



Safetronic

AUTHENTICATION SERVER

Salt Safetronic Authentication Server User Guide

Version 21.1

June 2021

© 2021 Salt Group Proprietary Limited. All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished under a licence agreement or nondisclosure agreement. Copies of software supplied by Salt Group must not be released to any party without written authorisation from Salt Group and remain the property of Salt Group for all time. They may not be transferred to any computer without both a service contract for the use of the software on that computer being in existence and written authorisation from Salt Group.

Copies of software released by Salt Group to academic establishments may not be used for any commercial purpose without written authorisation from Salt Group and royalty payments made to the company.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Salt Group.

Whilst Salt Group has made every effort in the preparation of this manual to ensure the accuracy of the information, the information contained in this manual is delivered without warranty, either express or implied. Salt Group will not be held liable for any damages caused, or alleged to be caused, either directly or indirectly by this manual.

Licences and Trademarks

All other brands and their products are trademarks or registered trademarks of their respective holders and should be noted as such. All other trademarks acknowledged.

Contents

CHAPTER 1: ABOUT THIS GUIDE	17
AUDIENCE.....	17
DOCUMENTATION SET	17
TYPOGRAPHICAL CONVENTIONS	18
CHAPTER 2: SETUP STEPS	18
PRODUCTION AND TEST ENVIRONMENTS.....	20
CHAPTER 3: PREREQUISITES AND SYSTEM REQUIREMENTS.....	21
LICENCES.....	21
License configuration for RMI.....	22
License configuration for SOAP	22
Adding client licenses when using SOAP.....	22
SUPPORTED PLATFORMS AND CRYPTO MODULES	23
DATABASES.....	23
JAVA ENVIRONMENT	24
Oracle Java components.....	24
OpenJDK Java components.....	24
SOFTWARE TOKEN	24
CRYPTO MODULES.....	25
nShield Solo and nShield Connect.....	25
OTHER SYSTEMS	26
AUTHENTICATION TOKENS	26
CHAPTER 4: INSTALLING AND UPGRADING SSAS	27
UPGRADING SSAS.....	27
SETTING UP YOUR JAVA ENVIRONMENT	27
Installing Java	27
Preparing Oracle Server JVM.....	28
Testing the JDK environment.....	29
INSTALLING SSAS	29
Normal mode.....	29
Console mode.....	30
Silent mode	30
INSTALLED FILES AND LIBRARIES	30
Library details.....	32
INSTALLING SSAS AS A WINDOWS SERVICE.....	34
INSTALLING SSAS AS SERVICE ON LINUX.....	34
UPDATING SSAS WITH EXISTING SALT MOBILE/STANDARD CONNECTOR INSTALLATION	34
REMOVING SSAS.....	35
CHAPTER 5: CONFIGURING THE SERVER AND DAEMON INTERFACES	37
SERVER INTERFACE.....	37
NAMING SERVICE AND NUMBER OF DAEMONS	37
Multiple daemons sharing an external RMIRegistry.....	42
Single daemon using JNDI.....	44
Multiple daemons using JNDI.....	46
TLS (SSL).....	47
Privacy.....	48
Request authentication	49

CHAPTER 6: CONFIGURING THE CRYPTO MODULE (CHANNEL TOKEN)	50
nSHIELD SOLO AND nSHIELD CONNECT	51
<i>Using an nShield HSM to protect Vasco token blobs</i>	51
<i>Environment variables</i>	52
TOKEN SHARING	53
REMOTE TOKENS	53
CHAPTER 7: SETTING UP THE DATABASES	54
GETTING THE JDBC DRIVERS	54
CONFIGURING THE CLASSPATH	54
PREPARING THE DATABASE	55
<i>Database schema</i>	55
TABLES REQUIRED BY SSAS	56
<i>Event and error log tables</i>	56
<i>Tamper evidence tables</i>	56
<i>Application-specific logging table</i>	56
<i>Service tables</i>	57
CHAPTER 8: SALT MOBILE SERVICE	60
CHAPTER 9: EMV AUTHENTICATION	61
MAIN CONFIGURATION PROCEDURES	61
CONFIGURING THE MASTERCARD EMV SECURECODE AUTHENTICATION MECHANISM	62
<i>Providing the EMV card information</i>	62
<i>Configuring the EMV CAP AAC Issuer profile</i>	63
<i>Setting EMV CAP card resynchronisation</i>	64
CHAPTER 10: VASCO SERVICE	65
MAIN CONFIGURATION PROCEDURES	65
CONFIGURING ADDITIONAL SETTINGS FOR VASCO	66
<i>Configuring the VACMAN Controller</i>	66
<i>Activating the Java interface to the hardserver</i>	66
<i>Configuring the VACMAN storage key</i>	67
IMPORTING AND MANAGING VASCO TOKENS	67
<i>Overview of the Vasco token process</i>	67
<i>Configuring the IMC to show the required token types</i>	68
<i>Importing the tokens</i>	68
<i>Managing the tokens</i>	68
<i>Vasco token fields</i>	69
CHAPTER 11: TEMAC SERVICE	70
MAIN CONFIGURATION PROCEDURES	70
CONFIGURING THE DATABASE FOR TEMAC	71
CHAPTER 12: OATH SERVICE	72
MAIN CONFIGURATION PROCEDURES	72
CONFIGURING ADDITIONAL SETTINGS FOR THE OATH SERVICE	73
IMPORTING AND MANAGING OATH TOKENS	73
<i>Overview of the OATH token process</i>	73
<i>Configuring the IMC to show the required token types</i>	73
<i>Importing the tokens</i>	74
<i>Managing the tokens</i>	74
<i>OATH token fields</i>	74
CHAPTER 13: ACTIVIDENTITY SERVICE	76
MAIN CONFIGURATION PROCEDURES	76
CONFIGURING ADDITIONAL SETTINGS FOR THE ACTIVIDENTITY SERVICE	76
IMPORTING AND MANAGING ACTIVIDENTITY PROFILES AND TOKENS	77
<i>Overview of the ActivIdentity token process</i>	77

Configuring the IMC to show the required token types.....	77
Creating the Issuer profiles.....	78
Importing the tokens.....	78
Managing the tokens.....	79
ActivIdentity profile fields.....	79
ActivIdentity token fields.....	82
ACTIVITY SERVICE KEY ROLLOVER.....	83
CHAPTER 14: PASSWORD, SAML, AND GENERIC MAC SERVICES	85
PASSWORD SERVICE.....	85
Main configuration procedures	85
Configuring additional settings for the Password Service.....	86
SAML GENERATION SERVICE.....	86
Main configuration procedures	86
SAML service configuration	87
Setting up the SAML signing key.....	88
Configuring additional settings for the SAML generation Service.....	88
GENERIC MAC SERVICE.....	88
Main configuration procedures	89
Configuring additional settings for the Generic MAC Verification Service.....	89
CHAPTER 15: PKI SERVICES	90
PKI SIGNATURE GENERATION SERVICE.....	90
Main configuration procedures	90
Generating and processing a key pair and certification request.....	91
Configuring additional settings for the Signature Service	92
PKI SIGNATURE VERIFICATION SERVICE.....	93
PKI signature verification for IdenTrust.....	93
PKI signature verification for BankID.....	94
XML digital signature (XMLDSig) verification	94
Configuring PKI trust.....	94
Main configuration procedures	95
Revocation checking guidance.....	96
Adding CA or FI certificates to a token.....	97
Setting certificate security levels.....	98
Verifying XML documents containing multiple digital signatures	100
Configuring additional settings for the Verify Signature and Random number services.....	101
CHAPTER 16: WEBPIN SERVICE	102
CHAPTER 17: ADES SERVICES	103
ADES SIGNATURE GENERATION SERVICE.....	103
MAIN CONFIGURATION PROCEDURES	103
REVOCATION CHECKING GUIDANCE.....	104
CONFIGURING ADDITIONAL SETTINGS FOR THE ADES VERIFICATION SERVICE.....	105
CHAPTER 18: SSAS MANAGEMENT CONSOLE	106
ABOUT THE CONSOLE.....	106
User interface.....	106
Menus and user tasks.....	109
DAEMON AND CONSOLE.....	110
Starting the SSAS daemon.....	110
Starting the Management Console	110
Configuring the daemon	111
SERVER PROPERTIES FILES.....	113
Mandatory and optional configuration settings	114
Sample SSAS configurations.....	114
Opening an existing server properties file	115
Copying a server properties file.....	115
SERVER.....	115

Creating a server.....	115
Starting and stopping the server.....	117
Deleting a server.....	118
CHANNEL.....	118
Creating a channel.....	118
Setting channel passwords.....	121
Deleting a channel.....	122
SERVICE.....	122
Adding a Service to a channel.....	122
Creating, selecting, and managing Key Encryption Keys (KEKs).....	127
Creating, selecting, and managing audit keys (TEMAC keys).....	128
Removing Services from a channel.....	128
CHANNEL SELECTORS.....	129
Assigning services to the channel selectors.....	130
Mapping channel selectors to the required channels.....	132
Reassigning services to different channel selectors.....	132
DATABASE CONNECTION POOLS.....	133
Configuring the default pool properties.....	133
Configuring a database connection pool.....	133
CHANGING AN OBJECT TYPE.....	134
CONFIGURING X.509 STORES.....	135
CONFIGURING USER AUTHENTICATION.....	135
User Authentication using the SOAP API.....	136
User Authentication using the RADIUS API.....	136
Analysing test results.....	160
Checking a token.....	160
Checking TLS connectivity.....	160
Checking a signature message file.....	161
Verifying logs.....	162
CHAPTER 19: TOKEN MANAGEMENT CONSOLE.....	164
ABOUT THE CONSOLE.....	164
User interface.....	164
Icons and user tasks.....	168
OPENING THE TMC AND LOGGING INTO A TOKEN.....	170
STANDARD TASKS.....	171
Generating a key pair and certification request for a token.....	171
Processing a certification response for a token.....	172
Importing a certificate from a file to a token.....	173
Importing a certificate from an LDAP server to a token.....	173
Verifying that a key is usable.....	175
Backing up a token.....	175
Restoring a token from a backup.....	175
Deleting an object from a token.....	175
ADVANCED TASKS.....	176
Changing the password for a token.....	176
Generating a secret key (symmetric key).....	176
Importing a secret key (symmetric key) to a token.....	177
Importing a wrapped key to a token.....	179
Exporting a key (wrapped) from a token.....	180
Exporting a certificate from a token.....	181
Importing a data object to a token.....	181
Exporting a data object from a token.....	181
Generating a new certification request for an existing private key (re- certifying).....	182
Importing a key and its certificates from a PKCS #12 file into a token.....	183
SETTING UP CLIENT CRYPTOGRAPHIC TOKENS.....	183
Starting a stand-alone TMC.....	183
Recommendations for configuring TLS between clients and servers.....	185
Setting up a client cryptographic token for digest authentication.....	185
Setting up a client cryptographic token for signature authentication.....	186

CHAPTER 20: TOKEN MANAGEMENT COMMAND LINE INTERFACE	187
ABOUT THE TMCLI	187
USING THE TMCLI	187
<i>Properties</i>	188
<i>Prefix</i>	189
COMMANDS	189
<i>Summary</i>	189
<i>List all keys and certificates</i>	190
<i>Generating a secret key</i>	190
<i>Exporting a secret key</i>	191
<i>Importing a secret key (symmetric key) to a token</i>	192
<i>Generate a key pair and certification request</i>	194
<i>Process a certification response</i>	194
<i>Add a certificate</i>	194
<i>Key test</i>	195
<i>Delete a key or certificate</i>	195
CHAPTER 21: IDENTITY MANAGEMENT CONSOLE	196
ABOUT THE CONSOLE	196
<i>Properties files</i>	196
<i>User interface</i>	197
<i>Menus and user tasks</i>	198
OPENING THE IMC	198
<i>Starting a stand-alone IMC</i>	198
CONFIGURING THE IMC TOKEN TYPE TO CHANNEL MAPPINGS	199
VIEWING, FILTERING, AND SORTING THE TOKENS	199
<i>Refreshing the list pane</i>	200
IMPORTING TOKENS FOR A CHANNEL	201
VIEWING DETAILS FOR A TOKEN	201
UPDATING A TOKEN	201
DELETING A TOKEN	201
CHAPTER 22: LOG VIEWING CONSOLE	202
ABOUT THE CONSOLE	202
<i>User interface</i>	202
<i>Menus and user tasks</i>	203
OPENING THE LVC	203
OPENING A SAVED LVC WORKSPACE	204
SAVING THE LVC WORKSPACE	204
PERFORMING A LOG SEARCH	204
PERFORMING A REAL-TIME (UPDATING) LOG SEARCH	210
VERIFYING THE INTEGRITY OF A LOG	211
CHAPTER 23: CONFIGURING THE CLIENT- SERVER COMMUNICATION	212
CONFIGURING THE CLIENT MAPPINGS	212
CONFIGURING TLS (SSL)	213
<i>Configuring server-side TLS</i>	213
<i>Configuring client-server TLS</i>	215
<i>Configuring the client</i>	215
CHAPTER 24: CONFIGURING FOR OPTIMUM SECURITY	217
GENERAL	217
<i>Server</i>	217
<i>Logs and token data</i>	217
<i>Crypto module</i>	225
SETTINGS	225
<i>Server</i>	225
<i>Environment settings</i>	225
<i>Channel</i>	226

Services.....	227
CHAPTER 25: CONFIGURING SSAS FOR SQL SERVER	229
USING MICROSOFT SQL SERVER.....	229
USING SQL SERVER MIRRORING	229
Types of mirroring.....	229
Configuring SQL server mirroring.....	230
High performance (asynchronous) mirroring issues.....	230
CHAPTER 26: CONFIGURING THE RADIUS INTERFACE	231
RADIUS DEPLOYMENT SCENARIOS.....	231
Third-party RADIUS server already exists	231
Clean setup: end-server	232
Clean setup: front-server	232
PERFORMANCE CACHING	233
RADIUS PROXY SERVER	233
IMC SETTINGS	234
IMC PLUG-IN SETTINGS.....	234
Identifying which plug-in files need to be modified.....	234
Identifying the key values to use.....	236
IMC RADIUS ENTITIES.....	236
RadiusClient.....	237
RadiusUser	237
LDAP	238
LDAPSchema.....	238
RadiusProxyCriteria	239
MANAGEMENT CONSOLE PROPERTY SETTINGS.....	239
APPENDIX A: SSAS COMMAND LINE UTILITIES	240
UTILITIES.....	240
SSAS Consoles.....	240
Running the server and daemon	240
Simple client applications.....	241
Additional utilities.....	241
PARAMETERS	242
APPENDIX B: TROUBLESHOOTING SSAS	243
ISSUES STARTING THE DAEMON	243
The daemon does not start.....	243
The daemon cannot be located.....	244
ISSUES STARTING THE SERVER	244
Missing or incorrect licences.....	240
Incorrect token settings.....	240
ISSUES WITH CLIENT- SERVER COMMUNICATION.....	241
GLOSSARY	242

Chapter 1: About this guide

This guide describes:

- The Salt Safetronic Authentication Server (SSAS) system requirements and how to install, upgrade, and remove SSAS
- The common service implementations and how to set them up
- The SSAS management consoles
- The main user tasks for configuring SSAS and your services
- How to troubleshoot common situations

Audience

This guide is for SSAS administrators. Read this guide to learn about:

- Installing and configuring the server
- The SSAS management consoles and the main user tasks you can perform using them

Documentation set

This guide is part of the main SSAS documentation suite:

Guide	Format	Content
<i>SSAS Concepts Guide</i>	PDF	Introduction to SSAS, its components, and the supported authentication tokens and methods. Deployment examples.
<i>SSAS User Guide</i>	PDF	Information for installing SSAS and preparing it for configuration. Procedures for configuring SSAS and setting up the authentication services, using the SSAS management consoles.
<i>SSAS Reference Guide</i>	PDF	Additional technical and reference information for advance configuration.
<i>SSAS Developer Guide for SOAP</i>	PDF	Programming information for the SSAS SOAP interface.
<i>SSAS API Reference</i>	HTML	Technical and reference information for the SSAS software API. Programming information for the SSAS RMI interface.

We welcome your feedback about any omissions or enhancements to these guides. It helps us improve them. Please provide your feedback to Salt Support.

Typographical conventions

Note: The word Note indicates important supplementary information.



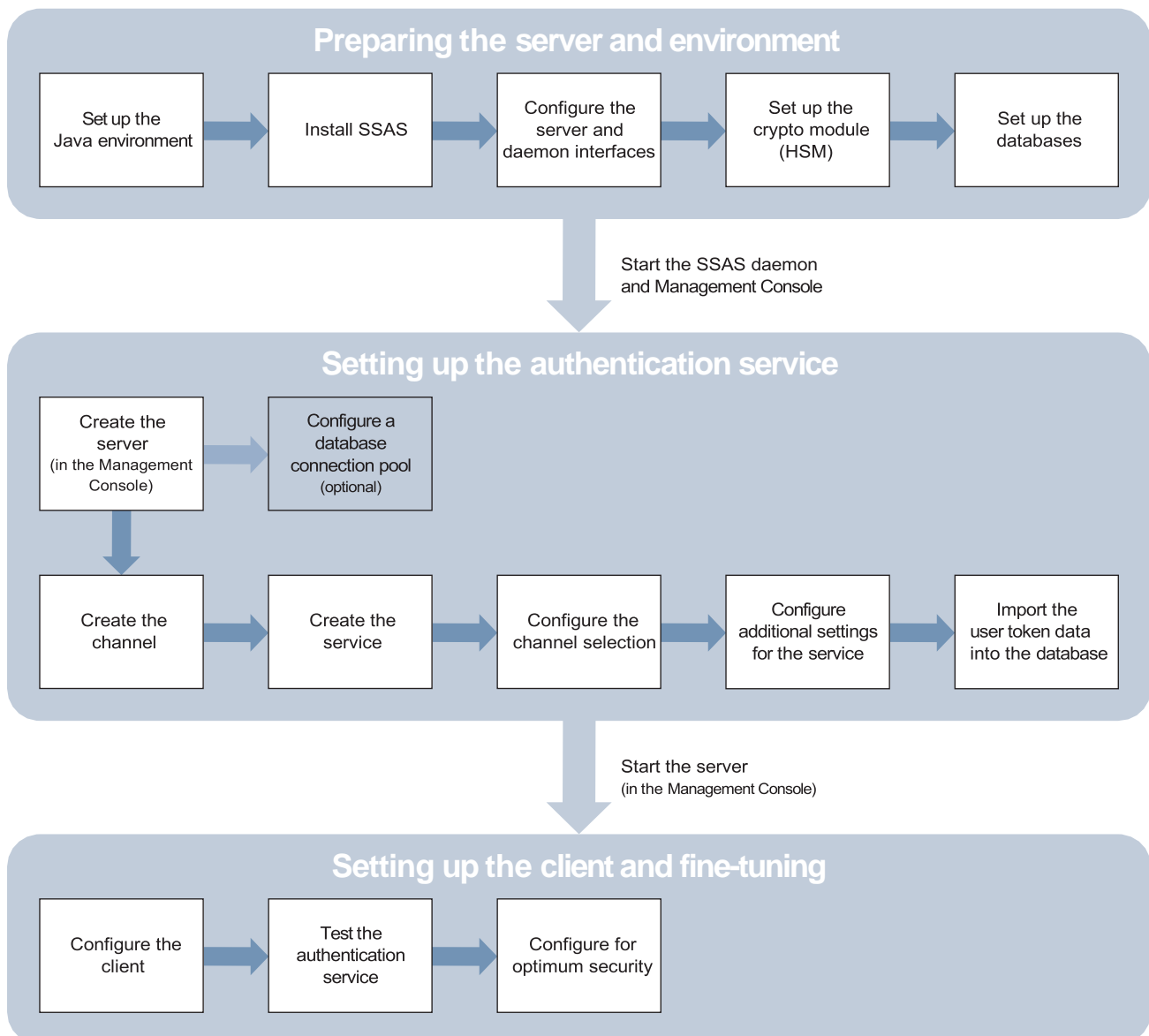
If there is a danger of loss or exposure of key material, or any other security risk, this is indicated by a security triangle in the margin.

Replaceable content is shown as: *replaceable*.

Chapter 2: Setup steps

Figure 1 shows the main stages and tasks for installing and configuring SSAS for a service implementation.

Figure 1. SSAS main setup steps



The chapters in this guide are organised according to this flow:

- Procedures for installing SSAS, preparing the server and daemon interfaces, and setting up the crypto module
- Guidance for each of the common service implementations
- Descriptions and user task procedures for each of the SSAS management consoles

We recommend that you read the guidance for your desired service implementation to understand what needs to be configured *before* configuring those elements within SSAS. However, if you want to explore the SSAS management consoles first, there are some sample properties files to help you do that. See *Sample SSAS configurations*.

Production and test environments

You can use the configuration procedures described in subsequent chapters to configure SSAS for both a production environment and a testing environment. We recommend that you test your configuration setup initially in a pre-production setting, allowing time to verify connections between participants and troubleshoot application interactions.

You can set up multiple configuration directories using server properties files customised for each purpose. For example, if you want to create a pre-production environment, you can create a configuration directory called `Preproduction` and import the appropriate server properties file into the Management Console. You can then make the necessary setting changes, and save it as `Preprod-SSASServer.properties` (to prevent any naming confusion). For more information, see *Testing your configuration*.

Chapter 3: Prerequisites and system requirements

Licences

To use SSAS, you must install a runtime license file that contains licences for the product categories you are going to use. To obtain the license file:

1. Check that the paper license supplied with SSAS matches your requirements and read the terms and conditions.
2. Send an email to Salt, requesting the license file and including:
 - The IP address (or DNS name) of the computer that is to run the server or client
 - The order number
 - The part numbers shown on the paper licence
3. If you need to upgrade a license file, attach the license file to the email.

When you receive an email response with an attached license file, copy the file to your SSAS machine and specify its location and name when creating a server (in the SSAS Management Console).

For evaluation licences, you must obtain approval from your Account Manager, and then contact Salt Support for a time-limited evaluation licence.

The license file defines the categories listed below.

Category	Relatest to...
AtClient	Client functionality
AtServer	The server, including the TC2/OCSP revocation checkers and the Signing and Random Number services
AtGenMacService	The EMV authentication service and others
AtSAML	The SAML authentication generation services

The license file also controls the following features:

- Number of allowed channels per server
- Privacy of client-server communication
- Authentication of client requests by the server
- Purchased services
- Client signing key selector, which selects signing keys depending on the calling application's certificate

License configuration for RMI

You can place the license file in any location accessible by the applications. You must specify the file location in the properties files for each server and client.

License configuration for SOAP

When using SOAP, the client licences are checked by, and consequently need to be accessible to, the server. You must set the following properties in the server configuration:

Property	Description
Soap Client Licenses File	The file containing one or more client licences
Soap License Refresh Interval	How often the client license file is checked for new licences

Adding client licenses when using SOAP

When using the SOAP interface, client licences are stored in a license file on the server side. The license file is an XML file where each license element is enclosed by a <License> tag.

To add licences for more clients, you must add the new license elements to this file. To do this, simply copy the new <License> element (contained in the <License> tags) and add it to the list of license elements in the license file.

The **!!ADD YOUR NEW LICENSE ELEMENT HERE!!** statement in the following sample license file shows where a new license element is inserted:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Licenses PUBLIC "-//THALES//SAFESIGN AUTHENTICATION SERVER LICENSE DTD//EN"
"http://www.thales-esecurity.com/SSAS/license/1.1/license.dtd">
<?xml-stylesheet type="text/xsl" href="http://www.thales-esecurity.com/SSAS/license/1.1/license.xsl"?>
<Licenses version="1.0">
  <License><LicenseData><LicenseHolder>client </LicenseHolder>
    <ProductName>AtClient 4.1</ProductName>
    <MachineID>192.168.0.23</MachineID></LicenseData>
    <Verification>pNjv6J575z6gWqoHnhdPTCnz9l7TM/otjTNJxYdkpdE85y
ZiZ9fNeYa6HejvmDQ+VFVDnMD/+J0nO8upoWA5Fd8yWDHz/IJ9ZrN38MOoIkC2fX+COSu2m5VJ
PDx2rsYXp863L1YgdTrs=</Verification>
  </License>
  <License>
    !!ADD YOUR NEW LICENSE ELEMENT HERE!!
  </License>
  ...
</Licenses>
```

Supported platforms and crypto modules

We have tested SSAS with these operating systems, Java versions, and Thales crypto modules and Hardware Security Modules (HSMs):

Operating system	Java 1.8	nShield [®] HSMs
Microsoft Windows 2012 Server R2 (x64)	Oracle 64-bit OpenJDK 64-bit	Y
Microsoft Windows 2016 Server (x64)	Oracle 64-bit OpenJDK 64-bit	Y
Microsoft Windows 2019 Server (x64)	Oracle 64-bit OpenJDK 64-bit	Y
Oracle Solaris x86 v10	Oracle 64-bit	Y
Red Hat Enterprise Linux 6.10 kernel 2.6.32-754.33.1.el6.x86_64	Oracle 64-bit OpenJDK 64-bit	Y
Red Hat Enterprise Linux 7.9 kernel 3.10.0-1160.11.1.el7.x86_64	Oracle 64-bit OpenJDK 64-bit	Y
Red Hat Enterprise Linux 8.4 kernel 4.18.0-305.el8.x86_64	Oracle 64-bit OpenJDK 64-bit	Y

This release supports other versions of the above operating systems as well as other mainstream operating systems, but Salt cannot guarantee compatibility. For more information, contact Salt Support.

Note: In Headless and Server environments, some of the SSAS administration tasks require a GUI. On Unix-based platforms, you can achieve this by exporting an X11 display for the graphical Java applications.

Databases

We have tested SSAS with these databases. If you require support for these databases, contact the database supplier.

Database	Notes
Microsoft SQL Server 2012, 2016, 2017, and 2019	Supports all SSAS functionality
Oracle 11g R2, 18c, and 19c	Does not support SafeSign Personal Security Module (PSM)
PostgreSQL 13	Supports all SSAS functionality

When using Oracle 18c/19c, the following JVM option needs to be added to disable autocommit:
`-Doracle.jdbc.autoCommitSpecCompliant=false.`

This can be added to the **THALES_SSAS_JVM_OPTIONS** in the `ssas-run-daemon.sh/bat` script.

This release may support other database versions. For more information, contact Salt Support.

Java environment

For information about the supported Java versions, see the *Supported platforms and crypto modules* table above.

Oracle Java components

The Java components for Windows, Solaris or Linux are:

- Oracle Java Runtime Environment (JRE) 1.8.0 update 291
- For Oracle JRE 1.8.0 updates earlier than 162, Oracle Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files is required. These components are available from <https://www.oracle.com/java/technologies/javase-jce8-downloads.html>

OpenJDK Java components

The Java components for Windows, Solaris or Linux are:

- OpenJDK 1.8.0 update 292

Software token

SSAS allows the use of a software token, an encrypted file, for testing your applications and verifying aspects of your configuration before you enter a production environment. However, we recommend you use hardware tokens (crypto modules), not software tokens, in production environments.

To use a software token with a Vasco service, you require the VACMAN Controller software version 3.19, which you can order from Vasco.

SSAS provides a sample software token in the `<SSAS install>\samples` directory. The password for this token is: **password**.

Crypto modules

For compliance with some authentication systems, such as the IdenTrust specification, the Relying Customer must use a hardware token (crypto module). SSAS behaves the same regardless of whether you use hardware or software tokens.

The nShield crypto modules and HSMs can support the cryptographic requirements of all SSAS services.

nShield Solo and nShield Connect

To use these nShield HSMs with SSAS, you must order the following options with the HSMs from Entrust:

- **NClient Licenses**, where *N* is the number of clients able to connect to the HSM, such as the SSAS machine, key management devices, and clients requiring key access.
- For the Vasco service **SEE Activation (EU+10)**, which is only available to customers in the Community General Export Area (CGEA, also known as EU+10). This feature includes the Thales CodeSafe® developer product, which enables you to develop, sign, and run SEE™ applications on the HSM.
Alternatively, **SEE Activation (Restricted)** is available to customers in any part of the world. This feature requires you to send your SEE application to Salt Group for signing so that you can use it on the HSM.

To use the HSM install the Security World Software™ (provided with the HSM) on the SSAS machine, which provides the appropriate nShield PKCS #11 implementation library. SSAS is compatible with Security World Software version 11.72 and later. When using the EMV service, only Security World 12.60.11 or later is supported.

You use the Security World Software to create a Security World™ with the HSM. However, you must perform all the key management operations for the HSM using the standard Token Management Console (TMC), which is included with SSAS.

Vasco service

To use the HSM with the Vasco service, you use CodeSafe to create a published *Secure Execution Engine* (SEE) application, known as a *SEE machine*, on the HSM and use it to protect the Vasco token blobs.

This deployment requires the VACMAN Controller software version 3.19, which you can order from Vasco.

For more information, see *Using an nShield HSM to protect Vasco token blobs*.

OATH service

Oath tokens are protected by a DES3 or AES key when using the nShield HSM.

For more information, see *Chapter 12: OATH service*.

Other systems

We have tested SSAS with the following components and systems. If you require support for these components, contact the supplier of these components.

Component	Tested system
Card management system	MyID
Certification Authority (CA, or Certificate Authority)	Conforming to the X.509 Certificate format defined in RFC 2459 and RFC 3280
LDAP directory	iPlanet
OCSP responder	Valicert, EJBCA

SSAS might support other system components, such as alternative LDAP servers. Contact Support for confirmation concerning your desired system components.

Authentication tokens

We have tested SSAS with the following authentication tokens. This is not intended to be an exhaustive list and we are updating this list on a regular basis. For further details on the specifications of the tokens supported, contact Salt Group.

- OATH tokens: HMAC-based one time password (HOTP), Time-based one time password (TOTP), OATH Challenge Response Algorithm (OCRA), OCRA + TOTP (for tokens supporting OTP generation as well as C/R)
- Salt Mobile tokens: Salt mSign, Salt mCode
- EMV CAP / Visa DPA tokens and smart cards, including AA4C
- Vasco DigiPass tokens
- ActivIdentity tokens, subject to the provision of the ActivIdentity SDK

Note: We have tested Vasco tokens and ActivIdentity tokens on Windows.

Chapter 4: Installing and upgrading SSAS

Upgrading SSAS

Check the Release Notes for information on updating SSAS from one version to another.

The SSAS installation program checks for existing files in the intended installation directory and, if any are present, moves them to a backup directory of your choosing. See *Installing SSAS*. The installation program does not modify or delete properties files. This enables you to install a new version of SSAS and then transfer your files and settings across in a controlled manner.

Setting up your Java environment

Before you install SSAS, you must download and install the appropriate Java components. See *Chapter 3: Prerequisites and system requirements*. The following sections provide information on setting up and testing the JDK environment.

Installing Java

On machines where SSAS is to be deployed, but no application development is taking place, install the JRE. Development machines require the JDK, whose installation includes JRE.

When installing the JDK, use the guidelines in the following subsections to facilitate your installation for the appropriate platform.

Installing the JDK on Windows

We recommend that you use the default JDK root directory of `C:\jdk<version_number>_<build_number>`. By default, installing JDK on Windows installs the JRE into `C:\Program Files\Java\JRE\<version>`.

Do not set the CLASSPATH variable. Ensure that you include the JDK bin directory in the PATH variable.

For Oracle JDK 8 updates earlier than 162, download the *JCE Unlimited Strength Jurisdiction Policy* files and place the following files into the JRE security directory (`C:\Program Files\Java\JRE\<version>\lib\security`):

- `local_policy.jar`
- `US_export_policy.jar`

Installing the JDK on Linux

We recommend that you use the RPM distribution of the JDK of your choice and install it to the default JDK installation directory: `/usr/java`.

For Oracle JDK 8 updates earlier than 162, download the *JCE Unlimited Strength Jurisdiction Policy* files and place the following files into the JRE security directory (`/usr/java/jre/lib/security`):

- `local_policy.jar`
- `US_export_policy.jar`

The policy files must be owned by the root user, but readable by all users of the system, including **others**. Set appropriate access rights using the `chmod` command. We recommend using **444** as the protection mask.

Installing the IBM Java SDK on AIX

Download the *JCE Unlimited Strength Jurisdiction Policy* files and place the following files into the JRE security directory (`/usr/<Java_directory>/jre/lib/security`):

- `local_policy.jar`
- `US_export_policy.jar`

The policy files must be owned by the root user, but readable by all users of the system, including **others**. Set appropriate access rights using the `chmod` command. We recommend using **444** as the protection mask.

Preparing Oracle Server JVM

If you are using one of the Oracle Java implementations, we recommend that you run the SSAS daemon process with the server JVM (by default, Oracle Java runtime uses the client JVM). While immediately accessible on other platforms, the server JVM on Windows platforms is only installed as part of the JDK, but not the JRE.

To make the server JVM accessible from the JRE:

1. In your JDK installation directory (for example `C:\<version_number>_<build_number>`), navigate to the `jre\bin` directory. One of the subdirectories there is typically named `server`.
2. Locate the directory with the JVM executables, which is typically `C:\Program Files\Java\JRE\<version>\bin`.
3. Copy the `server` directory from the JDK to the JVM executables directory. JVM is now enabled for running in server mode.

Testing the JDK environment

To ensure that your JDK environment installed properly:

1. Compile and run the following *HelloWorld* Java program to verify that Java runs from the command line:

```
public class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
```

2. To ensure that the Java standard library extension directory is being used, enter `javac HelloWorld.java`, followed by `java HelloWorld` from the directory where the source is located. For testing the server JVM, enter `java -server HelloWorld`.

If any of these tests fail, the JDK setup is not working properly. Refer to the standard JDK installation documentation for further information.

Installing SSAS

Before starting the installation:

- Close any other programs that are running
- Ensure you have read and followed the guidance in *Chapter 3: Prerequisites and system requirements* and *Setting up your Java environment*

On the computer where you are installing SSAS, log on as the local administrator (on Windows platforms) or root user (on Unix-based platforms).

SSAS provides the following installation modes:

- Normal mode is a graphical installation program that provides installation choices
- Console mode is a console program that provides the same installation choices as normal mode
- Silent mode installs SSAS with the default settings, without user interaction

The console and silent modes do not require graphical support on the target platform and are useful when installing SSAS on remote servers.

Normal mode

To install SSAS using the normal installation mode:

1. Start the installation program:
 - *Windows platforms:* Select **Start > Run** from the Windows Start menu, and then enter:
Windows\Disk1\InstData\VM\ssas.installer.exe

- *Linux:* As the root user:
 - a. Copy the following directory and its contents to your server's /root directory:
Linux/Disk1
 - b. On your server, change the permissions of the following file to allow execution, and then run the file:
/Disk1/InstData/NoVM/ssas.installer.bin
- 2. Proceed through the installation program, defining the required installation:
 - a. Read the License Agreement and, if you agree to the terms of the license, accept it.
 - b. Review the JDK installation prerequisites.
 - c. Choose the type of installation you require, and then specify the installation directory. The full installation set includes the client and server components and the SSAS Management Console.
 - d. If the intended installation directory contains existing files, specify the backup directory you want those files moved to.
 - e. Verify your choices in the pre-installation summary.
- 3. When you are ready to install the files, click **Install**.
The status screen shows the progress of the installation.
- 4. When the installation is complete, click **Done**.
- 5. *Unix-based platforms:* As the root user, set the appropriate ownership and access rights for the SSAS files and directories.

Console mode

To install SSAS using the console installation mode, run the installation program using the **-i console** switch:

- *Windows platforms:* `ssas.installer.exe -i console`
- *Unix-based platforms:* `ssas.installer.bin -i console`

The console mode presents similar installation options to the normal mode and can upgrade an existing installation of SSAS.

Silent mode

To install SSAS using the silent installation mode, enter:

- *Windows platforms:* `ssas.installer.exe -i silent`
- *Unix-based platforms:* `ssas.installer.bin -i silent`

Installed files and libraries

This section describes the organisation of files and directories in the SSAS installation directory. A breakdown of the libraries included in the SSAS installation is also included.

Note: The SSAS installation directory is represented as *<SSAS install>* in all procedures in the SSAS guides. Replace *<SSAS install>* with your own installation directory.

The SSAS installation directory on Windows is organised into the directories listed below.

Directory	Contents
-----------	----------

bin	<ul style="list-style-type: none"> • Several Windows batch files or Unix shell scripts for running SSAS components from the command line • The batch files or shell scripts for the SimpleClient, HttpServer, SimpleEMVClient, Log Viewing Console (LVC), VacmanImport, and VacmanManage application examples • Properties files and resources for the user interfaces of the various SSAS management consoles • Binaries and scripts for installing SSAS as a Windows service
config	<p>Several subdirectories:</p> <ul style="list-style-type: none"> • aitokenexport: Configuration information for ActivIdentity tokens. • client: The client properties file for the application examples included in SSAS. • database: A complete sample of database creation scripts for Oracle and Microsoft SQL databases. For example, tamper-evident logging, EMV, WebPIN, and DigiPass token databases. • imc: Configuration information for the Identity Management Console (IMC). • pskc: A mapping configuration file for the IMC. • saltMobile: Configuration information for Salt Mobile. • soap: A WSDL file that is required to access SSAS built-in SOAP services and WSDD files for the MyID and VACMAN SOAP plug-ins. You can add other plug-ins to this subdirectory
dtd	<ul style="list-style-type: none"> • The Document Type Definitions (DTDs) used by SSAS • Other files required for XML processing performed by SSAS, for example namespace to schema mappings
lib	The SSAS jar files. See <i>Library details</i> .
samples	<ul style="list-style-type: none"> • Several sample server properties files. See <i>Sample SSAS configurations</i>. • A sample software cryptographic token for testing. The password for this token is: password.
src	The source files for the SSAS example code
tmp	Temporary files for both the server and the SimpleClient example code, when used
UninstallerData	<ul style="list-style-type: none"> • The files required to perform a clean uninstall • Uninstall logs providing useful information about the uninstall

Library details

The Salt libraries located in the `lib` directory are listed below.

Library	Description
<code>imc.jar</code>	Salt IMC user interface
<code>lvc.jar</code>	Salt LVC user interface
<code>jsdp.jar</code>	Salt low-level crypto library JSDP (Java Secure Development Platform) and JCE Security Provider. Incorporates the IAIK JCE package.
<code>ssas.jar</code>	Salt high-level SSAS classes. Both the server and client packaged together.
<code>ssas-mc.jar</code>	SSAS Management Console user interface
<code>tmc.jar</code>	Salt TMC user interface. For managing client and server hardware tokens and software tokens. Can be started from the main Management Console or as a stand-alone application.

Dependencies

`jsdp.jar` depends on the libraries listed below.

Library	Description
<code>activation.jar</code>	Oracle JavaBeans Activation Framework is used by JavaMail to manage MIME data. Used by JSDP for parsing MIME and S/MIME data.
<code>iaik_cms.jar</code>	IAIK Cryptographic Message Syntax (CMS) library
<code>iaik_jce.jar</code>	IAIK JCE library
<code>iaik_jsse.jar</code>	IAIK Java Secure Socket Extension
<code>iaik_ssl.jar</code>	IAIK SSL support
<code>mail.jar</code>	Oracle JavaMail package. Used by JSDP for parsing MIME and S/MIME data.
<code>xalan.jar, xalan-serializer.jar</code>	HTML, text, or other XML document types. Used by JSDP and IXSIL (IAIK XML Signature Library).
<code>xercesImpl.jar</code>	Apache Xerces XML Parser used by Xalan. Used by JSDP and IXSIL.
<code>xmlParserAPIs.jar</code>	Used by JSDP and IXSIL / Xerces
<code>w3c_http.jar</code>	IAIK customised version of the W3C HTTP implementation. Used by IAIK for HTTPS support.

The `imc.jar`, `lvc.jar`, `ssas.jar`, `ssas-mc.jar`, and `tmc.jar` files depend on the low-level JSDP package and the libraries listed below.

Library	Description
<code>axis-*.jar</code>	Apache Axis Web Services. Used by SSAS for SOAP processing only.
<code>xss4j.jar</code>	Apache Web Service Security. Used for WSS processing.
<code>Jetty-*.jar, javax.servlet-*.jar</code>	Jetty HTTP server. Used by SSAS for processing HTTP requests.
<code>*-slf4j-*.jar</code>	Simple Logging Facade For Java. Provides logging support for third-party libraries.
<code>hibernate.jar, jta-*.jar, javassist.jar, dom4j.jar</code>	Hibernate Persistence Layer. Used within User Authentication for database persistence.
<code>castor-xml.jar</code>	XML Data Binding. Used for PSKC processing.
<code>commons-*</code>	Apache Commons Utilities. Provide common utility classes for

	SSAS and third-party libraries.
jmxremote.jar, jmxri.jar	Oracle Java Management Extensions. Provides JMX support within SSAS.
radius.jar	JRadius. Used by SSAS for processing RADIUS requests.
opensaml.jar, joda-datetime.jar, open-ws.jar, xmltooling.jar, xmlsec.jar, jargs.jar, velocity.jar	Shibboleth OpenSAML. Used for processing SAML2 requests.
SaltMobile*.jar, NetComponents.jar	Salt Mobile Service Implementation. Used for processing Salt Mobile requests.
SpinningProgressBar.jar, datetimepicker*.jar	Custom Swing Components. Used within the IMC.
HTTPClient.jar	Fully-featured HTTP client library. See: http://www.innovation.ch/java/HTTPClient/

Installing SSAS as a Windows Service

It is recommended to configure SSAS as a Windows service once the required SSAS server configuration has been established. To do this, open a command prompt with Administrator privileges and execute the following command:

