

OpenDNSSEC Documentation v1.3

Released July 2011

1. OpenDNSSEC Documentation Home	3
1.1 Getting Started	3
1.2 Overview of OpenDNSSEC	5
1.2.1 Key States	7
1.3 Installation	8
1.3.1 Dependencies	10
1.3.2 Hardware Security Modules	13
1.3.2.1 Using SoftHSM	13
1.4 Configuration	16
1.4.1 Configuration files	16
1.4.1.1 conf.xml	18
1.4.1.2 kasp.xml	22
1.4.1.3 zonelist.xml	28
1.4.1.4 zonefetch.xml	29
1.4.1.5 signconf.xml	30
1.4.1.6 Date Time durations	33
1.4.2 Zone content	34
1.4.3 Migrating to OpenDNSSEC	34
1.5 Running OpenDNSSEC	36
1.5.1 Key Management	39
1.5.2 Zone Management	41
1.5.3 Logging	42
1.6 Command Utilities	43
1.6.1 ods-ksmutil	46
1.7 Troubleshooting	51
1.7.1 Frequently Asked Questions	54
1.8 Reporting bugs	55
1.9 Reference Material	56
1.9.1 Downloads	56

OpenDNSSEC Documentation Home

Welcome

About

The **OpenDNSSEC documentation** gives information on how to install, configure, and run **OpenDNSSEC**. There might still remain some questions, so we try to reflect them in our a growing list of [frequently asked questions](#).

Remember that you also need an HSM, which uses the PKCS#11 interface. We do provide the [SoftHSM](#), a software-only implementation of an HSM. Read the [HSM Buyer's Guide](#) for more information and consult the list of HSM vendors.

The latest version of OpenDNSSEC is 1.3.

Scope

The goal of OpenDNSSEC is to have a complete DNSSEC zone signing system which maintains stability and security of signed domains. DNSSEC adds many cryptographic concerns to DNS; OpenDNSSEC automates those to allow current DNS administrators to adopt DNSSEC. This document provides DNS administrators with the necessary information to get the system up and running with a basic configuration.

OpenDNSSEC Documentation

Getting Started

Overview of OpenDNSSEC

Installation

Configuration files

Running OpenDNSSEC

Command Utilities

Troubleshooting

Downloads

A PDF version is available on our [Downloads](#) page, or export selected pages [here](#) (it is recommended to exclude the `_Inclusions_Library`).

Export individual pages using the 'Tools->Export as PDF' option.

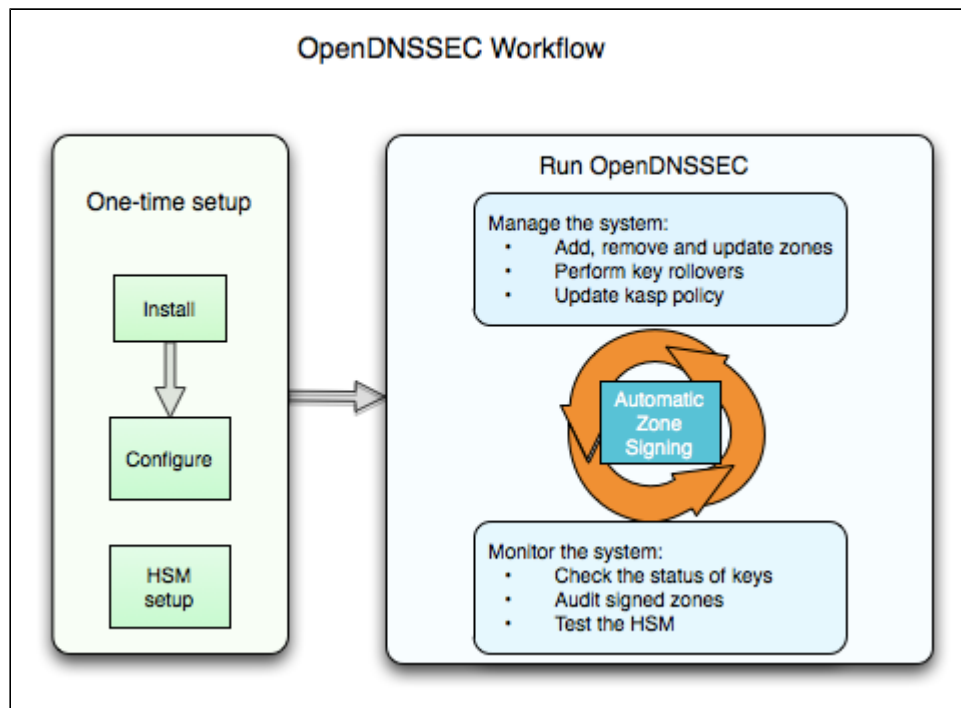
Getting Started

This page describes the key stages in getting up and running with OpenDNSSEC.

Experts can dive into the Basic Steps of setting up the system.

More detailed information on various elements of OpenDNSSEC is also listed including an overview of how OpenDNSSEC works.

Resources such as FAQ and reporting BUGS are also available.



On this Page

- Basics Steps
 - Installation
 - Configuration
 - Running OpenDNSSEC
 - Command utilities
- More Details...
 - Overview
 - Key states
 - Migrating to OpenDNSSEC
 - Hardware Security Modules
- Resources
 - Troubleshooting
 - FAQ
 - Reporting bugs

Basics Steps



Installation

Describes how you [install OpenDNSSEC](#) from source. It also gives some hints on how you can choose your hardware set-up to match your zone signing requirements.

Configuration

The next step after installation is to [configure OpenDNSSEC](#). Remember that you also need an HSM with the PKCS#11 API.

Running OpenDNSSEC

With the software installed and configured, it is time to [start using the system](#). Adding/removing zones, rolling keys, etc.

Command utilities

The software package comes with a number of [command line tools](#).

More Details....

Overview

Read an [overview](#) of how OpenDNSSEC works.

Key states

OpenDNSSEC maintains different [key states](#).

Migrating to OpenDNSSEC

It is possible to [migrate](#) a DNSSEC signed zone over to OpenDNSSEC, while keeping it secure.

Hardware Security Modules

Read about [HSM](#) and the OpenDNSSEC implementation of a soft HSM - [SoftHSM](#).

Resources



Troubleshooting

This chapter describes some [troubleshooting tips and experiences](#).

FAQ

This chapter describes some [frequently asked questions](#) from the users.

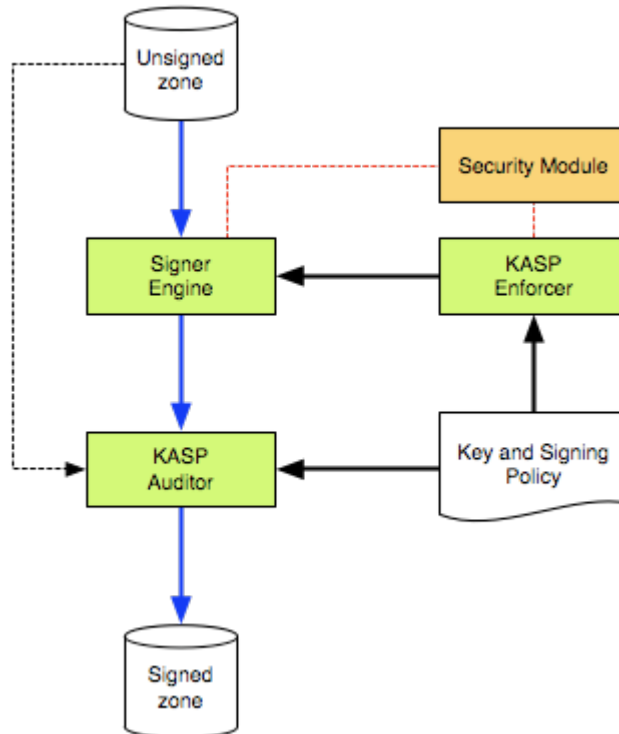
Reporting bugs

If you encounter any bugs: [Reporting Bugs](#)

Overview of OpenDNSSEC

OpenDNSSEC takes in unsigned zones, adds the signatures and other records for DNSSEC and passes the zones on to the authoritative name servers for that zones.

