NAME

sn – Digitally sign/verify/compare strongnames on CLR assemblies.

SYNOPSIS

sn [-q | -quiet] [options] [parameters]

DESCRIPTION

Digitally sign, verify or compare CLR assemblies using strongnames.

You can use the sn command to create "snk files" using the -k option described below.

CONFIGURATION OPTIONS

Configuration options are stored in the machine.config configuration file under /configuration/strongNames.

-c provider
: Change the default CSP (Crypto Service Provider). Currently not supported in Mono.

-m [y|n]
: Use a machine [y] key container or a user [n] key container. Currently not supported in Mono.

-Vl
: List the verification options. The list is kept under /configuration/ strongNames/verificationSettings in machine.config.

-Vr assembly [userlist]
: Exempt the specified assembly from verification for the specified user list. Currently not supported by sn. You must edit machine.config manually if you require this.

-Vu assembly
: Remove the exemption entry for the specified assembly. Currently not supported by sn, you must edit machine.config manually if you require this.

-Vx
: Remove all exemptions entries. Currently not supported by sn, you must edit machine.config manually if you require this.

CSP RELATED OPTIONS

-d container
: Delete the keypair present in the specified key container.

-i keypair.snk container
: Import the specified strongname file into the specified container.

-pc container publickey
: Export the public key from the specified CSP container to the specified file.

CONVERSION OPTIONS

-e assembly output.pub
: Export the assembly public key to the specified output file.

-p keypair.snk output.pub
: Export the public key from the specified strongname key file (SNK) or from a PKCS#12/PFX password protected file to the specified output file.

-o input output.txt
: Convert the input file to a CSV file (using decimal).

-oh input output.txt
: Convert the input file to a CSV file (using hexadecimal).

STRONGNAME SIGNING OPTIONS

-D assembly1 assembly2
: Compare if assembly1 and assembly2 are the same except for their signature. This is done by comparing the hash of the metadata of both assemblies.

-k [size] keypair.snk
: Create a new strongname keypair in the specified file. The default key length is 1024 bits and MUST ALWAYS be used when signing 1.x assemblies. Any value from 384 to 16384 bits (in

increments of 8 bits) is a valid key length to sign 2.x assemblies. To ensure maximum compatibility you may want to continue using 1024 bits keys. Note that there's no good reason, even if it's possible, to use length lesser than 1024 bits.

-R assembly keypair.snk
Re-sign the specified assembly using the specified strongname keypair file (SNK) or a PKCS#12/PFX password protected file. You can only sign an assembly with the private key that matches the public key inside the assembly (unless it's public key token has been remapped in machine.config).

-Rc assembly container
Re-sign the specified assembly using the specified strongname container.

-t file Show the public key token from the specified file.

-tp file Show the public key and the public key token from the specified file.

-T assembly
Show the public key token from the specified assembly.

-Tp assembly
Show the public key and the public key token from the specified assembly.

-v assembly
Verify the specified assembly signature.

-vf assembly
Verify the specified assembly signature (even if disabled).

HELP OPTIONS

-h , -? Display basic help about this tool.

-h config , -? config
Display configuration related help about this tool.

-h csp , -? csp
Display Cryptographic Service Provider related help about this tool.

-h convert , -? convert
Display conversion related help about this tool.

-h sn , -? sn
Display strongname related help about this tool.

CONFIGURATION FILE

Strongnames configuration is kept in "machine.config" file. Currently two
features can be configured.

/configuration/strongNames/pubTokenMapping
This mechanism lets Mono remap a public key token, like the ECMA token, to another public key for verification. This is useful in two scenarios. First, assemblies signed with the "ECMA key" need to be verified by the "runtime" key (as the ECMA key isn't a public key). Second, many assemblies are signed with private keys that Mono can't use (e.g. System.Security.dll assembly). A new key cannot be used because it should change the strongname (a new key pair would have a new public key which would produce a new token). Public key token remapping is the solution for both problems. Each token must be configured in a "map" entry similar to this one: <map Token="b77a5c561934e089" PublicKey="00..." />

/configuration/strongNames/verificationSettings
It is often useful during development to use delay signed assemblies. Normally* the runtime wouldn't allow delay-signed assemblies to be loaded. This feature allows some delay-signed assemblies (based on their public key token, optionally assembly name and user name) to be used like they were fully signed assemblies. [*] Note that Mono 1.0 "runtime" doesn't validate strongname signatures so this option shouldn't be required in most scenarios.

AUTHOR

Written by Sebastien Pouliot

COPYRIGHT

MAILING LISTS

Visit http://lists.ximian.com/mailman/listinfo/mono-list for details.

WEB SITE

Visit http://www.mono-project.com for details

SEE ALSO

secutil(1)