default value.
`adsiGroupBindDn`	Optional. The Personal Bind Distinguished Name (DN) for LDAP required for accessing a group. Use it only when all the groups that are accessed are under the same node. Do not use it in cases of normal authentication. Valid values (example): `ou=groups,ou=germany,dc=eur,dc=sa,dc=com` No default value.
`adsiForestDn`	Optional. The name of the forest. You use this value when accessing the Active Directory. Valid values (example): `dc=myorg,dc=com` No default value.

Parameters for LDAP Configuration

The following configuration parameters are used for authentication against an LDAP server.

Parameter	Description
`authType`	The user database type. The required value is `LDAP`. No default value.

Parameter	Description
`serverHost`	The name or IP address of the server. It may optionally be followed by a colon (:) and the port number. In the latter case, the `serverPort` parameter is ignored. A valid value is any valid server name and any valid IP address. No default value.
`serverPort`	Optional. The port of the server. A valid value is any valid port number. The default value is 389.
`serverType`	Optional. The type of the LDAP server. Valid values: - `OpenLdap` - Default value - `ActiveDirectory` - `SunOneDirectory` - `Novell` - `ApacheDS` - `Tivoli`
`personBindDn`	The Distinguished Name where the authentication information is stored. This value will be prefixed with the value of the `userIdField` parameter when issuing the authentication call. Valid values (example): `ou=users,ou=germany,dc=sa,dc=com` No default value.
`groupBindDn`	The Group Root Distinguished Name (DN) for LDAP where the search for group names starts. This value will be prefixed with the

Parameter	Description
	value of the `groupIdField` parameter when issuing the authentication call. Valid values (example): `ou=groups,ou=germany,dc=sa,dc=com` No default value.
`personObjClass`	Optional. The object classes of the user entries. Valid values: *`String_Value1`*`, `*`String_Value2`*`, ..., `*`String_ValueN`* The default value depends on the `serverType` parameter. See "LDAP Defaults for the Different Server Types" on page 34.
`groupObjClass`	Optional. The object classes of the group entries. Valid values: *`String_Value1`*`, `*`String_Value2`*`, ..., `*`String_ValueN`* The default value depends on the `serverType` parameter. See "LDAP Defaults for the Different Server Types" on page 34.
`personGrpAttr`	Optional. The property name of a user entry that points to the group in which the user is a member. Valid values: *`String_Value`* The default value depends on the `serverType` parameter. See "LDAP Defaults for the Different Server Types" on page 34.

Parameter	Description
`groupPrsAttr`	Optional. The property name of a user entry that points from the group to the respective users. Valid values: *`String_Value`* The default value depends on the `serverType` parameter. See "LDAP Defaults for the Different Server Types" on page 34.
`userIdField`	Optional. The property name that denotes the user ID. Valid values: *`String_Value`* The default value depends on the `serverType` parameter. See "LDAP Defaults for the Different Server Types" on page 34.
`passwdField`	Optional. The property name that denotes the password field of a user entry. Valid values: *`String_Value`* The default value depends on the `serverType` parameter. See "LDAP Defaults for the Different Server Types" on page 34.
`groupIdField`	Optional. The property name that denotes the group ID. Valid values: *`String_Value`* The default value is `cn`.

Parameter	Description
addPersonAttr	Optional. May contain additional fields and values that are used when a new user is added. The string %% will be replaced by the actual user name parameter. Valid values: *String_Value* The default value depends on the serverType parameter. See "LDAP Defaults for the Different Server Types" on page 34.
addGroupAttr	Optional. May contain additional fields and values that are used when a new group is added. The string %% will be replaced by the actual group name parameter. Valid values: *String_Value* The default value depends on the serverType parameter. See "LDAP Defaults for the Different Server Types" on page 34.
allowDomainAsBaseBindDn	Optional. Boolean. If the domain name is not specified explicitly and the defaultDomain parameter is set, this value is interpreted as BaseBindDN. Valid values: ■ true - The domain is interpreted as a BaseBindDN (for example, ou=People,dc=myorg,dc=com). ■ false - Default value.
personPropAttr	Optional. The user's properties of interest.

Parameter	Description
	A valid value is a comma-separated list that contains the property names: `String_Value1, String_Value2, ..., String_ValueN` The list with property names for a user entry is empty in the following cases: ■ All specified properties do not exist. ■ All specified properties are binary properties. The default value depends on the `serverType` parameter. See "LDAP Defaults for the Different Server Types" on page 34.
`groupPropAttr`	Optional. The group's properties of interest. A valid value is a comma-separated list that contains the property names: `String_Value1, String_Value2, ..., String_ValueN` The list with property names for a group entry is empty in the following cases: ■ All specified properties do not exist. ■ All specified properties are binary properties. The default value depends on the `serverType` parameter. See "LDAP Defaults for the Different Server Types" on page 34.
`ldapStartTls`	Optional. Boolean. If true, try to set up an encrypted communication over the plain LDAP port, if the LDAP server supports it. Valid values: ■ true

Parameter	Description
	■ false - Default value
`resolveGroups`	Optional. The method for finding the groups of a user using the LDAP authentication type. Valid values: ■ `CP` - This method uses a computed property field that contains all of the groups (virtually) in the user record. ■ `RU` - Default value. The recurse up method looks for a particular field (`personGrpAttr`) to find the groups in which the current entry is a direct member. ■ `RD` - The recurse down method performs an LDAP search to find all groups that have the particular user as a member. There are no more recursions performed at this time.
`computedGroupProp`	Optional. The name of an LDAP property. It is activated if `resolveGroups` is set to `CP`. Valid values: *`String_Value`* No default value.
`ldapSSLConnection`	Optional. Boolean. If true, a TLS/SSL secured communication to the LDAP server is enforced. Valid values: ■ true ■ false - Default value
`followReferrals`	Optional. Boolean. If true, try to follow LDAP server referrals.