It does this according to a Key and Signing Policy (KASP) that describes how an organisation wants their DNSSEC configured.



On this Page

- [What does OpenDNSSEC do automatically?](#)
- [What can be done manually?](#)
- [What must be done manually?](#)
- [What are the key components of OpenDNSSEC?](#)

What does OpenDNSSEC do automatically?

Once [installed](#), [configured](#) and running OpenDNSSEC will do the following:

- Generate, publish and retire keys held in an HSM according to policy. See the full key lifecycle in the [Key States] guide.
- Automatically (re-)sign zones according to policy.
- Audit the signed zones against policy (if configured).
- Receive unsigned zones from and provide signed zones to nameservers via AXFR (if configured).

What can be done manually?

- Zones can be added, updated and removed.
- Keys can be backup and exported or managed manually.
- Manual key rollovers can be performed to cater for emergencies.
- The Key and Signing policy can be updated.
- Signed zones can be audited against policy.

See the [Running OpenDNSSEC](#) guide for more details

What must be done manually?

[Uploading the Trust Anchor](#) to the parent and notifying OpenDNSSEC that this has been done is a manual operation.

What are the key components of OpenDNSSEC?

- **KASP** - is the set of user defined policies to be used for signing and maintaining zones managed by this system.
- **Enforcer** - is responsible for enforcing the policy by managing the keys and orchestrating zone signing.
- **Signer** - is responsible for performing zone signing according to the instructions from the Enforcer. Is also responsible for ensuring that the zone is secure i.e. it will validate correctly.
- **Auditor** - performs an independent check of the signed zone before it is published. It does this by checking the zone against the KASP database using a different implementation of the audit logic to that used by the signer.

A **HSM** is also required for key management and storage.

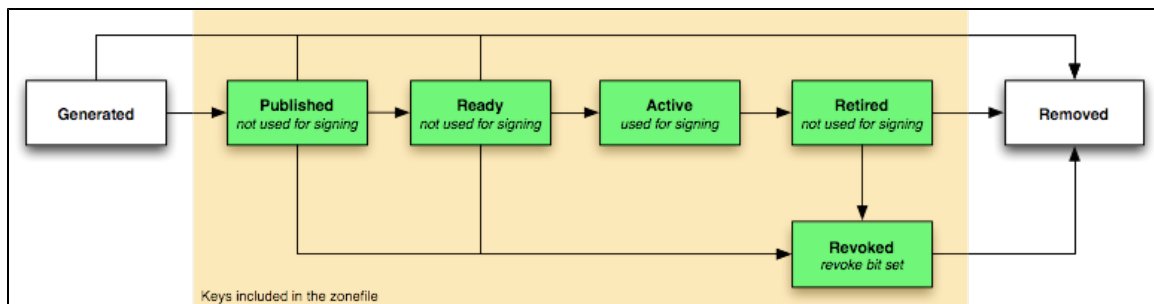
Key States

Most of the key states are as described in the DNSSEC key timing draft: <http://tools.ietf.org/html/draft-morris-dnsop-dnssec-key-timing-02>

The diagram below shows the states described in this draft.

On this Page

- [Generate](#)
- [Publish](#)
- [Ready](#)
- [Active](#)
- [Retired](#)
- [Dead](#)
- [KSK key states](#)
 - [DSSUB](#)
 - [DSPUBLISH](#)
 - [DSREADY](#)
 - [KEYPUBLISH](#)



Below is a brief describe of what the states mean:

Generate

Keys in the generate state have been created and stored but not used yet.

Publish

Keys in the publish state have been published in the zone, but are not yet considered safe to use. (i.e. They have not been in the zone long enough to have propagated through the system.)

Ready

Keys in the ready state have been published long enough that we could safely start to use them.

Active

Keys in the active state are those that are in use.

Retired

Keys in the retire state have been in use but have been replaced by a successor, they are post-published while signatures generated with them might still be in the system.

Dead

Keys in the dead state have been retired long enough for them to be safely removed from the zone.

KSK key states



For standby KSKs there are some extra states which replace Published and Ready. This is because the standby keys are not introduced into the zone until they are needed. Instead their DS record is submitted to the parent. (The idea is that if the key is needed in an emergency the shortest timescale that it can be used in is the publication through the child system.)

DSSUB

The DS has possibly been submitted (if it happened automatically) but in any case we are waiting for the ds-seen command.

DSPUBLISH

The ds-seen command has been given, and we are now waiting for the various propagation delays and safety margins to pass.

DSREADY

The DS record is now considered safe to use, so the standby key is ready.

KEYPUBLISH

We have been asked to use the standby key so we have published it in the zone. Once the key has propagated through the system it will move into the active state.

Installation

Before you start to use OpenDNSSEC in your production environment you must first decide which hardware you going to run on.

When you have a good system to run on, then it is time to install the software that OpenDNSSEC depends on, and finally installing OpenDNSSEC.

On this Page

- [Hardware set-up](#)
- [Platform support](#)
- [Dependencies](#)
- [Pre-built Binaries](#)
- [Obtaining the Source Code](#)
- [Building & Installing](#)
- [Post-installation](#)

Hardware set-up

Here are some short recommendations if you are planning to use OpenDNSSEC with many zones or a single large zone, and where the time is important.

- OpenDNSSEC is multi-threaded when it concerns the handling of multiple zone. But it is not currently multi-threaded in the handling of a single zone. So a multi-core machine will not give any benefits if you plan to only run a single large zone.



For handling on a single large zone is therefore more important to go with a CPU that is fast rather than a CPU with many cores.

- The OpenDNSSEC signer engine makes backup files to recover your zone data with no loss. These backup files will use up approximately three times the size of the signed zone on the HDD. The zones are also stored in memory. To keep track of updates, OpenDNSSEC maintains a previous, current and a new version of the zone.

Platform support

OpenDNSSEC has been tested on the following platforms:

- Debian Linux 5.0
- Mac OS X 10.5
- NetBSD
- OpenBSD 4.4
- Red Hat Enterprise Linux 5
- Solaris 10
- Ubuntu Linux 10.04

Dependencies

OpenDNSSEC depends on a number of open-source packages, all of which must be installed on your system for OpenDNSSEC to build successfully.

The installation of [dependencies guide](#) shows which packages are required and how to download/install them.



You also need a [Hardware Security Module](#).

Choose from any vendor that uses the [PKCS#11](#) interface. Or the software-only implementation of an HSM called SoftHSM created by the OpenDNSSEC project. Follow these instructions on [how to install SoftHSM](#).

