Adabas SAF Security Configuration

This document describes the ADASAF configuration.

- ADASAF Configuration Parameters
- Resource Profiles
- Using Adabas SQL Server Versions 1.4.2 and 1.4.3

ADASAF Configuration Parameters

Site-dependent ADASAF configuration parameters are specified using an assembled configuration module called SAFCFG, which is supplied as part of the SAF Security Kernel on the Adabas limited libraries.

Some ADASAF options can be overridden on a nucleus-by-nucleus basis by providing them in a dataset referenced by the DD name DDSAF.

Because of the sensitivity of SAF security, the ability to change the configuration module or the DDSAF dataset must be tightly controlled by the external security system.

For a complete description of ADASAF parameter usage, see the section ADASAF Parameters.

Resource Profiles

In order to secure Adabas, it is necessary to define resource profiles in the SAF repository. Each SAF-based security system provides the facilities required for maintaining resource profiles.

RACF enables the grouping of similar resource profiles into a resource class. CA-Top Secret and CA-ACF2 provide resource types which give equivalent functionality.

The default name of the resource class/type for Adabas is ADASEC. This can be changed by the parameter `DBCLASS` for normal checks and `NWCLASS` for cross-level checks. The maximum length of an Adabas resource name is 17, or 26 if the parameter `XLEVEL` is set to `XLEVEL=3`.

- Defining Resource Profiles
- Defining Resources to RACF
- Defining Resources to ACF2 Version 5
- Defining Resources to ACF2 Version 6
- Defining Resources to CA-Top Secret

Defining Resource Profiles

Before defining resource profiles, you need to decide which ADASAF features you will use, what format the resource names should take and which users and jobs should be allowed to access or update which resources.

At the very least you need to define, for each database protected by ADASAF:

- the class used for standard ADASAF checks (`DBCLASS`)
- nucleus execution profile, such as NUC00001.SVC249, to which the user assigned to the nucleus job has read access (to run in WARN mode) or update access (to run in FAIL mode)
- resource profiles to protect Adabas files (such as CMD00001.FIL01234). It may not be necessary to define every file individually – the various security systems allow definition of generic or wildcard resource names which can reduce the number of definitions required. The appropriate users or groups must then be granted the necessary access permissions for these resources.

If you wish to protect utilities, you must define

- resource profiles to protect each utility (such as REP00001.SVC249, SAV00001.SVC249 and so on) and ensure that User IDs which need to submit a utility for a database have read access to that database/utility's resource profile.

If you wish to use level 1 or 2 cross-level checking, you must define

- the class used for cross-level ADASAF checks (`NWCLASS`)
- resource profiles to protect Adabas files (such as CMD00001.FIL01234) and ensure that the User IDs of the *jobs* from which users are allowed to access or update these files have the correct access permissions

If you wish to protect Adabas Basic Services, you must define

- the general Adabas Basic Services profile (for example, ABS00001.GENERAL) and grant appropriate users read access to it
- the function profiles, such as ABS00001.SESSION and grant appropriate users read access to them
- if required, the subfunction profiles, such as ABS00001.PARM and grant appropriate users read and/or update access to them

If you wish to protect nucleus operator commands, you must define

- resources to protect command types (for example OPR00001.DISPLY) or individual commands (for example OPR00001.DSTAT), as required, and grant the User ID under which the nucleus job runs read access to those operator commands which are to be allowed.

The installation options defined in the configuration module (SAFCFG) determine the actual format of the resource profiles. For example, whether leading zeroes are included or not. Resources are defined using upper case characters only.

Adabas calls are protected by defining resource profiles denoting DBID and FNR. Valid access levels are READ and UPDATE.

By default, the resource class ADASEC is used for all Adabas profiles.

DBID and FNR resource profiles can be either three or five digits. Leading zeroes can be omitted. These options are set in the configuration module (SAFCFG).