```
cd <SSAS_HOME>\bin
powershell .\install-service.ps1 -serverfile '<SSAS properties file with full path>' -serverpass '<server master password>' -serverprefix '<server prefix>'
```

To start the service, open Windows Services and start the service named “Salt Safetronic Authentication Server v<version>”.

If you get a powershell policy error, run the following command to set the execution policy to unrestricted:

```
powershell Set-ExecutionPolicy -scope CurrentUser unrestricted
```

Don't forget to restore the setting after installing the service:

```
powershell Set-ExecutionPolicy -scope CurrentUser restricted
```

When running SSAS with Standard Connector, it is recommended to increase the default Java memory settings. Review the memory settings in <SSAS_HOME>\bin\ssas-run-daemon.bat script. It is also recommended to set the JVM to the JDK8 server jvm.dll (jvm.so).

Installing SSAS as Service on Linux

SSAS can be started as a background process through SysV init. To do this, run the install-service.sh script located in SSAS_HOME\bin:

```
$ cd <SSAS_HOME>\bin
$ ./install-service.sh <SERVER_PROPERTIES_FILE> <SERVER_PREFIX>
<MASTER_PASSWORD>
```

Run the following command to start SSAS service:

```
$ service <service_name> start
```

Where <service_name> is “ssas-<server_prefix>”. For example, if your SSAS prefix is “EMV” then your service name is “ssas-EMV”.

This service is started for run levels 2, 3, 4, and 5. If you wish to remove the SSAS service, run the remove-service.sh script located in SSAS_HOME\bin.

This Linux service installation was certified on RHEL 6 and 7. Please contact Salt Group support for other platforms.

Updating SSAS With Existing Salt Mobile/Standard Connector Installation

The SSAS installer creates a backup of the existing SSAS installation to the chosen folder before upgrading. The following steps are necessary after upgrading SSAS, to restore the Salt Mobile/Standard Connector services:

1. Run Salt Mobile Starter Kit (*SaltMobileSSASStarterKit-jvm*.jar*).
 - a. Select the following in the *Select Components to Install* window:
 - FCM Support JARs
 - SaltMobile Service JARs
 - Server Configuration Files
 - Token Type Selection
 - SaltMobile Documentation
 - SaltMobile Test Client
 - b. In the *Select Token Types to Install* window, uncheck all boxes.
 - c. In the *Installation Task Summary* window, uncheck *DEMO_PROPS*.
 - d. Click *Install Now*.
 - e. Save Settings and Exit.
2. Copy the following files from the SSAS backup folder:
 - a. Standard Connector folder
 - b. aludest folder
 - c. bin\provision*.properties
 - d. bin\serverkeys.properties
 - e. lib\<VASCO libraries> or any other plugin libraries
 - f. lib\<database libraries>
 - g. tmp\<files>
 - h. Safetronic\<files>
3. Compare the bin\ssas-setenv.bat from the backed up ssas-setenv.bat and update as necessary.

Removing SSAS

If you need to remove an existing version of SSAS for any reason, such as to overcome upgrade issues, use the SSAS uninstall program.

Note: It is recommended to stop the server before uninstalling SSAS. If you have installed SSAS as a Windows or Linux service you should also delete the service which can be done using the *remove-service.bat* script on windows; and *remove-service.sh* on Linux. Otherwise, the installer will remove the service during uninstallation.

The uninstall program removes the files and directories that the installation program created. However, it does not remove any files that you have added.

To uninstall SSAS, do one of the following:

- In the operating system, navigate to the program directory and select **Uninstall**
- In the <SSAS install>\UninstallerData directory, enter:
 - *Windows platforms:* "uninstallSafetronic.bin.exe"
 - *Unix-based platforms:* "uninstallSafetronic.bin"

Where <version> is the SSAS version number.

You can use the *-i console* switch to run the uninstall program in console mode, for example:

- *Windows platforms:* "uninstallSafetronic.bin.exe" -i console
- *Unix-based platforms:* "uninstallSafetronic.bin" -i console

Chapter 5: Configuring the server and daemon interfaces

You must configure the server interface and the daemon naming service so that clients (including the Management Console) can communicate with the server.

Server interface

The server provides three types of interfaces to clients:

- A native Java RMI interface
- A Web Services SOAP interface
- A REST interface

While the RMI interface is always enabled, the SOAP interface must be explicitly enabled in the server configuration. When enabled, the server provides a set of built-in SOAP services as described in the WSDL file located in the `<SSAS install>\config\soap` directory. These are functionally equivalent to the services offered through the RMI interface.

The server can also be configured with additional SOAP services that are generated through the Apache Axis framework by specifying their Web Service Deployment Descriptor (WSDD) files in the server configuration, and making their implementation classes available in the **THALES_SSAS_CLASSPATH** of the daemon.

Likewise, the REST interface must be explicitly enabled. When enabled, the server provides the following services through the REST interface:

- IMC
- EMV
- OATH
- OneSpan/VASCO DigiPass
- Random Number

REST API specification in the OpenAPI 3.0 format is accessible via the `/safetronic/openapi.api` endpoint. This can be used with the Swagger UI to view and test the API.

Note that the Salt Mobile APIs already support REST interface via the Standard Connector component of SSAS.

Naming service and number of daemons

You can configure the SSAS daemon to use either the RMIRegistry or the JNDI instance as a mechanism for enabling clients to locate the daemon and server objects. The daemon and the clients must use the same instance of the RMIRegistry or JNDI context.

Over its lifetime, any instance of a SSAS daemon attempts to register and unregister the two objects listed below (with optional unique suffixes) with the RMIRegistry or JNDI context.

Object	Description
com/thalesesecurity/at/daemon	Responds to start , stop , and is-server-running messages
com/thalesesecurity/at/server	The actual server

Notes for RMIRegistry:

- If two SSAS daemons are to share a single instance of the RMIRegistry, the unique suffix must be used to distinguish between them in the namespace.
- An RMIRegistry is uniquely identified by the host it is executing on and the port it listens to. As such, there can be multiple RMIRegistry instances executing concurrently on the same machine. The daemon, the server properties, and the client properties, must all point to the same RMIRegistry (that is, to the same host and port).
- There are two possible deployment configurations, depending on whether a common RMIRegistry is available in the runtime environment:
 - *No common RMIRegistry*: You can configure the daemon to start one internally. The internal RMIRegistry requires a free port in the runtime environment that it can listen to. This is the default configuration.
 - *Common RMIRegistry*: You can configure the daemon to use it.
- In addition to the port required by the RMIRegistry, each of the two objects registered by the daemon requires a port. By default, these ports are assigned by the RMI subsystem at random.
- Every RMIRegistry instance requires a free port to listen for bind and lookup events. Consider this if you are planning to run concurrent registries.

Notes for JNDI:

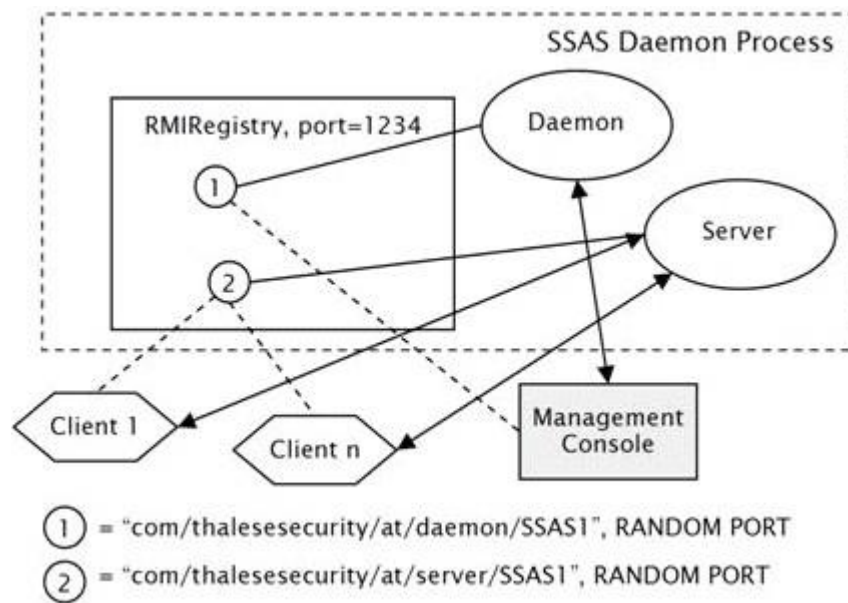
- If no optional unique suffix is specified, the daemon uses the **AtDaemon** and **AtServer** suffixes, resulting in the object names:
 - **com/thalesesecurity/at/daemon/AtDaemon**
 - **com/thalesesecurity/at/server/AtServer**
- For correct operation, the root JNDI context used by the SSAS daemon, server, and clients must be the same.
- Each of the two objects registered by the daemon requires a port. By default, these ports are assigned by the RMI subsystem at random.
- Whereas RMIRegistry uses a flat namespace, JNDI uses a hierarchical one. For RMIRegistry, an entry registered under the name **a/b/c** has that name in the global namespace. However, for JNDI, there is a node **a**, containing node **b**, containing leaf **c**.

The following sections outline some common configurations. For information on configuring the client properties file, see *Client configuration properties* in the *SSAS Reference Guide*.

Single daemon with internal RMIRegistry

The configuration shown in Figure 2 is the simplest of all configurations and the most common in production environments. It is also the one used in the sample properties files, and is the type generated by the Management Console's configuration wizards. We recommend that you use this configuration as the starting point when exploring SSAS because this enables you to learn more about the advantages and disadvantages of each configuration option before making a final decision about the production environment.

Figure 2. Simple SSAS configuration



The default daemon start and stop commands use this configuration with port 1234 for an internal RMIRegistry started by the SSAS daemon. Exported daemon and server objects are assigned random ports to listen to by the RMI subsystem, and no unique suffix is used.

The following table shows the configuration options that must be passed on to each of the entities in a simple SSAS configuration. For a description of all daemon options, see *Starting the SSAS daemon*. For a description of server configuration options, see *Server properties files*.

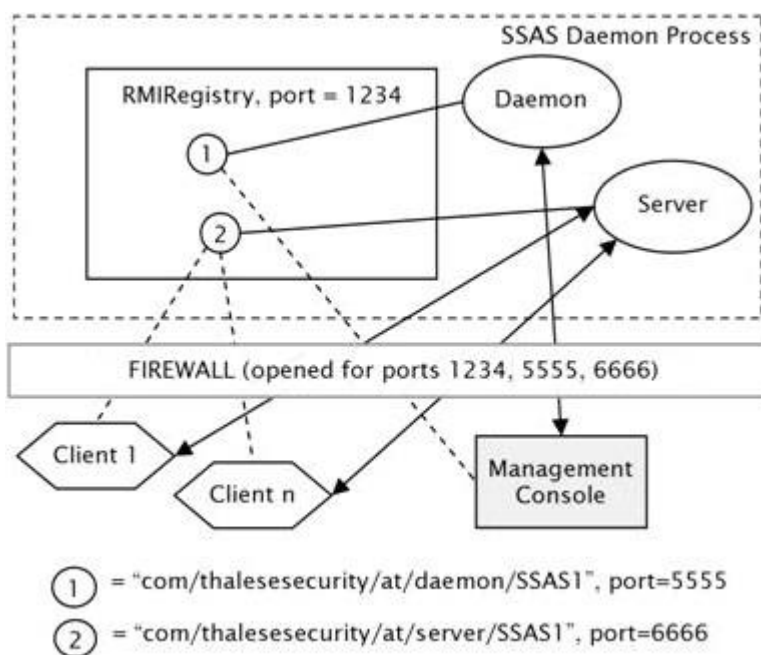
Entity	Configuration options
Daemon (command line parameters)	-rmiStartServer
	Omitted or set to true.
	-rmiPort= <port>
	Determines the port number the daemon uses to start the internal RMIRegistry. If the parameter is omitted, the daemon attempts to first start, then connect to an RMIRegistry on port 1099.
	-remoteName=AT1
	A unique name within the namespace. This can be any string. AT1 is used as an example and matches the sample properties files.
Server (from the Management Console)	Server->RMIRegistry Host= <host> Server->RMIRegistry Port= <port>
	If omitted, the Management Console attempts to locate the daemon that hosts the server execution in the RMIRegistry running on localhost: 1099.
Client (client properties file)	access.host= <host> access.port= <port>

	If both are omitted, the client attempts to locate the server that handles the request in the RMIRegistry running on localhost: 1099.
--	---

If a firewall exists between the clients and the daemon process, or between the Management Console and the daemon process, the ports that the SSAS daemon and server listen to can be fixed.

In Figure 3, the SSAS daemon is configured to listen to port 5555 (for start, stop, and **is-server-running** events, issued by the Management Console) and the server is responding to client requests on port 6666. The firewall must therefore have ports 1234, 5555 and 6666 opened.

Figure 3. Simple SSAS configuration in the presence of a firewall



In addition to the properties in the table above, you must apply the following properties:

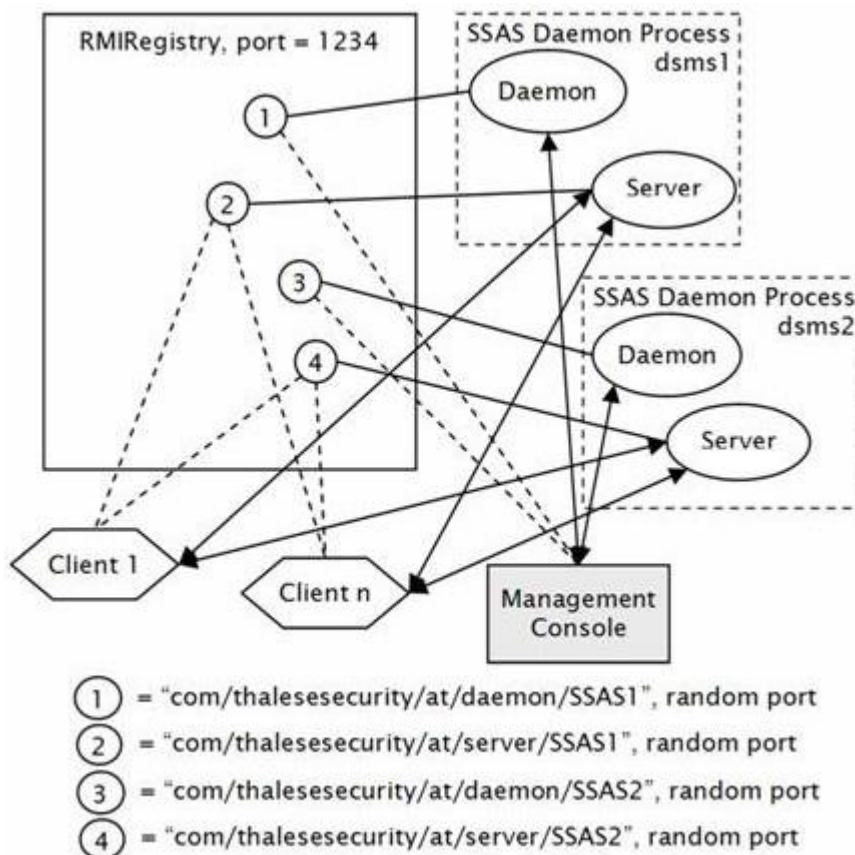
Entity	Configuration options
Daemon (command line parameters)	-atDaemonPort=5555
	The port the registered daemon object uses to listen for RMI connections from the Management Console. You can use any free port. 5555 is an example matching Figure 3.
Server (from the Management Console)	Server->Server RMI Port=6666
	The port the registered server object uses to listen to RMI connections from clients. You can use any free port. 6666 is an example matching Figure 3.

Multiple daemons sharing an external RMIRegistry

If an external RMIRegistry is already executing in the target environment, a daemon can be configured to use it, rather than starting its own internal instance. The information about the host and port of the RMIRegistry must be passed to every daemon instance (unless all daemons and the RMIRegistry are executed on the same machine, in which case the default options suffice). As Figure 4 shows, the server and client configurations must also be modified to point to the correct host and port of the external registry.

Also, if multiple daemons are registering with the same RMIRegistry instance, the unique suffix feature must be used.

Figure 4. Multiple daemons running against a single RMIRegistry



The following table shows the configuration options that must be passed on to each of the entities in a multi-server environment using the RMIRegistry.

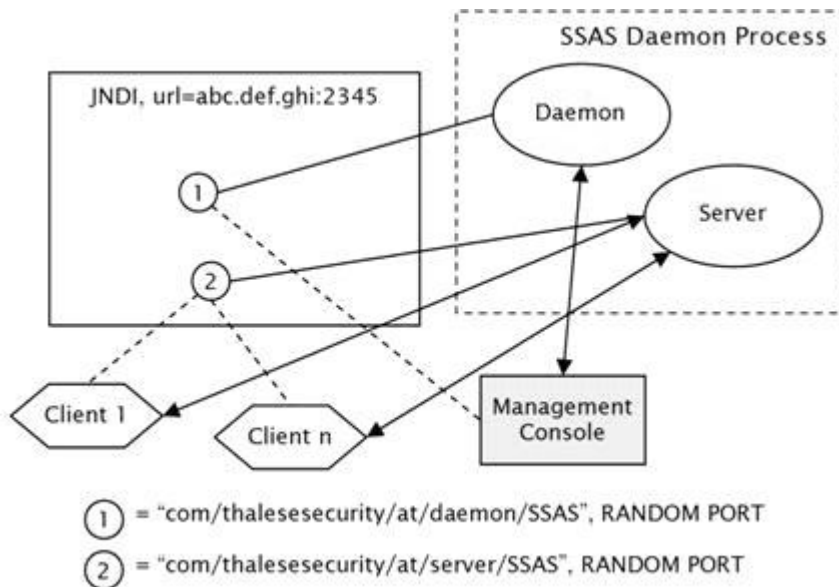
Entity	Configuration options
Daemon (command line parameters)	-rmiStartServer=false -rmiStartServer=false -rmiStartServer=false
	Does not start an internal RMIRegistry. Note: An external registry must be started before starting the daemon.

	-rmiHost=host
	Determines the machine running the external RMIRegistry. If the parameter is omitted, the daemon attempts to connect to an RMIRegistry running on localhost.
	-rmiPort=1234
	Determines the port number the daemon uses to connect to the external RMIRegistry. If the parameter is omitted, the daemon attempts to connect to an RMIRegistry listening on port 1099 . Any free port can be used. 1234 is an example matching Figure 4.
	-remoteName=SSAS1 or SSAS2
	The unique suffix for registering the daemon with the registry. This can be any string. SSAS1 and SSAS2 are examples matching Figure 4.
Server (from the Management Console)	Server->RMIRegistry Host= <host>
	Server->RMIRegistry Port= <port>
	If omitted, the Management Console attempts to locate the daemon that hosts the server execution in the RMIRegistry running on localhost: 1099 . Any free port can be used. 1234 is an example matching Figure 4.
	Server->Remote Name=SSAS1 or SSAS2
Client (client properties file)	access.rmiHost.1= <host> access.rmiPort.1=1234 access.remoteName.1=SSAS1 access.rmiHost.2= <host> access.rmiPort.2=1234 access.remoteName.2=SSAS2
	Host values must be specified in this case, even if they default to localhost. Port number values can be omitted, in which case they default to 1099 . Any unique names can be used. SSAS1 and SSAS2 are examples matching Figure 4.

Single daemon using JNDI

Figure 5 shows a simple JNDI configuration.

Figure 5. Simple JNDI configuration



Whereas the RMIRegistry is a part of the standard JRE, JNDI is an interface specification that enables different providers to register their implementations. As such, there is no *standard* executable to start on a given host and port. Instead, JNDI clients rely on the existence of the following properties among the System properties:

- **java.naming.factory.initial**, an appropriate initial context factory class
- **java.naming.provider.url**, a provider-specific repository address

Note: Some providers might require additional parameters. Consult the documentation for your JNDI provider.

These parameters can be passed on to SSAS in one of the following ways:

- As system properties through Java command line parameters using the **-D** flag to the **java** command. For example:

```
java -Djava.naming.factory.initial=<provider class>
    -Djava.naming.provider.url=<provider url>
    <class to execute>
```

- If not present, SSAS looks for a `jndi.properties` containing these properties along the **THALES_SSAS_CLASSPATH**

Note: If both a `jndi.properties` file exists in the CLASSPATH and also Java command line parameters are passed, the latter takes precedence.

Also, the SSAS RMI client accepts the JNDI configuration as a sub-tree of the properties in its properties file below **access.jndi**. If present, these take precedence over both command-line and settings specified in any `jndi.properties` along the **THALES_SSAS_CLASSPATH**.

The following table shows the configuration options that must be passed on to each of the entities in a multi-server environment.

Entity	Configuration options
CLASSPATH	In addition to all SSAS jars, the minimum requirement is that the configuration must contain a directory or jar where the JNDI provider class defined below might be located. If no system properties are passed as Java command line parameters to any of the components, the CLASSPATH must also contain a directory containing the <code>jndi.properties</code> file.
Daemon (command line parameters)	<code>-jndi=true</code>
	Indicates that the daemon object is registered using JNDI. If omitted, it defaults to false , indicating that the RMIRegistry is used instead. If <code>jndi.properties</code> is not used, system properties must be passed to the Management Console according to the description above. You do this by editing the <code>ssas-run-daemon.bat/.sh</code> file (similar changes have to be made in the <code>ssas-stop-daemon.bat/.sh</code> file).
Server (from the Management Console)	<code>Server->jndi=true</code>
	Indicates that JNDI is used for locating the daemon object that hosts the server. If omitted, defaults to false , indicating that the RMIRegistry is used instead. If <code>jndi.properties</code> is not used, system properties must be passed to the Management Console according to the description above. You do this by editing the <code>ssas-run-mgmtconsole.bat/.sh</code> file.
Client (client properties file)	<code>access.jndi=true</code>
	Indicates that JNDI is used for locating the daemon object that hosts the server. If omitted, defaults to false , indicating that the RMIRegistry is used instead.
	<code>access.jndi.java.naming.factory.initial= <provider class></code> <code>access.jndi.java.naming.provider.url= <provider url> access.jndi.x.y.z1=ABC</code>
	The access.jndi prefixed properties are stripped from the access.jndi prefix and passed as parameters to the construction of the SSAS client's Initial JNDI context. In other words, the JNDI context is initialised with the following properties:
	<code>java.naming.factory.initial= <provider class></code> <code>java.naming.provider.url= <provider url></code>

	x.y.z=ABC
	If no sub-tree is present below access.jndi in the client configuration, either a valid jndi.properties file must be present along the THALES_SSAS_CLASSPATH , or appropriate system properties must be defined in the client's environment prior to initialising it.

Multiple daemons using JNDI

If multiple daemons are required in the runtime environment (for redundancy or load-balancing of servers) unique suffixes must be used. The layout is exactly the same as shown in Figure 4.

The following table shows the configuration options that must be passed on for each component in a multi-server environment (with multiple daemons) using JNDI.

Entity	Configuration options
CLASSPATH	In addition to all SSAS jars, the minimum requirement is that the configuration must contain a directory or jar where the JNDI provider class defined below might be located. If no system properties are passed as Java command line parameters to any of the components, the CLASSPATH must also have a directory containing the jndi.properties file.
Daemon (command line parameters)	-jndi=true
	Indicates that the daemon object is registered using JNDI. If omitted, it defaults to false, indicating that the RMIRRegistry is used instead.
	-remoteName=SSAS1 or SSAS2
	The unique suffix for registering the daemon with JNDI. This can be any string. SSAS1 and SSAS2 are examples matching Figure 4.
Server (from the Management Console)	Server->jndi=true
	Indicates that JNDI is used for locating the daemon object that hosts the server. If omitted, it defaults to false , indicating that the RMIRRegistry is used instead.
	Server->remoteName=SSAS1 or SSAS2
	The unique suffix for locating the daemon to host server execution within the JNDI provider. The same string is also used as the unique suffix of the server that is started. This can be any string. SSAS1 and SSAS2 are examples matching Figure 4. If jndi.properties is not used, system properties must be passed to the Management Console according to the description above. You do this by editing the ssas-run-mgmtconsole.bat/.sh file.
Client (client properties file)	jndi=true
	Indicates that JNDI is used for locating the daemon object that hosts the server. If omitted, defaults to true , indicating that the

	RMIRRegistry is used instead.
	access.remoteName.1=SSAS1 access.remoteName.2=SSAS2
	Any unique names can be used. SSAS1 and SSAS2 are examples matching Figure 4.
	access.jndi.java.naming.factory.initial= <provider class> access.jndi.java.naming.provider.url= <provider url> access.jndi.x.y.z1=ABC
	The access.jndi prefixed properties are stripped from the access.jndi prefix and passed as parameters to the construction of the SSAS client's initial JNDI context. In other words, the JNDI context is initialised with the following properties:
	java.naming.factory.initial= <provider class> java.naming.provider.url= <provider url> x.y.z=ABC
	If no sub-tree is present below access.jndi in the client configuration, either a valid jndi.properties file must be present along the THALES_SSAS_CLASSPATH , or appropriate system properties must be defined in the client's environment prior to initialising it.

TLS (SSL)

You can configure TLS for your client-server communication to provide:

- Privacy
- Request authentication

The following sections describe these aspects. For a general overview, see the *SSAS Concepts Guide*. For more information on configuring and testing the client-server communication, see:

- *Chapter 22: Configuring the client-server communication*
- *Checking TLS connectivity*
- *Issues with client-server communication*

Privacy

Regardless of whether the RMIRegistry or JNDI is used as a naming mechanism, the RMI communication between the clients and servers can be privacy protected using TLS.

Use this option in cases where the network that the entities communicate over is not trusted. Be aware that using TLS decreases the performance of the overall system. Configuring the system is also more difficult, particularly if client authentication is enabled: every client must be equipped with a token containing keys and certificates for TLS communication. The same applies to the Access PKI Environment of all the servers.

If you use TLS, the clients always authenticate the server certificate and perform endpoint verification, but the server accepts connections from *any* client with whom the TLS handshake succeeds. As a further security precaution, you can configure the server to perform client authentication with endpoint verification.

To summarise, you need to make the following decisions:

- Whether to use TLS for protecting the RMI traffic between the clients and server

If you do use TLS, whether the server performs client authentication with endpoint verification. The configuration settings for privacy (RMI over TLS) are:

Entity	Configuration options
Server (from the Management Console)	Server->Use SSL=true
	If not specified, defaults to false , indicating that the RMI communication between the client and server is not protected.
	Server->Access PKI Environment-> Token must be correctly configured Server->Access PKI Environment-> Use Client Authentication=true
	If not specified, it defaults to false , which means that no client authentication is required by default.
	Server->Access PKI Environment-> Endpoint Identification Map
	If client authentication is set to true , the map must be configured with Distinguished Names of accepted client TLS certificates.
Client (client properties file)	access.useSSL=true
	If not specified defaults to false, indicating that the RMI communication between the client and server is not protected.
	access.pkiEnv.serverDnMap.dn1= <DN of server TLS certificate 1> access.pkiEnv.serverDnMap.dn2= <DN of server TLS certificate 2>
	One or more entries holding TLS certificate subject Distinguished Names that the client accepts during the TLS handshake.

	pkiEnv.keyStore.token.*
	For details of configuring a client cryptographic token, see <i>Setting up client cryptographic tokens</i> .

Request authentication

If the server is to perform signatures, or generate random material on behalf of the clients, consider enabling client request authentication. Request authentication prohibits the servicing of requests from unauthorised clients. The authentication can take one of two forms:

- Digest authentication, using a shared key
- Signature authentication, where the client signs the request using a dedicated signature key

In general, signature authentication requires more configuration, but allows for non-repudiation of requests to be claimed. The configuration settings for request authentication are:

Entity	Configuration options
Server (from the Management Console)	Server->Client Request Authentication= digestAuthentication signedAuthentication
	If clients must authenticate requests, this property must point to the selected authentication scheme.
	Server->Client Map
	Depending on the selected type of authentication, this property maps client names to client Distinguished Names (for signature authentication), or client names to shared keys (for digest authentication).
Client (client properties file)	access.clientRequestAuthenticationType= digestAuthentication signedAuthentication
	If request authentication is used, this setting must specify the selected authentication scheme. If not specified, defaults to none.

Chapter 6: Configuring the crypto module (channel token)

We recommend that you use a crypto module (*hardware token*) with SSAS to protect the sensitive material for a channel and perform the required cryptographic processing when handling authentication or digital signing requests. The crypto module is known as the *channel token*.

Note: SSAS allows the use of a software token (encrypted file), which you can use to test your applications and verify aspects of your configuration before you enter a production environment. However, we recommend you do not use software tokens in production environments.

You must install and configure the crypto module *before* you configure SSAS. For instructions on installing and configuring, refer to the documentation that accompanies your module.

Your crypto module must remain operational and accessible to the server. When configuring the SSAS channel using the Channel Wizard (Token page), you configure the settings for the module and set the channel password, see *Channel*. The table below identifies the token type and PKCS #11 implementation library that you specify in SSAS to use the module.

Thales module	Token type and library
nShield Solo and nShield Connect	Hardware(PKCS#11): <ul style="list-style-type: none">• cknfast.dll (32-bit Windows platforms)• libcknfast.so (Unix-based platforms)

Location of platform dependent components

SSAS is delivered with one platform dependent component: the PKCS #11 bridge to HSMs (crypto modules). It is located in the `<SSAS install>\lib\native` directory.

PKCS #11 bridge platform	Notes
Windows	<p>The bridge is implemented through the jsdp_pkcs11.dll file. The JVM on Windows expects the dynamic link libraries to be present along the PATH variable, which is why ssas-run-daemon.bat appends the directory above to the system PATH variable.</p> <p>If you intend to use third-party JNDI or JDBC providers that require platform- dependent libraries at runtime with the server, the directories of the platform- dependent libraries must either be present in the system PATH, or appended to the PATH within the script ssas-run-daemon.bat.</p>
Linux, AIX, and Solaris	<p>The bridge is implemented through the file:</p> <ul style="list-style-type: none"> • <i>Solaris 10 64-bit</i>: libjsdp_pkcs11_64.so • <i>Other Unix-based platforms</i>: libjsdp_pkcs11.so <p>If you intend to use third-party JNDI or JDBC providers that require platform- dependent libraries at runtime with the server, the directories of the platform- dependent libraries must either be present in the system LD_LIBRARY_PATH, or amended to the LD_LIBRARY_PATH within the script ssas-setenv.sh.</p> <p>An external dependency exists on the jsdp_pkcs11 native layer, which can be resolved by including the GCC library on the library path. If this cannot be resolved, java.lang.UnsatisfiedLinkError occurs when using a PKCS #11 device.</p>

nShield Solo and nShield Connect

Using an nShield HSM to protect Vasco token blobs

To install and configure the software so that SSAS uses an nShield Solo or nShield Connect HSM to protect the Vasco token blobs:

1. Install the nShield license and features on the HSM.
2. Install the CodeSafe software on the SSAS machine.
 When installing CodeSafe on Linux, the environment variables are not set automatically. You must configure the following variables:
 - **\$NFAST_HOME**: /opt/nfast
 - **\$NFAST_KMDATA**: /opt/nfast/kmdata
 The installer adds the following libraries to the %NFAST_HOME%\java\classes directory (Windows) or \$NFAST_HOME/java/classes directory (Linux):
 - nfjava.jar
 - kmjava.jar
 - jhsee.jar
3. Install and configure the VACMAN Controller for HSM. See *Configuring the VACMAN Controller*.
4. Use the CodeSafe software to create, publish, and sign a SEE machine on the HSM. For information on doing this, see the *CodeSafe Developer Guide*, which

accompanies the CodeSafe software.

5. Activate the Java interface to the hardserver that the SEE machine is using. See *Activating the Java interface to the hardserver*.
6. After setting up the Vasco service and database tables, configure the **DIGIPASS_CONFIG** table with the appropriate values. See *Configuring the VACMAN storage key*.

Environment variables

When you install the Security World Software on the SSAS machine, it provides the appropriate nShield PKCS #11 implementation library. The library uses several environment variables, which enable you to tailor it to your needs. You can set environment variables in the `cknfastc` file. If the file does not exist, create it as necessary. For more information, see the *Cryptographic API Integration Guide* provided with the Security World Software.

To ensure that the nShield HSM and SSAS integrate correctly, be aware of the variables below.

Environment variable	Description
CKNFAST_ASSUME_SINGLE_PROCESS	To ensure that the SSAS daemon can see the keys generated by the TMC, set this variable to 0 or N . It is set to 1 by default.
CKNFAST_DEBUG	If you are having problems integrating the nShield HSM with SSAS, set this variable and use the debug output to determine the issue and solution.
CKNFAST_FAKE_ACCELERATOR_LOGIN	To use the HSM, instead of a card set, to protect the key material that you create or import: <ul style="list-style-type: none"> Set this variable to 1 or Y Do not set CKNFAST_NO_ACCELERATOR_SLOTS the variable The key material is encrypted under the HSM's module key.
CKNFAST_NO_ACCELERATOR_SLOTS	Set this variable to 1 or Y . SSAS does not use the accelerator slot on the nShield HSM.
CKNFAST_LOADSHARING	Load-sharing enables you to load a single PKCS #11 token on to several nShield HSMs to improve performance. To use several nShield HSMs in a clustered, load-sharing environment, set this variable to 1 or Y (or another value other than 0 , N , or n). The nShield PKCS #11 library creates one virtual slot for each Operator Card Set (OCS) and, optionally, one slot for the HSM (or HSMs). If load-sharing is not enabled, you see two slots for every HSM connected. The OCS cards in each HSM must have the same pass phrase. Otherwise, SSAS and the TMC cannot log into each of the HSMs. The use of this load-sharing variable also enables you to use pass phrase-protected softcards to protect your keys.
CKNFAST_OVERRIDE_SECURITY_ASSURANCES	To ensure the TMC can perform particular key operations on the HSM, use the required

	<p>parameters to override specific security assurances:</p> <ul style="list-style-type: none"> • To import a private key and its certificates from a PKCS #12 file into the HSM, use import • To generate single DES keys, use tokenkeys and weak_des • To generate extractable (wrappable) keys, use explicitness • To generate key pairs, use shortkey_rsa=512 (for example) <p>For example, to enable the TMC to perform all the key operations above, use the following setting in the <code>cknfastrc</code> file:</p> <pre>CKNFAST_OVERRIDE_SECURITY_ASSURANCES=import;tokenkeys;weak_des;explicitness;shortkey_rsa=512</pre> <p>Note: When configuring the nShield to create weak keys, warning messages will be output by the PKCS#11 library. These warning messages are visible to the user on the TMC CLI.</p>
--	---

Token sharing

If you are sharing a crypto module (token) between channels, as a Filtered Key Store, set the **Key Store Type** to **JSDPFiltered** for those channels, and specify the **Visible Certificates** and **Visible Trusted Certificates**. To access these properties, select **PKI Environment** within the channel in the Management Console tree view. The properties appear in the property editor. For information about these properties, see the *SSAS Reference Guide*.

Remote tokens

If SSAS is accessing the crypto module remotely, from a platform without the appropriate PKCS #11 drivers, ensure that the Remote Token Server is:

- Started on a host that supports the appropriate PKCS #11 drivers
- Configured to access the crypto module

Chapter 7: Setting up the databases

This chapter contains information on setting up the databases that SSAS uses for token data, and event and error logs.

To set up the database, you must:

1. Get the JDBC drivers.
2. Configure the CLASSPATH.
3. Prepare the database.

Note: Do not import any token data into the database until you have configured SSAS. For configuration guidance for your service implementation, see the chapter in this guide for that implementation.

After setting up the databases, use the SSAS Management Console to:

1. Specify the **DB Driver** when creating the server. See *Creating a server*.
2. Create database connection pools for the databases. See *Database connection pools*.

Getting the JDBC drivers

We have tested SSAS with these JDBC drivers:

Database type	JDBC driver
Oracle	12.1.0.1.0, 12.1.0.2.0
Microsoft	Microsoft JDBC Driver 4.1 for SQL Server

Download the appropriate JDBC jar file from the manufacturer's website and copy the file into the `<SSAS install>\lib` directory. This file manages the communication between the database and SSAS.

Configuring the CLASSPATH

Add the JDBC driver to the CLASSPATH, which can be configured using the `ssas-setenv.bat` file (Windows platforms) or `ssas-setenv*.sh` file (Unix-based platforms). Ensure that you define the following paths in the **THALES_SSAS_CLASSPATH** section in the file:

Windows format:

Database type	Paths to define
Oracle	set THALES_SSAS_CLASSPATH=%THALES_SSAS_CLASSPATH%; %THALES_SSAS_HOME%\lib\ojdbc5.jar
Microsoft	set THALES_SSAS_CLASSPATH=%THALES_SSAS_CLASSPATH%;

	%THALES_SSAS_HOME%\lib\sqljdbc41.jar
--	--------------------------------------

Unix format:

Database type	Paths to define
Oracle	\$THALES_SSAS_HOME/lib/ojdbc5.jar\
Microsoft	\$THALES_SSAS_HOME/lib/sqljdbc41.jar\

If the .jar file name is different from that provided, change the path shown above accordingly.

Preparing the database

You must create the relevant tables on the chosen database, which you can do by using the scripts provided in the `<SSAS install>\config\database` directory. Both Oracle and Microsoft versions of the SQL commands are provided.

Select the correct file, depending on the required service. For example, to create the Vasco DigiPass tables, select the `create-digipass-db-tables.sql` file or `create-oracle-digipass-db-tables.sql` file accordingly.

You can insert the contents of these files into a suitable SQL tool to create the required tables.

Database schema

The `<SSAS install>\config\database` directory contains many useful SQL command file examples:

- `create-activid-db-tables.sql`
- `create-digipass-db-tables.sql`
- `create-emv-db-tables.sql`
- `create-ldap-db-tables.sql`
- `create-log-db-tables.sql`
- `create-oath-db-tables.sql`
- `create-oracle-activid-db-tables.sql`
- `create-oracle-digipass-db-tables.sql`
- `create-oracle-emv-db-tables.sql`
- `create-oracle-ldap-db-tables.sql`
- `create-oracle-log-db-tables.sql`
- `create-oracle-oath-db-tables.sql`
- `create-oracle-password-db-tables.sql`
- `create-oracle-radius-db-tables.sql`
- `create-oracle-saltmobile-db-tables.sql`
- `create-oracle-temac-db-tables.sql`
- `create-oracle-user-authentication-db-tables.sql`
- `create-password-db-tables.sql`
- `create-radius-db-tables.sql`
- `create-saltmobile-db-tables.sql`
- `create-temac-db-tables.sql`
- `create-user-authentication-db-tables.sql`
- `create-webpin-db-tables.sql`
- `create-ww-db-tables.sql`
- `run-create-ww-db.bat` (see the note below)

Note: You might need to modify the SSAS scripts used for starting and stopping servers and running the Management Console, to include appropriate database drivers in

the Java class path. For more information, see *Configuring the CLASSPATH*.

Before running the `run-create-ww-db.bat` script, you must ensure that SQL server language setting is **English** (not **British English**). You can set the default language for all users with the SQL Server Enterprise Manager by selecting the **Properties of the SQL Server** node and then using the **Server Settings** tab. Alternatively, you can set the language on a per-user basis under the **Security > Logins** node.

Tables required by SSAS

Event and error log tables

The default database logging requires at least two tables created in the database:

Database table	Columns
EVENTLOG	LOG_ID NUMBER(*,0) LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) EVENT VARCHAR2(4000) NOT NULL STACK_TRACE CLOB STATUS VARCHAR2(20)
ERRORLOG	LOG_ID NUMBER(*,0) LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) SERVICE VARCHAR2(50) ERROR_DESC CLOB STACK_TRACE CLOB ERROR VARCHAR2(4000) NOT NULL

Tamper evidence tables

If tamper-evident logging is required, an additional two tables are required:

Database table	Columns
EVENTLOG_ID	LOG_ID NUMBER(*,0) TABLE_ID NUMBER(*,0) NOT NULL
ERRORLOG_ID	LOG_ID NUMBER(*,0) TABLE_ID NUMBER(*,0) NOT NULL

Application-specific logging table

An application can send its own data that is logged with each request to the server. Application-specific data is recorded in the following table:

Database table	Columns
APPLICATION_LOG	LOG_ID NUMBER(*,0) LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40)

	TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) APPLICATION_DATA VARCHAR2(4000) NOT NULL
--	---

Service tables

Depending on the configured services, you must also create one or more of the following tables in the database:

Service	Database table	Columns
Verify Signature	CERT_STATUS_CHECK	LOG_ID NUMBER LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) MESSAGE CLOB XML_SIGNATURE_PATH VARCHAR(255) XML_NS_PREFIXES VARCHAR2(50) REQUEST_SIGNER_COUNTRY CHAR(2) REQUEST_SIGNER_SECURITY_LEVEL VARCHAR(50) MSG_SIGNER VARCHAR2(400) CONTENT CLOB DIGEST VARCHAR2(15) RESPONSE CLOB CERTIFICATE_STATUS CLOB NONCE VARCHAR(50) STATUS VARCHAR2(20)
Signature	SIGNATURE	LOG_ID NUMBER(*,0) LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) DTS CLOB DIGEST VARCHAR2(15) SIGNATURE CLOB STATUS VARCHAR2(100)
Random Number	RANDOM_NUMBER	LOG_ID NUMBER(*,0) LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) RANDOM_NUMBER CLOB STATUS VARCHAR2(100)
EMV Authentication Generic MAC	EMV_AUTHENTICATION	LOG_ID NUMBER LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40)

service handler		TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) SECURECODE VARCHAR2(16) SECURECODE2 VARCHAR2(16) PAN VARCHAR2(19) PANSEQ VARCHAR2(2) CHALLENGE VARCHAR2((16) CHALLENGE2 VARCHAR2(16) AMOUNT NUMBER(*,0) OTHER_AMOUNT NUMBER(*,0) CURRENCY_CODE NUMBER(*,0) ATC NUMBER(*,0) TRANSACTION_DATA VARCHAR2(200) REQUEST_TYPE VARCHAR(16) STATUS VARCHAR2(20)
SAML Authentication Assertion production service	SAML_ASSERTION_PRODUCTION	LOG_ID NUMBER LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) SAML_ASSERTION CLOB STATUS VARCHAR2(20)
TEMAC	TE_MAC	LOG_ID NUMBER LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) REQUEST_MODE NUMBER(*,0) NOT NULL KEY_REFERENCE VARCHAR2(50) NOT NULL KEY_ID VARCHAR2(40) NULL DATA_TO_MAC CLOB NOT NULL TE_MAC VARCHAR2(90) NULL STATUS VARCHAR2(100)
WebPIN Verification	WEBPIN_VERIFICATION	LOG_ID NUMBER LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) STATUS VARCHAR2(100) REQUESTMODE NUMBER UID VARCHAR2(32) MESSAGE CLOB
Vasco	DIGIPASS_AUTHENTICATION	LOG_ID NUMBER LOG_MAC VARCHAR2(90)

		LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) SERIAL_NUMBER VARCHAR2(22) PASSWORD VARCHAR2(17) ITIME_WINDOW NUMBER STIME_WINDOW NUMBER DATA VARCHAR2(177) CHALLENGE VARCHAR2(17) STATUS VARCHAR2(20) RESPONSE CLOB SIGNATURE CLOB
OATH	OATH_AUTHENTICATIO N	LOG_ID NUMBER LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CHANNEL VARCHAR2(40) REQUEST VARCHAR2(4000) STATUS VARCHAR2(100)
RADIUS	RADIUS_AUTHENTICAT ION	LOG_ID NUMBER LOG_MAC VARCHAR2(90) LOG_MAC_KEYID CHAR(40) TIMESTAMP VARCHAR2(40) NOT NULL TRANS_ID VARCHAR2(56) CLIENT_NAME VARCHAR2(40) NOT NULL USER_NAME VARCHAR2(128) NOT NULL PROXY_NAME VARCHAR2(40) RADIUS_REQUEST CLOB NOT NULL RADIUS_RESPONSE CLOB NOT NULL STATUS VARCHAR2(20)

Chapter 8: Salt Mobile service

This chapter provides configuration information and procedures to help you set up a Salt Mobile service in SSAS. For general information about this service, such as how the authentication works, see the *SSAS Concepts Guide*. For information on using the SSAS Management Console, see *SSAS Management Console*.

Please refer to the *SSAS Salt Mobile User Guide* for instructions on how to setup the Salt Mobile service in SSAS.

Chapter 9: EMV authentication

This chapter provides configuration information and procedures to help you set up an EMV CAP authentication service in SSAS. For general information about this service, such as how the authentication works, see the *SSAS Concepts Guide*. For information on using the SSAS Management Console, see *Chapter 17: SSAS Management Console*.

EMV CAP authentication is implemented as a Generic MAC Verification Service.

Note: The SSAS product and documentation refer to *EMV CAP* in many places. However, the SSAS CAP functionality also supports DPA, which is the Visa equivalent of CAP. For more information on using DPA, contact Support.

Main configuration procedures

To configure SSAS to provide an EMV CAP authentication, you must perform the following procedures:

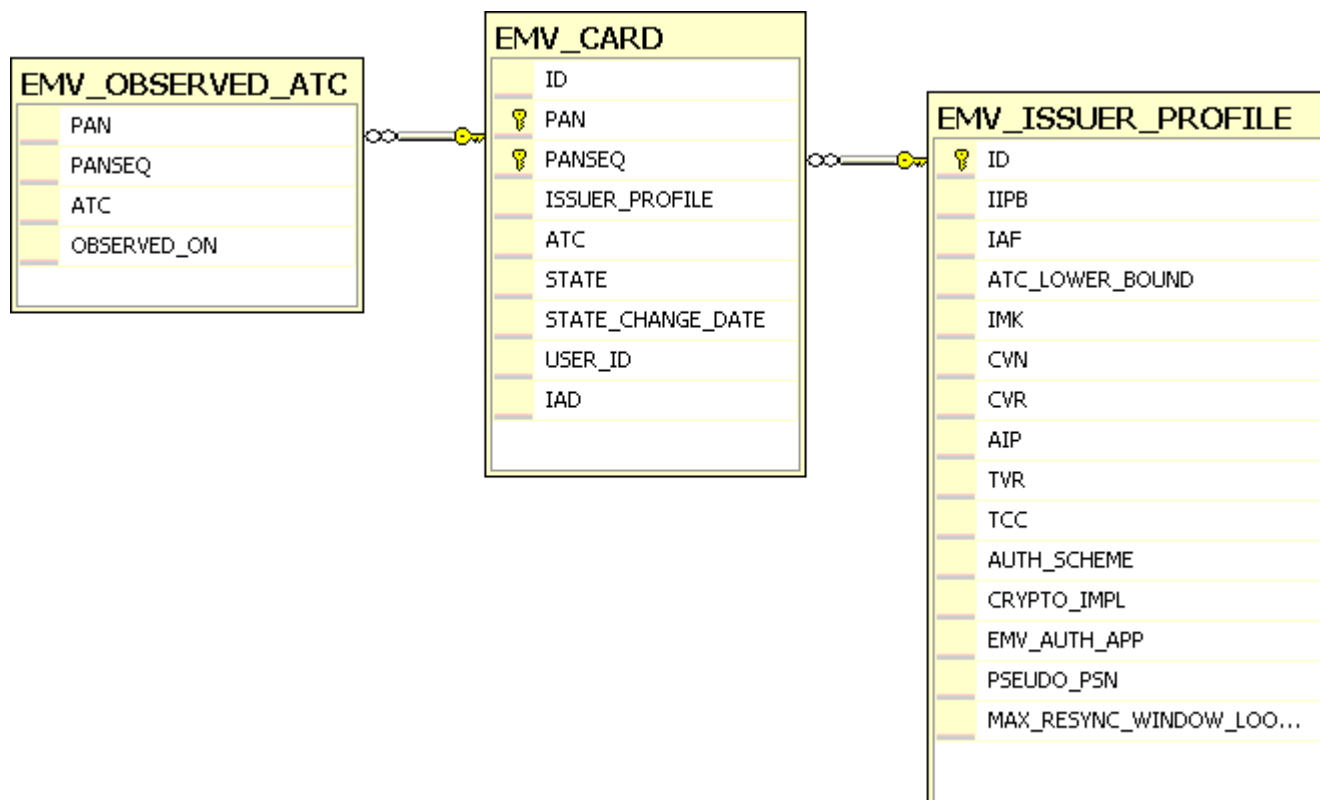
Procedure	For instructions, see:
If you need to, set up the SSAS environment by configuring: 1. <ul style="list-style-type: none"> The server and daemon interfaces. The crypto module (channel token). The database for the card details. 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i> <i>Chapter 7: Setting up the databases</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create a symmetric channel (using the Channel Wizard).	<i>Creating a channel</i>
4. Add the Generic MAC Verification Service to the channel (using the Services Wizard). When configuring the service, specify the database connection or file that holds the keys or key derivation information.	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> In the server, assign the Generic MAC Verification Service to the required channel selector. Map the channel selector to the channel. 	<i>Channel selectors</i>
6. Configure the Mastercard EMV SecureCode authentication mechanism.	<i>Configuring the Mastercard EMV SecureCode authentication mechanism</i>
7. Configure any additional settings for the Generic MAC Verification Service in the channel, as required.	<i>Configuring additional settings for the Generic MAC Verification Service</i>

Configuring the Mastercard EMV SecureCode authentication mechanism

Providing the EMV card information


The EMV authentication service depends on a database of EMV card information being populated according to the logical model shown in Figure 6.

Figure 6. EMV SecureCode authentication model



Alternatively, the calling application might supply the ATC and card details, with a reference to a particular Issuer profile, as shown in Figure 7.

Figure 7. Alternative EMV SecureCode authentication model

EMV_ISSUER_PROFILE	
	ID
	IIPB
	IAF
	ATC_LOWER_BOUND
	IMK
	CVN
	CVR
	AIP
	TVR
	TCC
	AUTH_SCHEME
	CRYPTO_IMPL
	EMV_AUTH_APP
	PSEUDO_PSN
	MAX_RESYNC_WINDOW_LOO...

For this logical model, only a database of the EMV Issuer profiles needs to be populated. Card information and observed ATC is supplied by the calling application.

Configuring the EMV CAP AAC Issuer profile

The EMV database model includes the following additions, which accommodate EMV CAP AAC.

Field/column	Description
IAF	This is a 4-digit decimal value, which SSAS reads as a binary byte. Essentially it is 8 bits/flags, which can set and enable (or disable) certain EMV functionality. SSAS uses IAF bit 5, the Token Formatting Indicator (TFI), to determine if alphanumeric CAP tokens are being used instead of standard numeric CAP tokens.
PSEUDO_PSN	You can set this field/column, within the EMV_ISSUER_PROFILE , to true or false depending on whether this batch of cards is configured to use the Pseudo PSN functionality within CAP AAC. If it is not required, omit the field or leave it as NULL in the database.
TOKEN_DATA_LENGTH	You can set this field/column, within the EMV_ISSUER_PROFILE , to the length of TokenData SSAS can expect within the EMV CAP token. Only use this if the CAP AAC specification cards are being used and the IPB IAD length is longer than the TokenData IAD length, meaning that the TokenData IAD must be padded to the same size. SSAS needs this field/column to determine how much padding has been added so it can remove it and allow the verification to succeed. If it is not required, omit the field or leave it as NULL in the database.

Setting EMV CAP card resynchronisation

EMV CAP resynchronisation is part of the EMV service. It enables the resynchronisation of an EMV card that has become out-of-sync with SSAS.

For enhanced security, SSAS only looks ahead one *EMV window*, by default. However, you can change this to any number of windows on a per issuer basis by updating the **MAX_RESYNC_WINDOW_LOOKAHEAD** column in the **EMV_ISSUER_PROFILE** table.

You can invoke the EMV CAP resynchronisation functionality through any of the server interfaces.

Chapter 10: Vasco service

This chapter provides configuration information and procedures to help you set up a Vasco service in SSAS. For general information about this service, such as how the authentication works, see the *SSAS Concepts Guide*. For information on using the SSAS Management Console, see *Chapter 17: SSAS Management Console*.

The Vasco service is implemented as a Generic MAC Verification Service and requires the use of a database connection. We have extensively tested the Vasco service with a range of Vasco tokens and application types. If you have specific requirements, contact Support or your Salt representative.

You can use an nShield Solo or nShield Connect HSM to protect the Vasco token blobs. To do this, you must order particular features for the HSM, install the appropriate software, and create and publish a SEE machine *before* setting up the Vasco service. See *nShield Solo* and *nShield Connect*.

Main configuration procedures

To configure SSAS to provide a Vasco service, you must perform the following procedures:

Procedure	For instructions, see:
Set up the SSAS environment by configuring: 1. <ul style="list-style-type: none"> The server and daemon interfaces. The crypto module (channel token). The database for the Vasco token details. 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i> <i>Chapter 7: Setting up the databases</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create a symmetric channel (using the Channel Wizard).	<i>Creating a channel</i>
4. Add the Generic MAC Verification Service to the channel (using the Services Wizard).	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> In the server, assign the Generic MAC Verification Service to the required channel selector. Map the channel selector to the channel. 	<i>Channel selectors</i>
6. Configure additional settings for Vasco.	<i>Configuring additional settings for Vasco</i> <i>Configuring additional settings for the Generic MAC Verification Service</i>
7. Import and manage the Vasco tokens.	<i>Importing and managing Vasco tokens</i>

Configuring additional settings for Vasco

Configuring the VACMAN Controller

For the Vasco service to work correctly, you must install and configure the correct VACMAN Controller. This is supplied in either a Windows installer or ZIP file. Extract or install these files to a directory of your choice, typically:

- **Windows:** C:\Program Files\Vacman Controller <version>
- **Linux:** /opt/Vacman Controller <version>

When installed, you must make the following VACMAN Controller libraries available:

- **Windows:** aal2sdk.dll / aal2sdk.lib
- **Linux:** libaal2sdk.so / libaal2sdk-<version>.so

To make the libraries available:

1. Locate the library files within the <VACMAN install directory> and note the path to those files. The path varies for different VACMAN Controller versions.
2. Either:
 - Copy the VACMAN Controller library files to the <SSAS install>\lib\native directory (Windows) or <SSAS install>\lib\native directory (Linux).
 - Edit the ssas-setenv.bat file (Windows platforms) or ssas-setenv*.sh file (Unix-based platforms) and add the path for the VACMAN Controller library files. Ensure that you define the path in the **THALES_SSAS_CLASSPATH** section in the file. For example, on a Windows machine the path might be:

```
set THALES_SSAS_CLASSPATH=%THALES_SSAS_CLASSPATH%;"C:\Program
Files\VASCO\VACMAN Controller 64 bit nCipher HSM 3.19.0\Bin"
```

3. Locate and copy the aal2sdk.jar file from the <VACMAN install directory> to the <SSAS install>\lib directory.

SSAS is now configured to work successfully with the VACMAN Controller.

Activating the Java interface to the hardserver

If you are using an nShield Solo or nShield Connect HSM, you must activate the Java interface to the hardserver that the published SEE machine, on the HSM, is using. To do this:

1. At a command prompt, enter: **config-serverstartup.exe -s -p**
2. Restart the hardserver.

For more information, see the User Guide that accompanies the HSM.

Configuring the VACMAN storage key

The Vasco DigiPass tables include a **DIGIPASS_CONFIG** table, which contains **KEY_ALIAS** and **SEE_NAME** columns. Use this table to identify the storage key that the VACMAN Controller uses to securely store the token blobs in the token database.

By default, the **KEY_ALIAS** value is set to **NULL**, which means that no encryption takes place.

nShield Solo and nShield Connect

If you are using an nShield Solo or nShield Connect HSM, you can use the published SEE machine on the HSM to protect the token blobs. To do this:

- Set the **KEY_ALIAS** to the file name of the storage key in **%NFAST_KMDATA%** (Windows) or **\$NFAST_KMDATA** (Linux)
If the storage key is in a different location, specify the full file path.
- Set the **SEE_NAME** to the name of the published SEE machine

Fail-over cluster configuration

If you have multiple nShield Connect HSMs, you can configure SSAS to use the SEE machines on the HSMs in a fail-over configuration. SSAS uses the SEE machine on one of the HSMs. If that HSM becomes unavailable, SSAS fails over to the next HSM, and so on.

To configure this:

1. Load the published SEE machine on all HSMs.
2. In SSAS, set the **Initial Size**, **Maximum Size**, and **Timeout** properties of the general pool. For information about configuring the general pool properties, see the *SSAS Reference Guide*.
3. Save the SSAS configuration, and stop and restart the SSAS daemon.

For example, to use five HSMs in a round-robin manner, set both the **Initial Size** and **Minimum Size** values to **5**. SSAS uses one of the HSMs. If it becomes unavailable for the **Timeout** period, SSAS fails over to the next HSM.

SSAS checks the pool once, when the daemon starts, to determine which HSMs are available from the pool. If you add HSMs, they are not added to the pool until you restart the daemon.

Importing and managing Vasco tokens

Note: You must configure and finalise the Vasco channel and service before beginning the token import process using the IMC.

Overview of the Vasco token process

To import and manage Vasco tokens, you carry out the following steps, which are described in the sections that follow:

1. Configure the IMC to show the required token types.
2. Import the tokens.
Vasco tokens are typically supplied as a batch of tokens in a DPX file.
Note: The DPX file data is encrypted. You must have the associated Init Key to decrypt the DPX file and import the token data into SSAS.
3. Manage the tokens.
When imported, the Vasco tokens are ready for use for authentication with SSAS.
You can manage the status of individual tokens, such as enabling or disabling them.

You use the IMC to manage the Vasco tokens. For information on starting the IMC, see *Opening the IMC*. When configured, the IMC groups the Vasco tokens under the **Vasco** token type.

For a description of the fields that appear when you import, view, and update a token, see *Vasco token fields*.

Configuring the IMC to show the required token types

You must configure the IMC mappings to show the Vasco token type within the IMC. You can configure the mappings either when you start the IMC or by using the IMC Preferences dialog.

To configure the mappings, follow the instructions in *Configuring the IMC token type to channel mappings*. Map the **Vasco** token type to your Vasco channel.

When you have configured the mappings, the token type appears under the channel in the IMC tree view. You can now import, view, and edit this token type for the channel.

Importing the tokens

To import a batch of Vasco tokens, you must have the DPX file and the associated Init Key. Follow the instructions in *Importing tokens for a channel*.

When imported, you can use the tokens for authentication with SSAS.

Managing the tokens

To view all the tokens, select the **Vasco** token type in the IMC tree view. The tokens appear in the list pane. You can filter and sort the tokens. See *Viewing, filtering, and sorting the tokens*.

You can view, update, and delete tokens by selecting them in the list pane and performing the appropriate action (using the **Tools** menu or the icons). For more information, see:

- *Viewing details for a token*
- *Updating a token*
- *Deleting a token*

Updating a token enables you to change its state. A Vasco token's state can be **Enabled** or **Disabled**.

Vasco token fields

The following tables describe the fields that appear when you import, view, and update a token.

Import

Field	Description
DPX File	The location of the .dpx file containing a collection of Vasco tokens
Init Key	A transport key used to decrypt the encrypted .dpx file

View and update

Field	Description
Dpblob	The token data blob. If you are using a hardware token, this is the encrypted value.
Token Type	The type of Vasco token. This is generally the model number of the token.
Authorisation Mode	The authorisation mode, which can be RO (Response Only), CR (Challenge Response), or SG (Signature).
State	The token state, which can be either Enabled or Disabled .
Static Vector	The static vector of the token, if supported.
Kernel Parameters	The ID of the kernel parameters in use by this token.
Id	The token serial number and application.

Chapter 11: TEMAC service

This chapter provides configuration information and procedures to help you set up a TEMAC service in SSAS. For general information about this service, such as how the authentication works, see the *SSAS Concepts Guide*. For information on using the SSAS Management Console, see *Chapter 17: SSAS Management Console*.

The TEMAC service is implemented as a Generic MAC Verification Service.

Main configuration procedures

To configure SSAS to provide a TEMAC service, you must perform the following procedures:

Procedure	For instructions, see:
Set up the SSAS environment by configuring: 1. <ul style="list-style-type: none"> The server and daemon interfaces. The crypto module (channel token). The database for the TEMAC service. 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i> <i>Chapter 7: Setting up the databases</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create a symmetric channel (using the Channel Wizard). Add the Generic MAC Verification Service to the channel (using the Services Wizard).	<i>Creating a channel</i>
4. When configuring the service, specify the database connection or file that holds the keys or key derivation information.	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> In the server, assign the Generic MAC Verification Service to the required channel selector. Map the channel selector to the channel. 	<i>Channel selectors</i>
6. Configure any additional settings for the TEMAC service.	<i>Configuring the database for TEMAC</i>
7. Configure any additional settings for the Generic MAC Verification Service in the channel, as required.	<i>Configuring additional settings for the Generic MAC Verification Service</i>

Configuring the database for TEMAC

The TEMAC generation and authentication services depends on a database of TEMAC keys being populated. The following is an example of the TEMAC database table:

```
KEY_REFERENCE varchar(50) NOT NULL,  
KEY_ALIAS varchar(50) NOT NULL,  
PRIMARY KEY (KEY_REFERENCE)
```

The NOT NULL command ensures the field cannot be left blank.

The TEMAC_Keys table contains KEY_REFERENCE and KEY_ALIAS columns. Use this table to identify the TEMAC keys that the TEMAC service uses.

Option	Description
KEY_REFERENCE	The Key Reference is the identifier of the key which must be supplied by the client when generating or verifying a MAC
KEY_ALIAS	The Alias of the key stored on the crypto module

Chapter 12: OATH service

This chapter provides configuration information and procedures to help you set up an OATH service in SSAS. For general information about this service, such as how the authentication works, see the *SSAS Concepts Guide*. For information on using the SSAS Management Console, see *Chapter 17: SSAS Management Console*.

The OATH service provides for authentication of tokens supporting HOTP, TOTP, and OCRA.


Main configuration procedures

To configure SSAS to provide an OATH Service, you must perform the following procedures:

Procedure	For instructions, see:
Set up the SSAS environment by configuring:	<i>Chapter 5: Configuring the server and daemon interfaces</i>
1. <ul style="list-style-type: none"> The server and daemon interfaces The crypto module (channel token) The database for the OATH token details 	<i>Chapter 6: Configuring the crypto module (channel token)</i> <i>Chapter 7: Setting up the databases</i>
2. Create a server in the Management Console (using the Server Wizard). Create a symmetric channel (using the Channel Wizard).	<i>Creating a server</i>
3. Do not choose software token as the token type. The OATH service uses clear keys in the database when configured with a software token. This feature exists for test purposes and must not be used as part of a production system.	<i>Creating a channel</i>
4. Add the OATH Service to the channel (using the Service Wizard).	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> In the server, assign the OATH Service to the required channel selector. Map the channel selector to the channel. 	<i>Channel selectors</i>
6. Configure any additional settings for the OATH Service in the channel, as required.	<i>Configuring additional settings for the OATH Service</i>
7. Import and manage the OATH tokens.	<i>Importing and managing OATH tokens</i>

Configuring additional settings for the OATH Service

You can access the service properties for a channel by selecting the service in the tree view.

In the property editor, use  to display all properties. The values shown reflect the settings you configured when adding the service to the channel and any changes made since.

To change a value, click it in the property editor. Help text appears in the Help pane. For information about the service properties, see the *SSAS Reference Guide*.

Importing and managing OATH tokens

Note: You must configure and finalise the OATH channel and service before beginning the token import process using the IMC.

If the OATH service is to be configured to use an nShield HSM, the OATH token properties file needs to be changed, see section *OATH Tokens* in the *SSAS Reference Guide*.

Overview of the OATH token process

To import and manage OATH tokens, you carry out the following steps, which are described in the sections that follow:

1. Configure the IMC to show the required token types.
2. Import the tokens.
3. Manage the tokens.

When imported, the OATH tokens are ready for use for authentication with SSAS. You can manage the status of individual tokens, such as enabling, blocking, or disabling them.

You use the IMC to manage the OATH tokens. For information on starting the IMC, see *Opening the IMC*. When configured, the IMC groups the OATH tokens under the **OATH** token type.

For a description of the fields that appear when you import, view, and update a token, see *OATH token fields*.

Configuring the IMC to show the required token types

You must configure the IMC mappings to show the OATH token type within the IMC. You can configure the mappings either when you start the IMC or by using the IMC Preferences dialog.

To configure the mappings, follow the instructions in *Configuring the IMC token type to channel mappings*. Map the **OATH** token type to your OATH channel.

When you have configured the mappings, the token type appears under the channel in the IMC tree view. You can now import, view, and edit this token type for the channel.

Importing the tokens

You import OATH tokens into the channel one at a time. Follow the instructions in *Importing tokens for a channel*.

When imported, you can use the tokens for authentication with SSAS.

Managing the tokens

To view all the tokens, select the **OATH** token type in the IMC tree view. The tokens appear in the list pane. You can filter and sort the tokens. See *Viewing, filtering, and sorting the tokens*.

You can view, update, and delete tokens by selecting them in the list pane and performing the appropriate action (using the **Tools** menu or the icons). For more information, see:

- *Viewing details for a token*
- *Updating a token*
- *Deleting a token*

Updating a token enables you to change its state. An OATH token's state can be **Enabled**, **Blocked**, or **Disabled**.

OATH token fields

The fields that appear when you import, view, and update a token are:

Field	Description
User	The user name associated with a token.
Try Counter	The Try Counter of the token (defaults to 0).
Counter	The 16-character hex encoded counter of the token (defaults to 16 zeros). For HOTP or OCRA, this is the event counter of the token. For TOTP, the default value of all zeros results in the token being synchronised on first use. Alternatively, this can be set to the hex encoded current time step of the token in seconds.
Id	The ID of the token, which is typically the token serial number.
Auth Type	The authentication type of the token. This can be HOTP , TOTP , or OCRA . Note: This returns an error if not supplied during token import. (Required for backwards compatibility.)
Key	The 16-character hex encoded key of the token.
State	The state of the token (defaults to Enabled). This can be Enabled , Blocked , or Disabled .
Mac	The tamper-evident MAC of the token.
Key Encryption Key	The Key Encryption Key is used to protect the OATH token key (DES3/AES key alias when using nShield HSMs).
Challenge	The last challenge sent by the server as part of an OCRA

	authentication or resynchronisation.
Next Challenge	The last challenge sent by the server as part of an OCRA resynchronisation.
Last Verified Timestamp	The time of the last successful TOTP authentication (measured in time steps).
Drift Steps	The time drift between the server's time and the time steps.

Chapter 13: ActivIdentity service

This chapter provides configuration information and procedures to help you set up an ActivIdentity service in SSAS. For general information about this service, such as how the authentication works, see the *SSAS Concepts Guide*. For information on using the SSAS Management Console, see *Chapter 17: SSAS Management Console*.


Main configuration procedures

To configure SSAS to provide an ActivIdentity service, you must perform the following procedures:

Procedure	For instructions, see:
Set up the SSAS environment by configuring: 1. <ul style="list-style-type: none"> The server and daemon interfaces The crypto module (channel token) The database for the ActivIdentity token details 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i> <i>Chapter 7: Setting up the databases</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create a symmetric channel (using the Channel Wizard).	<i>Creating a channel</i>
4. Add the ActivIdentity Service to the channel (using the Services Wizard).	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> In the server, assign the ActivIdentity Service to the required channel selector. Map the channel selector to the channel. 	<i>Channel selectors</i>
6. Configure any additional settings for the ActivIdentity Service in the channel, as required.	<i>Configuring additional settings for the ActivIdentity Service</i>
7. Import and manage the ActivIdentity tokens.	<i>Importing and managing ActivIdentity profiles and tokens</i>

Configuring additional settings for the ActivIdentity Service

You can access the service properties for a channel by selecting the service in the tree view.

In the property editor, use  to display all properties. The values shown reflect the settings you configured when adding the service to the channel and any changes made since.

To change a value, click it in the property editor. Help text appears in the Help pane. For

information about the service properties, see the *SSAS Reference Guide*.

Importing and managing ActivIdentity profiles and tokens

Note: You must configure and finalise the ActivIdentity channel and service before beginning the token import process using the IMC.

Overview of the ActivIdentity token process

To import and manage ActivIdentity tokens, you carry out the following steps, which are described in the sections that follow:

1. Configure the IMC to show the required token types.
2. Create the Issuer profiles.

An Issuer profile typically details the general configuration settings associated with all ActivIdentity tokens issued by a particular Issuing party. However, one Issuer can potentially use several Issuer profiles, depending on the configuration settings required for various batches of tokens.
3. Import the tokens.

Each ActivIdentity token contains several symmetric keys, unique to that token, that are used for the various cryptographic operations carried out by the token. These keys must be duplicated and imported into SSAS, along with the corresponding token serial number.

Each Issuer typically provides a batch of ActivIdentity tokens in an SDS file. The file contains the token data for each token: a token key and serial number data (collectively known as *token blobs*).

Note: The SDS file data is encrypted. You must have the associated SDS archive encryption key (usually obtained from a separate SDS manifest file) to import the token data into SSAS.
4. Manage the tokens.

When imported, the ActivIdentity tokens are ready for use for authentication with SSAS. You can manage the status of individual tokens, such as enabling, suspending, or revoking them.

You use the IMC to manage the ActivIdentity profiles and tokens. For information on starting the IMC, see *Opening the IMC*. When configured, the IMC groups the ActivIdentity profiles and tokens under the following token types:

- **ActivIdentityProfiles:** profile
- **ActivIdentity:** token

For a description of the fields that appear when you create, view, and update a profile or token, see *ActivIdentity profile fields* and *ActivIdentity token fields*.

Configuring the IMC to show the required token types

You must configure the IMC mappings to show the ActivIdentity token types within the IMC. You can configure the mappings either when you start the IMC or by using the IMC Preferences dialog.


To configure the mappings, follow the instructions in *Configuring the IMC token type to channel mappings*. Map both the **ActivIdentityProfiles** and **ActivIdentity** token types to your ActivIdentity channel.

When you have configured the mappings, the token types appear under the channel in the IMC tree view. You can now import, view, and edit these token types for the channel.

Creating the Issuer profiles

You use the Import dialog to create an Issuer profile. You must create an Issuer profile before you can import that Issuer's tokens.

To create an Issuer profile for a channel:

1. In the IMC tree view, select the **ActivIdentityProfiles** token type under the required channel.
2. Select **Tools > Import** or click .
The Import dialog appears. The mandatory fields are coloured yellow, the other fields are optional. For descriptions of the fields, see *ActivIdentity profile fields*.
3. Provide the required values for the Issuer profile you are creating, and click **OK**.
4. If the profile creation is successful, a success message appears. Click **OK**.
The profile is available for use within the channel. You can now import ActivIdentity tokens using that profile.

Viewing and managing the Issuer profiles

To view all the profiles, select the **ActivIdentityProfiles** token type in the IMC tree view. The profiles appear in the list pane. You can filter and sort the profiles. See *Viewing, filtering, and sorting the tokens*.

You can view, update, and delete profiles by selecting them in the list pane and performing the appropriate action (using the **Tools** menu or the icons). For more information, see:

- *Viewing details for a token*
- *Updating a token*
- *Deleting a token*

Note: You cannot delete an Issuer profile while there are tokens associated with it. Attempting to do so results in an error.

Importing the tokens

To import a batch of ActivIdentity tokens, you must have:

- The token SDS file, which contains the encrypted token blob data for the tokens
- The SDS manifest file, which specifies the SDS archive encryption key and token service names. To import the tokens, follow the instructions in *Importing tokens for a channel*.

The imported tokens all have the status that you selected during import. You can now use them for authentication with SSAS.

Managing the tokens

To view all the tokens, select the **ActivIdentity** token type in the IMC tree view. The tokens appear in the list pane. You can filter and sort the tokens. See *Viewing, filtering, and sorting the tokens*.

You can view, update, and delete tokens by selecting them in the list pane and performing the appropriate action (using the **Tools** menu or the icons). For more information, see:

- *Viewing details for a token*
- *Updating a token*
- *Deleting a token*

Note: We recommend that you do not delete tokens from the token database. If deleted, all token status information is lost, and it is possible for the original token key blob to be re-imported. If a token was previously set as **Revoked** prior to deletion, that same token can be reinstated and set to **Enabled**.

Token states and associations

Updating a token enables you to change its state or associate it with a different profile ID.

An ActivIdentity token's state can be **Enabled**, **Double_Code**, **Second_Code_Pending**, **Resync_ Pending**, **Suspended**, or **Revoked**. For descriptions, see *ActivIdentity token fields*. You cannot change the state after it has been set to **Revoked**.

Token reassignment

If a physical ActivIdentity token is reassigned by the token Issuer, its existing keys and service are deleted. It is then reconfigured with a new set of services and keys, while still maintaining the same serial number.

You can import the new SDS file, with the reassigned token data, into the IMC. As long as the existing token entry within the SSAS token database (with a matching serial number) has a state of **Revoked**, the IMC allows the new SDS token data to overwrite the old data. If the existing token entry is not in the state **Revoked**, the import of the new token data fails.

ActivIdentity profile fields

The fields that appear when you create, view, and update a profile are:

Field	Description
State	Enable or disable this Issuer profile. If disabled, token authentications for tokens that use this Issuer profile are refused. However, any tokens using the disabled Issuer can still be imported and managed using the IMC.

Profile ID	This must be unique or an error occurs on profile creation.
Failure Threshold	The number of times an incorrect response attempt is permitted before the token is automatically suspended. Default value is 3.
Double Code Threshold	<p>The number of times an incorrect response attempt is permitted before the token is put into a <i>double code</i> state. When in this state, two sequentially correct responses are required for a successful token verification. The token user has Failure Threshold number of attempts to enter the correct codes.</p> <p>This mode of operation can require a greater level of user education and management. To enable, ensure Double Code Threshold is less than Failure Threshold. To disable, ensure Double Code Threshold is more than Failure Threshold. Default value is 5 (double code mode is disabled if the default Failure Threshold of 3 is used). If enabled, the recommended value is 3.</p>
Challenge Size	<p>The challenge size, in number of digits. Defaults to 6.</p> <p>With a larger challenge, it is harder for an attacker to guess the response. However, a legitimate token user might find the challenge too long to enter with any reliability, resulting in incorrect responses and potentially suspended tokens (which then requires associated token management).</p>
Service Name Async	<p>The name of the ActivIdentity authentication service that SSAS uses to authenticate an <i>asynchronous</i> authentication response for a given token.</p> <p>An asynchronous authentication is, in ActivIdentity terminology, an authentication code generated using two variables: the challenge and the token secret key. It is therefore used by CR token service types.</p> <p>Each ActivIdentity token can support several cryptographic services. When receiving tokens from ActivIdentity, a manifest file accompanies the SDS token data file. A typical manifest file contains information about the services each token supports, as shown in the following manifest extract example (SDBauth.sdb):</p> <pre>Encryption key: 1234567812345678 serial = 2222222222 pininitval = 1254 pinposition = after pincheck = yes Services available: Service: 1, Name: KernelIdentifier, Type: 32768 Service: 2, Name: PASSWORD, Type: 201 (SDB_SERVICETYPE_ SYNC_AUTHENTICATION) Service: 3, Name: PASSWORD, Type: 200 (SDB_SERVICETYPE_ ASYNC_AUTHENTICATION) Service: 4, Name: PASSWORD, Type: 401 (SDB_SERVICETYPE_ SYNC_CERTIFICATION) Service: 5, Name: PASSWORD, Type: 400 (SDB_SERVICETYPE_ ASYNC_CERTIFICATION) Service: 6, Name: PASSWORD, Type: 500 (SDB_SERVICETYPE_ HOST_VERIFICATION) Service: 7, Name: Unlock, Type: 100 (SDB_SERVICETYPE_ TOKEN_UNLOCK)</pre> <p>Here all <i>_ASYNC_</i> based services use the PASSWORD service, as shown by services 3 and 5 in the example. This is the service name to enter.</p> <p>Defaults to PASSWORD.</p>
Service Name Sync	The name of the ActivIdentity authentication service that SSAS uses to authenticate a synchronous authentication response for a given token.

	<p>A synchronous authentication is, in ActivIdentity terminology, an authentication code generated using three variables: a time clock, an event counter, and the token secret key. It is therefore used by one time password (OTP) token service types.</p> <p>Each ActivIdentity token can support several cryptographic services. When receiving tokens from ActivIdentity, a manifest file accompanies the SDS token data file. A typical manifest file contains information about the services each token supports, as shown in the following manifest extract example for Service Name Async (above).</p> <p>Here all _SYNC_ based services use the PASSWORD service, as shown by services 2 and 4 in the example. This is the service name to enter.</p> <p>Defaults to PASSWORD.</p>
Service Name Unlock	<p>The name of the ActivIdentity authentication service that SSAS uses to unlock a given token. A token might be locked (suspended) if several incorrect authentication responses have been recorded sequentially for the token.</p> <p>When receiving tokens from ActivIdentity, a manifest file accompanies the SDS token data file. A typical manifest file contains information about the services each token supports, as shown in the manifest extract example Service Name Async (above).</p> <p>Here all _UNLOCK based services use the Unlock service, as shown by service 7 in the example. This is the service name to enter.</p> <p>Defaults to Unlock.</p>
Certificate mode	<p>This determines the type of certification service that has been used by the token to create the Message Authentication Code (MAC) received by SSAS. Can be Sync (synchronous) or Async (asynchronous).</p>
Resync Clock Range (secs)	<p>When attempting to resynchronise a token, this is the allowable clock counter window to search plus and minus the supplied value in seconds. Therefore, the total window size is twice the supplied value.</p> <p>When an ActivIdentity device is used for synchronous authentication, the device time clock and device event counter are used in the authentication process. SSAS keeps track of the values of the clock and event counter for each device, as they are used in the authentication process. When several OTPs are generated on the ActivIdentity token and not sent to the server, the device and the server become out-of-sync.</p> <p>A larger value results in an increasing probability of potential false-positive matches, reducing the security of the token OTP. However, SSAS is more likely to resynchronise with the token without manual intervention.</p> <p>Defaults to (+/-) 60 seconds.</p>
Resync Counter Range	<p>When attempting to resynchronise a token, this is the allowable event counter window to search.</p> <p>A larger value results in an increasing probability of potential false-positive matches, reducing the security of the token OTP. However, SSAS is more likely to resynchronise with the token without manual intervention.</p> <p>Defaults to 30 (up to <i>n</i> offline responses can be generated and SSAS can still resynchronise).</p>

ActivIdentity token fields

The following tables describe the fields that appear when you import, view, and update a token.

Import

Field	Description
SDS File	The .SDS archive file of a specific Issuer's token blob(s) and artefacts to be imported.
Encryption Key	Specify the clear encryption key of the archive, as received within the archive
KEK	Read-only field that displays the wrapping key (KEK) associated with the channel that the token(s) are being imported into.
Profile	Drop-down list of all configured Issuer profiles for the channel.
State	Initial state of the tokens when imported. Set this according to your token management strategy. Defaults to Enabled . Note: Although it is available, do not set an initial state of Revoked . The token state cannot be changed after a token is set as revoked.

View and update

Field	Description
Token Type	The token type: ActivIdentity.
Token Model	The token model number.
Serial Number	The token serial number.
State	<p>You can select one of the following states:</p> <ul style="list-style-type: none"> • Enabled: The token is active (authentication requests for that token are processed) and the token can be used as normal (one successful response required to authenticate). • Double_Code: The token is active, but two sequential correct responses must be received for an authentication to succeed. • Second_Code_Pending: The token is active, but it was previously in a Double_Code state. One correct authentication response has been received, the next correct response is awaited. • Resync_Pending: The token is active, but a re-synchronisation procedure is currently in progress. • Suspended: The token is inactive and authentication requests for that token are refused. The token can be unsuspended at a later date by moving it to a state other than Revoked. • Revoked: The token is inactive and can no longer be used for authentication purposes. <p>Note: When the token is set as Revoked, the token state cannot be changed using the Update Token dialog. However, there is a special circumstance whereby this can be achieved using token reassignment.</p>
Profile	Can be changed to any of the Issuer profiles within the drop-down list.

ActivIdentity service key rollover

Depending on the service administrator's key management policy, you might need to periodically change the cryptographic keys used to encrypt and, if enabled, provide tamper evidence for the token blob data. Key rollover involves changing either the Key Encryption Key (KEK) or audit key (TEMAC key). This involves:

- *KEK*: The token blob encryption keys within the token database are encrypted under the KEK. Therefore, this data must be translated from encryption under the old key to encryption under the new key.
- *TEMAC key*: With tamper evidence enabled, the token record data is protected by a keyed hash, which is generated using the TEMAC key. Therefore, this hash must be regenerated using the new TEMAC key (after first verifying the old hash using the old key).

To roll over to using a new KEK or TEMAC key:

1. Select the ActivIdentity service in the tree view.
2. In the property editor, depending on which key you want to roll over, do one of the following:
 - Select the **Key Encryption Key Label** value and click ...
 - Select the **Audit Key Label** value and click ...
3. Log into the channel token.

The dialog that appears allows you to generate and select a new key. For information on using the dialog, see *Creating, selecting, and managing Key Encryption Keys (KEKs)* or *Creating, selecting, and managing audit keys (TEMAC keys)*.
4. Select the new key, and click **OK**.

Note: Do not delete the old key. Doing so can render token data unusable. Key rollover occurs only for an individual token record when that individual record is accessed by SSAS (when an authentication request is received for that token). Currently there is no option to force an update for all token records attached to a channel.

Chapter 14: Password, SAML, and Generic MAC services

This chapter provides configuration information and procedures to help you set up the password, SAML, and Generic MAC services in SSAS. For general information about these services, such as how the authentication works, see the *SSAS Concepts Guide*. For information on using the SSAS Management Console, see *Chapter 17: SSAS Management Console*.

Password service

The Password service provides for authentication of user names with a static password.

Main configuration procedures

To configure SSAS to provide password authentication, you must perform the following procedures:


Procedure	For instructions, see:
Set up the SSAS environment by configuring: 1. <ul style="list-style-type: none"> The server and daemon interfaces. The crypto module (channel token). The database for the password information. 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i> <i>Chapter 7: Setting up the databases</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create a symmetric channel (using the Channel Wizard).	<i>Creating a channel</i>
4. Add the Password Service to the channel (using the Services Wizard). When configuring the service, select the database (or database connection pool) that holds the password information.	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> In the server, assign the Password Service to the required channel selector. Map the channel selector to the channel. 	<i>Channel selectors</i>

6. Configure any additional settings for the Password Service in the channel, as required.

Configuring additional settings for the Password Service

Configuring additional settings for the Password Service

You can access the service properties for a channel by selecting the service in the tree view.

In the property editor, use  to display all properties. The values shown reflect the settings you configured when adding the service to the channel and any changes made since.

To change a value, click it in the property editor. Help text appears in the Help pane. For information about the service properties, see the *SSAS Reference Guide*.

SAML generation service

The Security Assertion Markup Language (SAML) is an XML-based framework for communicating user authentication, entitlement, and attribute information. The XML standard allows secure web domains to exchange user authentication and authorisation data. Using SAML, an online service provider can contact a separate online identity provider to authenticate users who are trying to access secure content.

The SAML generation Service creates signed SAML messages conforming to SAML specifications versions 1.1 and 2.0.

Main configuration procedures

To configure SSAS to provide a SAML generation service, you must perform the following procedures:

Procedure	For instructions, see:
1. Set up the SSAS environment by configuring: <ul style="list-style-type: none"> The server and daemon interfaces. The crypto module (channel token). 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create an asymmetric channel (using the Channel Wizard). You do not need to choose and configure a verification mechanism (revocation checker) unless you are going to use the channel to provide a signature verification service. See <i>Revocation checking guidance</i> .	<i>Creating a channel</i>
4. Add the SAML generation Service to the channel (using the Services Wizard).	<i>SAML service configuration</i> <i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> In the server, assign the SAML 	<i>Channel selectors</i>

generation Service to the required channel selector. <ul style="list-style-type: none"> Map the channel selector to the channel. 	
6. Set up the SAML signing key.	<i>Setting up the SAML signing key</i>
7. To verifying documents containing multiple signatures, set the XPath.	<i>Verifying XML documents containing multiple digital signatures</i>
8. Configure any additional settings for the SAML generation Service in the channel, as required.	<i>Configuring additional settings for the SAML generation Service</i>

SAML service configuration

To produce SAML assertions, you must provide the following information. SAML 1.1 support:

- Whether to log the produced assertions.
- The Assertion Issuer (**Default Issuer**). If this is left blank, the distinguished name of the certificate used to sign the assertion is used.
For more information about issuers, see the SAML specification: <http://www.oasis-open.org/standards#samlv1.1>.
- The validity period of the produced assertions. If this is left blank, the value defaults to 300. This value is only used if the validity is not specified in the service request message.
The **Allow Client Issuer Override** check box, **Default Issuer Format** field, and **Allow Client Validity Override** check box are not used for SAML 1.1 assertions and can be ignored.

SAML 2.0 support:

- Whether to log the produced assertions.
- Whether to allow the client to override the server-configured Issuer and Issuer Format.
- The Assertion Issuer (**Default Issuer**). If this is left blank, the distinguished name of the certificate used to sign the assertion is used.
For more information about issuers and formats, see the SAML specification: <http://www.oasis-open.org/standards#samlv2.0>.
- The Assertion Issuer format (**Default Issuer Format**).
- Whether to allow the client to override the server-configured assertion validity.
- The validity period of the produced assertion. If this is left blank, the value defaults to 300. This value is used to calculate the **notOnOrAfter** attribute of the **<conditions>** element by adding the validity period to the message types **IssueInstant** attribute.

When you have provided the information, you can implement these additional procedures:

Item	Description
SAML 2.0 Authentication Request	An <AuthnRequest> message element can be sent to a SAML authority when attempting to obtain assertions containing authentication statements. This then returns a <Response> message containing one or more such assertions.
SAML 2.0 Logout Request	A session participant or session authority sends a <LogoutRequest> message to indicate that a session has been terminated.
SAML 2.0 Response	Generates a response when zero or more assertions satisfy a


	request. All SAML responses are of types that are derived from the StatusResponseType complex type. This defines common attributes and elements that are associated with all SAML responses.
SAML 2.0 Logout Response	Generates a logout response to be created by a session authority in response to a logout request from the principle. A SAML logout response is derived from the StatusResponseType complex type. This defines common attributes and elements that are associated with all SAML logout responses.

Setting up the SAML signing key

SAML messages must be signed with an RSA public key in accordance with the algorithm identified in the *XML Digital Signature Specification*. You can use the TMC to produce key requests for RSA public keys. These key requests must be RSA 1024-bit. For more information, see *Generating a key pair and certification request for a token*.

Configuring additional settings for the SAML generation Service

You can access the service properties for a channel by selecting the service in the tree view.

In the property editor, use  to display all properties. The values shown reflect the settings you configured when adding the service to the channel and any changes made since.

To change a value, click it in the property editor. Help text appears in the Help pane. For information about the service properties, see the *SSAS Reference Guide*.

Generic MAC service

The Generic MAC service provides verification of generic MAC or MAC-based messages. The type of MAC verification provided depends on the request and response classes. Common to all of these services is that they receive some form of MAC information about locating or deriving the corresponding key based on which verification is performed. The exact algorithm for verification is derived from the request type.

For example, the standard SSAS product provides implementations of a MasterCard EMV SecureCode authentication mechanism (compatible with CAP 2004) and service for Vasco DigiPass tokens:

- *EMV*: Verification of EMV CAP OTPs and CR codes as produced by EMV CAP 2004/2007 compliant cards and readers. Verification can be performed in software or hardware (using the HSM).
For information about this implementation, see *Chapter 9: EMV authentication*.
- *Vasco*: Verification and management of Vasco DigiPass tokens, including OTP, CR, and Vasco Signature Algorithms. Verification can be performed in software or hardware (using the HSM). For information about this implementation, see *Chapter 10: Vasco service*.

- **TEMAC:** Generation/Verification of a MAC from client data to provide tamper evidence protection.

Generation & verification can be performed in software or hardware (using the nShield Solo, or nShield Connect). For information about this implementation, see *Chapter 11: TEMAC service*.


Main configuration procedures

To configure SSAS to provide a Generic MAC verification service, you must perform the following procedures:

Procedure	For instructions, see:
1. Set up the SSAS environment by configuring: <ul style="list-style-type: none"> • The server and daemon interfaces • The crypto module (channel token) • The database for the card or token details, depending on the authentication service required 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i> <i>Chapter 7: Setting up the databases</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create a symmetric channel (using the Channel Wizard).	<i>Creating a channel</i>
4. Add the Generic MAC verification Service to the channel (using the Services Wizard).	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> • In the server, assign the Generic MAC Verification Service to the required channel selector. • Map the channel selector to the channel. 	<i>Channel selectors</i>
6. Configure any additional settings for the Generic MAC Verification Service in the channel, as required.	<i>Configuring additional settings for the Generic MAC Verification Service</i>

Configuring additional settings for the Generic MAC Verification Service

You can access the service properties for a channel by selecting the service in the tree view.

In the property editor, use  to display all properties. The values shown reflect the settings you configured when adding the service to the channel and any changes made since.

To change a value, click it in the property editor. Help text appears in the Help pane. For information about the service properties, see the *SSAS Reference Guide*.

Chapter 15: PKI services

This chapter provides configuration information and procedures to help you set up PKI signature generation and verification services in SSAS. For general information about these services, such as how the signing works, see the *SSAS Concepts Guide*. For information on using the SSAS Management Console, see *Chapter 17: SSAS Management Console*.

PKI signature generation service

The PKI signature generation service creates digital signatures using the owning channel's non-repudiation private key.

Main configuration procedures

To configure SSAS to provide a PKI signature generation service, you must perform the following procedures:

Procedure	For instructions, see:
1. Set up the SSAS environment by configuring: <ul style="list-style-type: none"> The server and daemon interfaces The crypto module (channel token) 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create an asymmetric channel (using the Channel Wizard). You do not need to choose and configure a verification mechanism (revocation checker) unless you are going to use the channel to provide a signature verification service. See <i>Revocation checking guidance</i> .	<i>Creating a channel</i>
4. Add the Signature Service to the channel (using the Services Wizard).	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> In the server, assign the Signature Service to the required channel selector. Map the channel selector to the channel. 	<i>Channel selectors</i>
6. Generate a key pair and certification request for the channel, and send it to the CA or Financial Institution (FI).	<i>Generating a key pair and certification request</i>
7. Process the certification response.	<i>Processing a certification response</i>

8. Configure any additional settings for the Signature Service in the channel, as required.	<i>Configuring additional settings for the Signature Service</i>
---	--

Generating and processing a key pair and certification request

Generating and processing a certification request for a channel token involves several steps:

1. Generate a certification request in SSAS.
2. Send the request to your CA or FI (in the case of IdenTrust) in a safe, out-of-band manner.
3. Receive a certificate response from your CA or FI in a safe, out-of-band manner.
4. Process the response in SSAS.

These following subsections describe how to generate a request and process the response.

Note: You must have access to the root certificate of your PKI hierarchy to perform these tasks. In the case of IdenTrust, this is the root of the hierarchy you want to work within (pre-production or production).

Generating a key pair and certification request

To generate a certification request for a channel token:

1. If you are planning to share the token between channels (as a Filtered Key Store), select the **PKI Environment** of your channel in the tree view. Ensure that the **Key Store Type** property has the value of **JSDPFiltered**.
2. Start the TMC and log into the channel token. See *Opening the TMC and logging into a token*.
3. Add your trusted root certificate to the token by following the instructions in *Importing a certificate from a file to a token*.
4. Generate the keys and certification request by following the instructions in *Generating a key pair and certification request for a token*.
5. Copy the certification request file to a disk (or other media) and send it to your CA or FI a safe, out- of-band manner.

If no errors were encountered during the process, and provided you started from a clean token, the channel's path view now shows a single trusted certificate and a private key with no associated certificates.

Processing a certification response

When your CA or FI receives the certification request, it first creates and signs your certificate. It might then:

- Create a certification response as a signed PKCS #7 file (of the type **Certificate and CRL dissemination**), which includes both the created certificate and its own certificate. This is then returned as the certification response.
- Return your certificate and their own certificate (issued by the Root Certificate Authority (CA), whose certificate you added to the token in the first step) as two separate certificate files.

Processing a PKCS #7 certification response

To process the certificate response:

1. Copy or move the certification response to a location accessible to your SSAS machine.
2. Select the appropriate channel or server access environment in the tree view.
3. Start the TMC and log into the channel token. See *Opening the TMC and logging into a token*.
4. Follow the instructions in *Processing a certification response for a token*.

Extracting individual certificates from a PKCS #7 file

Because of interoperability issues, the TMC is sometimes unable to process a certification response using the method described above. In this situation, you must extract the certificates from the PKCS #7 response and then add the individual files to the token.


To extract individual certificates from a PKCS #7 response on Windows 2008 R2 or 2012 R2:

1. Copy the PKCS #7 file to the Windows system and ensure the file has a .p7s or .p7c extension.
2. Double-click the file.
A Certificates viewer appears.
3. Expand the tree view to locate and select the **Certificates** directory.
4. To save your certificate to a file:
 - a. Double-click the certificate. The Certificate dialog appears.
 - b. Select the **Details** tab, and then click **Copy to File**. The Certificate Export Wizard appears.
 - c. Select **DER encoded binary X.509 (.CER)**, and then click **Next**.
 - d. Click **Browse**, specify an appropriate file name, and click **Save**.
 - e. Click **Next**.
 - f. Click **Finish**, and then click **OK** to close the confirmation message.
 - g. Click **OK** to close the Certificate dialog.
5. Repeat the previous step to save your CA's (or FI's) certificate as a file.
6. Close the Certificates viewer.
7. Copy both certificate files to a location accessible to the TMC.

You can now add the individual files to the token, by following the instructions in *Adding CA or FI certificates to a token*.

Configuring additional settings for the Signature Service

You can access the service properties for a channel by selecting the service in the tree view.

In the property editor, use  to display all properties. The values shown reflect the settings you configured when adding the service to the channel and any changes made

since.

To change a value, click it in the property editor. Help text appears in the Help pane. For information about the service properties, see the *SSAS Reference Guide*.

PKI signature verification service

The PKI signature verification service performs a full cryptographic check of a digital signature. The service uses the owning channel's revocation checking mechanism for performing a revocation check for certificates in the signer's certificate chain.

PKI signature verification for IdenTrust

IdenTrust is a PKI scheme created by the IdenTrust organisation (www.IdenTrust.com). To use the IdenTrust service in SSAS, you must configure the PKI Verify Signature Service to suit the IdenTrust system.

In one of its configurations, SSAS provides IdenTrust member Financial Institutions (FIs) and corporate customers with a cost-effective integration with the IdenTrust e-commerce trust infrastructure.

Developed to IdenTrust specifications and certified to the Network 3.0 specification, SSAS enables rapid integration of digital certificates issued by an IdenTrust approved CA into B2B applications. These applications can then inter-operate with the IdenTrust infrastructure and customers of other IdenTrust members.

IdenTrust OCSP verification mechanism

IdenTrust has several special requirements on OCSP, which the SSAS implementation supports:

- All OCSP requests must be signed with the Relying Customer's (RC) identity key.
- The AIA extension of the verified certificate must be ignored and requests always directed to the Relying Participant's (RP) OCSP responder.
- All connections to the RP's OCSP responder must utilise TLS.
- The Digital Signature Messaging System (DSMS) must accept that the OCSP responder's signing certificate is issued directly under the IdenTrust Root and not by the Relying Participant's CA (which is typical for an OCSP deployment).

To use this type of revocation checking, you only need to know the URL of your FI's OCSP responder.

IdenTrust TC verification mechanism

The IdenTrust Transaction Coordinator (TC) revocation checker internally employs OCSP as the revocation-checking request and response mechanism, but uses a more efficient protocol, based on XML, for communication. As a result, fewer requests are made to the RP (implying fewer digital signatures with the RC identity key) resulting in better performance overall. Not all FIs currently support TC services, so verify with your FI whether TC support is available.

To use this type of revocation checking, you only need to know the URL of your FI's TC.

PKI signature verification for BankID

BankID is a joint bank IT infrastructure currently run by several banks for digital identification and signing over the Internet using PKI (www.BankID.com). It was launched in 2003 to allow members of the public to authenticate themselves to government authorities, banks, and organisations remotely over the Internet, reducing the opportunity for digital fraud.

SSAS can provide the security infrastructure for all the digital identities issued over the BankID banking network. With SSAS, participating organisations can:

- Verify digital signatures generated under the BankID scheme
- Authenticate users
- Provide a strong audit trail for all transactions

You can use SSAS as a BankID Control Server (BICS).

SSAS no longer supports the BankID based client of previous versions. Instead, SSAS supports the Nexus Personal client and validates the XML digital signature created by the Personal client, known as the *BankID Signature Profile* (BISP).

SSAS also supports the ActiveX client that the Nexus CA (running under BICS) uses to sign transactions.

XML digital signature (XMLDSig) verification

XML documents can be signed to provide integrity and message authentication. The digital signature is created using an asymmetric key algorithm. A hashing function is applied to the part of the XML document being signed, and the result is encrypted by a private key. During signature verification, the corresponding public key and the hash function are used to check the signature. If the hash matches the signed part, integrity is assured.

The XML signature can be located with, or separately from, the item that is signed:

- *Enveloped signature*: Located within the XML document, it signs some part of the document.
- *Enveloping signature*: Located within the XML document, it contains the signed data.
- *Detached signature*: Located outside the XML document that contains the signed item. SSAS does not support detached signatures.

To verify signatures, ensure that the certificate of the trusted root CA is stored in the channel token.

Signed XML documents can contain a full or partial certificate chain. When verifying signatures, SSAS uses path validation to build a chain back to the trusted root certificate. See *PKI path validation* in the *SSAS Concepts Guide*.

SSAS uses an X.509 store as a cache, to save the certificates and CRLs that it obtains, so that it can re- use the cached objects when verifying other signed documents. For information on setting up an X.509 store, see *Configuring X.509 stores*.

Configuring PKI trust

SSAS supports different PKI trust levels when verifying signatures. To configure the trust levels:

1. Add the trusted Root CA certificate to the channel token.
The entire PKI hierarchy is trusted: all end-entity certificates issued under all sub-CAs.
2. Optional. To achieve a more detailed level of trust, use one of these channel selectors:

Channel selector	Add to the selector...	Result
Issuer Selector	The trusted Issuing CA DN	Only sub-domains of the PKI hierarchy are trusted: end-entity certificates issued under the specific sub-CAs
Subject Selector	The trusted end-entity certificate DN	Only individual end-entity certificates are trusted

Main configuration procedures

To configure SSAS to provide a PKI signature verification service, you must perform the following procedures:

Procedure	For instructions, see:
1. Set up the SSAS environment by configuring: <ul style="list-style-type: none"> • The server and daemon interfaces • The crypto module (channel token) 	<i>Chapter 5: Configuring the server and daemon interfaces</i> <i>Chapter 6: Configuring the crypto module (channel token)</i>
2. Create a server in the Management Console (using the Server Wizard).	<i>Creating a server</i>
3. Create an asymmetric channel (using the Channel Wizard). When configuring the channel, select the appropriate verification mechanism. See <i>Revocation checking guidance</i> .	<i>Creating a channel</i>
4. Add the following services to the channel (using the Services Wizard): <ul style="list-style-type: none"> • Verify Signature • Random number 	<i>Adding a Service to a channel</i>
5. Configure the channel selection for the channel: <ul style="list-style-type: none"> • In the server, assign both the Verify Signature and Random number services to the required channel selector. • Map the channel selector to the channel. 	<i>Channel selectors</i>
6. Add the CA or FI certificates to the channel token.	<i>Adding CA or FI certificates to a token</i>
7. Set the certificate security levels.	<i>Setting certificate security levels</i>
8. Set up the X.509 stores.	<i>Configuring X.509 stores</i>
9. To verifying documents containing multiple signatures, set the XPath.	<i>Verifying XML documents containing multiple digital signatures</i>
10. Configure any additional settings for the Verify Signature and Random number services	<i>Configuring additional settings for the Verify Signature and Random number</i>

in the channel, as required.	<i>services</i>
------------------------------	-----------------

Revocation checking guidance

When you create an asymmetric channel, you can select and configure the verification mechanism that is used, during signature verification, to check whether any certificates in the signer's certificate chain have been revoked.

Note: If the channel is not going to provide a signature verification service, you do not need to choose and configure a revocation checker.

As a rule:

- For an IdenTrust service configuration, use **Identrus OCSP** or **Identrus TC2**
- For a BankID service configuration, use **BIDT OCSP**
- For a general PKI signature verification service configuration, use the most appropriate mechanism depending on your needs

The verification mechanisms are:

Revocation checker	Description
None	No revocation checking of the certificates is performed on this channel. This is the default setting.
Generic OCSP	<p>This revocation checker conforms to RFC 2560 and requires the following information:</p> <ul style="list-style-type: none"> • Whether your OCSP service is designed to use a default OCSP server, in case no Authority Information Access (AIA) extension is present in the certificate checked, and if so, the address of your OCSP responder • Whether your OCSP responder allows plain text connections • Whether your OCSP responder requires requests to be signed (using the channel's identity key) • What is an acceptable time tolerance for responses from the OCSP responder • The host name/IP pairs of BankID application codebases, if you are using the service within the BIDT infrastructure
Identrus OCSP	<p>This revocation checker has several special requirements on OCSP, which this implementation supports:</p> <ul style="list-style-type: none"> • All OCSP requests must be signed with the Relying Customer's (RC) identity key • The AIA extension of the certificate verified must be ignored and requests must always be directed to the Relying Participant's (RP) OCSP responder • All connections to the RP's OCSP responder must utilise TLS • The Digital Signature Messaging System (DSMS) must accept that the OCSP responder's signing certificate is issued directly under the IdenTrust Root and not by the Relying Participant's CA (which is typical for an OCSP deployment) <p>Note: You can configure SSAS to perform the role of the DSMS.</p> <p>To use this type of revocation checking, you only need to know the URL of your FI's OCSP responder.</p>

BIDT OCSP	<p>This is a special revocation checker used only to verify BankID messages in the BIDT hierarchy. The following settings, special for BIDT, are relevant to this revocation checker mechanism:</p> <ul style="list-style-type: none"> • All OCSP requests are signed with the RC's identity key • A nonce is used in every request • Only the AIA extension in the signing certificate is used to find the OCSP responder • Only the signing (end-entity) certificate is checked for revocation • The certificates in the OCSP response are never checked for revocation • The Relax Signing Path option is OFF
Identrus TC2 (Transaction Coordinator)	<p>This revocation checker employs OCSP internally as the revocation-checking request and response mechanism, but uses a more efficient protocol (based on XML) for communication.</p> <p>As a result, fewer requests are made to the RP, which implies fewer digital signatures with the RC identity key and results in better overall performance.</p> <p>Not all FIs currently support TC services so you must verify with your FI whether TC support is available.</p> <p>To use this type of revocation checking, you only need to know the URL of your FI's Transaction Coordinator.</p>
CRL (Certificate Revocation List)	<p>This revocation checker can also be used to verify the revocation status of any certificate. If this method is selected, you must supply details of where the CRL can be found, and how it is downloaded or updated.</p>

Adding CA or FI certificates to a token

Manually adding certificates to a token

SSAS supports manual addition of certificates to a token. Use this option if any of the following circumstances are true:

- Rather than returning PKCS #7 certification responses, your CA or FI sends back two certificates: one of its own, the other corresponding to your certification request
- The CA or FI has created your certificate with an identity different from the one you requested
- You have received a PKCS #12 file with a private key and its certificate, and you need to add certificates to the token to complete the certificate chain stored on the token



SSAS treats manually added self-signed certificates as trusted certificates for the channel. Therefore, exercise great caution when manually adding certificates.

To add certificates to the token:

1. Ensure that both your certificate and your CA's or FI's certificate are accessible to the Management Console.
2. In the Management Console, select the appropriate channel or server access

environment in the tree view.

3. Start the TMC and log into the channel token. See *Opening the TMC and logging into a token*.
4. Follow the instructions in *Importing a certificate from a file to a token* to:
 - Import the CA or FI certificate
 - Import the end-entity certificate

Importing PKCS #12 files

Rather than allowing for a certification request/certification response sequence, some CAs generate a key pair and issue a certificate for SSAS at their own premises. These are then distributed inside a PKCS #12 file, which you can import into a channel token.

Note: FIPS 140-1 level 3 compliant hardware tokens do not allow importing of externally generated private or secret keys. If you are using such a device, you cannot import the PKCS #12 file and must perform a proper key generation, certification request, and certification response cycle as described above.

The import function supports only PKCS #12 files that contain exactly one private key and the certificate associated with it.

To import a private key and certificate from a PKCS #12 file into a channel token:

1. Because the import function supports only PKCS #12 files containing one private key and its associated certificate, ensure that you import to the token other certificates required to build a proper certificate path in your PKI first. See *Manually adding certificates to a token*.
2. In the Management Console, select the appropriate channel or server access environment in the tree view.
3. Start the TMC and log into the channel token. See *Opening the TMC and logging into a token*.
4. Follow the instructions in *Importing a key and its certificates from a PKCS #12 file into a token*.

Setting certificate security levels

About security levels

Certificate security systems, such as the BankID Control Server (BICS) and IdenTrust, require validation of certificates based on their security level. SSAS decides whether to accept a signature depending on the security level of the transaction and the security level of the certificate: the certificate's security level must be equal to, or above, the security level of the transaction.

Certificates are assigned a security level that consists of either a single letter (from A to Z) or two letters from (AA to ZZ). A given security level can comprise of any number of certificate policy IDs, issuers, or both. The name of a security level has a limit of two

characters. The security level of a certificate is indicated either by the certificate policy ID or the certificate issuer. The certificate issuer is identified by the certificate issuer DN and serial number.

A certificate policy ID has a single security level, in that all certificates issued under that policy must have the same security level. However, several policy IDs and issuers can be mapped to one security level.

For example, if you have a security level identified as *A4*, several different types of certificate can be mapped to this security level as long as all these certificates meet the requirements of this security level (even if they have been issued by different CAs/policies).

Security levels database table

The following table shows the security levels, which are pre-configured within the SSAS security levels database. The entries in this database are MAC protected.

Level	Policy ID	Issuer
A	1.2.752.78.1.1	File
B	1.2.752.78.1.2	Smart card
<i>IdenTrust: B</i> <i>Other: C</i>	1.2.752.78.1.3	WPKI Certificate on a SWIM-Chip

SSAS provides a method that returns the level of the certificate, if valid. SSAS uses the information specified above and the security levels database to determine the security level.

The Management Console enables you to add and delete new security levels. However, you cannot remove the pre-configured security levels shown in the table above.

Setting the security level

To set the certificate security level:

- Set the Issuer security level
- Set the policy ID security level

Setting the Issuer security level

To set the Issuer security level:

1. Under the **Services** node in the tree view, select **Verify Signature Service**.
2. Select the **Security Levels For Issuer** value in the property editor and click ...
The Security Level Issuer Mappings dialog appears, which contains a row of fields for each mapping. The dialog is blank when there are no mappings.
3. To add a mapping, click **New**.
A row of fields appears for the new mapping.
4. In **Security Level**, specify the appropriate letter. This can be one or two letters from A to Z.
5. Click ... (to the left of **Security Level**) and select the certificate file.
The certificate is parsed and the **Issuer DN** and **Serial Number** fields are automatically filled in.

6. When ready, click **Done**.
7. When prompted, type the master password for the server, and click **OK**.
The **Security Levels For Issuer** property value reflects the number of mappings.

Setting the policy ID security level

To set the policy ID security level:

1. Under the **Services** node in the tree view, select **Verify Signature Service**.
2. Select the **Security Levels For Policy ID** value in the property editor and click ...
The Mappings from Policy ID to Security Level dialog appears, which contains a row of fields for each mapping. The dialog is blank when there are no mappings.
3. To add a mapping, click **New**.
A row of fields appears for the new mapping.
4. In **Security Level**, specify the appropriate letter. This can be one or two letters from A to Z.
5. Click ... (to the left of **Security Level**) and select the certificate file. The certificate is parsed and the **Policy ID** field is automatically filled in.
6. When ready, click **Done**.
7. When prompted, type the master password for the server, and click **OK**.
The **Security Levels For Policy ID** property value reflects the number of mappings.

Verifying XML documents containing multiple digital signatures

To verify XML documents containing multiple signatures, the *XPath* is required to locate the signature that is provided to SSAS within the request message. For more information about XPath, see

<http://www.w3.org/TR/xpath>.

Example

To validate a SAML2Response message, containing both signed assertions and the signature of the response, individual calls must be made to SSAS to validate each signature within the message. The following XPath examples can be used to verify the signatures of the response and the first assertion.

The XPath to validate the response signature:

```
/*[node()]/node()[2]
```

The XPath to validate the first assertion signature:

```
/*[node()]/node()[4]/node()[2]
```

SSAS can validate the signature using either:

|SAML2Response RMI / AtServices
|SAML2Response SOAP

SAML2Response RMI / AtServices

When running the **doXMLDsigStatusCheck** call within **AtServices** , ensure that the parameter **signatureXPath** contains the XPath of the signature to verify:

```
AtRespMessage mresp = null;  
mresp = m_atSvc.doXMLDsigStatusCheck (xmldsigBytes,  
  
signatureXPath,  
  
nsprefixes);
```

SAML2Response SOAP


When running the **doCSCXML** call within the SOAP interface, ensure that the parameter **AdditionalXMLData** contains the XPath of the signature to verify. To do this:

1. Create an **AdditionalXMLData** object.
2. Add the XPath to the **setSignaturePath** of the **AdditionalXMLData** object.
3. Call **doCSCXML**:

```
AdditionalXMLData additionalXMLData = new AdditionalXMLData();  
additionalXMLData.setSignaturePath(signatureXPath);  
CSCResponse reszp = getClientSoapService().doCSCXML (null, null, xml, additionalXMLData);
```

Configuring additional settings for the Verify Signature and Random number services

You can access the service properties for a channel by selecting the service in the tree view.

In the property editor, use  to display all properties. The values shown reflect the settings you configured when adding the service to the channel and any changes made since.

To change a value, click it in the property editor. Help text appears in the Help pane. For information about the service properties, see the *SSAS Reference Guide*.

Chapter 16: WebPIN service

The WebPIN service can verify WebPIN passwords using a database, or convert PINs for external verification.

This WebPIN service is a legacy service that is not being actively developed. If you are using this service, contact Support for configuration information.

Chapter 17: AdES Services

This chapter provides configuration information and procedures to help you set up AdES signature generation and verification services in SSAS. For general information about these services, such as how the signing works, see the SSAS Concepts Guide. For information on using the SSAS Management Console, see Chapter 18: SSAS Management Console.

AdES signature generation service

The AdES signature generation service can be used to supply a document and create a cross-border compatible electronic signature of its contents

Main configuration procedures

To configure SSAS to provide a AdES signature generation service, you must perform the following procedures:

Procedure	For instructions, see:
1. Set up the SSAS environment by configuring: <ul style="list-style-type: none"> • The server and daemon interfaces • The crypto module (channel token) 	Configuring the server and daemon interfaces Configuring the crypto module (channel token)
2. Create a server in the Management Console (using the Server Wizard).	Creating a server
3. Create an asymmetric channel (using the Channel Wizard). When configuring the channel, select the appropriate verification mechanism. See Revocation checking guidance on page 83.	Creating a channel
4. Add the Verify Signature service to the channel (using the Services Wizard).	Adding a Service to a channel
5. Configure the channel selection for the channel: In the server, assign the Verify Signature to the required channel selector. Map the channel selector to the channel.	Channel selectors
6. Add the CA or FI certificates to the channel token.	Adding CA or FI certificates to a token
7. Set the certificate security levels.	Setting certificate security levels
8. Set up the X.509 stores.	Configuring X.509 stores

9. Configure any additional settings for the AdES Verification Service in the channel, as required.

[Configuring additional settings for the AdES Verification Service](#)

Revocation checking guidance

When you create an asymmetric channel, you can select and configure the verification mechanism that is used, during signature verification, to check whether any certificates in the signer's certificate chain have been revoked.

Note: If the channel is not going to provide a AdES signature verification service, you do not need to choose and configure a revocation checker.

The verification mechanisms for AdES are:

Revocation checker	Description
None	No revocation checking of the certificates is performed on this channel. This is the default setting.
AdES OCSP	<p>This revocation checker conforms to RFC 2560 and requires the following information:</p> <ul style="list-style-type: none"> • Whether your OCSP service is designed to use a default OCSP server, in case no Authority Information Access (AIA) extension is present in the certificate checked, and if so, the address of your OCSP responder • Whether your OCSP responder allows plain text connections • Whether your OCSP responder requires requests to be signed (using the channel's identity key) • What is an acceptable time tolerance for responses from the OCSP responder
AdES CRL	This revocation checker can also be used to verify the revocation status of any certificate. If this method is selected, you must supply details of where the CRL can be found, and how it is downloaded or updated.

Configuring additional settings for the AdES Verification Service

You can access the service properties for a channel by selecting the service in the tree view.

In the property editor, use to display all properties. The values shown reflect the settings you configured when adding the service to the channel and any changes made since.

To change a value, click it in the property editor. Help text appears in the Help pane. For information about the service properties, see the SSAS Reference Guide.

Chapter 18: SSAS Management Console

This chapter introduces the SSAS Management Console and provides procedural information for the user tasks you can perform using this console.

Note: To set up a common service implementation, follow the configuration information provided in the chapter for that implementation.

About the console

The SSAS Management Console is a graphical user interface that enables you to:

- Set up and manage the required authentication services, by configuring the server's properties files
- Start and stop the servers

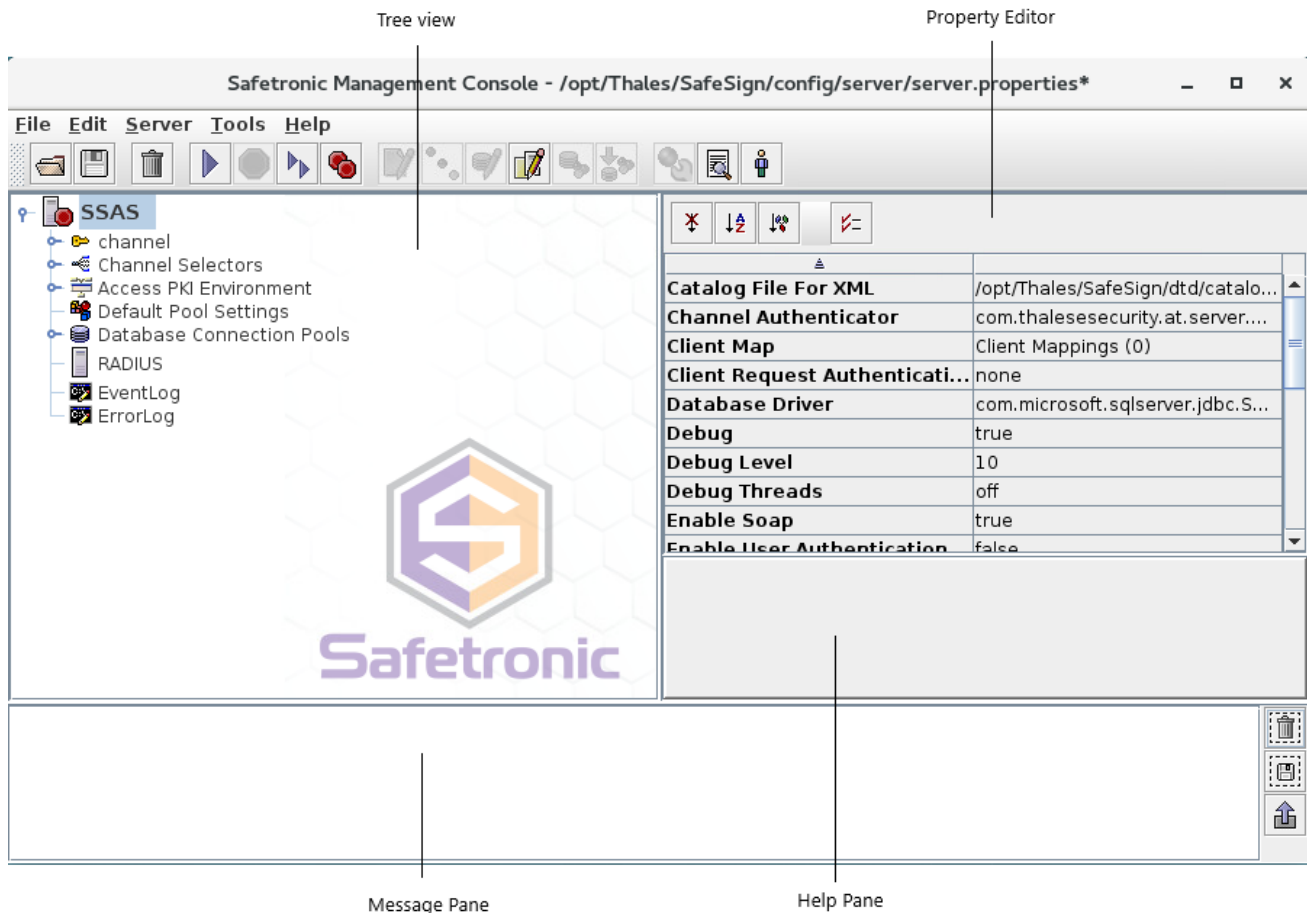
User interface

Figure 12 shows the SSAS Management Console, which has three main panes: tree view (left), property editor (right), and message pane (bottom).

Note: You can use keyboard shortcuts to cut, copy, and paste text between entry fields in the Management Console dialogs.

The Management Console prompts you to save your changes before starting a server or before exiting the Management Console. If you exit the Management Console or perform other Management Console functions without saving any modifications, your changes do not take effect.

Figure 12. SSAS Management Console



Most console controls and options have useful tool tips, which appear when you position your mouse pointer over those items. The Help pane provides guidance for the properties selected in the property editor.

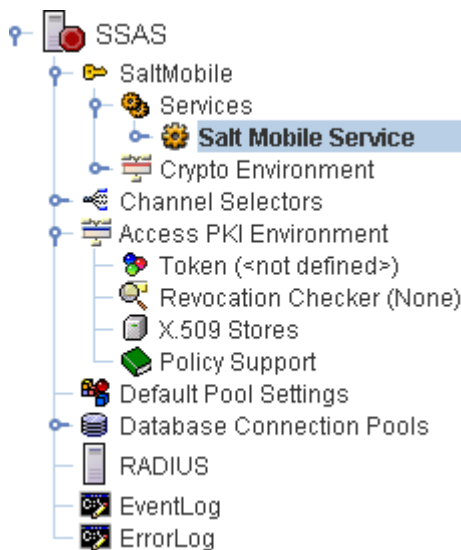
Configurations and the tree view

On first use, the Management Console shows an empty configuration. On subsequent starts, the console opens the previously edited server properties file.

You can use the console to administer one or more servers. The applicable nodes appear in the tree view when you add the servers, channels, and services. You also use the tree view to stop and start the servers.

Figure 13 shows a sample tree view, which displays configurable entities for a sample server, named *SSAS* in this example.

Figure 13. SSAS Management Console tree view



Selecting a node in the tree view displays applicable fields in the property editor, enabling you to alter those values. Right-clicking a node provides a context-sensitive menu offering actions relevant to that type of node.

Property editor

Most of the nodes in the tree view have configuration properties. The property editor shows the properties for the selected node, enabling you to edit them directly. By default, only the properties that have been set (assigned a value) are shown. The *SSAS Reference Guide* describes the properties for the various SSAS nodes.

The property editor has its own toolbar:

Icon	User tasks
	Displays the properties in their default order for the node.
	Sorts the properties into alphabetical order by property name. Toggles between ascending and descending order.
	Sorts the properties into alphabetical order by property value. Toggles between ascending and descending order.
	Displays <i>all</i> the properties for the node. Click again to show only the properties that have been set (assigned a value).



You can also click at the top of a column to change the order.

To edit a property directly, click its value (or navigate to it using the keyboard and press **F2**). The editing method depends on the type of property value to be entered. This simplifies value entry and assists with correct configuration by syntactically and semantically checking your input. The input methods are:




- Typing directly into the value box
- Selecting a value from a drop-down list of the available values
- Clicking the ... button that appears to the right of the value, which opens a separate dialog for value entry, such as a browse dialog for locating and selecting a file

Message pane

If any action conveys important information, it appears in the message pane:

- Information is in black text and preceded with 
- Errors are in red text and preceded with 

The message pane has its own toolbar:

Icon	User tasks
	Clears the message pane
	Saves the message pane contents to a text file
	Expands the message pane

You can also select text in the message pane and use **Control+C** to copy it to your computer's clipboard.

Getting started quickly

The easiest way to get started quickly is to open one of the server properties files shipped with SSAS (see *Sample SSAS configurations*), modify the settings for your own environment, and then save the changes. Saving the changes is important because modifications are not written to the active server properties file until saved.

Alternatively, you can use the Management Console Wizards, described in this chapter, to create a new configuration.

Menus and user tasks

The following table shows the main menu items and the associated user tasks, which are described in this chapter. (A subset of the menu items and user tasks are shown.)

Menu	User tasks
File	<i>Opening an existing server properties file</i> <i>Copying a server properties file</i>
Edit > New > Server	<i>Creating a server</i>
Edit > New > Channel	<i>Creating a channel</i>
Edit > New > Service	<i>Adding a Service to a channel</i> <i>Removing Services from a channel</i>
Edit > New > X509 store	<i>Configuring X.509 stores</i>
Edit > Delete	<i>Deleting a server</i> <i>Deleting a channel</i>
Edit > Configure > Configure Channel Selector	<i>Assigning services to the channel selectors</i>
Edit > Configure > Change Type	<i>Changing an object type</i>

Edit > Configure > Database Connection Pools Management	<i>Configuring a database connection pool</i>
Edit > Configure > Configure Channel Passwords	<i>Setting channel passwords</i>
Server	<i>Starting and stopping the server</i>
Tools > Manage Token	<i>Opening the TMC and logging into a token</i>
Tools > Log Viewer	<i>Opening the LVC</i>
Tools > Signature Check	<i>Checking a signature message file</i>
Tools > SSL Check	<i>Checking TLS connectivity</i>
Tools > Reset log rollback state	<i>Resetting the log rollback state</i>

Daemon and console

The SSAS daemon initialises communication between SSAS components so you must start this *before* starting the Management Console. The Management Console frequently probes the daemon for the status of the configured server. If the daemon is not started, there is a noticeable delay (due to network timeouts) when starting the Management Console and selecting a server node in the tree view.

Note: If the SSAS daemon does not start, there might be a port conflict. For information on modifying your port configuration, see *The daemon does not start*.

On Windows platforms, the SSAS installation program adds a **Salt Safetronic** item to the Windows **Start** menu, which includes some SSAS components such as the Management Console and the SSAS daemon.

On Unix-based platforms, the SSAS installation program adds shell scripts to the `<SSAS install>/bin` directory for starting the Management Console and SSAS daemon.

Starting the SSAS daemon

To start the daemon:

- *Windows platforms:* Either select **Start > Programs > Salt Safetronic > Run Daemon** or run `<SSAS install>\bin\ssas-run-daemon.bat`. A DOS command window appears, which can be minimised on your desktop.
- *Unix-based platforms:* Run `<SSAS install>/bin/ssas-run-daemon.sh`

For information on command line options for configuring the daemon, see *Configuring the daemon*.

Starting the Management Console

To start the Management Console:

- *Windows platforms:* Either select **Start > Programs > Salt Safetronic > Management Console** or run `<SSAS install>\bin\ssas-run-mgmtconsole.bat`
- *Unix-based platforms:* Run `<SSAS install>/bin/ssas-run-mgmtconsole.sh`

Configuring the daemon

This section provides information about the command line options that you can use to configure the SSAS daemon.

Scripts

SSAS installs two scripts related to the SSAS daemon: one to start the daemon, the other to stop it. It is through editing these files that the configuration options of a given SSAS daemon instance can be modified.

Note: The batch scripts assume a single instance of the daemon on any machine. If multiple instances are required, we recommend copying the batch files, then making changes to the copies as required by the second instance configuration.

It is also possible to create start and stop *super scripts* that, in turn, invoke start and stop scripts for all configured SSAS daemons.

Scripts on Windows

The two SSAS daemon related batch scripts on Windows platforms are:

- `<SSAS install>\bin\ssas-run-daemon.bat`
- `<SSAS install>\bin\ssas-stop-daemon.bat`

Note: The two start menu shortcuts in the SSAS program group, **Run SSAS Daemon** and **Stop SSAS Daemon**, are linked to these files.

Scripts on Unix

The two SSAS daemon related shell scripts on Unix-based platforms are:

- `<SSAS install>/bin/ssas-run-daemon.sh`
- `<SSAS install>/bin/ssas-stop-daemon.sh`

Both files share common environment variable settings from a third script, `<SSAS install>/bin/ssas-setenv.sh`.

Running SSAS with the Server JVM

When using the Oracle JRE we recommend running the SSAS daemon process with the Server JVM. See *Preparing Oracle Server JVM*. When you have verified that your environment supports using the Server JVM, do one of the following:

- Open the `ssas-run-daemon` script on your platform in an editor and uncomment the line where `THALES_SSAS_JVM_OPTIONS` is defined
- Define `THALES_SSAS_JVM_OPTIONS` as a system environment variable with the value `-server`

Starting the daemon

The types of parameters affecting daemon execution that can be passed on from the start script

are:

Parameter type	Description
Java parameters	<p>These include flags and parameters recognised by the Java interpreter used by the installed JVM. They are placed on the command line after the Java command keyword, but before the name of the class that is executed. An example of such a flag is the -cp flag, which is used for specifying the CLASSPATH.</p> <p>You can list the options recognised by the installed JVM's Java interpreter by entering java -help at the command prompt.</p>
Application parameters	Parameters typed on the command line after the class that is executed are passed on as arguments to the main method of the class.

The subsections that follow list possible parameters that can be passed to the SSAS daemon through the start script.

In general, the SSAS daemon Java implementation class, **com.thalesecurity.at.server.AtDaemon**, accepts application parameters in the following formats:

- **-longOptionName=value**
- **-longOptionName value**
- **-shortOptionName value**

Editing the start script to start the daemon with **-h** as an option lists all the recognised parameters and their default values.

CLASSPATH

SSAS scripts are configured to ignore any CLASSPATH defined in the environment. Instead, they utilise a variable **THALES_SSAS_CLASSPATH** that is passed to the JVM on runtime. As configured during installation, the **THALES_SSAS_CLASSPATH** passed on to the Java command executing the daemon is sufficient for most purposes.

However, if JNDI is to be used, the JNDI provider's jar and/or `jndi.properties` must be present in the **THALES_SSAS_CLASSPATH**. Similarly, if database logging or Generic MAC services are used, you must also add appropriate JDBC driver and/or data source jars to the **THALES_SSAS_CLASSPATH**.

JNDI options

The application option related to JNDI is:

Application option	Description
jndi	If JNDI is to be used, set this to true (or on or yes). Otherwise, if omitted or set to false , the RMIRegistry is used.

If JNDI is to be used, you must pass information about the JNDI provider and root context to the daemon (see *Single daemon using JNDI*). There are two ways to achieve this:

1. Modify the daemon start script to include the following Java parameters:

```
java -Djava.naming.factory.initial=<my provider class>
-Djava.naming.provider.url=<my url> <class to execute>
```

2. Supply `jndi.properties` containing these properties along the **THALES_SSAS_CLASSPATH**.

If both a `jndi.properties` file exists in the **THALES_SSAS_CLASSPATH** and Java command line parameters are passed, the latter take precedence.

RMI options

The application options related to RMI are:

Application option	Description
<code>rmiStartServer</code>	Whether a Java RMI registry is started within the <code>AtDaemon</code> . If you have other Java RMI applications that utilise an existing RMI registry, you might want to set this property to off . The default setting is on .
<code>rmiHost</code>	The RMI host name. The default setting is localhost .
<code>rmiPort</code>	The RMI port number. Ensure that this number does not conflict with other system port numbers in use.

Daemon options

The application option related to the daemon is:

Application option	Description
<code>atDaemonPort</code>	<p>The port that the daemon instance uses for communication. Use this application option if you need the daemon to use a fixed port for listening to start, stop, and is-server-running messages.</p> <p>If not present, the RMI system allocates a random free port when the daemon is started.</p>

Server properties files

To configure SSAS for your particular environment, you must specify the settings necessary for your implementation. The settings for SSAS are created and maintained in Java properties files, which contain a list of properties and their associated values.



Do not edit the server properties files manually. Doing so can cause the Management Console to read them incorrectly and lose any unsupported settings.

Using the Management Console, you need to set the property values for such SSAS functions as system configuration, PKI configuration, logging and pool management, and Remote Method Invocation (RMI) configuration.

Mandatory and optional configuration settings

SSAS provides numerous properties for flexibility and customisation. However, only a few properties are actually required to be set or modified for your SSAS implementation. To simplify set up and configuration, the SSAS Management Console filters the properties into two categories: Mandatory and Optional. The Mandatory properties can be configured from both the wizards and property editor. The Optional properties are often implicitly set to default values by the wizards, but they can be configured through the property editor.

By default, the mandatory properties display initially after a server properties file has been opened. The Mandatory properties are the minimum set of properties that need to be configured for SSAS.

To see all the properties available for configuration, or to delete or add properties, use



in the Management Console property editor.

Sample SSAS configurations

SSAS ships with the following sample configurations:

- Four server properties files, all pre-configured for revocation checking services
- One client properties file, which is for use by the SimpleClient example application included in SSAS Each configuration is described in *Sample properties files* in the *SSAS Reference Guide*.

Server properties files

The following files are located in `<SSAS install>\samples`. To simplify your SSAS configuration, you can open the appropriate file in the Management Console and modify it for your own environment. When you save the changes you have made, the file is saved as the system's active server properties file.

File	Description
csc-tc42-soft.properties	A sample configuration using the IdenTrust Transaction Coordinator for Verify Signature (VS) revocation checking. To customise this configuration for your own SSAS implementation, change the TLS host name and port number, and the Transaction Co-ordinator's Uniform Resource Identifier (URI).
multi-channel.properties	A sample multi-channel configuration with four channels, each with three services (Verify Signature, Signature, and Random Number Generation). It utilises a Signer channel selector for Verify Signature services, and a Sponsor channel selector for Signature and Random Number services.
ocsp-identrus42-soft.properties	A sample configuration using an OCSP responder for revocation checking. To customise this configuration for your own SSAS implementation, change the OCSP responder's URI.
sign-and-verify.properties	A sample signature with one channel and two services (Signature and Verify Signature), providing PKI signature

	generation and verification.
--	------------------------------

SimpleClient configuration

Located in `<SSAS install>\config\client`, the `simple-client.properties` file is used when running the SimpleClient sample application included with SSAS.


A client typically requires a configuration to establish its own PKI environment. It uses its PKI settings to perform, for example, a basic check on a digital signature message (such as PKCS #7 or BankID XML). A common component of a PKI configuration is a token. A token must always be configured in a client configuration although it is only used if signed client request authentication or client authenticated TLS for communication to the server is used.

Note: Be careful not to confuse the token specified in the SimpleClient configuration with the Relying Customer's token used by the server. They are different tokens, providing credentials for different entities.

For information on running this application, see *SOAP application examples* in the *SSAS Developer Guide for SOAP*.

Opening an existing server properties file

To open an existing properties file:

1. In the Management Console, select **File > Open** or click .
2. Locate and select the properties file, and then click **Open**.
The elements corresponding to the properties file appear in the tree view.

Copying a server properties file

To save a copy of the current properties file:

1. In the Management Console, select **File > Save As**.
2. Specify the location and file name for the new properties file, and then click **Save**.

Server

Creating a server

To create a server:

1. In the Management Console, select **Edit > New > Server**.
The Server Wizard appears, which contains several pages of fields that you use to configure the new server. The wizard fields and parameters are described in the subsections that follow.
2. Provide the required values and use **Next** to proceed through the wizard.
3. At the end of the wizard, click **Finish** to create the server. The server appears in the tree view.

You are asked if you want to start the Channel Wizard, which creates a channel for the server. See *Creating a channel*. You can start the Channel Wizard at any time.

General info page

Field	Description
Name	The name of the server being configured. A console friendly name, used as a prefix in the generated properties files. It is not the server name in the RMIRegistry or JNDI (Remote Name is used for this purpose).
Temp dir	The name and location of the directory to be used by SSAS for storing temporary files.
DB driver	<p>The database drivers (which are loaded at system startup) for the databases you have set up for SSAS to use.</p> <p>Using a comma- or space-separated list, specify the Fully Qualified Name (FQN) of the relevant driver class name for each database. Example FQNs:</p> <ul style="list-style-type: none"> <i>Oracle:</i> <code>oracle.jdbc.OracleDriver</code> <i>Microsoft:</i> <code>com.microsoft.sqlserver.jdbc.SQLServerDriver</code> <p>FQN syntax can vary slightly depending on the database version used.</p>
XML Catalog File	The full path name of an XML Catalog file. This file is used to map system and public identifiers to cached versions of the corresponding DTDs for faster XML parsing. This value is normally set to the <code><SSAS install>dtd\catalog.xml</code> file.

Daemon Connection page

You must select **RMI** or **JNDI** as the naming mechanism for the server. If RMIRegistry is selected as a mechanism, also supply the following information:

Field	Description
Host	The name of the machine RMIRegistry is running on
Port	The port number RMIRegistry is listening to

Event Logging and Error Logging pages

These wizard pages contain the same fields.

Field	Description
Logging mechanism	<p>Select the type of logging to use:</p> <ul style="list-style-type: none"> None: No logging Daemon Standard Output: Logging output is directed to the standard output of the SSAS daemon that is hosting the server Database: Logging output is directed to a JDBC compliant database File: Logging output is directed to a file Weblogic: Logging output is directed to a BEA WebLogic data source
Database connection	If Database logging is selected, a database connection must be specified as well.

	Clicking Select opens the Database Connection Pool dialog. See <i>Configuring a database connection pool</i> .
Log File	If File logging is selected, you must enter the log file name. Alternatively, use Browse to specify the file name through a file selector.

Starting and stopping the server

Before starting a server, ensure that:

- Its corresponding SSAS daemon is running. See *Starting the daemon*.
- Its corresponding databases are running, if you intend to use any to provide logging for the server or any of its channels.



or



is enabled depending on whether the selected server is running.

Starting a server

To start a server:

1. In the Management Console tree view, select the server and do one of the following:
 - Select **Server > Start Server**
 - Click
 - Right-click the server and select **Server > Start Server**. A dialog appears, requesting the password for the server.
2. Type the master password, and click **OK**. If the server is started correctly:
 - An appropriate message is displayed in the message pane.
 - The server and its nodes are greyed out in the tree view. While a server is running, you can view the configuration of its elements but you cannot modify them.

Stopping a server

To stop a server:

1. In the Management Console tree view, select the server and do one of the following:
 - Select **Server > Stop Server**
 - Click
 - Right-click the server and select **Server > Stop Server**. A dialog appears, requesting the password for the server.
2. Type the master password, and click **OK**.


When the server is stopped, its nodes in the tree view becomes enabled again, enabling you to alter its configuration.

Starting all servers

The Management Console enables you to start all configured servers. This applies even if some of the servers are already started. The console confirms the status of the running


servers and starts the other servers.

Note: To use this functionality, all servers must be protected with the same master password. To start all configured servers:


1. In the Management Console, select **Server > Start All** or click . A dialog appears, requesting the password for all the servers.
2. Type the master password, and click **OK**.

Stopping all servers

To stop all configured servers:

1. In the Management Console, select **Server > Stop All** or click . A dialog appears, requesting the password for all the servers.
2. Type the master password, and click **OK**.

Deleting a server

To delete a server, select the server in the tree view, and then select **Edit > Delete** or click . Confirm the deletion.

Channel

Creating a channel

To create a channel for a server:

1. Start the Channel Wizard in the Management Console. You can start the wizard at the end of the Server Wizard or do one of the following:
 - Select the required server in the tree view, and then select **Edit > New > Channel**.
 - Right-click the required server, and then select **New > Channel**.
2. Choose whether the channel is symmetric or asymmetric, and click **OK**. Depending on your choice, the Symmetric Channel Wizard or Asymmetric Channel Wizard appears. The wizard contains several pages of fields that you use to configure the new channel. The wizard fields and parameters are described in the subsections that follow.
3. Provide the required values and use **Next** to proceed through the wizard.
4. At the end of the wizard, click **Finish** to create the channel. The channel appears in the tree view.

You are asked if you want to start the Services Wizard, which creates services for the channel. See *Adding a Service to a channel*. You can start the Services Wizard at any time.

General info page

Field	Description
Name	The channel name, which can contain spaces. Channel names must be unique within any server.
Description	A textual description of the channel.
Use channel logging	Keep the check box clear if you want the channel to share the logging destination of event and error logs with the server. Selecting the check box adds two additional pages to the wizard, enabling you to define separate logs for the channel.

Token page

You can use the nShield Solo and nShield Connect HSMs for all asymmetric services and the symmetric Vasco service.

Field	Description
Token Type	Determines the type of token the channel uses: <ul style="list-style-type: none"> • Software: A Salt proprietary software token format. • Hardware (PKCS#11): A hardware token with a PKCS #11 interface. Use this option for nShield HSM. • Hardware (HSM8000): A hardware token with an associated keystore file. Use this option for an HSM 8000 or payShield HSM.
Channel password	Click ... to set the channel password. The Change Channel Password dialog appears. Type the master password, and then type and confirm the new channel password. Click OK .

Additional field for the **Software** token type:

Field	Description
Token File	The location of the software token file. You can use Browse to locate and select the file.

Additional fields for the **Hardware (PKCS#11)** token type:

Field	Description
Library	The location of the PKCS #11 library supplied to you by the hardware token (HSM) manufacturer. The library depends on the platform and HSM, see <i>Chapter 6: Configuring the crypto module (channel token)</i> . You can use Browse to locate and select the library.
Slot Ordinal	If the PKCS #11 library supports multi-slot devices, the slot ordinal selects which slot the channel uses. It is a zero-based integer corresponding to the ordinal of slots as listed by the PKCS #11 library.
Token Label	Specifies the token label to identify the PKCS#11 slot. It corresponds to the softcard name in nShield. This may be used in place of Slot Ordinal.

Additional fields for the **Hardware (HSM8000)** token type:

Field	Description
-------	-------------

Key Store	The keystore file location. All relevant cryptographic objects are encrypted under the HSM LMK. You can use Browse to locate and select the keystore file.
HSM Units	The HSM 8000 or payShield HSM units attached to the keystore. All units are configured with the same LMK, header length and character set. Click Configure to view and select the units.

Revocation Checking page (asymmetric channel only)

Field	Description
Verification mechanism	<p>The verification mechanism that the channel uses:</p> <ul style="list-style-type: none"> • None • Generic OCSP • Identrus OCSP • BIDT OCSP • Identrus TC2 • CRL <p>For an explanation of each of these mechanisms, see <i>Revocation checking guidance</i>.</p>

Additional fields for the **Generic OCSP** verification mechanism:

Field	Description
Default OCSP URL	The URL of the OCSP responder to be used for certificate verification if a certificate has no AIA extension, or if Always use default URL is selected
Always use default URL	Whether the default URL must be used even if the certificate has an Authority Information Access (AIA) extension
Sign Requests	Whether OCSP requests are signed or not
Always use HTTPS	Forces SSAS to use a TLS connection to the OCSP server

Additional field for the **Identrus OCSP** verification mechanism:

Field	Description
Financial OCSP URL	The URL of the OCSP responder to be used for certificate verification if a certificate has no AIA extension

Additional field for the **Identrus TC2** verification mechanism:

Field	Description
Financial TC URL	The Uniform Resource Locator (URL) of the Relying Participant's Transaction Coordinator

Additional fields for the **CRL** verification mechanism:

Field	Description
Use LDAP	Whether an LDAP directory is contacted for downloading CRLs. Provide the URL to the LDAP directory storing CRLs appropriate for

	the channel's PKI hierarchy. If the LDAP directory requires login for read access, provide an appropriate User DN (user name) and User password .
Use SSL	(Field available if Use LDAP is selected.) Whether TLS is used for communication with the directory.
Force CRL Download	(Field available if Use LDAP is selected.) Whether the latest CRL is always obtained, even if a valid one exists in the cache.
Use Delta CRLs	(Field available if Use LDAP is selected.) Whether the usage of delta CRLs is allowed.

HTTP&SSL page

Field	Description
Socket timeout	The maximum time (in milliseconds) that a read operation from a socket is allowed to block, before the operation is interrupted. 0 is default and means no timeout (infinite wait).
Use proxy	If you must use a proxy for outward HTTP and HTTPS traffic, select this check box, and specify the Proxy Host and Proxy Port .

Note: SSAS has separate configurations for the HTTP and HTTPS proxies. The proxy values that you enter here are used to configure both of these. However, any subsequent modifications to one of the configurations are not propagated to the other configuration. Therefore, if you are using both proxy mechanisms and want to make changes, you must remember to change both.

Event Logging and Error Logging pages

These wizard pages contain the same fields as the Server Wizard. See *Event Logging and Error Logging pages*. However, these fields relate to the logging of channel-level data.


Setting channel passwords

There is a channel password for each channel's cryptographic token. SSAS encrypts each channel password with the master password and stores it in the server properties file. For more information, see *Channel tokens and passwords* in the *SSAS Concepts Guide*.

SSAS requires the channel passwords to be set in advance so that you only need to enter the master password every time you start the server. You can set and change the channel passwords using the following methods.

Changing all passwords for a server

To change the passwords for a server and its channels:

1. Select the server in the tree view, and then click  or select **Edit > Configure > Configure Channel Passwords**.
2. Type and confirm the new master password, then click **OK**.
A New Password dialog appears for each channel within the server.
3. On each dialog, type and confirm the new password for that channel, and click **OK**.
To retain the existing password, click **Cancel**.

Note: If two channels share a token (their tokens have identical configurations), only one password dialog is presented. To override this behaviour, use the procedure for changing individual passwords (in the next section).

If the server has an access environment configured, its password is also set.


If a channel token is annShield HSM, observe the requirements for password entry in *Opening the TMC and logging into a token*.

Changing individual channel passwords

To change the password for a channel:

1. Select the channel in the tree view.
2. Select the **Password** value in the property editor and click ... The Change Channel Password dialog appears.
3. Type the master password.
4. Type and confirm the new channel password, and click **OK**.

Deleting a channel

To delete a channel, select the channel in the tree view, and then click  or select **Edit > Delete**. Confirm the deletion.

Service

Adding a Service to a channel

You add Services to a channel by using the Services Wizard. The wizard shows only the appropriate services for the selected channel type (symmetric or asymmetric). There is a wizard page for each of the Services that can be added, allowing you to enable and configure the required services.



The Services Wizard affects *all* the services for a channel, not only the new ones you are adding.

Therefore, do not deselect the check boxes for the existing services unless you want to remove them from the channel.

To add a Service to a channel:

1. Start the Services Wizard in the Management Console. You can start the wizard at the end of the Channel Wizard or do one of the following:
 - Select the required channel in the tree view, and then select **Edit > New > Service**
 - Right-click the required channel, and then select **New > Service**

The Services Wizard appears, which contains several pages of fields. Each page relates to a service that you can configure for the channel. The wizard fields and parameters are described in the subsections that follow.

Depending on your choice, the Symmetric Channel Wizard or Asymmetric Channel Wizard appears.
2. Use **Next** to proceed through the wizard, and locate and select the services that you require. Provide the required values to configure each service.
3. When you have finished configuring the services, click **Finish** to add them to the channel. The services appear in the tree view.

Symmetric channel service pages

Random number service page

Field	Description
Use Random Numbers Generation Service	Whether this service is used in the channel
Log Random Response	Whether the generated random material is recorded in the event log

Generic MAC verification service page

Field	Description
Use generic MAC Verification Service	Whether this service is used in the channel.
Database connection File	The database connection, or a file, that holds keys and/or key derivation information. Select either Database connection or File and then use the appropriate button (Select or Browse). Clicking Select opens the Database Connection Pool dialog. See <i>Configuring a database connection pool</i> . Clicking Open opens a file browsing dialog.

WebPIN Verification service page

Field	Description
Use WebPIN Verification Service	Whether this service is used in the channel.
Database connection	The database connection to the WebPIN token database. Clicking Select opens the Database Connection Pool dialog.

	See <i>Configuring a database connection pool</i> .
Private key identifier	The key identifier (the alias of the token key as presented by the TMC) of the private key.
ZPK ID	The identifier of the ZPK, which is used to return data.
PEK identifier	The identifier of the PEK, which is used to store passwords in the database.

OATH service page

Field	Description
Use OATH Service	Whether this service is used in the channel.
Log Request	Whether the request data is recorded in the event log.
Database connection	The database connection to the OATH database. Clicking Select opens the Database Connection Pool dialog. See <i>Configuring a database connection pool</i> .
Window	The required counter window. For more information, see <i>OATH reference</i> in the <i>SSAS Reference Guide</i> .
Throttle Value	The maximum number of failed OTP validation attempts, after which the token becomes blocked.
Maximum Resynchronisation Window	The temporary window value that is used during attempts to resynchronise the SSAS counter with the token counter if it has incremented beyond value set for Window above. For more information, see <i>OATH reference</i> in the <i>SSAS Reference Guide</i> .
Audit records	Whether OATH database record auditing is enabled. This field is not available if you are using a software token.
Audit key identifier	The audit (TEMAC) key to use for tamper evidence protection of OATH records. Clicking Select enables you to generate and select a TEMAC key. See <i>Creating, selecting, and managing audit keys (TEMAC keys)</i> .

Password service page

Field	Description
Use Password Service	Whether this service is used in the channel.
Database connection	The database connection to the password database. Clicking Select opens the Database Connection Pool dialog. See <i>Configuring a database connection pool</i> .
Audit records	Whether password database record auditing is enabled. This field is not available if you are using a software token.
Audit key identifier	The audit (TEMAC) key to use for tamper evidence protection of password service records. Clicking Select enables you to generate and select a TEMAC key. See <i>Creating, selecting, and managing audit keys (TEMAC keys)</i> .

Salt Mobile service page

Field	Description
Use Salt Mobile Service	Whether this service is used in the channel.
Database connection	The database connection to the Salt Mobile database. Clicking Select opens the Database Connection Pool dialog. See <i>Configuring a database connection pool</i> .
Log Request	Whether the request data is recorded in the event log.

ActivIdentity service page

Field	Description
Use ActivIdentity Service	Whether this service is used in the channel.
Log Request	Whether the request data is recorded in the event log.
Database connection	The database connection to the ActivIdentity database. Clicking Select opens the Database Connection Pool dialog. See <i>Configuring a database connection pool</i> .
Key Encryption Key Label	The Key Encryption Key (KEK) to use to encrypt the token encryption keys (TEKs). Clicking Select enables you to generate and select a KEK. See <i>Creating, selecting, and managing Key Encryption Keys (KEKs)</i> .
Audit records	Whether tamper evidence checking on ActivIdentity token records is enabled. This field is not available if you are using a software token.
Audit Key Label	The audit (TEMAC) key to use for tamper evidence protection of ActivIdentity token records. Clicking Select enables you to generate and select a TEMAC key. See <i>Creating, selecting, and managing audit keys (TEMAC keys)</i> .

Asymmetric channel service pages

Verify Signature service page

Field	Description
Use Verify Signature Service	Whether this service is used in the channel.
Log Message	Whether the message (whose certificates are to be checked) is recorded in the event log. BankID messages are always recorded.
Log Message Content	Whether the content of the message (whose certificates are to be checked) is recorded in the event log. This setting applies to PKCS #7 messages with detached content only

(implicit messages).

Random number service page

This wizard pages contain the same fields as the Random number service page for a symmetric channel. See *Random number service page*.

Signature service page

Field	Description
Use Signature generation Service	Whether this service is used in the channel
Log DTS	Whether the Data-To-be-Signed (DTS) is recorded in the event log
Log Signature	Whether the generated signatures are recorded in the event log

SAML Generation service page

Field	Description
Use SAML Generation Service	Whether this service is used in the channel.
Log SAML Response	Whether the produced SAML message (produced assertion) is recorded in the event log.
Allow Client Issuer Override (SAML2 Only)	Whether the client is allowed to override the server-configured Issuer and Issuer Format.
Default Issuer	The Assertion Issuer. If this is left blank, the distinguished name of the channel's identity key's certificate (which is used to sign the assertion) is used.
Default Issuer Format	The Assertion Issuer format.
Allow Client Validity Override (SAML2 Only)	Whether the client is allowed to override the server-configured assertion validity.
Default Validity (seconds)	The validity period of the message (produced assertion).

Creating, selecting, and managing Key Encryption Keys (KEKs)

You can create and use Key Encryption Keys (KEK) for the ActivIdentity service. KEKs are effectively wrapping keys, which are used to encrypt Token Encryption Keys (TEKs). KEKs are stored on the channel token.

A TEK is automatically generated for each ActivIdentity Issuer profile, and is used to encrypt the Issuer's token key data within the database. The TEK is then stored in the database, encrypted under the channel's ActivIdentity KEK.

To create or select a KEK:

1. Do one of the following:
 - In the ActivIdentity service page (of the Services Wizard), click **Select** next to the **Key Encryption Key Label** field
 - Select the **Key Encryption Key Label** value in the property editor and click ...
2. When requested, log into the channel token.

The Choose Key Encryption Key dialog appears, listing the existing KEKs and enabling you to generate, select, and manage keys. On first use, the list is empty and you must generate a key.

Note: If you are configuring SSAS to connect to an existing token database, you can select a previously used KEK or a new KEK. Token data imported with a previously used KEK is still usable (even if a new KEK is selected) as long as the previous KEK is present on the channel token.
3. To generate a new key, click **Generate Key** and, if required, log into the channel token. The new key appears in the list.
4. Select the required key, and click **OK**.

The Choose Key Encryption Key dialog also enables you to back up, restore, and delete KEKs.

To back up a key to a file encrypted under the channel token LMK, select the key and click **Backup**. Specify the location and file name of the backup file.

To restore a backed-up key, click **Restore**, and locate and select the key backup file.

Note: You can only restore the key to a channel token that uses the same LMK as the token used to create the initial key backup file.

To delete a key, select the key and click **Delete**. Confirm the deletion.

Note: You must not delete keys from the channel token unless you are sure they are not used by any channel with the ActivIdentity service configured. Deleting a key that is used by a channel renders any tokens associated with that key unusable. We recommend that you back up keys, and correctly administer the backup archive, before deleting any keys.

Creating, selecting, and managing audit keys (TEMAC keys)

You can create and use audit (TEMAC) keys with the ActivIdentity service to providing a secure audit trail. SSAS uses the keys to generate MAC values for each entry in the ActivIdentity database, and those values can be used later to verify the database entries.

To create or select a TEMAC key:

1. Click **Select** next to the **Audit key identifier** field.
2. When requested, log into the channel token.
The Choose Audit Key dialog appears, listing the existing audit keys and enabling you to generate, select, and manage keys. On first use, the list is empty and you must generate a key.
Note: If you are configuring SSAS to connect to an existing token database, you can select a previously used TEMAC key or a new TEMAC key. Token data imported with a previously used TEMAC key is still usable (even if a new TEMAC key is selected) as long as the previous TEMAC key is present on the channel token.
3. To generate a new key, click **Generate Key** and, if required, log into the channel token. The new key appears in the list.
4. Select the required key, and click **OK**.

The Choose Audit Key dialog also enables you to back up, restore, and delete audit keys.

To back up a key to a file encrypted under the channel token LMK, select the key and click **Backup**. Specify the location and file name of the backup file.

To restore a backed-up key, click **Restore**, and locate and select the key backup file.

Note: You can only restore the key to a channel token that uses the same LMK as the token used to create the initial key backup file.

To delete a key, select the key and click **Delete**. Confirm the deletion.

Note: You must not delete keys from the channel token unless you are sure they are not used by any channel with this service configured. Deleting a key that is used by a channel renders any tokens associated with that key unusable. We recommend that you back up keys, and correctly administer the backup archive, before deleting any keys.

Removing Services from a channel

You can remove Services from a channel by using the Services Wizard and deselecting the services you no longer require.



The Services Wizard affects *all* the services for a channel. Therefore, do not deselect the check boxes for the services that you want to keep.

Note: You cannot remove all services from a channel. At least one service must be left in the channel. To remove services from a channel:

1. Do one of the following:
 - Select the required channel in the tree view, and then select **Edit > New > Service**
 - Right-click the required channel, and then select **New > Service**
2. Use **Next** to proceed through the wizard, and locate and deselect the services that you want to remove.
3. When you are ready, click **Finish** to remove the services from the channel.

Channel selectors

When you have defined the channels and services you want to use for a server, you must configure the channel selectors so that SSAS knows how to distribute incoming requests to the channels.

Note: You must configure a channel selector even if you only use one channel. In this situation, use the Fixed Channel Selector.

Configuring channel selection is a two-step process, see *Channel selector examples* in the *SSAS Concepts Guide*. You must:

- Assign the used services to the required channel selectors
- Map each channel selector to the required channels

All services support the use of the Sponsor Selector, Client Selector, and Fixed Channel Selector. The Verify Signature and IdenTrust payment services are typically configured to use the Issuer Selector or Subject Selector. The Sponsor Selector is used in situations where SSAS is asked to do something on behalf of the client application (for example, sign some data with a channel's keys), so the client application needs to indicate which channel it wants to use.

Channel selector	To configure this selector, you must:
Issuer Selector	<ul style="list-style-type: none"> • Gather the DNs for all the issuers whose certificates you expect SSAS to perform Verify Signature checking. • Map the DNs to the required channel.
Subject Selector	<ul style="list-style-type: none"> • Gather the DNs for all the signers whose certificates you expect SSAS to perform Verify Signature checking. • Map the DNs to the required channel.
Client Selector	<ul style="list-style-type: none"> • Configure the server to use signed request authentication. This is required because the selector uses the subject DN of the client issuing the request. For information about request authentication, see <i>Request authentication</i>. • Gather the DNs for all clients that are sending requests to the server. • Map the DNs to the required channel.

Sponsor Selector	<ul style="list-style-type: none"> • Gather all the sponsor strings your application uses. • Map the sponsor strings to the required channel.
Fixed Channel Selector	Map the service to the required channel.
White-Label Channel Selector	<ul style="list-style-type: none"> • Gather the DNs for all the issuers whose PKCS #7 certificates you expect SSAS to perform Verify Signature checking. • For the issuers that you want the Sponsor Selector to handle: • Configure the White-Label Channel Selector, mapping the DNs to sponsor string prefixes. • Configure the Sponsor Selector, mapping the full sponsor strings (with prefixes) to the required channel. • For the other issuers, configure the Issuer Selector, mapping the DNs to the required channel.
Generic MAC Channel Selector	<ul style="list-style-type: none"> • Gather the specific MAC-based service type in use. This can either be EMV, TEMAC, DigiPass, Gemplus OTP or Salt Mobile. The implementation class names to be used in the channel selector configurations are: <ul style="list-style-type: none"> - com.thalesesecurity.at.service.impl.EMVGenMACVerification - com.thalesesecurity.at.service.impl.TEMACService - com.thalesesecurity.at.service.impl.DigiPassGenMACVerification - com.thalesesecurity.at.service.impl.DigiPassManagementService - com.thalesesecurity.at.service.impl.GemplusOTPVerification - au.com.saltgroup.mobile.service.SaltMobileServices

The subsections that follow provide information on configuring the channel selectors. For further configuration information, see *Channel selector settings* in the *SSAS Reference Guide*


Assigning services to the channel selectors

When configuring channel selection for a server, the first step is to assign the services you are using to the channel selectors that you want SSAS to use when directing requests.

Notes:

- You can assign multiple services to a channel selector.
- You cannot assign a service to multiple channel selectors.
- You must assign all the services that you are using in your channels. However, you do not need to assign any service types that you are not using.

To assign services to a channel selector:

1. Under the **Channel Selectors** node in the tree view, select the channel selector and do one of the following:
 - Select **Edit > Configure > Configure Channel Selector**
 - Click 
 - Right-click the channel selector and select **Configure > Configure Channel Selector**. The Choose Services dialog appears, listing all the available

services in the left-hand list box.

Note: Services already assigned to other channel selectors are not listed. For information on reassigning services, see *Reassigning services to different channel selectors*.

2. Move the services that you want the channel selector to handle to the right-hand list box, by selecting the services and using the arrow buttons. You can use **Shift** or **Control** to select multiple services. The single arrow moves the selected services in that direction. The double arrow moves all the services in that direction.
3. When the right-hand list box contains the required service types, click **OK**. The service types appear under the channel selector in the tree view.

Repeat the procedure until all the service types that you are using have been assigned to a channel selector.

Mapping channel selectors to the required channels

When configuring channel selection for a server, the second step is to map each channel selector to the required channels. This involves providing the channel selectors with sufficient information to direct requests among the configured channels.


Notes:

- You can map a channel selector to multiple channels.
- You can map multiple channel selectors to the same channel.
- You must map all the channel selectors that you assigned services. However, you do not need to map any of the other channel selectors.

To configure the mappings for a channel selector:

1. Under the **Channel Selectors** node in the tree view, select the channel selector.
2. Select the **Channel Map** value in the property editor and click ...
The Channel Mappings dialog appears, which contains a row of fields for each mapping. The dialog is blank when there are no mappings.
3. To add a mapping, click **New**. A row of fields appears for the new mapping.
Provide the required values to configure the mapping.
The fields depend on the type of channel selector:

Field	Description
Issuer DN	The distinguished name of the Issuer's verified signature signer's certificate
Subject DN	The distinguished name of the subject's verified signature signer's certificate
Sponsor	An arbitrary string agreed between the client and server
Service	A service fixed to a single channel
Channel	The name of the mapped channel
Prefix	A pre-defined prefix that is combined with the Issuer's distinguished name to create a unique mapping

4. Repeat the procedure to add further mappings. To remove a mapping, click  next to the mapping row.
5. When the dialog contains the required mappings for the channel selector, click **Done**.
6. The **Channel Map** value in the property editor shows the number of mappings.

Repeat the procedure until all the channel selectors with assigned service types have been mapped to the required channels.

Note: If you are using the White Label Selector, you must also configure the Sponsor Selector and Issuer Selector.

Reassigning services to different channel selectors

If you have already assigned a service to a channel selector, you cannot reassign it to a different channel selector without first removing it from the existing channel selector.

To reassign a service to a different channel selector:

1. Open the Choose Services dialog box for the *existing* channel selector, move the

service back to the left-hand list box and click **OK**.

2. Open the Choose Services dialog box for the *new* channel selector, move the service to the right-hand list box and click **OK**.

For information on using the Choose Services dialog box, see *Assigning services to the channel selectors*.

Note: Adding, removing, or reassigning services does not change the mappings for the channel selectors.


Database connection pools

Servers can potentially run many different pools of objects. For example, services, database connections, and signature objects are all pooled. While you can configure each pool separately to fine tune its own performance requirements, SSAS also allows you to change default pool properties for *all* pools across a server.

Naturally, local pool settings take precedence over the server-wide settings (which are specified in **Default Pool Settings**). For example, the **Maximum Size** property is set to **6** in the **Default Pool Settings**, and **10** in the **Signature Service Pool**. The **Signature Service Pool** setting overrides the server-wide setting, and it runs with 10 service object instances pooled.

Configuring the default pool properties

To configure the default pool properties for a server:

1. In the Management Console, select **Default Pool Settings** under the server in the tree view.
2. In the property editor, use  to display all properties. Set the properties as required. For information about the Default Pool Settings properties, see *Pool properties* in the *SSAS Reference Guide*.

Configuring a database connection pool

To create a database connection pool:


1. In the Management Console, either:
 - Select the server in the tree view, and then select **Edit > Configure > Database Connection Pools Management**
 - Right-click **Database Connection Pools**, and then select **Configure > Database Connection Pools Management**The Database Connection Pool dialog appears.
2. Click **New**.
3. In the New database pool dialog, provide the required values to configure the database connection.

The fields and parameters are:

Field	Description
Pool Name	A suitable name describing the database pool, which is used to identify the pool when creating and configuring services
Database URL	Specify the database connection string using the relevant JDBC format, for example: <ul style="list-style-type: none"> <i>Oracle</i>: jdbc:oracle:thin:@machineName:1521/databaseName <i>Microsoft</i>: jdbc:sqlserver://machineName;databaseName=databaseName
Username	The user name for the account that SSAS uses when connecting to the database server
Password	The password for the account that SSAS uses when connecting to the database server
Master password	The master password for the server (SSAS)

- Click **OK**.
- Close the Database Connection Pool dialog.
The pool appears under the **Database Connection Pools** node in the tree view.

To edit an existing database connection pool:


- Select the pool under the **Database Connection Pools** node in the tree view.
- In the property editor, use  to display all properties. Set the properties as required. For information about the Default Pool Settings properties, see *Pool properties* in the *SSAS Reference Guide*.

Changing an object type


Some objects in the Management Console tree view are polymorphic, which means that you can change their object type. The following objects are polymorphic:

- Tokens
- Revocation checkers
- Event logs
- Error logs

To change an object type:

- Select the object in the Management Console tree view, and do one of the following:
 - Select **Edit > Configure > Change Type**
 - Click 
 - Right-click the object, and then select **Configure > Change Type**
- Select the new object type and click **OK**.

After changing an object type, select the object and use the property editor to set the

required properties. Use  to display all properties. For information about the properties, see the *SSAS Reference Guide*.

Configuring X.509 stores

For a PKI environment, you can set up an ordered list of X.509 stores, which are used in conjunction with CRL revocation checking. The X.509 stores appear under the **Access PKI Environment** node in the Management Console tree view.

Note: The order that the stores are defined is important because it represents the search order. For more information, see *X.509 stores* in the *SSAS Reference Guide*.

To configure a new X.509 store:

1. In the Management Console tree view, select the **X.509 Stores** node under **Access PKI Environment**, and either:
 - Select **Edit > Configure > Change Type**
 - Right-click the node, and then select **New > X509 Store**
2. Select the type of store (**Memory Store** or **LDAP Store**), and click **OK**. The store appears under the **X.509 Stores** node.
3. Select the store and use the property editor to set the required properties.
For information about the properties, see *X.509 stores* in the *SSAS Reference Guide*.

Configuring User Authentication

The User Authentication component enables SSAS to query an external user store to retrieve user OTP token information. This functionality is primarily designed to be used with the MyID as the external user store but you can customise it to interact with external third-party user stores.

Note: Access to third-party user store formats is limited to the database types supported by the SSAS database connection pool implementation.


The information returned about a token usually includes the token serial number, token type, token model, and algorithm information. A user can have multiple tokens within the external user store and all tokens are returned by the User Authentication component.

In the situation when multiple tokens are returned, the User Authentication component attempts to prioritise the tokens into an order based on which tokens are most likely to be successful. The User Authentication component is supplied with a default token prioritiser but can also be extended to use a custom-written prioritiser. The default token prioritiser performs the ordering process based on the last time the token was used successfully. This information comes from a cache that SSAS stores securely to log all successful transactions.

The User Authentication component supports the following services (highest priority first):

- Vasco
- ActivIdentity
- EMV
- OATH
- Salt Mobile
- Password

To enable the User Authentication component:

1. In the Management Console, select the server in the tree view.
By default, the server's **Enable User Authentication** value in the property editor is set to **false**.
2. Set the **Enable User Authentication** value to **true**.
The **User Authentication** node appears under the server in the tree view.
3. Select User Authentication in the tree view.
4. In the property editor, use  to display all properties, and then set the values as required.
For information about the User Authentication properties, see *User Authentication* in the *SSAS Reference Guide*.

You can use either the SOAP API or the RADIUS API to support the User Authentication component.

User Authentication using the SOAP API

To support User Authentication, SSAS has a SOAP plug-in called **UserAuthenticationSoap**, which provides the following methods:

- **BasicResponse doVerify(String transID, String username, String password, ExtraData extraData)** throws **RemoteException**
- **BasicResponse doVerifyWithType(String transID, String username, String password, String tokenType, ExtraData extraData)** throws **RemoteException**

The **UserAuthenticationSoap** plug-in is automatically deployed when the User Authentication component is enabled within the SSAS Management Console, and undeployed when disabled.

The two **doVerify** methods defined within the SOAP plug-in are designed to allow a user to be authenticated without the client needing to specify any token-specific information other than the OTP (and optionally the token type, to narrow the authentication process).

With the **doVerify** methods:

1. The user name is looked up in the external user store configured for the User Authentication component.
2. The user's tokens are returned. If the **doVerifyWithType** method was called, the tokens are narrowed by token type.
3. The remaining tokens are prioritised based on the default prioritiser, unless a custom prioritiser has been created and configured for the User Authentication component.
4. Each token is taken in turn and authentication is attempted with the supplied OTP until either authentication succeeds or there are no more tokens to attempt.
5. SSAS updates its internal cache and the appropriate response is returned to the client.

User Authentication using the RADIUS API

The RADIUS solution supports authentication of OTPs through the User Authentication component. To enable RADIUS to pass all requests to the User Authentication component:

1. Configure the User Authentication component in the Management Console.
2. In the Management Console, select RADIUS under the server in the tree view.
3. In the property editor:
 - a. Set **Enable Radius** to **true**.
 - b. In **Radius Database Connection**, configure a new connection.
 - c. Set **Radius Users BackendType** to **internalUserAuthentication**.

Testing your configuration

The configuration procedures described in this guide can be used to configure SSAS for both a production environment and a testing environment. We recommend that you test your configuration setup initially in a pre-production setting, allowing time to verify connections between participants and troubleshoot application interactions.

The SSAS Management Console provides you with several diagnostic tests to ensure your configuration is complete and your connections have been set up correctly. This section provides information on running these tests.

Note: Diagnostic tests are independent of the running state of the server. The server can be started or stopped during the tests.

Analysing test results

The results of the diagnostic tests appear in the message pane. The information provided is useful for verifying the success of your infrastructure interactions, or troubleshooting any problems indicated by test failures.


Figure 14 shows an example of diagnostic test output. Each test is preceded by the phrase **START OF TESTS** with a brief description of the test purpose and its subsequent results. If the test passes successfully, an **OK** precedes the results. If the test fails, the text is displayed in red and preceded with 

Figure 14. Sample diagnostic output