Pre-built Binaries

You can find information about packages for your operating system here: <http://www.opendnssec.org/download/packages/>

Obtaining the Source Code

The latest version of OpenDNSSEC can be found as a tarball on <http://www.opendnssec.org>

The development (unstable) version of OpenDNSSEC is available from the Subversion repository and can be obtained using the following command:

```
svn co http://svn.opendnssec.org/trunk/OpenDNSSEC OpenDNSSEC
```

Building & Installing

1. If you downloaded the tarball then first untar it:

```
tar \-xzf opendnssec-<VERSION>.tar.gz  
cd OpenDNSSEC
```

or if you are working from the repository:

```
cd OpenDNSSECsh autogen.sh
```

2. Then it is time to configure the build scripts:

```
./configure
```

You may also need some other options to configure.

```
--disable-auditor      Disable auditor build (default enabled)
--enable-epclient     Enable epclient build (default disabled) (experimental)
--enable-timeshift    For debugging purposes
--with-database-backend Select database backend (sqlite3|mysql) (default sqlite)
```

Use the following command to find out which other options that are available:

```
./configure --help
```

The configure script defaults to `--prefix=/usr/local`, `--sysconfdir=/etc`, and `--localstatedir=/var`

3. Once configured, build OpenDNSSEC using:

```
make
```

... and install using ...

```
sudo make install
```



If the build fails it might be because of a missing software dependency. Please read the error messages carefully.

Post-installation

Depending on operating system, there may be a few additional steps required after installation.

Linux Users Linux users need to rebuild the dynamic linker caches. To do this, issue the command:

```
sudo ldconfig [library-path [library-path ...]]
```

If OpenDNSSEC or any of the pre-requisites were installed in non-standard directories, the list of library paths should be specified as arguments on the command line.

Dependencies

The following sections list the prerequisite software required for OpenDNSSEC.

On this Page

- [Software versions](#)
 - [ldns](#)
 - [libxml2](#)
 - [Ruby Gems](#)
 - [dnstruby](#)
 - [OpenSSL for Ruby](#)
 - [SQLite](#)
 - [MySQL](#)

Software versions

The following sections list the prerequisite software required for OpenDNSSEC. For Ubuntu users, the name of the package (where relevant) is listed.



The version of the package available from the Ubuntu download sites may not be compatible with OpenDNSSEC; in that case, the latest version of the package should be obtained (and built if required).

Users of operating systems with different software packaging should consult the appropriate documentation.

Implicit in these sections is the assumption that the operating system has the following languages installed: C, C++, Ruby. If any are absent, consult the documentation for your operating system.

In all cases, a location from where to get a copy of the package source code and build instructions for the package are given.



As there are some dependencies between the prerequisite components, they should be installed in the order listed here.

Note, where no version number is specified any fairly recent distribution (e.g. Ubuntu 10.04) will have a new enough version of the software in its standard repositories. Also note that these are minimum version numbers, so they provide API calls that we use; there may be bug fixes in later versions which are useful.

Software	min. version for 1.3.0	min. version for trunk
ldns	1.6.9	1.6.9
libxml2, libxml2-dev, libxml2-utils	2.6.16	2.6.16
ruby, rubygems		
dnstruby	1.52	1.52
libopenssl-ruby		
java	not required	
sqlite3, libsqlite3, libsqlite3-dev	3.3.9	3.3.9
(mysql-client, libmysqlclient15, libmysqlclient15-dev)	5.0.3	5.0.3

ldns

ldns is a DNS programming library used in the signer component.

Ubuntu Users

Make sure the the packages "libldns-x.z.y" (where "x.y.z" is the ldns version number) and "libldns-dev" are installed on your system.

Installing from Source Distribution

Download a copy of ldns from <http://www.nlnetlabs.nl/downloads/ldns>.

When downloaded and unpacked, "cd" into the directory into which you have unpacked the tar file and issue the following commands:

```
./configure (--disable-gost)
make
sudo make install
```

This installs the ldns library in `/usr/local/lib`. If you require the software to be installed elsewhere, add the switch `--prefix=<location>` to the `./configure` command.

libxml2

libxml2 is a C-library for handling XML. It is used in all parts of OpenDNSSEC.

Ubuntu Users

Install the packages "libxml2", "libxml2-dev", and "libxml2-utils".

Installing from Source Distribution

Download a copy of libxml2 from <http://xmlsoft.org/downloads.html>.

When downloaded and unpacked, "cd" into the directory into which you have unpacked the tar file and issue the following commands:

```
./configure
make
sudo make install
```

This installs the libxml2 library in */usr/local/lib*. If you require the software to be installed elsewhere, add the switch `--prefix=<location>` to the `./configure` command.

Ruby Gems

Ruby Gems is the standard way of installing Ruby packages. It is only used for the installation of the DNSRuby package.

Ubuntu Users

Install the package "rubygems".

Installing from Source Distribution

Download a copy of RubyGems from <http://rubyforge.org/projects/rubygems>.

When downloaded and unpacked, make sure that "ruby" is in your path, then issue the following command:

```
sudo ruby setup.rb
```

... to install RubyGems. The command "gem" will be installed in the same place as the "ruby" command.

dnstruby

dnstruby is a third-party DNS library used by the Auditor.

There is no Ubuntu package for dnstruby. Instead, on all operating systems, install dnstruby using the command:

```
sudo gem install dnstruby
```

OpenSSL for Ruby

The Auditor uses the OpenSSL library for some operations.

Ubuntu Users

Install the package "libopenssl-ruby".

Installing from Source Distribution

Most operating systems seem to include this software as part of the operating system. Consult your documentation for more information.

SQLite

SQLite is a cut-down SQL database system, used by the KASP component of OpenDNSSEC.

Ubuntu Users

Install the packages "sqlite3" and "libsqlite3-dev".

Installing from Source Distribution

Download a copy of SQLite from <http://www.sqlite.org/download.html>.

When downloaded and unpacked, "cd" into the directory into which you have unpacked the tar file and issue the following commands:

```
./configure
make
sudo make install
```

This installs sqlite in */usr/local/bin*. If you require the software to be installed elsewhere, add the switch `--prefix=<location>` to the

./configure command.

MySQL



You can choose to use MySQL instead of SQLite for the KASP database. This will give you better performance when handling thousands of zones.

Ubuntu Users

Install the packages "mysql-client", "libmysqlclient15", "libmysqlclient15-dev".

Installing from Source Distribution

Download it from <http://dev.mysql.com/downloads/mysql>

At this site there are links for various different systems, or to the source code if you want to build the code yourself. Full documentation is also available from the download page.

Hardware Security Modules

Documentation on Hardware Security Modules (HSMs).

Remember that to run OpenDNSSEC you also need an HSM, which uses the PKCS#11 interface. We do provide the SoftHSM, a software-only implementation of an HSM. Read the [HSM Buyer's Guide](#) for more information and consult the [list of HSM vendors](#).

- [Using SoftHSM](#)
- [HSM Buyer's Guide](#)
- [HSM Vendors](#)

Using SoftHSM

SoftHSM is an implementation of a cryptographic store accessible through a PKCS#11 interface. You can use SoftHSM as an HSM for OpenDNSSEC.