Examples

Example of a 3-digit resource profile protecting file 2 of database 195:

```
CMD195.FIL002
```

Example of a 5-digit resource profile protecting file 1053 of database 19:

```
CMD00019.FIL01053
```

Example of same resource name with leading zeroes suppressed:

```
CMD19.FIL1053
```

Defining Resources to RACF

This section explains how to add the above definitions to RACF. For more details of the procedures to be followed, consult the relevant IBM documentation.

- Add Classes to Class Descriptor Table
- Update MVS Router Table
- Activate New Classes
- Assign User ID for Started Task
- Permit Users Access to Resource Profiles

Add Classes to Class Descriptor Table

Resource classes used must be added to the RACF class descriptor table. For further details, refer to the *IBM SPLRACF* documentation; an example is given in IBM SYS1.SAMPLIB, member RACINSTL. Classes must be allocated the maximum lengths as described above and be defined for discrete and generic profile use. Other attributes control the level of RACF logging and SMF recording when executing RACROUTE calls.

Update MVS Router Table

Update the MVS router table; refer to the *IBM SPL RACF* documentation. An example is given in IBM SYS1.SAMPLIB, member RACINSTL, section RFTABLE.

Activate New Classes

Activate new resource classes with SETROPTS (see IBM *RACF Command Language Reference* documentation). For example, activate class ADASEC:

```
SETROPTS CLASSACT(ADASEC)
```

```
SETROPTS GENCMD(ADASEC)
```

```
SETROPTS GENERIC(ADASEC)
```

Assign User ID for Started Task

The Adabas nucleus started task or batch jobs require a User ID having the relevant RACF authorizations including the ability to perform `RACROUTE,TYPE=EXTRACT` calls on profiles belonging to the above classes.

Permit Users Access to Resource Profiles

Add profiles to protect the different resources and permit users the required level of access. The following RACF commands add resource profile CMD00001.FIL01234 and grant User ID user2 READ access:

```
RDEFINE ADASEC CMD00001.FIL01234 UACC(NONE)
```

```
PERMIT CMD00001.FIL01234 CLASS(ADASEC) ACCESS(READ) ID(user2)
```

Defining Resources to ACF2 Version 5

This section explains how to define resources to ACF2 version 5. For more details, consult the relevant ACF2 documentation.

- Add Task User ID to ACF2
- Activate the SAF interface
- Insert the SAFSAVE Record
- Insert the SAFPROT Record
- Insert the SAFMAPS Record
- Define Resource Rules

Add Task User ID to ACF2

Adabas normally executes as an MVS started task. Define user ID of started task to ACF2 with the following attributes. To avoid the `NON-CNCL` attribute, APAR TW95626 must be applied.

```
MUSASS,NON-CNCL,STC
```

Activate the SAF interface

```
GSO OPTS - SAF
```

Insert the SAFSAVE Record

Switch off all SAF checks:

```
SAFSAVE CLASSES(-) CNTLPTS(-) SUBSYS(-)
```

Insert the SAFPROT Record

Switch on the SAF security checks for Adabas:

```
CLASSES(-) CNTLPTS(-) SUBSYS(ADARUN)
```

Insert the SAFMAPS Record

For the general resource class names used by ADASAF define a 3-character ACF2 resource type code:

```
SAFMAPS MAPS(ADA/ADASEC)
```

Define Resource Rules

Define the required security profiles in ACF2 using the new type code.

The following example shows resource CMD00001.FIL01234 being added allowing read access for user ID user2:

```
$KEY(CMD00001.FIL01234) TYPE(ADA) UID(user2) ALLOW SERVICE(READ)
```

Defining Resources to ACF2 Version 6

This section explains how to define resources to ACF2 version 6. For details of the procedures required, consult the current ACF2 documentation.

- Add Task User ID to ACF2
- Insert the SAFDEF Record
- Insert the CLASMAP Record
- Define Resource Rules

Add Task User ID to ACF2

Adabas normally executes as an MVS started task. Define User ID of started task to ACF2 with the following attributes:

```
MUSASS,STC
```

Both ACF2 6.1 and 6.2 no longer need TW95626 since the current versions are more SAF-compliant.