```

** START OF SSL TESTS **
OK - Opened Token and Logged In
Connecting to identrus10.identrus.com:443
OK - Host reachable
Starting SSL Handshake
com.entegrity.jsdp.iaik.security.ssl.SSLException: Server certificate
rejected by ChainVerifier
Discontinuing further tests
** END OF SSL TESTS **

```

Checking a token

To verify the settings of a channel or access environment token, open the TMC, log into the token, and use the path view to determine the status of the token. For information on doing this, see *Interpreting a token using the path view*.

Checking TLS connectivity

The TLS check verifies:

- That a given host and port are reachable from your computer
- That a TLS handshake can be performed to the accessed host and port

Prerequisites

Before performing this test, you must run a successful token check on the channel.

Performing the test

To check the TLS connectivity for a channel:

1. In the Management Console tree view, select the channel and either:
 - Select **Tools > SSL Check**
 - Right-click the channel and select **Tools > SSL Check**
2. If requested, type the channel password (*not* the master password), and click **OK**.
3. If requested, type the master password for the server, and click **OK**. The Select Host and Port dialog appears.
4. Specify the **Host** and **Port** to connect to, and click **OK**. You can specify the host either by name (must be DNS resolvable) or by IP address.
The tests begin to run, with the results appearing in the message pane.

What is tested?

The test first checks that the computer running the Management Console can connect to a host. If the connection is successful, the test attempts a handshake using the settings of the channel's PKI environment, including the token and TLS specific settings.

Checking a signature message file

The signature check verifies:

- That the message (PKCS #7 or BankID XML) file can be decoded
- That the digital signature on the message file is valid within the context of the selected channel
- The revocation status of each certificate included on the message

To run the signature check, you must have access to a PKCS #7 or BankID XML file. If you do not have a PKCS #7 or BankID XML file, contact Salt Support for assistance.

Prerequisites

Before performing this test:

- You must run a successful token check on the channel
- If your channel's revocation checking utilises TLS to connect to revocation checking resources, perform a TLS check

Performing the test

To check the signature message file for a channel:

1. In the Management Console tree view, select the channel and either:
 - Select **Tools > Signature Check**

- Right-click the channel and select **Tools > Signature Check**
- 2. If requested, type the channel password (*not* the master password), and click **OK**.
- 3. If requested, type the master password for the server, and click **OK**.
The Select Signature File and Content dialog appears.
- 4. In **Signature File**, specify the full path to the signature message file you have captured for the test. You can use ... to locate and select the file.
- 5. For PKCS #7 files:
 - If the PKCS #7 file you have captured for the test has detached content (which is typical of IdenTrust), in **Content File** specify the full path to the content file. You can use ... to locate and select the file.
 - To display detailed information on certificates contained in the PKCS #7 message, select
Verbose output.
- 6. Click **OK**.
The tests performed by the Signature check can take up to a few minutes, depending on the checks that need to be made and the connections that need to be established. The results appear in the message pane.

What is tested?

This test first checks a PKCS #7 or BankID XML file and verifies that it can be decoded. It then extracts the certificate of the entity that digitally signed the message file and verifies that the digital signature for the message matches the information on the signer's certificate.

During this test, each certificate included in the message is verified and validated for its revocation status. Both PKCS #7 and BankID XML files typically contain the signer's certificate chain up to, but excluding, the root certificate. The chains displayed in the results window also include the root certificate picked up from the token. Typically, this results in a path length of three, so you see the following displayed:

- The root certificate of your hierarchy
- The message signer's issuing CA certificate
- The message signer's certificate

Each certificate (excluding the root) is also revocation checked according to the channel's configured revocation checking mechanism.

Verifying logs

If you have enabled tamper-evident logging for the logs, you can use this test to validate that the logs have not been modified.

Performing the test

To verify the logs:

1. In the Management Console tree view, select a log that has tamper-evident logging enabled, and do one of the following:
 - Select **Tools > Verify Log**

- Right-click the log and select **Tools > Verify Log**
- 2. If requested, type the channel password (*not* the master password), and click **OK**.
- 3. If requested, type the master password for the server, and click **OK**. The tests begin to run, with the results appearing in the message pane.

What is tested?

If rollback protection is enabled, the test first reads the last expected value from the token. It then reads entries in the database in their chronological order, verifying each entry's MAC, and verifying that no entries are missing. Finally, the largest sequential entry number is compared with that read from the token.

The verification process continues even if errors are detected, and information about every error found is displayed in the message pane.

Note: The maximum number of errors that the verification tolerates without stopping the check is 1000.

Resetting the log rollback state

If you have altered a log (with tamper-evident logging enabled) but are happy with the changes, you can reset the rollback state so that the log uses a new sequence of MAC values. You might want to do this if you have been testing and making log changes, but you now want to use the log for a production environment and do not want the testing changes showing in future log validation tests (for the production environment).

To reset the log rollback state for a log:

1. In the Management Console tree view, select the log and select **Tools > Reset log rollback state**.
2. If requested, type the channel password (*not* the master password), and click **OK**.
3. If requested, type the master password for the server, and click **OK**.

Chapter 19: Token Management Console

This chapter introduces the SSAS Token Management Console (TMC) and provides procedural information for the user tasks you can perform using this console.

About the console

The TMC enables you to view and manage the contents of the SSAS channel token. You can inspect and modify token contents, including managing the keys, certificates, and data objects on the token.

You can also use a stand-alone TMC to:

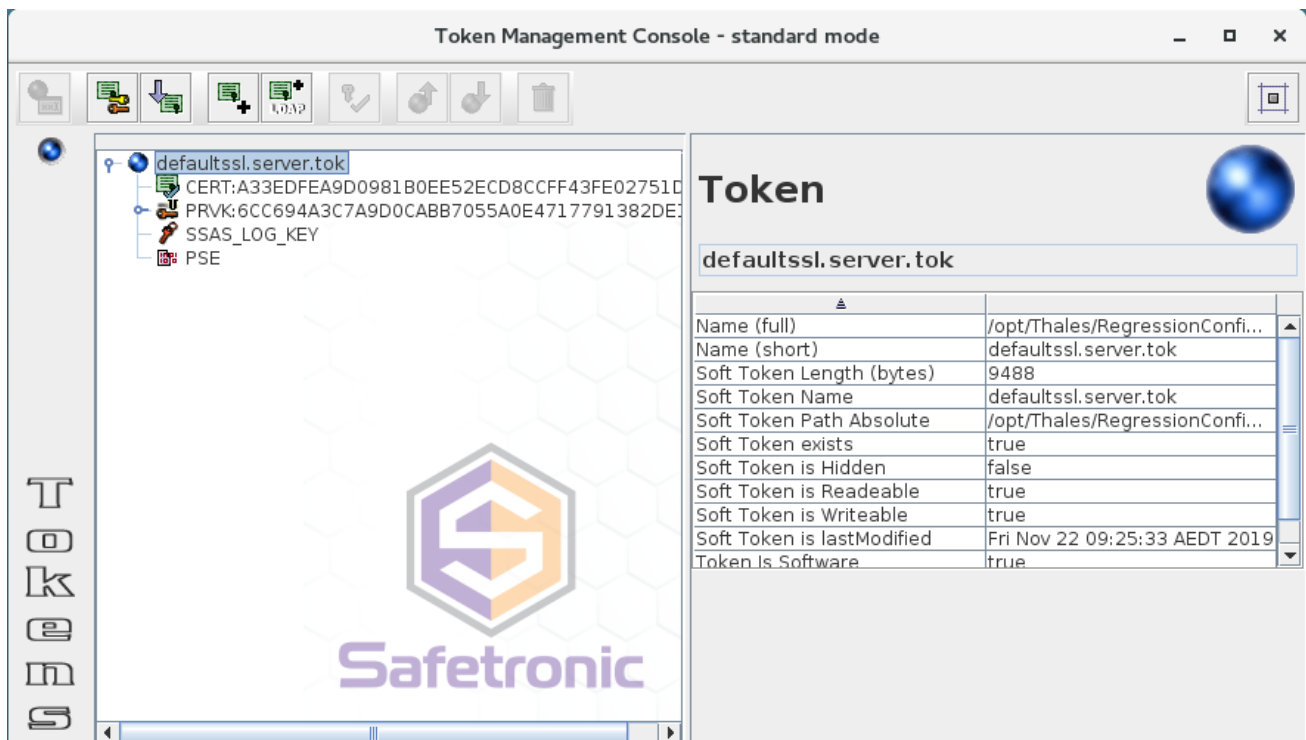
- Set up client cryptographic tokens for TLS communication with the server
- Enable client request authentication using signatures or digests

User interface

Standard mode

When started, the TMC appears in its standard mode, as Figure 15 shows. The standard mode provides a subset of the available functionality, which is sufficient for normal usage.

Figure 15. TMC standard mode






The standard mode presents a path view of the token. The left side contains a tree view of the token, displaying trusted certificates, private keys, and their certificate chains. The right side shows the properties for the object selected on the left side. Unlike the SSAS Management Console, you cannot edit the properties displayed on the right. You make modifications using the available TMC actions.

Object types shown

The standard mode uses the following icons when showing the token objects:


Icon	Object	Description
	Active certificate	A certificate within its validity period used in the currently active certificate chain of some private key.
	Active trusted certificate	A trusted self-signed certificate within its validity period used in the currently active certificate chain of some private key.
	Non-active certificate	A certificate within its validity period that participates in a non-active certificate chain (a chain superseded by a newer certificate chain) of some private key.
	Non-active trusted certificate	A trusted self-signed certificate within its validity period that participates in a non-active certificate chain (a chain superseded by a newer certificate chain) of some private key.
	Private key pending certification response	A private key whose associated certificates cannot be found on the token. In most cases, this is because a certification request has been issued for a newly

		generated key pair but no response has been processed yet.
	Identity private key	A private key whose associated certificate has non-repudiation as one of the allowed key usages.
	Utility private key	Utility private key A private key whose associated certificate does not have non-repudiation as one of the allowed key usages.

Advanced mode

The advanced mode provides:

- The standard mode functionality and extra advanced functionality
- The path view and an extra object view
- A more detailed view of the token

To access the advanced mode, or when asked by Salt Support, click  in the top right corner of the user interface to change mode.








To return to the standard mode, click .



Use the advanced mode with care. Changes made in this mode can severely damage your security configuration and render it unusable with your applications.


Object types shown

In addition to showing the same object types as the standard mode, the advanced mode uses the following extra icons when showing the token objects:

Icon	Object	Description
	Not yet valid or expired certificate	A certificate outside its validity period that forms part of a certificate chain for some private key. In effect, such a chain is unusable and can never become active.
	Not yet valid or expired trusted certificate	A trusted self-signed certificate outside its validity period that forms part of a certificate chain for some private key. In effect, such a chain is unusable and can never become active.
	Dangling valid certificate	A certificate within its validity period not a part of a certificate chain of any private key on the token. Consider removing such certificates from the token.
	Dangling not yet valid or expired certificate	A certificate outside its validity period not a part of a certificate chain of any private key on the token. Remove such certificates from the token.
	Public key	A public key.
	Secret key	A secret key.
	Data object	A data object stored on the token.

Path view

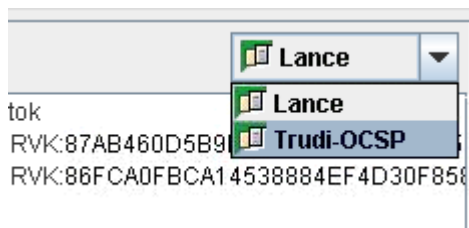
The standard mode shows only a path view of the token.

In advanced mode, the path view is indicated by  in the top right of the user interface. (You can use this icon to switch to the object view.)

The path view is a logical presentation of the token as perceived by the selected channel. It identifies the channel's *identity* and *utility* keys, and attempts to build their corresponding certificate chains. If the channel is using a Filtered Key Store, only certificates explicitly filtered by the Visible Certificates and Visible Trusted Certificates are used in this process.

Even if you started the TMC within the context of one channel, you can switch to any other channel that shares the same token. To do this, select the channel in the list box above the path view, as Figure 16 shows.

Figure 16. Selecting another channel



If multiple channels share the same token, as a Filtered Key Store, ensure those channels are configured appropriately. See *Token sharing*.


Interpreting a token using the path view

You can determine the status of a channel's token based on the objects shown in the path view:

Objects visible in the path view	Description
No private keys or trusted certificates	The token is empty and must be configured.
No keys, but at least one trusted certificate	A trusted root certificate has been added. You can now proceed with generating a key pair and certification request.
A trusted certificate and private key (with no associated certificates)	A key pair has been generated on the token, but no certification response has been processed.
A trusted certificate, private key, and associated certificates	The token is properly configured for usage within your PKI scheme.
A trusted certificate and private key (with multiple)	The token's private key has been re-certified and the newest certificate chain is active. The channel associated certificate chains, one of which is green) is properly configured for use with the Verify Signature or Generate Signature services.

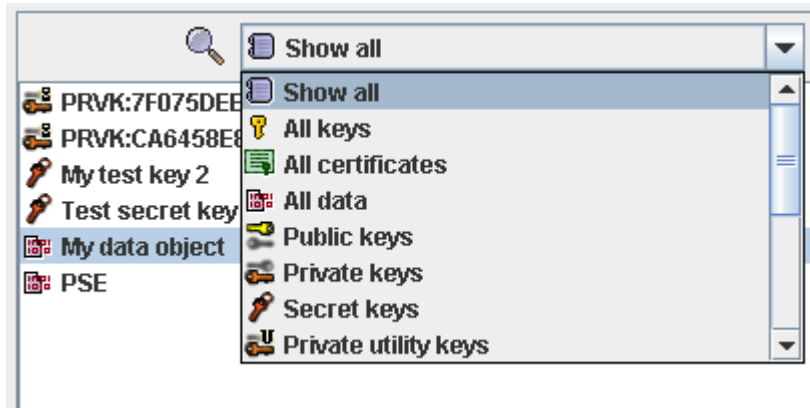
Object view

In addition to the path view, the advanced mode can show an object view of the token.

The object view is indicated by  in the top right of the user interface. (You can use this icon to switch to the path view.)

The object view presents an unsorted list of all the objects on the token. You can show a subset of the objects by selecting a pre-defined filter in the list box above the object view, as Figure 17 shows.

Figure 17. Filtering the object view












The **Certificates expiring in** option enables you to view the certificates due to expire within a specified number of days.

Icons and user tasks











Standard mode

The standard mode icons and the associated user tasks, which are described in *Standard tasks* are:

Icon	User tasks
	<i>Opening the TMC and logging into a token</i>
	<i>Generating a key pair and certification request for a token</i>
	<i>Processing a certification response for a token</i>
	<i>Importing a certificate from a file to a token</i>
	<i>Importing a certificate from an LDAP server to a token</i>
	<i>Verifying that a key is usable</i>
	<i>Backing up a token</i>
	<i>Restoring a token from a backup</i>
	<i>Deleting an object from a token</i>

Advanced mode


The *additional* advanced mode icons and the associated user tasks, which are described in *Advanced tasks*, are:

Icon	User tasks
	<i>Changing the password for a token</i>
	<i>Generating a secret key (symmetric key)</i>
	<i>Importing a secret key (symmetric key) to a token</i>
	<i>Importing a wrapped key to a token</i>
	<i>Exporting a key (wrapped) from a token</i>
	<i>Exporting a certificate from a token</i>
	<i>Importing a data object to a token</i>
	<i>Exporting a data object from a token</i>
	<i>Generating a new certification request for an existing private key (re-certifying)</i>
	<i>Importing a key and its certificates from a PKCS #12 file into a token</i>

Opening the TMC and logging into a token


This section describes how to start the TMC from the SSAS Management Console. You can also start a stand-alone TMC. See *Starting a stand-alone TMC*.

To open the TMC:

- In the Management Console tree view, select:
 - A channel or its **ATEnvironment**, **PKIEnvironment**, or token
 - A server's **Access PKI Environment** node or its token
- Do one of the following:
 - Select **Tools > Manage Token**
 - Click 
 - Right-click the node and select **Tools > Manage Token** The TMC appears.

Before you see any objects on the token, you must authenticate to the token by logging in. If you are viewing a software token for the first time, ensure that the **Create If Missing** property is set to **true** in the software token configuration.

To log into a token:

- In the TMC, select the token.
- Click .
- Type the token password and click **OK**.

Depending on your user authentication type, do one of the following:

- *No authentication:* Type the user name and click **OK**.
- *PIN authentication:* Type the user name followed by a space and the PIN, then click **OK**. For example, a user **james** with a PIN of **1234** types **james 1234**.
- *Smart card authentication:* Type the user name, click **OK**, and then type your PIN on the reader attached to the device when prompted by the reader.


The token remains in a logged in state until you close the TMC.

Standard tasks

Generating a key pair and certification request for a token

You can generate a key pair and PKCS #10-based certificate request for a token, and then send the request to your intended CA. Before doing this, consult your intended CA about how they process the key usages extension in a certification request.

To generate the key pair and certification request:

1. In the TMC, select the token in the path view, ensuring that you are within the correct channel. If you are using a Filtered Key Store, doing this ensures that visibility settings in the channel configuration are updated correctly.
Note: If you are generating a new key pair to replace an existing one, select the existing key or its certificate. By doing so, the **Requested identity** field (in the wizard that appears) is pre-populated with the subject distinguished name from the existing certificate.
2. Click .
A wizard appears, which contains several pages of fields that you use to generate the key pair and request. The wizard fields and parameters are described in the subsections that follow.
3. Provide the required values and use **Next** to proceed through the wizard.
4. At the end of the wizard, click **Finish** to generate the key pair and certification request. SSAS generates the key pair and stores the following files in the specified directory:
 - A certification request file with the name `request_<date and time of creation>`. This has either a .p10 extension (for raw PKCS #10 requests) or a .pem extension (for PEM encoded requests).
 - A SHA-1 thumbprint file, for a raw PKCS #10 request. This has a .txt extension.

You can now send the certification request to your intended CA. When the certificate is returned, follow the procedure in *Processing a certification response for a token*.

Step 1 page

Field	Description
Key type	The values for the key you want to generate.
Key length	
Requested identity (DN)	The valid distinguished name that you want the issued certificate to be associated with. For example, CN=Test, O=Salt Group, C=AU .

Step 2 page

This wizard page enables you to configure the key usages extension of your request. Consult your intended CA about how they process the key usages extension in a request, and use the following guidance on the settings for this wizard page:

CA type	Description	Settings to provide
Policy driven CA	Key usages in the issued certificates are driven by CA policy. Such CAs typically reject certification requests that have a key usages extension. This is mandatory for IdenTrust CAs and is also the most common practice in other environments.	Include key usages: not selected.
Policy driven CA requiring empty key usages in request	This is the same as above except that, due to different interpretation of the certification request format standards, the CA might require a key usages extension to be present in the request but rejects the request if it actually contains any key usages.	Include key usages: selected. All key usages: not selected.
CA honours key usages in request	The CA accepts the key usages extension.	Include key usages: selected. The required key usages: selected.

Step 3 page


Field	Description
Request format	Either a raw PKCS #10 format or a Privacy Enhanced Mail (PEM) encoded certification request
Output directory	Where you want the certification request stored

Processing a certification response for a token

SSAS can process certification responses in the following formats:


- Raw PKCS #7 binary format (also known as the *Certificate and CRL dissemination message type*)
- Base64
- PEM encoded

To process a certification response from a CA:

1. If the PKCS #7 response file does not have a .p7s, .p7c, or .p7b extension, change the file name to one of those extensions.
2. In the TMC, select the token associated with the certification response.
3. Click  .
The Choose Certification response file to process dialog appears.
4. Locate and select the file, and then click **Open**.
SSAS validates the response file, extracts the certificates, and adds them to the token.

Importing a certificate from a file to a token


To import a certificate from a DER or PEM file:

1. In the TMC, select the token.
2. Click .
3. Locate and select the certificate file, and click **Open**.
SSAS imports the certificate and adds it to the token. A success message appears, advising that the certificate might not be visible in the standard mode path view.
4. Close the message.

If SSAS can build a path from a private key up to a trusted root certificate, the certificate is displayed. If the certificate is not visible, change to the advanced mode, and then toggle between the path view and object view.

Importing a certificate from an LDAP server to a token

Some CAs prefer to publish certificates to a publicly accessible LDAP directory server. To obtain certificates from an online LDAP server:

1. In the TMC, select the token.
2. Click .
- The Import LDAP Certificate Wizard appears. The settings and search criteria that you provide for this wizard are stored for subsequent use in this wizard.
3. Click **Next**.
4. In **Server URL**, specify the URL of the LDAP server. For example, **ldap://directory.saltgroup.com.au**.
5. Specify the **Searchbase**. For example, **o=salt group, c=au**.
6. Click **Next**.
7. Provide the search criteria for locating the certificate:
 - **Issuer DN**: The distinguished name of the certificate issuer
 - **Serial Number (decimal)**: The unique serial number of the certificate
 - **Subject DN**: The distinguished name of the certificate subject
8. Click **Next** to start the certificate search.
SSAS contacts the LDAP directory and displays the published certificates matching your search criteria. The certificate list has the format: **<Subject DN> [<Issuer DN> --- <Serial Number>]**.
9. Select the required certificates from the list, and click **Finish**. You can use `Shift` or `Control` to select multiple certificates.
SSAS imports and processes the certificates.

Certificate processing


If a single matching certificate is returned, SSAS checks the signature. If the certificate is self-signed, SSAS stores it as the trusted root certificate. Otherwise, SSAS stores it as an untrusted end-entity certificate.

If the certificate is not self-signed (end-entity), SSAS attempts to build the proper certificate path for it using all private keys and matching certificates already present in the software token. If the full certificate chain is found for the private key and all chain signatures verified, the tree-like PKI structure appears.

Note: You must import the Root CA certificate first so any subsequent certificates can be successfully verified.

Verifying that a key is usable


To check that a key is usable:

1. In the TMC, select the key.
2. Click .
SSAS checks the key using random data to ensure the key is usable and displays whether it was successful.
3. Close the message.

Backing up a token


You can back up a token by storing it to a password-encrypted file. The file contains all the keys, certificates, and data objects stored on the token.

To back up a token:

1. In the TMC, select the token.
2. Click .
3. Specify the location and name of the file, and click **Save**.
4. Type the password for encrypting the token data and click **OK**. SSAS encrypts the data and stores it to the specified file.

Restoring a token from a backup

You can restore any token that you have previously backed up to a password-encrypted file. To restore a token:

1. In the TMC, click .
2. Locate and select the backup file, and click **Open**.
3. Type the password used to encrypt the token data and click **OK**.
SSAS imports the data, decrypts it, and restores the token in the required channel.
The restored keys, certificates, and data objects appear for the token.

Deleting an object from a token


To delete an object, select it in the TMC and click . Confirm the deletion.

You can only restore deleted objects if you previously backed up the token (containing the object) and later restore the token from the backup file.

Advanced tasks

Changing the password for a token

To change the password for a token:

1. In the TMC, select the token.
2. Click .
The Change token password dialog appears.
3. Type the existing password.
4. Type and confirm the new password for that channel.
5. Click **OK**.

Generating a secret key (symmetric key)

When generating a secret key on a token, you can choose whether the key is:

- Split over several components
- Extractable, allowing it to be wrapped under another key

These properties help keep the key secure when transferring it to another system.

You can display the generated key components on screen or save them to multiple files.

Note: It is important to protect the key components and to distribute them among several authorised individuals. If you choose to display the components, ensure the authorised individuals (component holders) are present and ready to record the components before starting the key generation.

To generate a secret key on a hardware token:

Note: The procedure is simplified for software tokens.


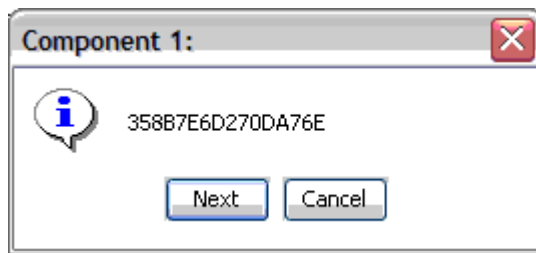
1. In the TMC, select the token.
2. Click .
The Generate Secret Key dialog appears.
3. In **Alias**, type a meaningful name for identifying the key.
4. Select the **Key type** and **Key length** for the key you want to generate.
5. To split the key over several components, select **Split into components** and specify the number of components.
6. To make the key extractable, select **Extractable**.
7. Click **OK** to generate the key components.
The Component dialog appears, asking you how you want to receive the key components.
8. To show each component (in turn) on separate dialogs:
 - a. The holder of the first component must click **Display** because the first component is displayed immediately, as Figure 18 shows. When they have recorded their component, they click **Next**.

Figure 18. Example key component



- b. For each subsequent component, a Next component preparation dialog appears. The next component holder clicks **Next** when ready. When they have recorded their component, they click **Next** again.

Note: The component holders cannot step back through the component display process to view previous dialogs. If they click **Cancel**, they are asked if they want to delete the key (stop the key generation).

9. To store each component to a separate file:
 - a. Click **Save in multiple files**.
 - b. For each component (in turn), specify the location for the file. A KCV is displayed.
10. Carefully record or store the KCV.

Importing a secret key (symmetric key) to a token


Importing a secret key as components

You can only import a secret key as a set of components to a hardware token. You can import the components:

- Manually, with the component holders typing their components.
- From the separate component files.

Ensure you have the component holders or files, and the KCV (generated with the key) ready before starting the key import.

To import a secret key into a hardware token:

1. In the TMC, click . The Import Secret Key dialog appears.
2. In **Alias**, type a meaningful name for identifying the key.
3. Select the **Key type** for the key.
4. Specify the number of components that were created when the key was generated.
5. To make the key extractable (allowing it to be wrapped under another key), select **Extractable**.
6. Click **OK**.

You are asked how you want to import the key components.

7. To import the components manually:
 - a. Click **Manual input**.
 - b. For each component (in turn), the component holder must type their component and click **OK**.

Note: The component holders must enter their components in the order they were

generated. The component holders cannot step back through the component display process to view previous dialogs. If they click **Cancel**, they are asked if they want to delete the key (stop the process).

A KCV is displayed.

- c. Compare the value against the KCV generated with the key. If they match, click **OK**. Otherwise, if you reject the key, you can choose to delete it from the token.
8. To import the components from separate files:
 - a. Click **From multiple files**.
 - b. For each component (in turn), locate and select the file.
Note: You must select the component files in the order they were generated.
 - c. Locate and select the KCV.

Importing a secret key as a shared secret

You can only import a secret key as a shared secret to a hardware token. To import a secret key into a hardware token:

1. In the TMC, click  .
The Import Secret Key dialog appears, as shown in Figure 19.

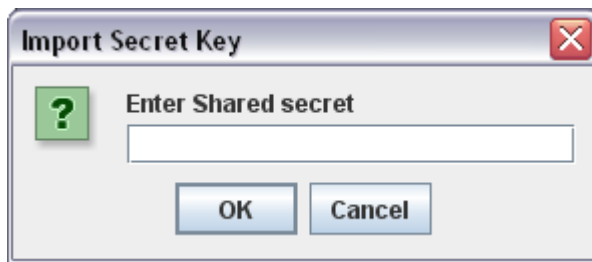
Figure 19. Import secret key dialog



2. In **Alias**, type a meaningful name for identifying the key.
3. Select **Shared secret**.
4. To make the key extractable (allowing it to be wrapped under another key), select **Extractable**.
5. Click **OK**.

- The shared secret dialog box appears, as shown in Figure 20.

Figure 20. Shared secret dialog



- Enter the **Shared secret**
- Click **OK** and the key will be imported.

Importing a wrapped key to a token

You can import a wrapped key from a file into a hardware token only. An unwrapping key matching the wrapping key (used to encrypt the key you are importing) must exist on the token.

Note: You can use the Safetronic Key Import Utility as an alternative method for importing keys. For information about this utility, see the *SSAS Reference Guide*.

An encrypted key is called a *wrapped* key. It is encrypted with a *wrapping* key and decrypted with an *unwrapping* key.

To import a wrapped key into a token:


- In the TMC, click . The Import key dialog appears, as shown in Figure 21.

Figure 21. Import key dialog



- For **Unwrapping key**, click ... and select a key from the list of suitable unwrapping keys on the token.
- For **File path**, click ... and locate and select the file containing the wrapped key.

4. Depending on the type of wrapping and unwrapping keys, provide the settings for the other dialog fields using the following guidance:

Wrapped key	Unwrapping key	Settings to provide
Any	LMK (SSCM only)	None. All the other fields are disabled.
RSA public or private key	Any, except LMK	Alias: A meaningful name used when storing the key on the token, which helps identify the key. Wrapped key type: Public or Private (as appropriate).
Secret key	Any, except LMK	Alias: A meaningful name used when storing the key on the token, which helps identify the key. Wrapped key type: Secret key . Key algorithm: DES , DES2 , DESede , AES or Generic (as appropriate).

5. Click **OK**.
SSAS decrypts (unwraps) the key and imports it into the token.

Exporting a key (wrapped) from a token


You can export a key to a file, encrypting (wrapping) it with another key present on the token. This functionality applies to secret, public, and private keys, and is only available for hardware tokens.

The token implementation largely determines the choice of keys that can be exported, and the keys that can be used for wrapping and unwrapping. Token implementations vary between manufacturers. It is unlikely that keys exported from one type of hardware token can be imported to another type of token. The key combinations that SSAS supports and the algorithms (formats) it uses are:

Wrapped key	Wrapping key	Unwrapping key	Format
Secret key with the isWrappable property set to true	RSA public key that supports wrapping	RSA private key that supports unwrapping	RSA with PKCS #1 padding
	DES, Triple DES and AES keys that support wrapping	DES, Triple DES and AES keys that support unwrapping	DES/ECB/NoPadding DESede/ECB/NoPadding
	ZMK (SSCM only)	ZMK (SSCM only)	ANSI X9.17
Private, public, or secret key with	LMK (SSCM only)	LMK (SSCM only)	Proprietary format wrapping and


the isWrappable property set to true			unwrapping both the key material and its associated attributes
--------------------------------------	--	--	--

To export a wrapped key from a token to a file:

1. In the TMC, select the key you want to export.
2. Click .
3. Specify the location and file name for the export key file.
A Select wrapping key dialog appears, showing the keys available on the token for the wrapping operation.
Subject to the rules in the table above, the dialog might show only a subset of the keys present on the token. Even if a key supports wrapping, it might be missing from the list if no suitable wrapping algorithm is implemented on the token. An unwrapping operation can fail for the same reason. For more information, refer to your hardware token manufacturer's documentation.
4. Select the wrapping key and click **OK**.
SSAS encrypts (wraps) the key and exports it to the specified key file.


Exporting a certificate from a token

To export a certificate from a token to a file:

1. In the TMC, select the certificate.
2. Click .
3. Specify the location and name of the file, and click **Save**.
SSAS stores the certificate to the specified file using binary DER encoding.


Importing a data object to a token

To import a data object from a file to a token:

1. In the TMC, select the token.
2. Click .
3. Locate and select the file containing the data object, and click **Open**.
4. In **Alias**, type a meaningful name for identifying the object, and click **OK**. SSAS imports the content of the data object and adds it to the token.

Exporting a data object from a token

To export a data object to a file:


1. In the TMC, select the object.
2. Click .

3. Specify the location and name of the file, and click **Save**. SSAS stores the content of the data object to the specified file.

Generating a new certification request for an existing private key (re-certifying)

If a certificate associated with one of the private keys reaches the end of its validity period, you can re- certify the key. You can generate a new certification request for the existing key, which you can send for certification to the same CA or a different CA. Before doing this, consult your intended CA about how they process the key usages extension in a certification request.

To generate the new certification request:

1. In the TMC, select the key in the path view, ensuring that you are within the correct channel. If you are using a Filtered Key Store, doing this ensures that visibility settings in the channel configuration are updated correctly.
2. Click .

A wizard appears, which contains several pages of fields that you use to the certification request. The wizard fields and parameters are described in the subsections that follow.
3. Provide the required values and use **Next** to proceed through the wizard.
4. At the end of the wizard, click **Finish** to generate the certification request. SSAS stores the following files in the specified directory:
 - A certification request file with the name `request_<date and time of creation>`. This has either a .p10 extension (for raw PKCS #10 requests) or a .pem extension (for PEM encoded requests).
 - A SHA-1 thumbprint file, for a raw PKCS #10 request. This has a .txt extension.

You can now send the certification request to your intended CA. When the certificate is returned, follow the procedure in *Processing a certification response for a token*.

Step 1 page

Field	Description
Requested identity (DN)	The valid distinguished name that you want the issued certificate to be associated with. Ensure that this is correct.

Step 2 page

This wizard page enables you to configure the key usages extension of your request. Consult your intended CA about how they process the key usages extension in a request, and use the following guidance on the settings for this wizard page:

CA type	Description	Settings to provide
Policy driven	Key usages in the issued certificates are	Include key usages: not


CA	driven by CA policy. Such CAs typically reject certification requests that have a key usages extension. This is mandatory for IdenTrust CAs and is also the most common practice in other environments.	selected.
Policy driven CA requiring empty key usages in request	This is the same as above except that, due to different interpretation of the certification request format standards, the CA might require a key usages extension to be present in the request but rejects the request if it actually contains any key usages.	Include key usages: selected. All key usages: not selected.
CA honours key usages in request	The CA accepts the key usages extension.	Include key usages: selected. The required key usages: selected.

Step 3 page

Field	Description
Request format	Either a raw PKCS #10 format or a PEM encoded certification request
Output directory	Where you want the certification request stored

Importing a key and its certificates from a PKCS #12 file into a token

To import a key and its certificates from a PKCS #12 file:

1. In the TMC, select the token into which you want to import the key and certificates.
2. Click .
3. When prompted, provide the password for the PKCS #12 file that was supplied to you in a secure, out-of-band way by the creator of the file.
The Select PKCS#12 file to import dialog appears.
4. Locate and select the file, and then click **Open**.
SSAS imports the key and certificates, and adds them to the token.

Setting up client cryptographic tokens

You can use a stand-alone TMC for configuring client cryptographic tokens for TLS communication with the server, and for enabling client request authentication using signatures or digests. You can also do this using the TMCLI. See *Token Management Command Line Interface*.

Starting a stand-alone TMC

You start the TMC by using `<SSAS install>\bin\tmc.bat` (Windows platforms) or `<SSAS install>/bin/tmc.sh` (Unix-based platforms).

Depending on the client's properties file, either a new (software) token is created or an existing (software or hardware) token is used.

When used in the context of configuring a client cryptographic token, you must pass the stand-alone TMC two parameters (in this order):

1. The client properties file.
2. The prefix `pkiEnv.keyStore` (case-sensitive).

For example, for a client properties file `c:\client_config.properties`, the correct command is:

```
<SSAS install>\bin\tmc.bat c:\client_config.properties pkiEnv.keyStore
```

Recommendations for configuring TLS between clients and servers

A good application design strives to minimise creation and destruction of `AtClientEnv` and `AtClient` instances. We recommend that you:

- Create only one `AtClientEnv` and then remain logged into it for the life of the client application
- Create only one `AtClient` object for the life of the application

Creating an `AtClientEnv` is expensive computationally because it involves logging into the underlying client cryptographic token. Also, when TLS is used between client and server, the very first `AtClientEnv` is the client cryptographic context. Logging it out later on invalidates all TLS sessions created between the client and server. This is due to the design of the underlying RMI stack and Java TLS.

Transient `AtClient` objects are possible, but we do not recommend them because of their impact on performance.

In essence, we recommend (and in case of TLS usage it is required) that the application treats `AtClientEnv` and `AtClient` objects as singletons. Synchronise the creation of `AtClientEnv` and `AtClient` objects to ensure only one instance, and to avoid racing conditions if the application is multi-threaded. Subsequent service calls need not be synchronised.

Setting up a client cryptographic token for digest authentication

If you are using the digest authentication scheme, the token is populated with a supplied shared key for the matching Server ID. To set up a client cryptographic token for digest authentication:

1. Make a note of the server node name in the server configuration (the server name as displayed in the tree view, also visible as the server Name property).
2. Open your client properties file and make a note of the value of the name property.
3. Create a text file containing the shared key.
4. Start the stand-alone TMC. See *Starting a stand-alone TMC*.
5. Log into the token.
6. Invoke the **Add Data Object** action.
7. In the File Open dialog, select the text file you created containing the shared key, and then click **OK**.
8. Enter the following value as the data object alias:

```
digestAuthenticationsharedSecret: client_name<===>server_name
```

1. For example, for a client name `Client_ABC` and server named `Server_01`, the alias is:

```
digestAuthenticationsharedSecret: Client_ABC<===>Server_01
```



If the shared secret (shared key) for the supplied Server ID already exists on the token, it is overwritten.
The alias is case-sensitive.

Setting up a client cryptographic token for signature authentication

To set up a client cryptographic token for signature authentication:

1. Start the stand-alone TMC. See *Starting a stand-alone TMC*.
2. Log into the token.
3. Invoke the **Generate Key Pair** and **Certification Request** action. For more information, see *Generating a key pair and certification request*.
4. Transfer the generated certification request to your CA in a safe, out-of-band manner.
5. When you receive the certification response, import it to the token by following the instructions in *Processing a certification response for a token*.

Chapter 20: Token Management Command Line Interface

This chapter introduces the SSAS Token Management Command Line Interface (TMCLI) and provides procedural information for the user tasks you can perform using this interface.

About the TMCLI

The TMCLI enables you to view and manage the contents of the SSAS channel token. You can inspect and modify token contents, including managing the keys, certificates, and data objects on the token.

Using the TMCLI

You run the TMCLI by using `<SSAS install>\bin\tmcli.bat` (Windows platforms) or `<SSAS install>/bin/tmcli.sh` (Unix-based platforms). The notation used to indicate command line syntax is described below:

Notation	Description
Text without brackets	Items you must type as shown
<Text inside angle brackets>	Placeholder for which you must supply a value
[Text inside square brackets]	Optional items, must be typed as shown
Vertical bar ()	Separator for mutually exclusive items; choose one

The command must be the first property provided to the command line. To perform a command, additional options may need to be passed to the TMCLI. For details of these additional options see *Commands*.

Note: If the syntax is entered incorrectly, the TMCLI will display an error message. All general messages from the TMCLI are sent to stdout, error messages are sent to stderr.

When using the command line interface for configuration, you must pass the following parameters to the TMCLI:

1. The properties file
2. The password of the cryptographic token to be configured

If you are using an existing server properties file, you must provide a prefix to identify which cryptographic module you wish to use.

The following table displays the generic options available when using the TMCLI:

Option	Description
-f, --property.file <properties>	Path and filename of the properties file
-h, --help	Outputs the syntax. If a command has been provided, this will display details of all available options for that command
-p, --prefix <prefix>	Prefix to use when parsing the properties
-x, --password <password>	Password of the cryptographic token to be configured

Properties

The token properties are stored in the properties file. The TMCLI requires the complete path and filename of the properties when configuring the token. For a properties file named *ssas.properties* located in */home/ssas/ssasconfig/* the syntax would be:

```
-f /home/ssas/ssasconfig/server.properties
```

Creating a new properties file

If there is not a properties file available, you can create a new properties file. The TMCLI script contains examples of the different types of properties required for each type of token. These properties are shown below.

Example for PKCS#11 token:

```
token.class=com.entegrity.jsdp.security.token.pkcs11.PKCS11TokenSpi
token.name=wspkcs11d.dll
token.slotOrdinal=0
```

Example for software token:

```
token.class=com.entegrity.jsdp.security.token.sdpsoft.SDPSoftTokenSpi
token.name=config/server/token.tok
```

Example for payShield token:

```
token.class=com.entegrity.jsdp.security.token.hsm.ThalesHSMTTokenSpi
token.enableHsmDebug=false
token.hsmCharacterSet=ASCII
token.hsmHeaderLength=4
token.hsmUnitList=<ipaddress>:<port
>:100
```

```
token.name=config/server/token.tok
token.timeout=15
token.downPeriod=30
```

Prefix

When using a SSAS server properties file, you can instruct the TMCLI to configure either the Access PKI crypto module or a channel crypto module. In either case, you must pass the TMCLI the prefix (-p).

The access PKI prefix will be:

```
<server name>.access.pkiEnv.keyStore
```

The channel crypto prefix will be:

```
<server name>.channel.<channel name>.pkiEnv.keyStore
```

For example to configure a channel token on channel *Symmetric* on a server called *SSAS Server* use:

```
<SSAS install>/bin/tmcli.sh <command> -f /home/ssas/ssasconfig/server.properties -x 1111 -p "SSAS
Server.channel.Symmetric.pkiEnv.keyStore"
```

Commands

This section explains how to use the TMCLI to view and manage the contents of the SSAS channel token. You can inspect and modify token contents, including managing the keys, certificates and data objects on the token.

Summary

The summary command will display a count of the keys, certificates and data objects on the token. The syntax to display the summary is:

```
<SSAS install> /tmcli.sh summary -f <properties> [-h] [-p <prefix>] -x <password>
```

For example, the command line will display:

```
Token: Software ssas1.tok Secret Keys 10
```

Private Keys 1
 Certificates 5
 Data Objects 1

List all keys and certificates

The list command will display the keys or certificates on the token. The user can provide one or more of the options [--SECRET|--PUBLIC|--PRIVATE|--CERTS] with the list command to specify the key types to list (if no type is specified then all types will be listed). The keys will be listed in a separate table from the certificates.

The syntax to display the list command is:

```
<SSAS install> /tmcli.sh list [--SECRET|--PUBLIC|--PRIVATE|--CERTS] -f <properties> [-h] [-p <prefix>] -x <password>
```

For example, the command line will display:

```
ALIAS | ALGORITHM | ID | LENGTH | TYPE
HMAC_KEY1 | DESede | 94F78F63871476B5101419C8BED539F52584FECB | 128 |
SECRET HMAC_KEY2 | DESede | C916F3096524F923FDF2987033229F54E5D87157 |
128 | SECRET
```

Option	Description
--SECRET	List secret keys
--PUBLIC	List public keys
--PRIVATE	List private keys
--CERTS	List certificates

Generating a secret key

The user can generate a secret key and optionally export it as components. The components must be output as files. If you provide the count option, then the output option must also be provided.

The syntax to generate a secret key in the command line is:

```
<SSAS install> /tmcli.sh generatesecretkey [-c <count> -o <output>] [--extractable] --keytype <type> [--keysize <size>] -z <alias> -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
-c, --components <count>	Allows the key to be split over several components. Specify the number
--extractable	Sets the key to extractable (default is not extractable). Use this option
--keytype <type>	Key type to generate e.g. generic, DES, DES2, DESede, RC2, RC4,

--keysize <size>	Key size in bits (dependent on keytype). Note: If no keysize is provided, the keysize will default to a suitable size
-o, --output <output>	Output location template as a file path. If the user supplies filename.ext then the output will be written as filename1.ext, filename2.ext, ...filename <i>n</i> .ext and filenamecheck.ext
--keytype <type>	Key type to generate e.g. DES DESede
-z <alias>	The key alias

For example, the command line will display:

```
Token: Software ssas1.tok
Defaulting to keysize: 128
Successfully generated secret key
```

Perform the list command to ensure the key was generated correctly.