On this Page

- [Download](#)
- [Dependencies](#)
- [Installing](#)
- [Key management](#)
- [Converting keys to/from BIND](#)
- [Backup](#)

Download

Release:

- Can be download on <https://www.opendnssec.org/download/>

SVN repository (Read the README.svn):

```
svn co http://svn.opendnssec.se/trunk/softHSM/
```

Dependencies



Updated post release

Botan dependency corrected from 1.8.0 to 1.8.9

SoftHSM depends on the Botan 1.8.9 or greater (a cryptographic library) and SQLite 3.3.9 or greater (a database library). But it is recommended to use Botan 1.8.5 or greater since there is a known issues on some OS which freezes the application when it tries to pull entropy.



If the packaged version for your distribution does not work try to compile the latest version from source. They can be found at

<http://botan.randombit.net>

and

<http://www.sqlite.org>

Installing

1. Configure the installation/compilation scripts.

```
tar -xzf softhsm-<version>.tar.gz
cd softhsm-<version>
./configure
```

Options:

```
--with-botan=PATH      Specify prefix of path of Botan
--with-sqlite3=PATH    Specify prefix of path of SQLite3
--enable-64bit         Compile a 64-bit version
--with-loglevel=INT    The log level. 0=No log 1=Error 2=Warning
                      3=Info 4=Debug (default INT=3)
--prefix=DIR           The installation directory
                      (default DIR=/usr/local)
```

For more options:

```
./configure --help
```

2. Compile the source code.

```
make
```

3. Install the library

```
sudo make install
```

4. Add the tokens to the slots.

The default location of the config file is `/etc/softhsm.conf`. This location can be change by setting the environment variable.

```
export SOFTHSM_CONF=/home/user/config.file
```

Open the config file and add the slots

```
pico /home/user/config.file

0:/home/user/my.db
# Comments can be added
4:/home/user/token.database
```



The token databases does not exist at this stage. The given paths are just an indication to SoftHSM on where it should store the information for each token.

Each token is now treated as uninitialized.

5. Initialize your tokens. Use either the softhsm tool or the PKCS#11 interface.

```
softhsm --init-token --slot 0 --label "My token 1"
```

Type in SO PIN and user PIN.

```
softhsm --init-token --slot 4 --label "A token"
```

Type in SO PIN and user PIN.

6. Link to this library and use the PKCS#11 interface

Key management

It is possible to export and import keys to libsofthsm.

1. Importing a key pair.

Use the PKCS#11 interface or the softhsm tool where you specify the path to the key file, slot number, label and ID of the new objects, and the user PIN.
The file must be in PKCS#8 format.

```
softhsm --import key1.pem --slot 1 --label "My key" --id A1B2 --pin 123456
```

Add, `--file-pin <PIN>`, if the key file is encrypted.
Use, `softhsm --help`, for more info.

2. Exporting a key pair.

All keys can be exported from the token database by using the softhsm tool. The file will be exported in PKCS#8 format.

```
softhsm --export key2.pem --slot 1 --id A1B2 --pin 123456
```

Add, `--file-pin <PIN>`, if you want to output an encrypted file.
Use, `softhsm --help`, for more info.

Converting keys to/from BIND

The softhsm-keyconv tool can convert keys between BIND `.private-key` format and PKCS#8 key file format.

1. Convert from BIND `.private` to PKCS#8

Keys used for DNSSEC in BIND can be converted over to PKCS#8. Thus possible to import them into SoftHSM.

```
softhsm-keyconv --topkcs8 --in Kexample.com.+007+05474.private --out rsa.pem
```

Add, `--pin <PIN>`, if you want an encrypted PKCS#8 file.
Use, `softhsm-keyconv --help`, for more info.

2. Convert from PKCS#8 to BIND `.private` and `.key`

PKCS#8 files can be converted to key used for DNSSEC signing in BIND. The public key is also saved to file.

```
softhsm-keyconv --tobind --in rsa.pem --name example.com. --ttl 3600 \  
                --ksk --algorithm RSASHA1-NSEC3-SHA1
```

Add, `--pin <PIN>`, if you the PKCS#8 file is encrypted.
Use, `softhsm-keyconv --help`, for more info.

The following files will be created in this example:

```
Kexample.com.+007+05474.private  
Kexample.com.+007+05474.key
```

Backup

A token can be backed up by issuing the command:

```
sqlite3 <PATH TO YOUR TOKEN> ".backup copy.db"
```

Copy the "copy.db" to a secure location. To restore the token, just copy the file back to the system and add it to a slot in the file `softhsm.conf`.

If you are using SQLite3 version < 3.6.11, then you have to use the command below. But it will not copy the "PRAGMA user_version", which is used by SoftHSM for versioning. So you have to do that manually. In this case the version number is 100.

```
sqlite3 <PATH TO YOUR TOKEN> .dump | sqlite3 copy.db  
sqlite3 copy.db "PRAGMA user_version = 100;"
```

Configuration

Configuring OpenDNSSEC has a number of aspects:

Configuration files

These can be customised for each installation, although the default configuration installs good default values. The individual files and tools to check the configuration files are described [here](#).

Keys

Configure your [HSM](#) as required.

Zones

Check the supported [zone formats](#).

Migration

If you are migrating to OpenDNSSEC read our [Migration guide](#).

Configuration files

The default configuration installs good default values for anyone who just wants to sign their domains with DNSSEC. There are four configuration files for the basic OpenDNSSEC installation. You have

- *conf.xml* which is the overall configuration of the system,
- *kasp.xml* which contains the policy of signing,
- *zonelist.xml* where you list all the zones that you are going to sign,
- *zonefetch.xml* (which is optional) for zone transfers.

Click on the filenames below to see details of the file contents.

On this Page

- Files
 - [conf.xml](#)
 - [kasp.xml](#)
 - [zonelist.xml](#)
 - [zonefetch.xml](#)
- Signer configuration
- Date/time durations
- Checking your configuration files

Files

[conf.xml](#)

The overall configuration of OpenDNSSEC is defined by the contents of the file `/etc/opendnssec/conf.xml`. In this configuration file you specify logging facilities (only syslog is supported now), system paths, key repositories, privileges, and the database where all key and zone information is stored.

[kasp.xml](#)

`kasp.xml` - found by default in `/etc/opendnssec` - is the file that defines policies used to sign zones. KASP stands for "Key and Signature Policy", and each policy details

- security parameters used for signing zones
- timing parameters used for signing zones

You can have any number of policies and refer to the proper one by name in for example the `zonelist.xml` configuration file.

[zonelist.xml](#)

The `zonelist.xml` file is used when first setting up the system, but also used by the `ods-signerd` when signing zones. For each zone, it contains a `Zone` tag with information about

- the zone's DNS name
- the policy from `kasp.xml` used to sign the zone
- how to obtain the zone
- how to publish the zone

[zonefetch.xml](#)

OpenDNSSEC can sign zonefiles on disk, but can also sign zones received from transfer (AXFR). If you configure a zone fetcher configuration, the Signer Engine will kick off the zone fetcher that will listen to NOTIFY messages from the parent and store AXFR messages on disk.

Information in this file details

- where to fetch zone data from
- protection mechanisms to be used

Signer configuration

There are also xml files for each of the zones that the user wants to sign, but those are only used for communication between the Enforcer and the Signer Engine. And they are created automatically by the Enforcer. The location of these files can be found in `zonelist.xml`.

Read more [details about Signer configuration](#)

Date/time durations



Please read this description of [how date/time durations are specified in the above files](#).