Insert the SAFDEF Record

The SAFDEF record must be inserted as follows:

```
SAFDEF FUNCRET(4) FUNCRSN(0) ID(SAFGWAY) MODE(GLOBAL)
      RACROUTE(REQUEST=AUTH SUBSYS=ADARUN- REQSTOR=- ) RETCODE(4)
```

Insert the CLASMAP Record

For the general resource class names used by ADASAF define a 3-character ACF2 resource type code:

```
CLASMAP ENTITYLN(0) MUSID() RESOURCE(ADASEC) RSRCTYPE(ADA)
```

Define Resource Rules

Define the required security profiles in ACF2 using the new type code.

The following example shows resource CMD00001.FIL01234 being added allowing read access for User ID user2:

```
$KEY(CMD00001.FIL01234) TYPE(ADA) UID(user2) ALLOW SERVICE(READ)
```

Defining Resources to CA-Top Secret

This section describes the definition of resources to CA-Top Secret. For more details, consult the relevant CA-Top Secret documentation.

- Defining Resources to CA-Top Secret Facility
- Assign User ID for Started Task
- Add Procedure Name for Started Task
- Add Resource Type to Resource Definition Table
- Assign Ownership of Resources
- Permit Access to Defined Resources

Defining Resources to CA-Top Secret Facility

ADASAF issues authorization checks against a specific CA-Top Secret facility. The default facilities are batch and STC. Different facilities can be defined enabling, for example, development and production environments to be secured separately. Additional facilities are added by modifying predefined models. The following attributes are important and should be assigned when modifying facilities for ADASAF:

```
NAME=fac,AUTHINIT,MULTIUSER,NONPWR,PGM=ADA,NOABEND
```

Assign User ID for Started Task

Add one User ID for each instance of the Adabas started task. If required, different facilities can be assigned to development and production tasks. The designated facility is assigned to the started task User ID:

```
TSS CRE(user-id) DEPT(dept) MASTFAC(fac)
```

Add Procedure Name for Started Task

The procedure name under which the Adabas started task executes must be defined to CA-Top Secret. Different procedure names are suggested when securing different environments separately with the use of non-default CA-Top Secret facilities:

```
TSS ADD(STC) PROC(proc) USER(user-id)
```

Add Resource Type to Resource Definition Table

Resource types must be added to the CA-Top Secret resource definition table (RDT). Resource definitions relating to Adabas are kept in resource type ADASEC. Refer to *CA-Top Secret Reference Guide* for a detailed explanation of the following command and arguments:

```
TSS ADD(RDT) RESCLASS(ADASEC) RESCODE(HEXCODE) ATTR(LONG)
      ACLST(NONE,READ,CONTROL) DEFACC(NONE)
```

Assign Ownership of Resources

Assign ownership to a particular resource as shown in the following example. This must be done before permitting access to defined resource profiles:

```
TSS ADD(user1) ADASEC(CMD00001.FIL01234)
```

This makes user user1 owner of resource CMD00001.FIL01234.

Permit Access to Defined Resources

Access to resource profiles is granted as follows. The example gives User ID user2 read access to database 1, file 1234.

```
TSS PER(user2) ADASEC(CMD00001.FIL01234) FAC(fac) ACCESS(READ)
```

The facility argument can be omitted if Adabas operates under the master facility Batch or STC.

Using Adabas SQL Server Versions 1.4.2 and 1.4.3

This section describes how to use Adabas SAF Security to protect calls received from Adabas SQL Server, based on the SQL table name. This feature is only available for Adabas SQL Server versions 1.4.2 and 1.4.3 operating on z/OS, OS390 or compatible platforms.