Exporting a secret key

The user can export a secret key to file, encrypting (wrapping) it with another key present on the token. This functionality applies to secret, public and private keys, and is only available for hardware tokens.

The token implementation largely determines the choice of keys that can be exported, and the keys that can be used for wrapping and unwrapping. Token implementations vary between manufacturers. It is unlikely that keys exported from one type of hardware token can be imported to another type of token.

The key combinations that SSAS supports and the algorithms (formats) it uses are:

Wrapped key	Wrapping key	Unwrapping key	Format
Secret key with the isWrappable property set to true	RSA public key that supports wrapping	RSA private key that supports unwrapping	RSA with PKCS #1 padding
	DES, Triple DES and AES keys that support wrapping	DES, Triple DES and AES keys that support unwrapping	DES/ECB/NoPadding DESede/ECB/NoPadding AES/ECB/NoPadding AES/CBC/PKCS5Padding

The syntax to export a secret key in the command line is:

```
<SSAS install> /tmcli.sh exportsecretkey -o <filename> -w <alias> -z <alias> -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
-o, --output <filename>	Output filename
-w, --wrapping <alias>	Alias of wrapping key
-z, --alias <alias>	Alias of the key to be exported

Importing a secret key (symmetric key) to a token

The user can import a secret key as file components or provide a shared secret on the command line.

Importing a secret key as components

You can only import a secret key as a set of components to a hardware token. You can import the components:

- Manually, with the component holders typing their components.
- From the separate component files.

Ensure you have the component holders or files, and the Key Check Value (KCV) (generated with the key) ready before starting the key import.

The syntax to import a secret key in the command line is:

```
<SSAS install> /tmcli.sh importsecretkey -c <count> [--extractable] --keytype <type> -i <input> [--nocheck] -z <alias> -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
-c, --components <count>	The number of key components (between 2-9)
--extractable	Key is extractable (default is not extractable). Use this option to allow the key to be wrapped under another key
--keytype <type>	Key type to generate e.g. generic, DES, DESede
-i, --input <input>	Input location template as a file path. If the user supplies filename.ext then the files read will be filename1.ext, filename2.ext, ...filename <i>n</i> .ext and filenamecheck.ext
--nocheck	Do not validate the key check value (the key check file is not required)
-z <alias>	The key alias

Importing a secret key as a shared secret

You can only import a secret key as a shared secret to a hardware token. The syntax

to import a shared secret secret in the command line is:

```
<SSAS install> /tmcli.sh importsharedsecret [--extractable] -i <input> -z <alias> -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
--extractable	Key is extractable (default is not extractable). Use this option to allow the key to be wrapped under another key
-i, --input <input>	The shared secret
-z <alias>	The key alias

Generate a key pair and certification request

You can generate a key pair and PKCS #10-based certificate request for a token, and then send the request to your intended CA. Before doing this, consult your intended CA about how they process the key usages extension in a certification request.

The syntax to generate a key pair and certification request in the command line is:

```
<SSAS install> /bin/tmcli.sh generatekeypaircertrequest --RSA | --DSA --keylen <length> --dn <dn> [keyusages] -pem | --binary -o output -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
--RSA	Key type is RSA
--DSA	Key type is DSA (software only)
--keylen <length>	Key size in bits (dependant on key type)
--dn <dn>	The DN for the certificate request e.g. CN=SOME NAME, O=YOUR ORGANIZATION, C=GB
--pem	PKCS#10 output is PEM
--binary	PKCS#10 output is binary (der encoded)
-o, output <output>	Output directory. Saves PKCS#10 and a text thumb-print file to the directory

You can now send the certification request to your intended CA. When the certificate is returned, follow the procedure in *Process a certification response*.

Process a certification response

SSAS can process certification responses in the following formats:

- Raw PKCS #7 binary format (also known as the *Certificate and CRL dissemination message type*)
- Base64
- PEM encoded

The syntax to process a certification response in the command line is:

```
<SSAS install> /tmcli.sh processcertificationresponse -i <response> -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
-i, --input <response>	The file containing the certificate

Add a certificate

The syntax to add a certificate to the token in the command line is:

```
<SSAS install> /tmcli.sh addcertificate -i <certificate> -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
-i, --input <certificate>	The file containing the certificate

Key test

The syntax to perform a key test in the command line is:

```
<SSAS install> /tmcli.sh keytest -z <alias> -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
-z <alias>	The key alias

If the test is successful, the following message will be displayed:

```
Token: Software ssas1.tok
Signature over arbitrary data: OK.
Decryption: OK.
```

Delete a key or certificate

You must tell the TMCLI if you want to delete a key or certificate. The syntax to delete a key or certificate in the command line is:

```
<SSAS install> /tmcli.sh delete --KEY | --CERT -z <alias> -f <properties> [-h] [-p <prefix>] -x <password>
```

Option	Description
-z, --alias <alias>	The key or certificate alias
--KEY	Delete a key
--CERT	Delete a certificate

For example, the command line will display:

```
Token: Software ssas1.tok
Successfully deleted key
```

Chapter 21: Identity Management Console

This chapter introduces the SSAS Identity Management Console (IMC) and provides procedural information for the user tasks you can perform using this console.

About the console

The IMC enables you to view and manage user tokens, such as OATH and Vasco tokens, either using a user interface or remotely using SOAP or REST. You can:

- View tokens for your channels, organised by type
- Import, update, and delete tokens for a channel

Properties files

The IMC stores and manages several properties:

- The list of token types and channels that have been added to the console
 - The location of the server properties file
 - The location of the individual token properties files
- The IMC uses the following properties files:

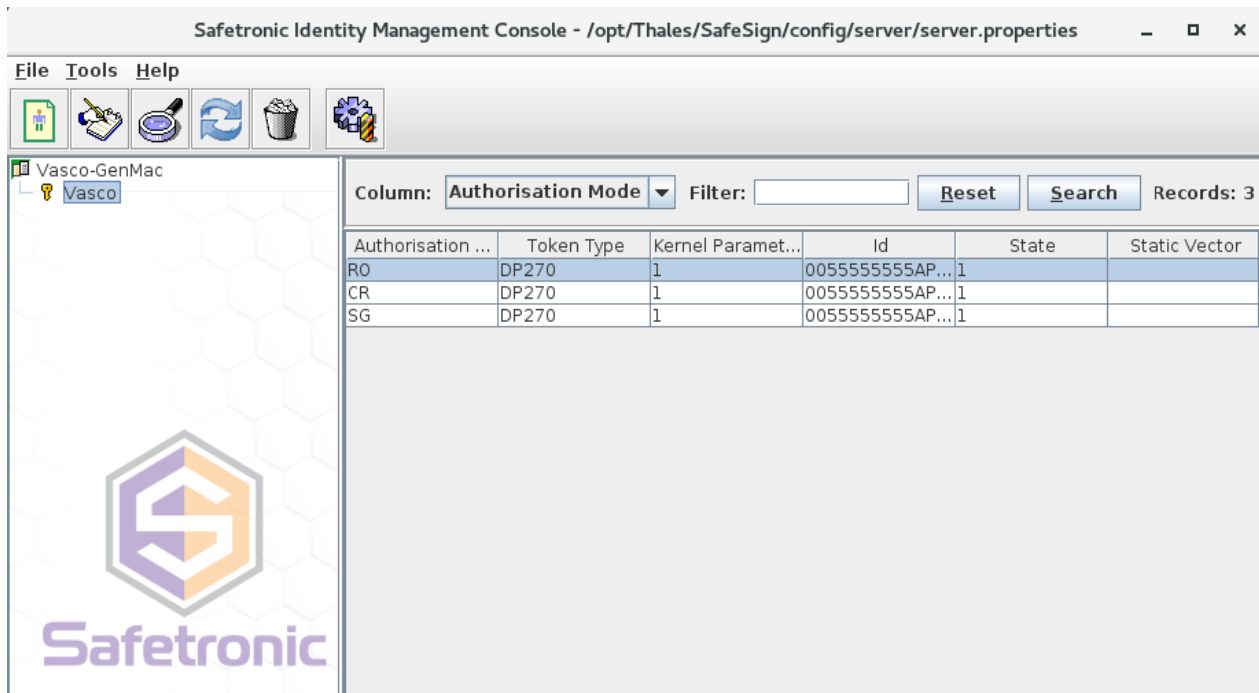
File	Location defined in .SSASPrefs	Description
IMC properties	IMCChooserFile	This file contains the token type to channel mappings. You can view and edit its contents using the IMC Preferences dialog. For more information, see <i>Configuring the IMC token type to channel mappings</i> . The file can be located anywhere. You determine the location of the file when saving the mappings in the IMC Preferences dialog.
Token properties	<i>Not applicable</i>	The property file for each token contains all the information required by the IMC to handle the management of tokens. The properties are listed, together with an example, in <i>Data/token property files</i> in the <i>SSAS Reference Guide</i> .
Server properties	ChooserFile	See <i>Server properties</i> in the <i>SSAS Reference Guide</i> . This file can be located anywhere.

The .SSASPrefs file is stored in the SSAS computer's user home directory.

User interface

The IMC has a tree view and list pane, as Figure 22 shows. Selecting a token type in the tree view displays applicable tokens in the list pane.

Figure 22. IMC main view



The channels and token types shown by the IMC depend on the mappings in the IMC properties file, which you configure using the IMC Preferences dialog. For more information, see *Configuring the IMC token type to channel mappings*.

If a service on a channel maps to only one configured token type, its mappings are added automatically. However, if there is more than one token type with a given service code, an automatic mapping is not possible. The service type to map can be gained from the property (in the IMC properties file):

```
[Server Name].channel.[Channel Name].poolManager.pool.[Pool Number].name= [Service Type]
```

The *Service Type* code is specified in the token properties file as the **ServiceType** property. For generic MAC services (which can include EMV and Vasco tokens), you must create the mapping using the IMC Preferences dialog.

Menus and user tasks


The following table shows the main menu items and the associated user tasks, which are described in this chapter. (A subset of the menu items are shown.)

Menu	User tasks
File > Preferences	<i>Configuring the IMC token type to channel mappings</i>
Tools > Import	<i>Importing tokens for a channel</i>
Tools > Update	<i>Updating a token</i>
Tools > View	<i>Viewing details for a token</i>
Tools > Refresh	<i>Refreshing the list pane</i>
Tools > Delete	<i>Deleting a token</i>

Opening the IMC

This section describes how to start the IMC from the SSAS Management Console. You can also start a stand-alone IMC. See *Starting a stand-alone IMC*.

To open the IMC:

1. In the Management Console tree view, select the server.
2. Either:
 - Click 
 - Right-click the server and select **Tools > Identity Manager**

The IMC uses a preferences file, which contains the token type to channel mappings.
3. If the file is empty, or you are opening the IMC for the first time, you are asked if you want to load an existing mappings file. To open an existing file, click **Yes** to locate and open the file.
If you click **No**, the Preferences dialog appears, enabling you to add mappings. For information on adding mappings, see *Configuring the IMC token type to channel mappings*.
4. When prompted, type the master password for the server associated with the mappings, and click **OK**.
The IMC appears.

Starting a stand-alone IMC

To start the IMC when the SSAS Management Console is not running:


1. In the `<SSAS install>\bin` directory, edit `imc.bat` (Windows platforms) or `imc.sh` (Unix-based platforms) to include an entry pointing to your database driver file.
2. Run `imc.bat` or `imc.sh`.
Because the IMC has not been started from within the Management Console, the IMC creates its own objects based on the properties contained within the server properties file.
3. Locate and specify the server properties file, and click **OK**.
4. If there is more than one server identified in the server properties file, choose the required server. The IMC uses a preferences file, which contains the token type to channel mappings.

5. If the file is empty, or you are opening the IMC for the first time, you are asked if you want to load an existing mappings file. To open an existing file, click **Yes** to locate and open the file.
If you click **No**, the Preferences dialog appears, enabling you to add mappings. For information on adding mappings, see *Configuring the IMC token type to channel mappings*.
6. When prompted, type the master password for the server associated with the mappings, and click **OK**.
The IMC appears.

Configuring the IMC token type to channel mappings

The IMC Preferences dialog enables you to view and edit the token type to channel mappings for each server. The mappings determine the channels and token types that appear in the IMC.

To edit the token type to channel mappings for a server:

1. Select **File > Preferences** or click .
The Preferences dialog appears, showing the existing token type to channel mappings for the server. On first use, the dialog is blank.
2. To add a mapping:
 - a. Click **Add**.
 - b. Choose the appropriate options to map a **Token Type** to a **Channel**, and then click **OK**.
The **Token Type** drop down list contains all the token types that can be mapped to a service on an existing channel. The **Channel** drop down list contains all the channels having a service that maps to a token type.
The new mapping appears in the Preferences dialog.
3. Add further mappings as required.
4. When you have finished, click **Save** to store the mappings.
5. If this is the first time you have edited the mappings, a dialog appears so you can specify a location for the preferences file. Specify the location and click **OK**.

Viewing, filtering, and sorting the tokens

To view all the tokens of a particular type for a channel, select the token type under the required channel in the IMC tree view. The tokens appear in the list pane.

Note: The IMC shows the channels and token types that relate to the token type to channel mappings in the IMC Preferences dialog. If the required channels and token types are not shown in the IMC, you might need to add the appropriate mapping.
See *Configuring the IMC token type to channel mappings*.

To sort the tokens by ascending or descending order, click the appropriate column heading.

To filter the tokens (to show a subset), select the required value for **Column** (at the top of the list pane) and specify the filter criteria. For example:


- Specifying **Column** as **State** and **Filter Value** as **Enabled** shows only those tokens that have the value **Enabled** in the **State** column
- Specifying **Column** as **PAN** and **Filter** as **5401** shows only tokens that have **PAN** numbers containing **5401**

Refreshing the list pane

To refresh the list pane, to account for any token changes, select **Tools > Refresh** or click .

Importing tokens for a channel

To import a token or batch of tokens for a channel:

1. In the IMC tree view, select the token type under the required channel.
2. Select **Tools > Import** or click .
The Import dialog appears. The fields and parameters depend on the token type. For descriptions of the fields, see:
 - *ActivIdentity token fields*
 - *Vasco token fields*
 - *Salt Mobile unallocated token fields*
 - *Salt Mobile allocated token fields*
 - *OATH token fields*
3. Provide the required values for the token or batch of tokens you are importing, and click **OK**.
4. If the import is successful, a success message appears. Click **OK**. The imported tokens are available for use within the channel.

Viewing details for a token


To view the complete details for a token, select the token in the list pane, and then click



or select **Tools > View**.


Updating a token

To update the details for a token:

1. Select the token in the list pane, and then select **Tools > Update** or click .
A dialog appears showing the current values for the token, enabling you to change some of them.
2. Make the required changes and click **OK**.
3. If the token update is successful, a confirmation message appears. Click **OK**.

Deleting a token

To delete a token from the database:

1. Select the token in the list pane, and then select **Tools > Delete** or click .
2. Use the information in the dialog that appears to check that this is the correct token to delete.
3. If required, provide a password.
4. Click **OK**.

Chapter 22: Log Viewing Console

This chapter introduces the Log Viewing Console (LVC) and provides procedural information for the user tasks you can perform using this console.

About the console

A server can produce event and error logs, which record all requests handled by the server and any processing errors.

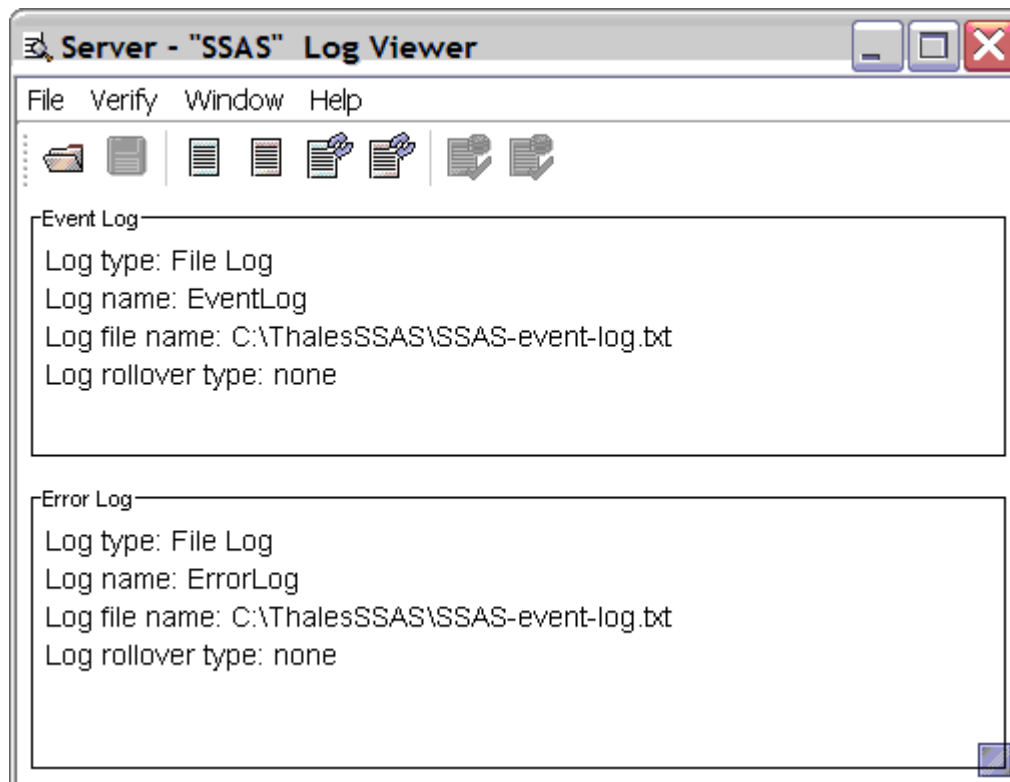
The LVC enables you to inspect the contents of the event and error logs. You can:

- Perform static and real-time (updating) searches of the logs
- View and save detailed information for log items
- Save and re-open your log searches
- Verify the integrity of the logs (checking for signs of tampering)

User interface

The LVC is a workspace. When it first opens, the LVC displays the names of the logs currently in use, as Figure 27 shows.

Figure 27. LVC initial view



All the log searches you perform appear as new tabs in the LVC workspace. You can save the workspace and open it another time to run the same searches.

You can rename a tab by selecting it and then selecting **File > Rename Tab**.

Menus and user tasks

The following table shows the main menu items and the associated user tasks, which are described in this chapter. (A subset of the menu items are shown.)

Menu	User tasks
File > Add Tab	<i>Performing a log search</i>
	<i>Performing a real-time (updating) log search</i>
File > Open Workspace	<i>Opening a saved LVC workspace</i>
File > Save Workspace	<i>Saving the LVC workspace</i>
Verify	<i>Verifying the integrity of a log</i>

Opening the LVC


To open the LVC, select a server in the SSAS Management Console tree view, and then do one of the following:

- Select **Tools > Log Viewer**
- Click 

- Right-click the server and select **Tools > Log Viewer**


Opening a saved LVC workspace

To open a previously saved LVC workspace:

1. Select **File > Open Workspace** or click .
2. Locate and select the workspace file, and click **Open**. The search tabs appear in the workspace.

Saving the LVC workspace



To save the log search tabs shown in LVC workspace:

1. Select **File > Save Workspace** or click .
2. Specify a file name and location for the workspace file, and click **Save**.

Performing a log search

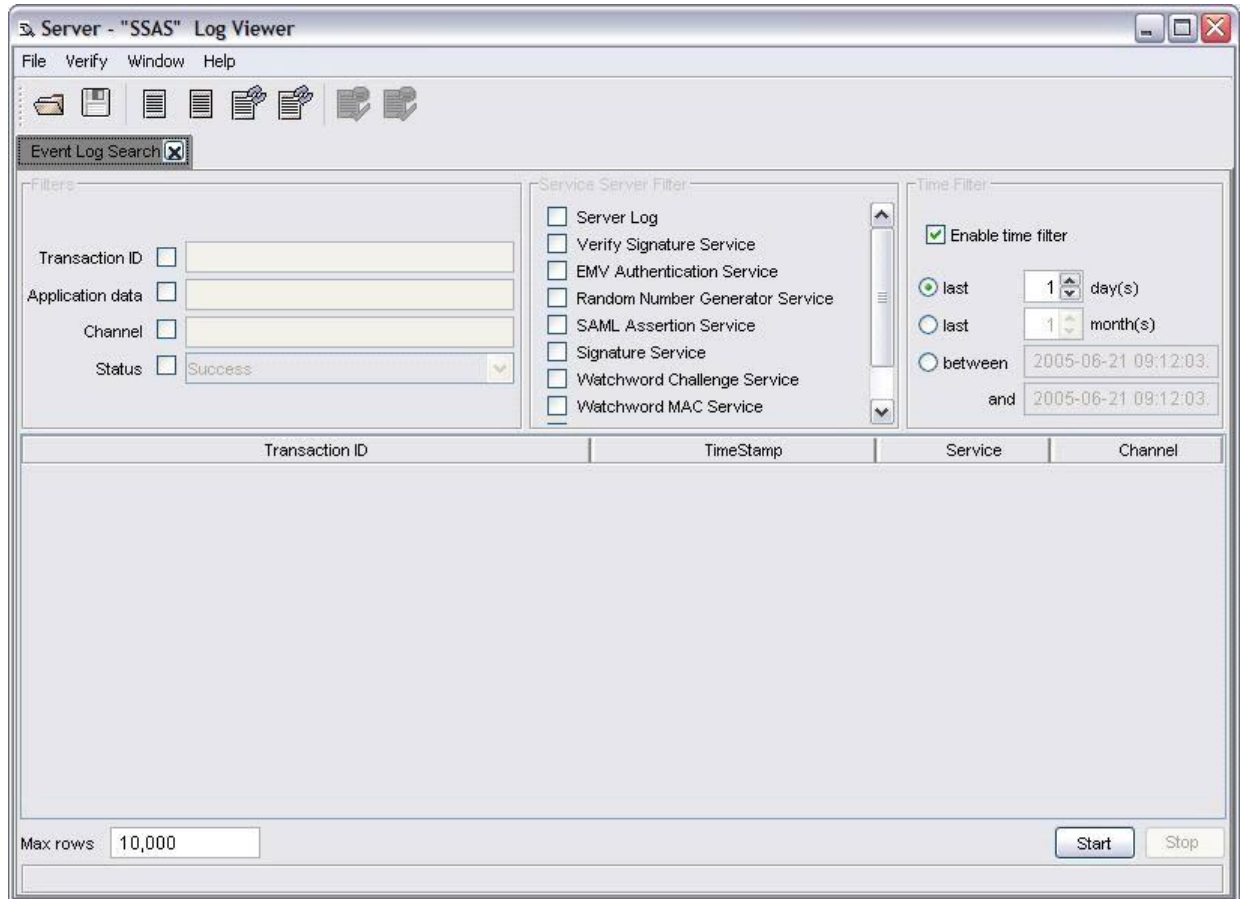
Each log search appears as a new tab in the LVC

workspace. To perform a log search:

1. Depending on the log you want to search, either:
 - *Event log*: Select **File > Add Tab > Event Log Search** or click .
 - *Error log*: Select **File > Add Tab > Error Log Search** or click .

A new tab appears, showing search filters and a results list (which is initially blank). The available filters depend on the type of log. Figure 28 shows an Event Log Search tab.

Figure 28. Event Log Search tab



2. Select and specify the required search filters:
 - **Filters** determines the type of results shown
 - **Service Server Filter** limits the results to the selected services
 - **Time Filter** limits the results to a specified time period
 - **Max rows** limits the number of results shown
3. To start the search, click **Start**.
The search results appear in the results list.

To sort the log items by ascending or descending order, click the appropriate column heading.

To view details for a log item, double-click the item in the results list. A dialog appears, showing the stored properties for the log item. Figure 29 shows an Event Log details dialog.

Figure 29. Event Log details dialog



You can:



- View full details for a property, by selecting the property value and clicking ...
- View the related log items in the other log, by clicking **Jump to Error Log** or **Jump to Event Log**
- Verify the log item, by clicking **Verify**
- Save the log item details to a text file, by clicking **Export**

To close all open detail dialogs, select **Window > Close All Detailed Views**.

Performing a real-time (updating) log search

A real-time log search is similar to a static log search, except that the search is repeated, and the results updated, at specified intervals. Each real-time log search appears as a new tab in the LVC workspace.



To perform a real-time log search:

1. Depending on the log you want to search, either:
 - *Event log*. Select **File > Add Tab > Real-time Event Log Search** or click 
 - *Error log*. Select **File > Add Tab > Real-time Error Log Search** or click 
2. Specify the search filters. For information on doing this, see *Performing a log search*.
Note: The **Time Filter** options are different, enabling you to limit the search to a specific period (up to the current time) or from a fixed date and time.
3. In the bottom right of the tab, specify the search refresh interval.
4. To start the search, click **Start**.

Verifying the integrity of a log

By default, the logs are not protected. SSAS supports tamper-evident logging, which enables you to detect whether the logs have been tampered with. For more information, see *Tamper-evident logging* in the *SSAS Concepts Guide*.

If you have enabled tamper-evident logging, you can check the integrity of the logs. To do this for a log, either:

- *Event log*. Select **Verify > Verify Event DB Log** or click .
- *Error log*. Select **Verify > Verify Error DB Log** or click . If there are any issues with the log, they are displayed.

Chapter 23: Configuring the client-server communication

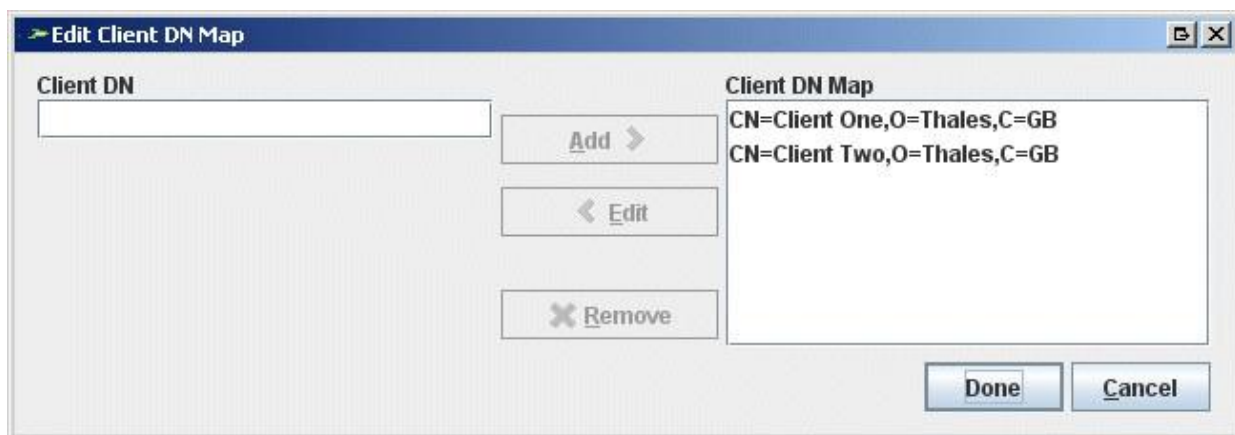
This chapter contains information about configuring both the server and the client to set up the client-server communication.

For information on potential issues and workarounds, see *Issues with client-server communication*.

Configuring the client mappings

The Edit Client DN Map dialog, Figure 30, enables you to configure the DNs of the clients allowed to connect to the server. You use this dialog when configuring the server for TLS communication. See *Configuring TLS (SSL)*.

Figure 30. Edit Client DN Map dialog



To access the Edit Client DN Map dialog:

1. Select **Access PKI Environment** in the Management Console tree view.
2. In the property editor, select the **Endpoint Identification Map** value and click ...

To add a new client, specify the DN of the client (which can be found in the client's end-entity certificate) in **Client DN**, and then click **Add**. The client is added to the list of configured clients.

Note: SSAS expects the DN attributes in the order specified in RFC 2253, for example CN, O, C as shown in Figure 30, rather than C, O, CN. Some previous versions of SSAS accepted the attributes in any order, so you might need to adjust your existing client DN entries, in the **Client DN Map** list.

To edit a client, select it in the **Client DN Map** list, and click **Edit**. The client is moved from the list to **Client DN** for editing. When you have made the required change, click **Add** to return the client to the list of configured clients.

To delete a client, to prevent it from accessing the server, select the client in the **Client DN Map** list, and then click **Remove**.

Configuring TLS (SSL)

SSAS enables you to secure TCP/IP communications between the client and server using the TLS (SSL) protocol through the Java Secure Socket Extension (JSSE) provided by the installed JRE.

To use TLS, either:

- Configure server-side TLS
- Configure client-server TLS

Both involve configuring the server and client accordingly.

Many of the actions require use of the TMC. For information on starting the TMC, see either:

- *Opening the TMC and logging into a token*
- *Starting a stand-alone TMC*

SSAS performs TLS using RSA only. The RSA keys are kept in the client and server cryptographic tokens. All TLS session keys are generated in software.

SSAS supports the following TLS cipher suites:

- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA

Configuring server-side TLS

To configure the server for server-side TLS:

1. Ensure the **Access PKI Environment** contains a public certificate and private key pair that can be validated up to a Trusted Root Certificate.
To configure the server's private key, do one of the following:

- *Generating a key pair and certification request for a token, and then Processing a certification response for a token*
- *Importing a key and its certificates from a PKCS #12 file into a token*

To add any extra certificates to the token, do one of the following:

- *Importing a certificate from a file to a token*
 - *Importing a certificate from an LDAP server to a token*
2. Configure the TLS (SSL) settings for the RMI or SOAP interface. These interfaces are configured independently.

To enable RMI over TLS, set the following properties (for the **server** node):

Property	Value
Use SSL	true
Server RMI Port	The port number that the server uses to listen for TLS connections

To enable SOAP over SSL, set the following property (for the **server** node):

Property	Value
SOAP Use SSL	true

The following properties (for the **Access PKI Environment** node) are common for RMI and SOAP. These all have default values specified:

Property	Value
SSL Client Protocol Maximum	The maximum acceptable TLS or SSL protocol when establishing a connection with the client. See the table below for acceptable values. The default value is VERSION_TLS12 .
SSL Client Protocol Minimum	The minimum acceptable TLS or SSL protocol when establishing a connection with the client. See the table below for acceptable values. The default value is VERSION_TLS10 .
Timeout	The maximum amount of time that a read operation is allowed to block, before the socket times out.

The table below shows the acceptable protocol values for the **SSL Client Protocol Maximum** and **SSL Client Protocol Minimum** properties, and which versions are supported.

Protocol	SSAS value	Supported?
SSL 3	VERSION_SSL30	Y
TLS 1.0	VERSION_TLS10	Y
TLS 1.1	VERSION_TLS11	Y
TLS 1.2	VERSION_TLS12	Y

You must now configure the client. See *Configuring the client*.

Configuring client-server TLS

To configure the server for client-server TLS:

1. Add any additional Root CA and sub CA certificates to the **Access PKI Environment** token to enable a valid path to be built from the client's end-entity certificate (received as part of the TLS handshake) up to a trusted Root Certificate. You can add these by either:
 - *Importing a certificate from a file to a token*
 - *Importing a certificate from an LDAP server to a token*
2. To enable Client Authentication, set the following properties for **Access PKI Environment**:

Property	Value
Use Client Authentication	true
Endpoint Identification Map	The DN of all clients allowed to connect to this server. See <i>Configuring the client mappings</i> .

You must now configure the client. See *Configuring the client*.

Configuring the client

Note: Use the information in this section in conjunction with the information in *Setting up client cryptographic tokens*.

SSAS provides the following examples and scripts:

- Example RMI clients, in `<SSAS install>\src\com\thalesecurity\atdemo\client`
- Batch or shell scripts, in `<SSAS install>\bin`
- Example client properties file, in `<SSAS install>\config\client\simple-client.properties`

Using the example properties file, you can configure the scripts for TLS. Change the following properties:

Property	Value
pkiEnv.keyStore.token.name	The location of the token used to store the certificates and private keys that are used as part of the TLS handshake.
pkiEnv.ssl.client.protocolVersion.maximum	The maximum acceptable TLS or SSL protocol allowed while establishing a connection with the server.
pkiEnv.ssl.client.protocolVersion.minimum	The minimum acceptable TLS or SSL protocol allowed while establishing a connection with the server.
pkiEnv.ssl.client.debugPrints	If TLS/SSL debugging is required, set to true . The default is false .

Server-side TLS

To configure the client for server-side TLS:

1. Use the TMC to add any extra certificates to the token by either:

- *Importing a certificate from a file to a token*
 - *Importing a certificate from an LDAP server to a token*
2. Set the following properties in the example `simple-client.properties` file:

Property	Value
<code>access.useSSL</code>	true
<code>pkiEnv.ssl.client.debugPrints</code>	The DN of all clients allowed to connect to this server. See <i>Configuring the client mappings</i> .

Client-server TLS

To configure the client for client-server TLS:

1. Use the TMC to configure the client's private key by either:
 - *Generating a key pair and certification request for a token, and then Processing a certification response for a token*
 - *Importing a key and its certificates from a PKCS #12 file into a token*
2. Add any extra certificates by either:
 - *Importing a certificate from a file to a token*
 - *Importing a certificate from an LDAP server to a token*
3. Set the following properties in the example `simple-client.properties` file:

Property	Value
<code>access.useSSL</code>	client
<code>access.pkiEnv.serverDnMap.dn1</code>	The DN of the server's end-entity certificate

For information on configuring SOAP for TLS, refer to your SOAP provider.

Chapter 24: Configuring for optimum security

This chapter considers the security of the SSAS system:

- Server
- Logs and token data
- Crypto module
- Clients

While physical security (such as a locked door) and network security (such as a firewall) are advised, we have not provided details on how to implement these specifically because the requirements vary between installations.

We recommend that you generate security procedures to cover the handling of key material, such as generation, loading, backup and storage of keys, selection of personnel, and PIN selection.

General

Server

SSAS protects the server with a master password, which encrypts the channel passwords and other sensitive data in the server properties file. The master password is therefore required to start and stop the server. Protect this password carefully. For information on changing it, see *Changing all passwords for a server*.

Although SSAS requires the use of the server master password to start and stop the server, anyone with physical access to the server can stop the server by shutting down the server processes directly.

Therefore, restrict physical access to the server to trusted personnel only. Similarly, save the server properties file to a secure location, such as local to the SSAS installation.

Logs and token data

SSAS can use a database both for logging and for storing token details, depending on the services used. You can configure SSAS to provide some protection to enable detection of unauthorised changes to the database. See *Services* and *Logs*. However, to prevent such damage, restrict access to this database.

SSAS records events (requests for a service) and errors. Check these logs periodically to ensure that no unexpected errors are occurring.

Crypto module

SSAS can use a crypto module (hardware token) for the channel token, which protects the keys. Protect the crypto module using suitable methods.

Settings

For each property specified, values are provided that you can use in a production environment (other settings might be appropriate while testing configurations). Those settings that are *Recommended* are those which have a serious effect on the security of SSAS. Those settings that are *Encouraged* might provide enhanced security, which can be useful for some applications.

Server

Property	Value	Advice	Information
Client Request Authentication	digestAuthentication or signedAuthentication	Encouraged	Ensures non-repudiation of client requests.
Debug	off (default)	Recommended	Outputs extra information. Use this only in a test environment.
Soap Use SSL	true	Recommended	If Enable Soap is set, ensures privacy of communication when using the SOAP interface. Use this setting unless both client and server are on the same trusted network.
Soap Use WSS	true	Recommended	If Enable Soap is set, ensures security of communication when using the SOAP interface. Use this setting unless both client and server are on the same trusted network.
Use SSL	true	Recommended	Ensures privacy of communication when using the RMI interface. Use this setting unless both client and server are on the same trusted network. <i>See Issues with client-server communication.</i>

Environment settings

The following property can be set for the server (the **Access PKI Environment**), for each asymmetric channel (the **PKI Environment**), and for each symmetric channel (the **Crypto Environment**):

Property	Value	Advice	Information
Token	Hardware	Recommended	Using a hardware token is generally

The following properties can be set both for the server (the **Access PKI Environment**) and for each asymmetric channel (the **PKI Environment**):

Property	Value	Advice	Information
Revocation Checker	Setting other than None as appropriate	Recommended	Setting this to none allows revoked certificates to be accepted as valid, so you must not use this in a production environment.
Relax BC Requirements	false (default)	Encouraged	This setting is provided for backwards compatibility. Do not set to true unless you are using old CA certificates that do not conform to RFC 2459.
Use Client Authentication	true	Encouraged	If Use SSL is set, this setting restricts TLS connections to those whose TLS certificate has a configured DN (distinguished name).
SSL Client Protocol Maximum	VERSION_TLS12	Recommended	If Use SSL is set to true , this setting restricts the maximum protocol that connecting clients can use when establishing secure connections to the server.
SSL Client Protocol Minimum	VERSION_TLS12	Encouraged	If Use SSL is set to true , this setting restricts the minimum protocol that connecting clients can use when establishing secure connections to the server. (By default, this is set to VERSION_TLS10 for backwards compatibility.)

Channel

Property	Value	Advice	Information
Remote Management Password	Any value	Recommended	The SOAP interface (if configured) allows some remote management of the channel. Set this password to protect unauthorised clients from using these functions (such as adding tokens or changing token state).

Services

OATH

Always configure channels to use a crypto module, because using a software token requires clear keys in the database. Further protection can be provided by setting the **Audit** property to **true**, which provides tamper evidence protection of the token records in the database.

WebPIN

Note: Even when using a crypto module, decryption is currently performed in software only.

Password

Property	Value	Advice	Information
Encryption	true	Encouraged	Encrypts the hashed password in the database. If set, an Encryption Key must be provided.
Audit	true	Encouraged	Provides tamper evidence protection of the password records in the database.

Verify Signature

Property	Value	Advice	Information
Revocation Check Client Bypass	False (default)	Recommended	This flag allows the client to pass in the bypassRevocationCheck setting. If this flag is set to true , you must always use a secure link between the client and the server. Any uses by the client are recorded in the log.

Salt Mobile

Property	Value	Advice	Information
Allow Gateway Override	False	Encouraged	If set to true , allows the IMC administrator or calling client to override the server gateway settings for a channel when provisioning a Salt Mobile token. This means that an SMS or push message sent as part of the token provisioning process can be sent using a gateway other than that specified server side.

ActivIdentity

Property	Value	Advice	Information
Audit	true	Encouraged	<p>If set to true, the token records are protected by a TEMAC on token import, and the TEMAC is checked each time the token record is read (such as during token authentication).</p> <p>Tamper evidence is only available if using a crypto module. There is a potential performance penalty when token record tamper evidence checking is enabled, in return for increased token record security.</p>

Logs

Property	Value	Advice	Information
Type	Database	Encouraged	Using a database log for the event log enables use of the other options to protect the logs. It is generally advisable to use a file log for the error log to enable warning logging of any problems with the event log.

When the **Type** setting is either **Database** or **Weblogic**, the following options are available:

Property	Value	Advice	Information
Authenticate Log	True	Encouraged	All log entries are amended with a sequence number and a Message Authentication Code (MAC). This does not prevent modification of logs, but does provide a mechanism of detecting modified log entries, or entries out of sequence.
Enable Rollback	True	Encouraged	<p>The last used sequence number is stored on the crypto module, which enables detection of the removal of the last entries.</p> <p>Authenticate Log must also be set to true.</p>

Chapter 25: Configuring SSAS for SQL server

Using Microsoft SQL Server

To ensure that SSAS can network with Microsoft SQL Server correctly:

1. On the SSAS computer, open the SQL Server Configuration Manager (in Microsoft Windows).
2. In the tree view, locate and select **Protocols for MSSQLSERVER**, and then double-click **Named Pipes**.
3. In the Named Pipes Properties dialog, set **Enabled** to **Yes**, and click **OK**.
4. In the tree view, double-click **TCP/IP**.
5. In the TCP/IP Properties dialog, select the **IP Addresses** tab.
6. For **IP1**, set **Enabled** to **Yes**, and click **OK**.

IP1 is set to port 1433 and the IP address of the local machine.

7. Close the SQL Server Configuration Manager.
8. Stop and then restart the SQL server to activate these changes.

Using SQL server mirroring

Database mirroring increases database availability by transferring data from the main server (the *principal*) to a standby server (a *mirror*) so that you can fail over to the mirror if the principal fails.

Types of mirroring

With a valid configuration, SSAS can support these types of mirror:

Mirroring type	Description
High performance (asynchronous)	This transfers data asynchronously to the mirror server. The principal server does not wait for an acknowledgement from the mirror that all transactions have been recorded. In the event of a forced failover is required, data can be lost.
High protection (synchronous)	This transfers data synchronously to the mirror server. The principal server waits for an acknowledgement from the mirror that all transactions have been recorded. A manual failover is required in the event of failure. Until a manual failover is triggered, the database is unavailable.
High availability (synchronous with automatic failover)	This transfers data synchronously to the mirror server. The principal server waits for an acknowledgement from the mirror that all transactions have been recorded. An automatic failover is performed in the event of failure where a witness server is present. Otherwise, a manual failover is required.

Configuring SQL server mirroring

SSAS supports SQL server mirroring by default. To use it, you only need to configure the correct **Database URL** for the database connection string (in the New database pool dialog). See *Configuring a database connection pool*. SSAS checks the database URL for the existence of the following values:

- jdbc:sqlserver
- failoverPartner

If both values are present, SSAS assumes that a mirrored SQL server database is being used. The following example shows a connection string for a mirrored SQL server database:

```
jdbc:sqlserver://<principalServerName/ip>:<serverPort>;databaseName=<databaseName>;failoverPartner=<mirrorServerName/ip>;
```

Note: databaseName is also a required parameter of the connection string. This is a requirement of the JDBC driver.

High performance (asynchronous) mirroring issues

Because asynchronous mirroring does not guarantee that all data is copied between the principal and mirror databases, data might not be present in the mirror. This can lead to some of the following issues:

- Logging or tamper-evident logging (database and hardware PKCS #11 only).
When rollback protection has been enabled and SSAS has detected the presence of missing logs (in other words, the next log ID retrieved from the database is less than that stored on the crypto module), the channels are disabled, and calls to SSAS are rejected. This is in keeping with the SSAS design, which is intended to protect rollback records from being compromised.
- Tokens data and token profiles/configuration data out of synch (Synchronous Services).
Because data (token/token profiles/token configuration) is not guaranteed to be in synch with the principal database, there is a chance that previous changes made between the token and the principal are not duplicated on the mirror. Examples are:
 - A token has previously been re-synchronised but the mirror database was not updated. The token needs to be re-synchronised again before it authenticates with the token.
 - A token's configuration has changed (for example, the authentication window was made bigger) and the mirror database has not been updated. The token needs to be authenticated under the new window but fails to authenticate with the value stored in the mirror database.

Chapter 26: Configuring the RADIUS interface

This chapter provides information for setting up the SSAS RADIUS interface. For general information about this interface, see the *SSAS Concepts Guide*.

In general, a RADIUS server startup includes the following steps:

1. Instantiate and initialise the RADIUS server.
2. Fetch RADIUS client details from the backend system.
3. If a proxy, fetch remote RADIUS server details from the backend system.

All necessary runtime information must be available from the backend system (prior to starting the RADIUS server), which is the SSAS database and, in some cases, external LDAP directories. You must configure the following entities in the SSAS database:

- RADIUS clients (always)
- RADIUS users (only when managed internally by SSAS)
- LDAP servers (only when some or all RADIUS users are located there)
- RADIUS remote servers (only when SSAS is acting as proxy)

The installation directory contains several SQL scripts that can help you create the tables. See *Database schema*. You can use the IMC afterwards to populate the tables with data.

Note: The PROTECTION_KEY is a DES or AES secret key that must be generated first using the HSM with the TMC. This key is used to protect the SHARED_SECRET value from being clear in the table.

RADIUS deployment scenarios

Third-party RADIUS server already exists

In this scenario, the SSAS RADIUS server can be placed between clients and existing servers and configured to proxy some of the requests. This caters for existing users with static passwords or third- party tokens (such as RSA SecurID).

The shared secret between the SSAS RADIUS server and clients can be the same one as used by the third-party RADIUS server. However, we recommend that you set up a new shared secret between SSAS and the remote server.

Perform the following actions:

1. Modify the client properties file to point to the SSAS host/port. You can re-use the shared secret.

2. Using the IMC, configure the SSAS RADIUS server with:
 - A new client (existing/new shared secret)
 - New users (local or external)
 - A new remote server (new shared secret)
 - A proxy forwarding criteria (to which requests are to be forwarded)
3. Reconfigure the third-party server:
 - Add new client details to point to the SSAS RADIUS server (acting as client)
 - Remove old client details if all traffic is expected to go through SSAS

Clean setup: end-server

In this scenario, the SSAS RADIUS server is either:

- Completely replacing the existing endpoint RADIUS server
- Installed as a new endpoint server in a new network setup There is no need for proxying.

Perform the following actions:

1. Configure the client to point to the SSAS host/port.
Invent a new shared secret that is at least 32 random characters long.
2. Using the IMC, configure the SSAS RADIUS server with:
 - A new client (shared secret, as set above)
 - New users (local or external)

Clean setup: front-server

In this scenario, the SSAS RADIUS server is either:

- Completely replacing the existing front-end RADIUS server
- Installed as a new front-end server in a new network setup

There is a need for *roaming*, which forwards some of the requests to the remote servers. Perform the following actions:

1. Configure the client to point to the SSAS host/port. You can re-use the shared secret.
2. Using the IMC, configure the SSAS RADIUS server with:
 - A new client (shared secret, as set above)
 - New users (local or external)
 - A new remote server (shared secret can be re-used, if required)
 - A proxy forwarding criteria (to which requests are to be forwarded)
3. Configure the third-party server, by adding new client details to point to the SSAS RADIUS server.

Performance caching

When the client, remote server, LDAP, and user details are loaded from the backend system to the RADIUS server memory space, they are cached. A configurable **Radius Data Refresh Interval** parameter (in the Management Console) facilitates the caching and caters for configuration changes, such as the removal or disabling of clients/users in the backend system.

Also, any replies sent by the RADIUS server to the RADIUS client equipment are cached internally for a few seconds to ensure that the RADIUS client receives the message. This happens because network problems, offline servers, and large traffic loads can sometimes prevent the client from picking up the packet. Therefore, the client receives a quick reply on its prompting for a second copy of the packet.

RADIUS proxy server

With a SSAS RADIUS proxy server, a RADIUS server:

1. Receives an authentication request from a RADIUS client (such as a NAS).
2. Forwards the request to a remote RADIUS server.
3. Receives the reply from the remote server.
4. Sends the reply to the client.

This functionality enables the use of an existing commercial RADIUS server while migrating to stronger authentication schemes, such as PKI and EMV. During the migration period, the SSAS RADIUS server can forward the requests to the existing *remote* RADIUS server.

The diversion choice depends on the parameter values from the request. Proxy forwarding criteria is based on *one* of these attributes:

Data	RADIUS standard attributes
Calling telephone number (from where the user dial-in call was placed)	Calling-Station-Id
Called telephone number (where the dial-in call was received)	Called-Station-Id
IP address of the RADIUS client	NAS-IP-Address
Unique ID of the RADIUS client (specified in the request by the client gear)	NAS-Identifier
Realm part of the supplied user name (userABC@saltgroup.com.au or Salt\userABC)	Realm (this is SSAS-specific)

You must use these attribute string values when configuring the forwarding criteria types in the IMC.

You can strip off the realm part of the user name *and* separator character (@, /, \, or %) before forwarding the raw user name. The realm string is either a prefix (realm/user_name) or postfix (user_name@realm).

IMC settings

Before enabling RADIUS server in the SSAS Management Console, you must use the IMC to set up various RADIUS system entities (such as clients, users, and proxies) in the SSAS database.

The following entities are *required*:

- RADIUS clients (defining who is allowed to talk to server)
- RADIUS users (defining who is the holder of the two-factor authentication token):
 - If users are located in the SSAS database, add the local users.
 - If users are located on an external LDAP server, add the LDAP details (LDAP server and LDAP schema mapping).
 - If existing users are located on an LDAP server, but new users are in the SSAS database, add *both* the LDAP details (LDAP server and LDAP schema mapping) and the local users.

A search for user data starts in the database and continues in the external directories.

The following entities are *optional*:

- RADIUS remote servers (used only when SSAS is proxying some of the client requests)
 - Proxy forwarding criteria (used to determine which requests are proxied)
- For information about the proxy forwarding criteria and required attribute strings values, see *RADIUS proxy server*.

For information about the RADIUS entities and attributes visible in the IMC, see *IMC RADIUS entities*.

IMC plug-in settings

To configure the IMC to manage RADIUS entities, you might need to manually edit several IMC plug-in property files using a text editor. The plug-in files are located in the directory `<SSAS install>\config\imc\plugins`.

Note: By default, the plug-in files contain values for **data.dbPoolIdentifier** and **serviceType** that correspond to the Generic Mac service. Therefore, if a Generic MAC service is configured within your server that uses the same database connection pool as that configured for RADIUS, you do not need to modify the plug-in files.

Identifying which plug-in files need to be modified

In the Management Console, select RADIUS in the tree view. The values of the **Radius Proxy Requests** and **Radius Users BackendType** properties (in the property editor) determine the plug-in files that you need to modify:

IMC plug-in property file	Modified when...
RadiusClientType.properties	Always
RadiusUserType.properties	Radius Users BackendType is internal or internalAndExternal
LDAPType.properties	Radius Users BackendType is internal or internalAndExternal

LDAPSchemaType.properties	Radius Users BackendType is internal or internalAndExternal
RadiusProxyType.properties	Radius Proxy Requests is true
RadiusProxyCriteriaType.properties	Radius Proxy Requests is true

When you have determined the files to modify, you must alter the value of two keys within those files: **data.dbPoolIdentifier** and **serviceType**.

Identifying the key values to use

The SSAS RADIUS server uses the database connection pool specified in the **Radius Database Connection** property (in the Management Console).

To manage RADIUS entities using the IMC, you must configure the server with at least one service that uses the same database connection pool as the RADIUS server to store token data. The service can be any one of:

- Generic MAC
- OATH
- Salt Mobile
- ActivIdentity

If one of these services is present in any channel, and it uses the same database connection pool as that specified for RADIUS, you must add the corresponding values shown in the following table for **data.dbPoolIdentifier** and **serviceType** to the plug-in property files.

Service type	data.dbPoolIdentifier value	serviceType value
Generic Mac	gen_mac.db.pool.name	3500
OATH	oath.pool.name	1800
Salt Mobile	saltmobile.pool.name	2100
ActivIdentity	actividentity.pool.name	2200

Example server configuration

In an example SSAS configuration:

- The RADIUS server uses the database connection pool **DBPool**, which you set in the **Radius Database Connection** property
- You configure an ActivIdentity service to use the *same* database connection pool, by setting it in the service's **Database Connection Pool** property

Based on the table above (which shows the values to set in each of the IMC plug-in files), for the ActivIdentity service you locate the keys and change their values as follows:

```
data.dbPoolIdentifier=actividentity.pool.name
serviceType=2200
```

In this example, when you have saved all changes to the plug-in files, you can manage all RADIUS entities, not only those (if any) associated with the ActivIdentity service and channel.

IMC RADIUS entities

The following tables list the RADIUS entities and attributes visible in the IMC.

RadiusClient

Field	Description
Shared Secret	Secret shared between the client and server.
Host	IP address or DNS name.
Protection Key	Triple DES or AES secret key alias. Used to encrypt the shared secret. This key must be generated first using HSM with the TMC.
State	The state of the token: enabled , disabled , suspended , or revoked .
Client Name	A descriptive name for the client.

RadiusUser

Field	Description
Username	Name used when providing RADIUS credentials.
Token Type	The type of token: EMV , VASCO_DIGIPASS , OATH , SALTMOBILE , or ACTIVITY .
Token Serial No	The unique serial number of the token.
Auth Scheme	One of: OTP, CR. If left empty, OTP is assumed.
Token Info	Additional information regarding the token and/or the token secret key: <ul style="list-style-type: none"> For EMV card, this can be PANSEQ value 01 - 99 For Salt Mobile, this is the mobile type, such as 010 for mCode OTP, and 014 for mCode CR. For ActivIdentity, this is the token model, such as AI1114. This must be set for CR and single-prompt MAC authentication schemes.
Token Challenge Length	Length of a challenge, in digits, a user token is expecting. (For Vasco it is 4, but does not need to be entered here.)
Token Password Length	Used to indicate the length of the token's password. Used for OATH AID value.
Token Password Encoding	Used to indicate if the token's password is encoded in hex or decimal. Values: HEX or DEC.
Token Password Checksum	Used to indicate that the last digit of the password is replaced with a checksum over remaining digits. Values: YES or NO.
Token Counter Lookahead	Used to indicate the look-ahead value to be used during verification process, for cases when the token counter is out of synch with the SSAS counter.
Token Issuer	Salt Mobile only: ID of Issuer profile to use from the Salt Mobile configuration table.
Token Issuer Password	Salt Mobile only: clear text password for the above profile.
Proxy Config Token	Some value meaningful only to a client that is actually a proxy RADIUS server. Described in the RADIUS RFC.
State	The state of the token: enabled , disabled , suspended , or revoked .

LDAP

Field	Description
LDAP Name	LDAP server user-friendly name.
LDAP Type	Can be GENERIC or MICROSOFT AD .
Host	IP address or DNS name.
Port	The LDAP server port.
Search Base DN	Base DN for search operation.
Authentication Mode	anonymous , simple , or tls
Login Name	User account with read-only privileges. Used for simple and tls authentication modes. Example: user@company.com
Password	User's password.
Timeout	Connection timeout in seconds. If missing, 5 seconds is the default value.

LDAPSchema

A mapping between SSAS-specific user parameters (such as token type and serial number) and LDAP schema parameters. Populate **LDAPSchema** if the LDAP parameter names are different from the SSAS parameter names.

Example using the existing attributes (in Microsoft Active Directory):

Field	Example
User name	CN (Common Name)
Token type	Description
Token serial no	Telephone number

Example with new *extended schema* attributes:

Field	Example
User name	cn=DirectoryManager
Token type	sSASTokenType Token serial no
User name	cn=DirectoryManager

RadiusProxy

Field	Description
Shared Secret	Secret shared between SSAS and remote server.
Host	IP address or DNS name.
Protection Key	Triple DES or AES secret key alias. Used to encrypt the shared secret. Can be the same key used for RadiusClient setup.
State	The state of the remote server.
Proxy Name	Descriptive name for the remote server.
Authentication Port	Remote server's listening port: 1645 or 1812.

RadiusProxyCriteria

Field	Description
Strip Username Realm	Indicates if the <i>named</i> realm is stripped from the user name before forwarding, or not. Values: 1 (for yes, do strip) and 0 .
Forwarding Criteria Value	Actual value that has to match the one in the received request message, for example: <ul style="list-style-type: none"> • 01234567890 (phone number, <i>numbered</i> realm) • 10.0.0.123 (client IP address) • UNIQUEID1234 (client Id) • company.com (named realm)
Proxy Name	Remote server's user-friendly name (as set in RadiusProxy).
Forwarding Criteria Type	One of the following strings: <ul style="list-style-type: none"> • Calling-Station-Id • Called-Station-Id • NAS-IP-Address • NAS-Identifier • Realm

The server can handle multiple criteria types (for a certain remote server) and performs the AND logical operation on them. This means that the request message must satisfy *all* forwarding requirements before it is proxied to the remote server. However, IMC supports *only* single entries, which limits the multiple proxy criteria types (for a certain remote server) to only one type. This can be bypassed by manually adding new rows to the database table (using a separate database user interface).

Management Console property settings

To access the RADIUS properties for a server, select RADIUS in the Management Console tree view.

In the property editor, use  to display all properties.

For information about the RADIUS properties, see *RADIUS properties* in the *SSAS Reference Guide*.

Appendix A: SSAS Command Line Utilities

Utilities

SSAS includes the following command-line utilities in the <SSAS install>\bin directory, which have either a .bat (Windows) or .sh (Unix) extension. Some of the utilities contain comments, viewable when editing them, that describe the available arguments and how you can use those utilities.

SSAS Consoles

Utility	Description
imc	Starts a stand-alone version of the Identity Management Console (IMC). Edit the file to include an entry pointing to your database driver file. See <i>Configuring the server and daemon interfaces</i>
ssas-run-mgmtconsole	Starts the SSAS Management Console. See <i>Starting the Management Console</i> You can edit the file to: <ul style="list-style-type: none"> • Pass system properties to the management console • Change the language resource bundle used for the UI and messages. See the SSAS reference Guide
tmc	Starts a stand-alone version of the TMC. Edit the file to include entries defining the channel token. See <i>Token Management Console</i>
tmcli	The SSAS channel token can be configured using the command line interface. See <i>Token Management Command Line Interface</i>

Running the server and daemon

Utility	Description
ssas-run-daemon	Starts the SSAS daemon. You can supply a set of arguments. For configuration options, see <i>Configuring the server and daemon interfaces</i> The run SSAS daemon start menu shortcut calls this utility Stops the SSAS daemon.
ssas-stop-daemon	The run SSAS daemon start menu shortcut calls this utility Stops the SSAS daemon. You can supply a set of arguments. For configuration options, see <i>Configuring the server and daemon interfaces</i> The stop SSAS daemon start menu shortcut calls this utility
ssas-start-server	Starts the server. Supply these parameters when running this utility: <ul style="list-style-type: none"> • ATSERVERCONFFILE • SERVERPREFIX

	<i>Parameters</i>
ssas-stop-server	<p>Stops the server. Supply these parameters when running this utility:</p> <ul style="list-style-type: none"> • ATSERVERCONFFILE • SERVERPREFIX • MASTERPASSWORD <p><i>Parameters</i></p>
ssas-server-status	<p>Queries the server as to whether it is running. Supply these parameters when running this utility:</p> <ul style="list-style-type: none"> • RETURNTYPE • ATSERVERCONFFILE • SERVERPREFIX <p><i>Parameters</i></p>

Simple client applications

Utility	Description
ssas-build-examples	<p>Builds the SSAS sample client applications:</p> <ul style="list-style-type: none"> • SimpleClient • SimpleEMVClient • SimpleGUIClient • Thales ISPI application (IspiServlet and HttpServer). <p>For information on running this application, see SOAP application examples in the SSAS Developer Guide for SOAP</p>
ssas-run-simple-client	Starts the SimpleClient application. Edit the file to point to your configuration file
ssas-run-emv-client	<p>Starts the SimpleEMVClient sample client application, which performs EMV signature verification</p> <p>Edit the file to point to your configuration file, and supply the required arguments when running the utility. See the file's comments</p>
ssas-run-gui-client	Starts the SimpleGUIClient, which is the GUI-enabled SSAS client application
ssas-run-httpserver	Starts the Thales ISPI application, which is a web server that uses the SimpleClient configuration

Additional utilities

Utility	Description
aitokenexport	<p>Starts the ActivIdentity Export Utility</p> <p>Only use this if instructed to do so by Salt</p>
encrypt-password	Encrypts a channel password with a master password and stores it in the server properties file. Enables you to do this without starting the Management Console
ssas-setenv	Sets the environment variables. Edit the file to change particular settings, as instructed by the SSAS user documentation or Salt

ssas-versions	Queries the server for the SSAS Management Console version number
---------------	---

Parameters

You must supply particular parameters when running some of the utilities. This table describes all the parameters:

Parameter	Description
RETURNTYPE	The type of response required from the server: <ul style="list-style-type: none">• boolean• binary• string
ATSERVERCONFFILE	The file path and name of the server configuration file, for example: /home/ssas/ssasconfig/ssas.properties
SERVERPREFIX	The prefix for the server, for example: SSAS Server
MASTERPASSWORD	The master password for the server

Appendix B: Troubleshooting SSAS

This chapter provides problem solving information for common situations.

Issues starting the daemon

The daemon does not start

The most likely causes of a SSAS daemon failing to start are:

- The RMIRegistry port is occupied by another application
- Another daemon is already running


When SSAS is installed, the daemon batch and shell scripts are configured to use an RMIRegistry as a publishing mechanism. They also instruct the daemon to start an internal registry on port 1234. If port 1234 is occupied by another application, the daemon fails to start.

To resolve the conflict, do one of the following:

- Modify the application by changing its port or stopping it
- Reconfigure the daemon to use a different port for the internal RMIRegistry
- Reconfigure the daemon to use an external RMIRegistry
- Reconfigure the daemon to use JNDI as a publishing mechanism

For more information, see *Chapter 5: Configuring the server and daemon interfaces*.

As an example, follow these steps to reconfigure the daemon to use an internal RMIRegistry on a different port:

1. Using a text editor, open one of the following files, depending on your platform:
 - *Windows platforms:* <SSAS install>\bin\ssas-run-daemon.bat
 - *Unix-based platforms:* <SSAS install>/bin/ssas-run-daemon.sh
2. Locate the line specifying the port value and change it to an available port with a value above 1024.
3. Close and save the file.
4. Open the SSAS Management Console and select the applicable server in the tree view.
5. In the property editor, change the **RMIRegistry Port** property value to be the same port number as above.
6. Select **File > Save** or click .

The daemon cannot be located

When starting a server from the Management Console, the following message is displayed:

ERROR - Failed to locate the AtDaemon:

Possible causes:

- AtDaemon not started
- AtDaemon started on a different host than specified by the current configuration
- AtDaemon listens to a different RMI port than specified in the current configuration

Detailed exception:

java.rmi.ConnectException: Connection refused to host: <host name>, nested exception is:
java.net.ConnectionException: Connection refused: connect

The most likely causes of this are:

- The **AtDaemon** is not started
- The RMIRegistry used by the **AtDaemon** is executing on a different host or different port than specified in the server properties file
- The machine running the RMIRegistry and/or **AtDaemon** cannot be reached from the one on which the Management Console is running
- A firewall restricts the traffic between the computers hosting the **AtDaemon**, RMIRegistry and/or the Management Console

To resolve the issue, do one of the following:

- If the SSAS daemon is not started, start it.
- Ensure that both the daemon and server use the same publishing mechanism (RMIRegistry or JNDI), and that these are identically configured for both.
- If you have network problems, fix them, then restart the daemon and Management Console before trying to start the server again.
- If a firewall restricts traffic, fix the ports used by the daemon and server. For more information, see *Chapter 5: Configuring the server and daemon interfaces*.

Issues starting the server

When starting the server, an exception message similar to the following is displayed in the Management Console message pane:

<Some date> - Starting server, please wait
(due to seeding of the server, this may take a while)
ERROR - Failed to start server
Exception received:
<exception stack trace ending with>: *<Error message>*

The following tables present some common errors and their solutions.

Missing or incorrect licences

Error	Description and solution
License file abc does not exist	The server's License File property is not defined, has an empty value, or points to a missing file. Check that the License File property for the server you are starting has a correct value.
License verification failure: <component>	The license component data (component name, allowed host, or validity) has been modified. Check whether the license file was unintentionally modified. Contact Salt Support to obtain a new license file.
Invalid PKCS#1 padding	The license component signature has been modified. Check whether the license file was unintentionally modified. Contact Salt Support to obtain a new license file.

Incorrect token settings

Error message contains	Description and solution
java.io.FileNotFoundException: <filename> (The system cannot find the file specified)	A channel's software token is not configured correctly. If the software token is being used for the first time, ensure that its Create If Missing property is set to true. If the software token has been used before, ensure that Token Location specifies a correct file name.
EMPTY Channel Selector table	The channel selector's map contains no services. Ensure that every service type is mapped to a configured channel selector. For more information, see <i>Mapping channel selectors to the required channels</i> .
Invalid PKCS#5 padding or C_Login failed	Possible issues: The wrong master password was entered when starting the server. One of the software passwords is not configured correctly (Invalid PKCS#5 padding). This could be a password for a channel token, the Access PKI Environment token, or a database. One of the PKCS #11 hardware token passwords is not configured correctly (C_Login failed). This could be a password for a channel token or the Access PKI Environment token. The error message indicates where the login failed, which helps identify which password is incorrect. Ensure you enter the correct master password when starting the server. If necessary, reset or reconfigure the appropriate password. If that fails, reset all passwords for the server as described in <i>Changing all passwords for a server</i> .

Issues with client- server communication

When configuring client-server communication, you might encounter the following issues.

Issue	Description and solution
KeyEncipherment and DataEncipherment key usages are not checked	When the Use SSL flag is set for the server, communications are protected using TLS. However, the KeyEncipherment and DataEncipherment key usages in the public key certificate are not checked so no errors are raised if these are not set.
SSL Debug settings do not take effect	<p>When the server has been started, any future change to the SSL Debug settings do not take effect until you restart the SSAS daemon. For example, if you start the server with SSL debugging enabled, and then stop the server to disable SSL debugging, the debugging is still enabled when you restart the server.</p> <p>Restart the SSAS daemon for SSL Debug settings to take effect.</p>
The existence of multiple signing keys prevents the selection of a particular key	<p>When the Access PKI Environment contains multiple private keys, you cannot select which private key is to be used as part of the TLS/SSL handshake.</p> <p>You can use a Filtered Key Store for the Access PKI Environment (setting the Key Store Type to JSDPFiltered) with only one certificate chain in it to force selection of the required key.</p>

Glossary

Term	Definition
2FA	See <i>two-factor authentication</i> .
AAA	Authentication, authorisation, and accounting.
AdES	Advanced Electronic Signature. An electronic signature that has met the requirements set forth under EU Regulation No 910/2014 (eIDAS-regulation) on electronic identification and trust services for electronic transactions in the internal market
asymmetric channel	A SSAS channel, in which you are setting up signature services for PKI (asymmetric) environments. Compare with symmetric channel.
asymmetric cryptography	A form of cryptography where different keys, a shared public key and a secret private key, are used for encryption and decryption. Also known as public key cryptography. See PKI.
audit keys	See <i>TEMAC keys</i> .
authentication token	A hardware token used by the holder to verify their identity.
bridge CA	A CA set up by a trusted entity to act as a trust anchor for other PKIs. When the PKIs exchange their certificates with the bridge CA, it allows their entities to build paths of trust with entities in the other PKI known to the bridge CA. See cross- certification.
CA	Certification Authority or Certificate Authority. A trusted or known entity in a PKI that validates the authenticity of entities within the PKI and signs their certificates.
CAdES	CMS Advanced Electronic Signatures. It is a set of extensions to Cryptographic Message Syntax (CMS) signed data making it suitable for advanced electronic signatures.
certificate	A digital certificate containing the public key for an entity, which binds the key to the entity. When the certificate is signed, by a trusted entity or known CA, it provides a level of assurance that the entity is trusted.
certificate path	A chain of certificates and CAs that allow users of a PKI to trace an entity's certificate back to the trusted root CA. This activity is known as building a path of trust. See also cross-certification and bridge CA.
certificate revocation	The act of withdrawing (revoking) certificates prior to their natural expiration if, for example, the certificate entity has been compromised or is no longer deemed trustworthy.
channel	A logical group in SSAS that contains the settings and objects for the required authentication or signing services. You can create as many channels as you need. There are two types of channel, which support different services: symmetric and asymmetric channels. All channels use a channel token.
channel mapping	IMC configuration settings that determine the channels and token types that appear in the IMC.
channel password	The password for logging into a channel token. SSAS encrypts the individual channel passwords with the master password and stores them in the server properties file.
channel selector	A server object that directs incoming requests to the appropriate channel

	and service. SSAS has several pre-defined channel selector implementations, which parse requests based on different attributes, such as the sending client's subject DN.
channel token	A token (usually a crypto module or HSM) used by a SSAS channel to protect its sensitive material and perform the required cryptographic processing when dealing with authentication or digital signing requests. SSAS can use software tokens as channel tokens, but we do not recommend this for production environments.
client	An application server that has an API or plug-in configured to send authentication or signing requests to SSAS (the server).
client cryptographic token	A token (usually a crypto module or HSM) used by a SSAS client to protect the TLS key material used to secure its communication with SSAS (the server).
client mappings	The configuration within SSAS that specifies the SSAS clients allowed to connect to the server.
client request authentication	A SSAS feature where the server requires the client to authenticate their requests in a manner that cannot be disputed later. There are two methods of request authentication: <i>digest authentication</i> and <i>signature authentication</i> .
CR	Challenge response. An authentication mechanism where one party presents a question (a challenge) and another party must provide a valid answer (a response).
CRL	Certificate Revocation List. A service that can be used by entities to determine whether certificates have been withdrawn (revoked). See also OSCP server and certificate revocation.
cross-certification	The process of two separate PKIs exchanging signed certificates, which allows each other's entities to build paths of trust between the PKIs. For example, if two separate bank PKIs cross-certify, a customer of one bank can trust certificates from entities of the other bank. Alternatively, they can use a bridge CA.
crypto module	See HSM.
daemon	The SSAS daemon is a process that executes (runs) in the target runtime environment and is responsible for starting and stopping a server. In essence, it hosts the execution of the server. The daemon publishes an interface implementation, RMIRegistry or JNDI, which SSAS clients use to locate the server.
database	SSAS can use one or more databases for error logs, event logs, and storing token data for the authentication tokens and devices. The databases can be on the SSAS machine or another machine.
database connection pool	The connection details for one of the databases that SSAS is configured to use.
digest authentication	A method of achieving request authentication, based on a configured shared key between the server and client. For each request, the client is required to present a valid MAC over a challenge generated by the server. Compare with signature authentication.
digital signature	A generated hash, of the message being signed, encrypted with the signer's private key. The encrypted signature and the signer's certificate are included with the message to allow the recipient to check the signature by comparing the hashed message with the encrypted hash.
DPA	The Visa equivalent of EMV CAP.
EMV	Europay, MasterCard, Visa. A global standard for smart card chip inter-operation.
EMV CAP	EMV Chip Authentication Program. The MasterCard and Visa initiative and technical specification for using EMV banking smart cards for

	authenticating users and transactions in online and telephone banking.
error log	A log recording SSAS processing errors. You can record error information for individual channels.
event log	A log recording the requests that SSAS handles. You can record event information for individual channels and services.
FI	Financial Institution.
Filtered Key Store	A SSAS feature that enables you to share channel tokens between channels. It provides channel-specific views of the token by filtering the certificates and keys present on the token.
hardware token	A device, such as a smart card, electronic key fob, or crypto module, capable of storing cryptographic keys and certificates, and performing cryptographic operations. See also authentication token, channel token, and client cryptographic token.
HMAC	Hash-based Message Authentication Code. A short value, derived from a message using a cryptographic hash function, which is used to determine the integrity and authenticity of that encrypted message. Compare with MAC.
HOTP	HMAC-based one time password. An authentication mechanism that uses a cryptographic hash function to generate a 6 - 8 digit number that is only valid for one login session or transaction.
HSM	Hardware Security Module or Host Security Module. Also known as a crypto module. A purpose-built device that protects key material and provides secure cryptographic processing.
identity keys	Cryptographic keys used for non-repudiation. Compare with utility keys.
IMC	Identity Management Console. The graphical user interface used to administer authentication tokens, such as OATH and Vasco tokens.
IMK	Issuer Master Key, which is used to generate the derived keys that are personalised onto the EMV smart cards.
JNDI	Java Naming and Directory Interface. One of interface implementations that the SSAS daemon can publish so that SSAS clients can locate the server. Compare with RMIRegistry.
KEK	Key Encrypting Key or Key Encryption Key. Also known as a wrapping key. A key used for encrypting another key so that you can transport it securely between systems.
key pair	A public key and private key. See asymmetric cryptography.
LDAP server	Lightweight Directory Access Protocol server. LDAP is an open industry standard for accessing and maintaining distributed directory information over an IP network.
LMK	Local Master Key. The top-level system key stored within the tamper proof memory of the HSM, which is used to encrypt key material.
LVC	Log Viewing Console. The user interface used to inspect the SSAS error and event logs.
MAC	Message Authentication Code. A short value, derived from a message, which is used to determine the integrity and authenticity of that encrypted message. Compare with HMAC.
MAC verification	An authentication mechanism where one party generates a MAC value for a message (derived from the message and a secret key), which another party uses to authenticate the message and check the integrity of its content.
master password	The password for starting the server, which also encrypts the individual channel passwords that are stored in the server properties file.
MC	SSAS Management Console. The graphical user interface used to administer SSAS by modifying the parameters of the server properties file, which is read when the server process starts.

mCode	Traditional 2FA security token for verifying user identities to access online services using contemporary biometric methods.
MFA	Multi-factor authentication.
mSign	Contemporary mobile security MFA token for authentication of user identity and verification of transactions independent of the delivery channel that initiated the request.
MyID	An identity management solution for managing users and their digital credentials and issuing authentication devices. Intercede MyID was the OEM of SSMS.
non-repudiation	The ability to prove, in a way that cannot be reasonably disputed, that particular users instigated or approved particular activities, such as money transactions. In digital security, non-repudiation refers to: A service that provides proof of the integrity and origin of data An authentication that, with high assurance, can be asserted to be genuine
nShield Connect	A network-attached nShield HSM that provides secure general-purpose cryptographic processing.
nShield Solo	An nShield PCI or PCIe card for servers, providing general-purpose cryptographic processing.
OATH	Open Authentication. An industry-wide collaboration using open standards authentication for the universal adoption of strong authentication.
OCRA	OATH Challenge Response Algorithm. An algorithm for authenticating OATH tokens.
one-factor authentication	See <i>SFA</i> .
OTP	One time password. An authentication mechanism that generates a password that is only valid for one login session or transaction.
OCSP server	Online Certificate Status Protocol server. A service that can be used by entities to
PAdES	PDF Advanced Electronic Signatures. It is a set of restrictions and extensions to PDF and ISO 32000-1 making it suitable for advanced electronic signatures.
payShield 9000	A Thales HSM designed specifically for payment applications, providing secure card issuing and payment processing cryptographic functions.
PKCS #7 message	Cryptographic Message Syntax Standard. Can be used to sign or encrypt messages under a PKI, and circulate certificates.
PKCS #12 file	Personal Information Exchange Syntax Standard. A file typically containing a private key and a public key certificate.
PKI	Public Key Infrastructure. A framework of entities that use asymmetric cryptography and signed certificates to provide a hierarchy of trust. The hierarchy and certificates allow users to verify the identity of another party over an insecure network, such as the internet.
PKI path validation	See certificate path.
public key cryptography	See asymmetric cryptography and PKI.
RADIUS	Remote Authentication Dial In User Service. A widely used networking protocol that provides centralised authentication, authorisation, and accounting (AAA) management for users connecting to a network service. SSAS supports RADIUS in the form of a RADIUS server offering authentication only.
REST	REpresentational State Transfer.
request	See client request authentication.

authentication	
revocation	See certificate revocation.
RMI	Remote Method Invocation. The protocol that SSAS uses for client-server communication.
RMIRegistry	One of interface implementations that the SSAS daemon can publish so that SSAS clients can locate the server. Compare with JNDI.
Root CA	The highest CA within a PKI, which has a self-signed certificate and is the trust anchor within the PKI hierarchy.
SAML	Security Assertion Markup Language. An XML-based framework for communicating user authentication, entitlement, and attribute information.
Salt mCode	See mCode.
Salt Mobile SDK	Embed the capability of Salt mSign and Salt mCode standalone security tokens within existing mobile apps to enable traditional offline OTPs for 2FA and contemporary biometric connected MFA verification of user identity for access control and transaction verification.
Salt mSign	See mSign.
secret key	A cryptographic key that you must protect and keep private. Can be used to refer to either a symmetric key or the private key in an asymmetric key pair.
Security World	The cryptographic environment for nShield HSMs, which provides multi-layer protection for key material and cryptographic processing operations.
server	A SSAS installation, containing channels, services, channel selectors, and other objects that are configured to service authentication and signature requests for one or more clients. The server receives the incoming requests through its client interface, and directs each request to the appropriate channel, which services the request. SSAS runs as a server process, controlled by the SSAS daemon.
service	A collection of parameters that SSAS uses to process authentication or signing requests. You can create as many services as you need. There are pre-defined service implementations, which relate to the authentication tokens and signing methods the SSAS supports.
SFA	Single-factor authentication. Also known as one-factor authentication. A method of authentication where the user possesses one factor, typically knowledge of a username and password. Compare with <i>two-factor authentication</i> .
signature	See digital signature.
signature authentication	A method of achieving request authentication, where the client is configured with a private key and must sign important fields of each request, enabling the server to verify the request origin. Compare with digest authentication.
SOAP	Simple Object Access Protocol. A protocol for exchanging structured information in an implementation of web services.
software token	An encrypted file used for secure storage.
SSAS	Salt Safetronic Authentication Server. Salt software providing a single authentication platform, which you can use to consolidate all your authentication and digital signing requirements and manage the distributed authentication devices centrally.
SSCM	SafeSign Crypto Module. A discontinued Thales HSM that provided optimised cryptographic functions specific to SSAS. It was based on the Thales WebSentry device.
SSL	Secure Sockets Layer. See TLS.
SSMS	SafeSign Management Server. A discontinued Thales e-Security

	product. Intercede MyID was the OEM of SSMS. An identity management solution for managing users and their digital credentials and issuing authentication devices. Intercede MyID is commercially available.
SWS	Security World Software. The nShield software that sets up a Security World for your nShield HSMs.
symmetric channel	A SSAS channel, in which you are setting up services for symmetric key tokens or symmetric cryptographic environments. Compare with asymmetric channel.
symmetric cryptography	A form of cryptography where the same key, known as a symmetric key, is used for encryption and decryption. Hardware tokens typically contain a symmetric key.
symmetric key	Also known as a secret key. See symmetric cryptography.
tamper-evident logging	A SSAS feature that amends log entries with a sequence number and MAC so that log entries cannot be manipulated or re-ordered without detection.
TEMAC keys	Tamper-evident MAC keys. Keys used by SSAS to generate the MAC values for log entries, to provide tamper evidence protection.
TLS	Transport Layer Security. A protocol that uses a mixture of asymmetric and symmetric cryptography to protect the network communication between two entities, such as a client and server, so that they can communicate without the risk of eavesdropping and tampering. TLS is an evolution of Secure Sockets Layer (SSL).
TMC	Token Management Console. The graphical user interface used to administer channel tokens and client cryptographic tokens.
token	See hardware token and software token.
TOTP	Time-based one time password. An authentication mechanism that uses a cryptographic hash function and the current timestamp to generate a password that is only valid for one login session or transaction.
trust anchor	See <i>bridge CA</i> .
trust granularity	The PKI trust level for signature verification. You can trust different levels of the PKI hierarchy: the entire hierarchy, only sub-domains of the hierarchy, or only individual end-entity certificates.
two-factor authentication	A method of authentication where the user possesses two factors: they have a token and know a password or PIN to use that token. Compare with SFA.
User Authentication	A SSAS feature that queries an external user store to retrieve user OTP token information.
utility keys	Cryptographic keys used for decryption and digitally signing data. Compare with identity keys.
wrapped key	A key encrypted by another key, which is known as a wrapping key or KEK.
wrapping key	See <i>KEK</i> .
WSDD	Web Service Deployment Descriptor. A configuration file that describes how the components of a web service are chained together to process requests and responses.
WSDL	Web Services Descriptor Language. An XML-based interface description language for defining the requests a web service can accept and the responses it can generate.
WSS	Web Services Security. A protocol that is an extension to SOAP and uses XML signatures and XML encryption to protect communication, at the application layer, between two entities.

XAdES	XML Advanced Electronic Signatures. It is a set of extensions to XML-DSig recommendation making it suitable for advanced electronic signatures.
X.509	A standard for PKI, which specifies formats for public key certificates, CRLs, and other PKI elements.
XMLDSig	XML digital signature. You can sign an entire XML document, or part of it, to provide integrity and message authentication. The XML signature can be enveloped, enveloping, or detached (SSAS does not support detached). See digital signature