Checking your configuration files

The OpenDNSSEC XML configuration files (`conf.xml` and `kasp.xml`) offer the user many options to customise the OpenDNSSEC signing system. Not all possible configuration texts are meaningful however.

A tool ([ods-kaspcheck](#)) is provided to check that the configuration files (`conf.xml` and `kasp.xml`) are semantically sane and contain no inconsistencies.



It is advisable to use this tool to check your configuration before starting to use OpenDNSSEC.

conf.xml

The overall configuration of OpenDNSSEC is defined by the contents of the file `/etc/opendnssec/conf.xml`. In this configuration file you specify logging facilities (only `syslog` is supported now), system paths, key repositories, privileges, and the database where all key and zone information is stored.

Date/time durations are as specified [here](#).

On this Page

- [The elements of the conf.xml file](#)
 - [Preamble](#)
 - [Configuration](#)
 - [Repositories](#)
 - [Common](#)
 - [Enforcer](#)
 - [Signer Configuration](#)
 - [Auditor Configuration](#)

The elements of the conf.xml file

Preamble

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: conf.xml.in 1143 2009-06-24 12:10:40Z jakob $ -->
```

Each XML file starts with a standard element "`<?xml...>`". As with any XML file, comments are included between the delimiters "`<!--`" and "`-->`".

Configuration

```
<Configuration>
```

The enclosing element of the XML file is the element `<Configuration>` which, with the closing element `</Configuration>`, brackets the configuration information.

Repositories

```
...
<RepositoryList>
```

OpenDNSSEC stores its keys in "repositories". There must be at least one repository, although the system can be configured to run with multiple repositories.



There are a number of reasons for running with multiple repositories, including:

- Temporarily running with multiple repositories whilst a switch is made from one repository to another, e.g. when replacing hardware.
- The chosen type of repository has a limit on the number of keys that can be stored, and multiple repositories are needed to handle the chosen number of keys.
- Different types of repository are needed for different security levels, e.g. a key-signing key may require a higher level of security than a zone-signing key.

In practice, all repositories are "HSM"s (hardware security modules) or an emulation of one.

SoftHSM

The software emulation of an HSM - SoftHSM - must be installed separately but is part of the OpenDNSSEC project, and its definition is shown below. The element names will be explained later.

```
...
<Repository name="SoftHSM">
  <Module>/usr/local/lib/libsoftsm.so</Module>
  <TokenLabel>OpenDNSSEC</TokenLabel>
  <PIN>1234</PIN>
</Repository>
```

HSM

As indicated, any number of repositories can be specified in the configuration file:

```
<!--
<Repository name="sca6000">
  <Module>/usr/lib/libpkcs11.so</Module>
  <TokenLabel>Sun Metaslot</TokenLabel>
  <PIN>test:1234</PIN>
  <Capacity>1000</Capacity>
  <RequireBackup/>
  <SkipPublicKey/>
</Repository>
-->
```

(The example above is commented out by the XML comment delimiters.)

- `<Repository>` - the definition of a repository is bracketed by the `<Repository>` and `</Repository>` elements. The name attribute must be supplied and must be unique. It is this name that is used in the `kasp.xml` file to identify which repository holds the keys. This field is limited to 30 characters.
- `<Module>` identifies the dynamic-link library that controls the repository. Each type of HSM will have its own library.
- `<TokenLabel>` identifies the "token" within the HSM that is being used - essentially a form of sub-repository. The token label is also used where there are two repositories of the same type, in that each repository should contain a different token label sub-repository. OpenDNSSEC will automatically go to the right HSM based on this. This field is limited to 32 characters.
- `<PIN>` is the password to the HSM. OpenDNSSEC have this stored *en-claire* in the configuration file. Later versions will allow the option of requiring the password to be typed in when OpenDNSSEC is started.
- `<Capacity>` indicates the maximum number of keys the HSM can store. It is an optional element - if there is no (realistic) limit to the number of keys, remove it.
- `<RequireBackup>` is an optional element that specifies that keys from this repository may not be used until they are backed up. If backup has been done, then use `ods-ksmutil` to notify OpenDNSSEC about this. The backup notification is needed for OpenDNSSEC to be able to complete a key rollover.
- `<SkipPublicKey>` is an optional element which specifies that the public key objects should not be stored or handled in the HSM. The public key is needed in order to create the DNSKEY RR. In theory, the public part of the key is also available in the private key object. However, the PKCS#11 API does not require the HSM to behave in this way. We have not seen a HSM where we cannot do this, but you should remove this flag if you are having any problem with it. The benefit of adding this flag is that you save space in your HSM, because you are only storing the private key object.

```
...
</RepositoryList>
```

The list of repositories ends with the closing tag.

Common

These are the configuration that is shared among the components of OpenDNSSEC.

```
...
<Common>
  <Logging>
    <Syslog><Facility>local0</Facility></Syslog>
  </Logging>

  <PolicyFile>/etc/opendnssec/kasp.xml</PolicyFile>
  <ZoneListFile>/etc/opendnssec/zonelist.xml</ZoneListFile>

  <!--
    <ZoneFetchFile>/etc/opendnssec/zonefetch.xml</ZoneFetchFile>
  -->
</Common>
```

All components of HSM log error, warning and progress information. This is configurable and defined in the <Logging> element. Currently, the only syslog() feature configurable via the OpenDNSSEC configuration file is the facility name under which messages are logged. This can be any of the names listed in the operating system's syslog header file (usually /usr/include/sys/syslog.h, but the location is system dependent).



Although any facility listed there can be used, it is recommended that one of the "local" facilities (usually "local0" through "local7") be used.

Then you also have pointers to where the policy and zone list files can be found. There are also an optional element where you specify the path of the zone fetch configuration used for inbound AXFR.

Enforcer

The KASP enforcer component of OpenDNSSEC - which deals with key rollover and key generation - has its own section in the configuration file:

```
...
<Enforcer>
  <!--
  <Privileges>
    <User>opendnssec</User>
    <Group>opendnssec</Group>
  </Privileges>
  -->

  <Datastore><SQLite>/var/opendnssec/kasp.db</SQLite></Datastore>
  <Interval>PT3600S</Interval>
  <!-- <ManualKeyGeneration/> -->
  <!-- <RolloverNotification>P14D</RolloverNotification> -->
  <!-- <DelegationSignerSubmitCommand>/usr/local/sbin/epclient</DelegationSignerSubmitCommand> -->
</Enforcer>
```

The section is bracketed by the <Enforcer> .. </Enforcer> tags.

The Enforcer can drop its privileges if specified.

The database used by the Enforcer is specified by the <Datastore> tag. OpenDNSSEC supports SQLite and MySQL, the choice being indicated by one of two mutually exclusive tags:

SQLite

If SQLite is the database in use, the <Datastore> tag must contain a single <SQLite> tag which specifies the database file in use (as shown above).

MySql



The support of MySQL is considered experimental

If MySQL is in use, then the <Datastore> contains a single <MySQL> tag, which in turn contains elements that describe the database connection. The following XML elements shows this:

```
...
<Datastore>
  <MySQL>
    <Host port="1213">dnssec-db</Host>
    <Database>KASP</Database>
    <Username>kaspuser</Username>
    <Password>abc123</Password>
  </MySQL>
</Datastore>
```

- <Host> is the name of the system on which the database resides. It is optional - if omitted, the database is assumed to run on the same system as OpenDNSSEC. The "port" attribute gives the port to which the MySQL connection is made. It too is optional, and defaults to 3306 if omitted.
- <Database> is the name of the database holding the KASP Enforcer data.
- <Username> is the username needed to connect to the database.
- <Password> is the password associated with the username.

Other Enforcer Parameters

- <Interval> is how often the Enforcer runs to check whether keys are coming up for expiry and should be rolled. The more frequently this is run, the closer will the usage of keys reflect the policy set for it. However, if the key lifetimes are in the order of months, an <Interval> of the order of a day to a week is sufficient.
- <ManualKeyGeneration/> will disable the automatic key generation in the Enforcer. The user have to generate the keys itself with the *ods-ksmutil key generate* command.
- <RolloverNotification> specifies how long before a KSK rollover the Enforcer should start logging messages about the future rollover.



Configure the <DelegationSignerSubmitCommand> if you want to have a program/script receiving the new KSK during a key rollover. This will make it possible to create a fully automatic KSK rollover, where OpenDNSSEC feed your program/script on *stdin* with the current set of DNSKEYs that we want to have in the parent as DS RRs.

There are two examples available: an *epplibclient* and a simple mail script. Remember that the *ods-ksmutil key ds-seen* must be given in order to complete the rollover. This should only be done when the new DS RRs are available on the parents public nameservers.

Signer Configuration

The Signer Engine of OpenDNSSEC - the part that constructs signature records to include in to the zone file - also has its own section in the configuration file:

```

...
<Signer>
<!--
  <Privileges>
    <User>opendnssec</User>
    <Group>opendnssec</Group>
  </Privileges>
-->
  <WorkingDirectory>/var/opendnssec/tmp</WorkingDirectory>
  <WorkerThreads>4</WorkerThreads>
  <SignerThreads>4</SignerThreads>

<!--
  <NotifyCommand>/usr/local/bin/my_nameserver_reload_command</NotifyCommand>
-->
</Signer>

```

Delimited by the `<Signer>` .. `</Signer>` tags, the components are:

- `<Privileges>` the Signer Engine can drop its privileges if specified.
- `<WorkingDirectory>` defines the location in which the Signer Engine will create temporary files.
- `<WorkerThreads>` specify the number of workers. One worker can handle one zone a time. When it is finished with the zone it takes the next one in queue.
- `<SignerThreads>` specify the number of threads that are dedicated to signing RRsets. Usually set to the number of parallel operations your HSM can handle. If the element is omitted from the configuration, the number of signer threads is equal to the number of workers.
- `<NotifyCommand>` optional element that will tell the Signer Engine to call this command when the zone has been signed. Will expand the following variables: %zone (the name of the zone that was signed) and %zonefile (the filename of the signed zone).

Auditor Configuration

The Auditor can check a signed zone against the policy and the unsigned zone. This is to verify that the everything is done correctly.

```

...
<Auditor>
<!--
  <Privileges>
    <User>opendnssec</User>
    <Group>opendnssec</Group>
  </Privileges>
-->

  <WorkingDirectory>/var/opendnssec/tmp</WorkingDirectory>
</Auditor>

```

Delimited by the `<Auditor>` .. `</Auditor>` tags, the components are:

- `<Privileges>` the Auditor can drop its privileges if specified.
- `<WorkingDirectory>` defines the location in which the Auditor will create temporary files.


```

</Configuration>

```

As there are no more elements, the `</Configuration>` tag closes the file.

kasp.xml

 KASP stands for "Key and Signature Policy"

kasp.xml (found by default in `/etc/opendnssec`) is the file that defines policies used to sign zones. Each policy comprises a series of parameters that define the way the zone is signed. This section explains the parameters by referring to the example kasp.xml file supplied with the OpenDNSSEC distribution.

Date/time durations are as specified [here](#).

On this Page

- [Elements of the kasp.xml file](#)
 - [Preamble](#)
 - [Policy Description](#)
 - [Signatures](#)
 - [Authenticated Denial of Existence](#)
 - [Key Information](#)
 - [Zone Information](#)
 - [Parent Zone Information](#)
 - [Auditing](#)

Elements of the kasp.xml file

Preamble

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: kasp.xml.in 1154 2009-06-24 13:02:25Z jakob $ -->
```

Each XML file starts with a standard element "`<?xml...>`". As with any XML file, comments are included between the delimiters "`<!-->`" and "`-->`".

Policy Description

```
<KASP>
```

The enclosing element of the XML file is the element `<KASP>` which, with the closing element `</KASP>`, brackets one or more policies.

```
...
<Policy name="default">
```

Each policy is included in the `<Policy>...</Policy>` elements. Each policy has a "name" attribute giving the name of the policy. The name is used to link a policy and the zones signed using it; each policy must have a unique name.

The policy named "default" is special, as it is associated with all zones that do not have a policy explicitly associated with them.

```
...
<Description>A default policy that will amaze you and your friends</Description>
```

A policy can have a description associated with it. Unlike XML comments, the description can be understood by programs and may be used to document the policy, e.g. a future GUI may display a list of policies along with their description and ask you to select one for editing.

Signatures

The next section of the file is the Signatures section, which lists the parameters for the signatures created using the policy.

```

...
<Signatures>
  <Resign>PT2H</Resign>
  <Refresh>P3D</Refresh>
  <Validity>
    <Default>P7D</Default>
    <Denial>P7D</Denial>
  </Validity>
  <Jitter>PT12H</Jitter>
  <InceptionOffset>PT300S</InceptionOffset>
</Signatures>

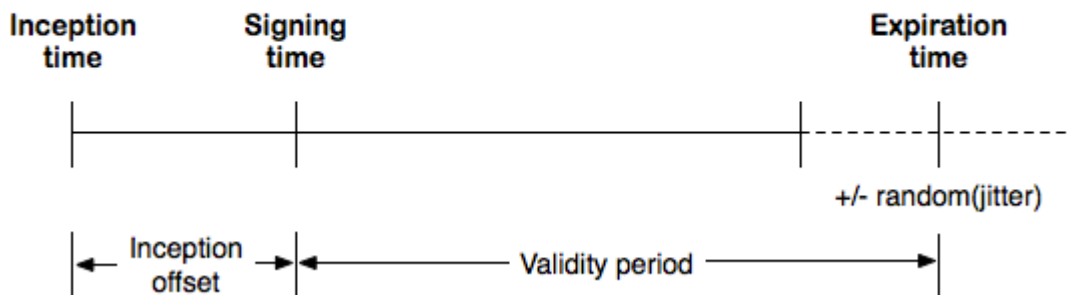
```

Here:

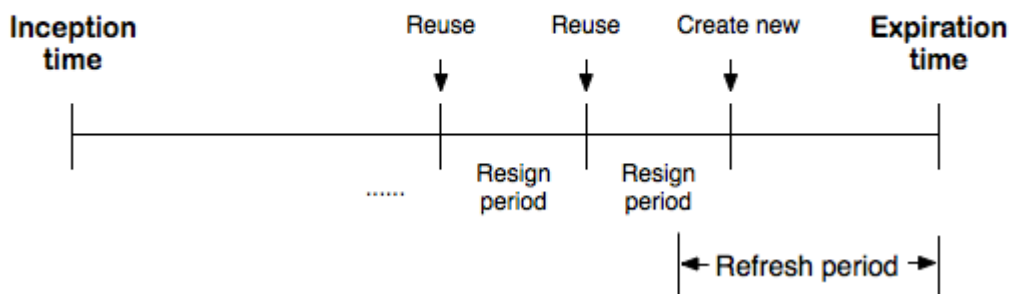
- <Resign> is the re-sign interval, which is the interval between runs of the Signer Engine.
- <Refresh> is the refresh interval, detailing when a signature should be refreshed. As signatures are typically valid for much longer than the interval between runs of the signer, there is no need to re-generate the signatures each time the signer is run if there is no change to the data being signed. The signature will be refreshed when the time until the signature expiration is closer than the refresh interval. Set it to zero if you want to refresh the signatures each re-sign interval.
- <Validity> groups two elements of information related to how long the signatures are valid for - <Default> is the validity interval for all RRSIG records *except* those related to NSEC or NSEC3 records. In this case, the validity period is given by the value in the <Denial> element.
- <Jitter> is the value added to or extracted from the expiration time of signatures to ensure that not all signatures expire at the same time. The actual value of the <Jitter> element is the $-j + r \% 2j$, where j is the jitter value and r a random duration, uniformly ranging between $-j$ and j , is added to signature validity period to get the signature expiration time.
- <InceptionOffset> is a duration subtracted from the time at which a record is signed to give the start time of the record. This is required to allow for clock skew between the signing system and the system on which the signature is checked. Without it, the possibility exists that the checking system could retrieve a signature whose start time is later than the current time.

The relationship between these elements is shown below.

Signature lifetime



Reuse of signatures



Authenticated Denial of Existence

Authenticated denial of existence - proving that domain names do not exist in the zone - is handled by the <Denial> section, as shown below:

```
...
<Denial>
  <NSEC3>
    <OptOut/>
    <Resalt>P100D</Resalt>
    <Hash>
      <Algorithm>1</Algorithm>
      <Iterations>5</Iterations>
      <Salt length="8"/>
    </Hash>
  </NSEC3>
</Denial>
```

<Denial> includes one element, either <NSEC3> (as shown above) or <NSEC>.

NSEC3

- <NSEC3> tells the signer to implement NSEC3 scheme for authenticated denial of existence (described in [RFC 5155](#)). The elements are:
- <OptOut/> - if present, enable "opt out". This is an optimisation that means that NSEC3 records are only created for authoritative data or for secure delegations; insecure delegations have no NSEC3 records. For zones where a majority of the entries are delegations that are not signed - typically TLDs during the take-up phase of DNSSEC - this reduces the number of DNSSEC records in the zone.
- <Resalt> is the interval between generating new salt values for the hashing algorithm.
- <Algorithm>, <Iterations> and <Salt> are parameters to the hash algorithm, described in [RFC 5155](#).

NSEC

Should, instead, NSEC be used as the authenticated denial of existence scheme, the <Denial> element will contain the single element <NSEC/> - there are no other parameters.

Key Information

Parameters relating to keys can be found in the <Keys> section.

```
...
<Keys>
```

Common Parameters

The section starts with a number of parameters relating to both zone-signing keys (ZSK) and key-signing keys (KSK):

```
...
<TTL>PT3600S</TTL>
<RetireSafety>PT3600S</RetireSafety>
      <PublishSafety>PT3600S</PublishSafety>
<ShareKeys/>
<Purge>P14D</Purge>
```

- <TTL> is the time-to-live value for the DNSKEY resource records.
- <PublishSafety> and <RetireSafety> are the publish and retire safety margins for the keys. These intervals are safety margins added to calculated timing values to ensure that keys are published and retired without there being a chance of signatures created with the keys being considered invalid.

If multiple zones are associated with a policy, the presence of <ShareKeys/> indicates that a key can be shared between zones. E.g. if you have 10 zones then you will only use one set of keys instead of 10 sets. This will save space in your HSM. If this tag is absent, keys are not shared between zones.

If <Purge> is present, keys marked as dead will be automatically purged from the database after this interval.

Key-Signing Keys

Parameters for key-signing keys are held in the <KSK> section:

```
...
<KSK>
  <Algorithm length="2048">7</Algorithm>
  <Lifetime>PLY</Lifetime>
  <Repository>softHSM</Repository>
  <!-- <Standby>1</Standby> (Experimental) -->
  <ManualRollover/>
</KSK>
```

- <Algorithm> determines the algorithm used for the key (the numbers reserved for each algorithm can be found in the appropriate [IANA registry](#)).
- <Lifetime> determines how long the key is used for before it is rolled.
- <Repository> determines the location of the keys. Keys are stored in "repositories" (currently, only hardware security modules (HSMs) or devices conforming to the PKCS#11 interface), which are defined in the [conf.xml](#). In the example above, the key is stored in softHSM - the example configuration file distributed with OpenDNSSEC defines this as being the software emulation of an HSM distributed as part of OpenDNSSEC.
- <Standby> **Experimental** (we are currently not supporting offline HSM, which is needed to get the security level needed to fulfill the idea behind standby keys. Will be fixed in a future version) Determines the number of standby keys held in the zone. These keys allow the currently active key to be immediately retired should it be compromised, so enhancing the security of the system. (Without an standby key, additional time is required to allow information about the new key to reach validator caches - see <http://tools.ietf.org/html/draft-ietf-dnsop-dnssec-key-timing-02> for timing details.)
- <ManualRollover/> is an optional tag. This tag indicate that the key rollover will only be initiated on the command by the operator. There is still a second step for the KSK, where the key needs to be published to the parent before the rollover is completed. Read more in the chapter "Running OpenDNSSEC". The ZSK rollover will although be fully automatic if this tag is not present.

Zone-Signing Keys

Parameters for zone-signing keys are held in the <ZSK> section, and have the same meaning as for the KSK:

```
...
<ZSK>
  <Algorithm length="1024">7</Algorithm>
  <Lifetime>P14D</Lifetime>
  <Repository>softHSM</Repository>
  <!-- <Standby>1</Standby> (Experimental) -->
</ZSK>
```

The ZSK information completes the contents of the <Keys> section.

```
...
</Keys>
```

Zone Information

General information concerning the zones can be found in the <Zone> section:

```
...
<Zone>
  <PropagationDelay>PT9999S</PropagationDelay>
  <SOA>
    <TTL>PT3600S</TTL>
    <Minimum>PT3600S</Minimum>
    <Serial>unixtime</Serial>
  </SOA>
</Zone>
```

- <PropagationDelay> is the amount of time needed for information changes at the master server for the zone to work its way through to all the secondary nameservers.
- The <SOA> element gives values of parameters for the SOA record in the signed zone.



These values will override values set for the SOA record in the input zone file.