Parameter	Description
	Valid values: ■ true - Default value ■ false
`refServerBindingType`	Optional. The kind of binding during "referral following". Valid values: ■ `same_creds` - Default value. Uses the same credentials for authentication to the next LDAP server. ■ `no_creds` - Uses anonymous binding to the next server.
`referralHopsCnt`	Optional. The count of the referral hops. If this parameter is not specified, the count is unlimited. A valid value is any positive integer. The default value is unlimited.
`useLdapTechUser`	Optional. Boolean. Enables the usage of a technical user. Valid values: ■ true ■ false - Default value
`techLdapUserCredFile`	Mandatory only if you enable the usage of a technical user. The path of the technical user credentials file. A valid value is any valid directory and file name on the file system. No default value.

Parameter	Description
	For more information, see "Creating Technical User Credential Files" on page 9.
`techLdapUserKeyFile`	Mandatory if a key file was used to create the technical user credentials file. The path of the key file. A valid value is any valid directory and file name on the file system. No default value. For more information, see "More About Key Files" on page 13.

LDAP Defaults for the Different Server Types

This section lists the default values for a number of LDAP parameters. These defaults vary according to the type of server that is used. The headings below represent the valid values for the `serverType` parameter.

Note that spaces have been added for readability.

OpenLDAP

- `personObjClass`

  `top, person`
- `groupObjClass`

  `top, groupOfUniqueNames`
- `personGrpAttr`

  `ou`
- `groupPrsAttr`

  `uniqueMember`
- `userIdField`

  `cn`
- `passwdField`

  `userPassword`
- `addPersonAttr`

```
sn:%%
```

- addGroupAttr

  no default

- personPropAttr

  ```
  cn, sn, description, telephoneNumber, seeAlso
  ```

- groupPropAttr

  ```
  uniqueMember
  ```

ActiveDirectory

- personObjClass

  ```
  top, person, organizationalPerson, user
  ```

- groupObjClass

  ```
  top, group
  ```

- personGrpAttr

  ```
  memberOf
  ```

- groupPrsAttr

  ```
  member
  ```

- userIdField

  ```
  cn
  ```

- passwdField

  ```
  unicodePwd
  ```

- addPersonAttr

  ```
  sAMAccountName:%%, userAccountControl:66048
  ```

- addGroupAttr

  ```
  sAMAccountName:%%, groupType:2
  ```

- personPropAttr

  ```
  cn, displayName, description, mail, telephoneNumber,
  physicalDeliveryOfficeName, givenName, sn, homeDirectory, ou, cn,
  description
  ```

- groupPropAttr

  ```
  member
  ```

SunOneDirectory

- `personObjClass`

  `top, person, organizationalperson, inetorgperson`

- `groupObjClass`

  `top, groupofuniquenames`

- `personGrpAttr`

  no default

- `groupPrsAttr`

  `uniqueMember`

- `userIdField`

  `uid`

- `passwdField`

  `userPassword`

- `addPersonAttr`

  `cn:%%, sn:%%`

- `addGroupAttr`

  no default

- `personPropAttr`

  `uid, cn, sn, title, description, telephoneNumber, seeAlso, postalAddress, postalCode, postOfficeBox`

- `groupPropAttr`

  `uniqueMember`

Novell

- `personObjClass`

  `top, person, organizationalPerson, ndsLoginProperties`

- `groupObjClass`

  `top, groupOfUniqueNames`

- `personGrpAttr`

  no default

- `groupPrsAttr`

  `uniqueMember`

- `userIdField`

  `cn`

- `passwdField`

  `userPassword`

- `addPersonAttr`

  `cn:%%, sn:%%`

- `addGroupAttr`

  `cn:%%, sn:%%`

- `personPropAttr`

  `cn, fullName, description, eMailAddress, telephoneNumber, departmentNumber, givenName, sn`

- `groupPropAttr`

  `uniqueMember`

ApacheDS

- `personObjClass`

  `top, person, organizationalPerson`

- `groupObjClass`

  `top, groupOfUniqueNames`

- `personGrpAttr`

  no default

- `groupPrsAttr`

  `uniqueMember`

- `userIdField`

  `cn`

- `passwdField`

  `userPassword`

- `addPersonAttr`

  `cn:%%, sn:%%`

- `addGroupAttr`

  `cn:%%, sn:%%`

- `personPropAttr`

```
cn, description, telephoneNumber
```

- groupPropAttr

```
uniqueMember
```

Tivoli

- personObjClass

```
top, person, organizationalPerson, user
```

- groupObjClass

```
top, group
```

- personGrpAttr

```
memberOf
```

- groupPrsAttr

```
member
```

- userIdField

```
cn
```

- passwdField

```
unicodePwd
```

- addPersonAttr

```
cn:%%, sn:%%
```

- addGroupAttr

```
cn:%%, sn:%%
```

- personPropAttr

```
top, person, organizationalPerson, user
```

- groupPropAttr

```
member
```