Using ADASAF, together with Entire Service Manager and Adabas SQL Server, it is possible to

- Authenticate Adabas SQL server clients
- Authorize SQL table access
- Authorize SQL table update

Security authorization is sought for all requests processed by Adabas SQL Server. Commands are routed to their final Adabas target only if the originating user's User ID and password are authenticated and the user has sufficient authorization for the requested SQL objects. Otherwise, the request is returned to the calling application with SQL response code –3004.

SQL objects are defined to the host security repository. Users are given the required access level. The SQL object name is prefixed with the name of the database as defined in the SQL catalog. Using this convention development users can, for example, be restricted to development databases. Access to data is controlled according to an individual's read/update authorization for each SQL object within the SQL database.

The following tasks are necessary when securing access through Adabas SQL Server:

- define Adabas SQL Server protection configuration options, as described below
- attach security component to Entire Service Manager (ESG)
- make the necessary definitions in your SAF security system, as described below
- configure an ADASAF-protected nucleus to provide SQL table security

Note:
The SQL `connect` statement must be utilized by the calling application to provide User ID and password

Configuration Options

In addition to the configuration options described in this document and the general options described in the *SAF Security Kernel* documentation, the following SAFCFG options are relevant for protection of Adabas SQL Server tables:

SAFCFG Option	Description
`SQNCU=0`	Specify the number of SQL table checks to be buffered per user, in the cache defined by `GWSIZE`.
`SQUNI=Y|N`	Whether or not to allow access to undefined tables (where the security system differentiates between undefined and disallowed resources).
`SQTBCAT=Y|N`	Determine table names from SQL catalog. You should set this to Y.
`SQCLASS=ADASEC`	Specify the name of the resource class to be used for checking SQL tables. This name should be up to eight alphanumeric characters and may be the same as or different from that used for database checks (as specified by the `DBCLASS` parameter).
`GWDBID=`*`nnnnn`*	Specifiy an ADASAF-protected database for which SQL table authorization is to be performed.

Entire Service Manager (ESG) Considerations

The security component attached to ESG requires 32K of storage from the ESG thread.

You will need a reentrant ADALNK (created from ADALNKR in the Adabas source library). Ensure that the equates LNUINFO, LRVINFO, LEXITB, LEXITA and LXITAFP are all 0 and that SVCNR is set to your Adabas SVC number.

You must then link the Adabas SQL Server security module, NA2PESQ, using the following linkage editor control statements:

Statement	Description
`INCLUDE AAFLIB(NA2PSQ1)`	supplied ADASAF load library
`INCLUDE CFGLIB(SAFCFG)`	your assembled SAFCFG module
`INCLUDE ADALIB(ADALNKR)`	reentrant ADALNK
`NAME NA2PESQ`	

Finally, ensure that NA2PESQ and ESQUEX5 (from the ADASAF load library) are available in one of the load libraries in ESG's COMPLIB concatenation and that they are both named in ESG's SYSPARM member as RESIDENTPAGE modules.

Configuring the ADASAF-Protected Nucleus

The nucleus specified with `GWDBID` must be configured to handle SQL table authorization requests. The SQL table names (together with their corresponding Database IDs and file numbers) must first be extracted from the SQL catalog. This can be done using ESQBIF in a batch job, with SYSIN similar to the following:

```
SELECT CAT02_TABLE_ID, CAT02_DB_NR, CAT02_FILE_NR, CAT02_SCHEMA_ID

 FROM DEFINITION_SCHEMA.ESQCAT1 WHERE CAT02_TABLE_TYPE EQ 'BASE TABLE';
```

Direct the output (SYSPRINT) to a dataset and allocate this dataset to DD name SEFDD1 in the JCL that executes the specified nucleus.

If your SQL catalog regularly changes, you can add a step to execute ESQBIF before executing the Adabas nucleus and pass the output through to the nucleus step's SEFDD1 DD statement.

Security Definitions

ADASAF authorizes SQL tables against the class specified for SQCLASS (default ADASEC), using resource names of the format:

```
Database-name.table-name
```

For example

```
ESQ1.CRUISE
```

Note:
ADASAF always operates in fail mode for SQL table protection.