The values are:

- <TTL> - TTL of the SOA record.
- <Minimum> - value for the SOA's "minimum" parameter.
- <Serial> - the format of the serial number in the signed zone. This is one of:
 - counter - use an increasing counter (but use the serial from the unsigned zone if possible)
 - datecounter - use increasing counter in YYYYMMDDxx format (xx is incremented within each day)
 - unixtime - the serial number is set to the "Unix time" (seconds since 00:00 on 1 January 1970 (UTC)) at which the signer is run.
 - keep - keep the serial from the unsigned zone (do not resign unless it has been incremented)

Parent Zone Information

If a DNSSEC zone is in a chain of trust, digest information about the KSKs used in the zone will be stored in DS records in the parent zone. To properly roll keys, timing information about the parent zone must be configured in the <Parent> section:

```
...
<Parent>
  <PropagationDelay>PT9999S</PropagationDelay>
  <DS>
    <TTL>PT3600S</TTL>
  </DS>
  <SOA>
    <TTL>PT3600S</TTL>
    <Minimum>PT3600S</Minimum>
  </SOA>
</Parent>
```

- <PropagationDelay> is the interval between the time a new KSK is published in the zone and the time that the DS record appears in the parent zone.
- The <DS> tag holds information about the DS record in the parent. It contains a single element, <TTL>, which should be set to the TTL of the DS record in the parent zone.
- <SOA> gives information about parameters of the parent's SOA record, used by KASP in its calculations. As before, <TTL> is the TTL of the SOA record and <Minimum> is the value of the "minimum" parameter.

Auditing

The zone will be audited before it is written to the signed directory, if the following tag is included.



If you are signing a large number of zones and have a high work load on your server, the memory resources might get exhausted because each instance of the auditor has its own Ruby VM.

```
...
<Audit />
```

If you are signing a very large zone (more than half a million records, for example), then you may wish to use the Partial Auditor. This checks a sample of the zone (rather than every RRSet cryptographic signature), and performs many of the same checks as the full auditor (including key lifetime tracking). To enable this, replace the above Audit tag with :

```
...
<Audit>
  <Partial />
</Audit>
```

This is the last section of the policy specification, so the next element is the policy closing tag:

```
...
</Policy>
```

If there are any additional policies, they could be entered here, starting with `<Policy>` and ending with `</Policy>`. However, in this case there are no additional policies, so the file is ended by closing the `<KASP>` tag:

```
</KASP>
```

zonelist.xml

The list of zones that OpenDNSSEC will sign is held in the zone list file `/etc/opendnssec/zonelist.xml`. As well as listing the zones, it also specifies the policy used to sign the zones.

This section explains the parameters of the zone list by referring to the example `zonelist.xml` file supplied with the OpenDNSSEC distribution.

Date/time durations are as specified [here](#).

On this Page

- [Elements of the zonelist.xml file](#)
 - [Preamble](#)
 - [Zone List](#)
 - [Zones](#)
- [Zone List File Creation](#)
- [File Names](#)

Elements of the zonelist.xml file

Preamble

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: zonelist.xml.in 1147 2009-06-24 12:18:17Z jakob $ -->
```

Each XML file starts with a standard element `<?xml...>`. As with any XML file, comments are included between the delimiters `<!-->` and `-->`.

Zone List

```
<ZoneList>
```

The enclosing element of the XML file is the element `<ZoneList>` which, with the closing element `</ZoneList>`, brackets the list of zones.

Zones

Each zone is defined by a `<Zone>` element:

```
...
<Zone name="example.com">
```

The mandatory attribute "name" identifies the zone. Each zone within the zone list must have a unique name. Use "." when signing the root.

Policy

```
...
<Policy>default</Policy>
```

The first element of the `<Zone>` tag is `<Policy>`, which identifies the policy used to sign the file. Policies are defined in the `kasp.xml` file, and

the name in this element must be that of one of the defined policies.

Configuration

Information from the Enforcer to the Signer Engine is passed via a special signer configuration file, the name of which is given by the `<SignerConfiguration>` section of the zone definition:

```
...
<SignerConfiguration>/var/opendnssec/signconf/example.com.xml</SignerConfiguration>
```

(Note that this file is a temporary file that passed between OpenDNSSEC components and is not intended to be edited by users.)

Adapters

The next part of the zone element specifies from where OpenDNSSEC gets the zone data and to where the signed data is put.

```
...
<Adapters>
  <Input>
    <File>/var/opendnssec/unsigned/example.com</File>
  </Input>
  <Output>
    <File>/var/opendnssec/signed/example.com</File>
  </Output>
</Adapters>
```

The `<Adapters>` element comprises an `<Input>` and `<Output>` element which (fairly obviously) identify the input source and output sink of the data.

Within each element is a tag defining the type of data source/sink and its parameters. At the moment, only the `<File>` element is defined, which takes as its only data the name of the input unsigned or output signed zone file.

In the future, it is planned to offer the ability to accept and output data in the form of AXFRs and IXFRs.

```
...
</Zone>
```

The `</Zone>` tag closes the definition of the zone. As indicated above, one or more zones can be defined in this file.

```
</ZoneList>
```

The `</ZoneList>` element closes the file.

Zone List File Creation

For a small number of zones, the zone list file can be easily edited by hand. Where the number of zones is large - for example, ISPs serving thousands of customers - the intention is that the file be generated by the zone manager's systems using e.g. the `ods-ksmutil zone add` command.

File Names

As can be seen in the example above, a number of elements that specify file names (`<SignerConfiguration>`, `<Adapter>/<Input>` and `<Adapter>/<Output>`) include the zone name in the name of the file.



Where there are multiple zones, this is strongly recommended as a way of avoiding confusion.

zonefetch.xml

OpenDNSSEC can sign zonefiles on disk, but can also sign zones received from transfer (AXFR). If you configure a zone fetcher configuration (see [XML / RelaxNG compact](#)), the Signer Engine will kick off the zone fetcher that will listen to NOTIFY messages from the parent and store AXFR messages on disk. The messages will be stored as the input file adapter plus an additional ".axfr" extension. If the transfer was

successful, the zone fetcher kicks the Signer Engine so that the incoming zone will be signed.

Date/time durations are as specified [here](#).

On this Page

- [Configuration](#)
- [Design](#)

Configuration

The zone fetcher configuration filename must be specified in [conf.xml](#) in the <ZoneFetchFile> element. An example zonelfetch.xml file is available [here](#). In the zonelfetch.xml file the following elements may be specified:

- <NotifyListen>: This defines which interface and port the must bind to listen NOTIFY messages on. You can specify an IPv4/IPv6 address plus port number.
- <Default>: This has the default values for master servers and tsig credentials.
 - <TSIG>: This configures your TSIG credentials. Name, Algorithm and Secret are required.
 - <RequestTransfer>: This configures your master servers to contact. You can specify multiple IPv4/IPv6 addresses. Unfortunately, only the first encountered port number will be used.

Design

The zone fetcher can run a single time, but the Signer Engine will start it as a daemon. As a daemon, it will accept NOTIFY messages for which it has master servers configured (withRequestTransfer). NOTIFY also does not make use of the TSIG credentials.

You can specify the listening interface and port with <NotifyListen>. By default, the zone fetcher will listen on any interface, port 53.

To listen on a specific address, use:

```
...
<NotifyListen>
  <IPv4>192.0.2.100</IPv4><Port>53</Port>
</NotifyListen>
```

Upon a valid NOTIFY, the zone fetcher sends a transfer request to one of the master servers. If configured, it adds the TSIG RR. A successful AXFR response will be stored on disk.

The Signer Engine will know if it has to check for an AXFR on disk before signing a new unsigned zone. Thus, the Signer Engine needs to be kicked with 'update <zone>' if a AXFR was received. Luckily, the zone fetcher will do that for you.

signconf.xml

There are xml files for each of the zones that the user wants to sign. Those define the API between the Enforcer and the Signer Engine. The Enforcer creates these files while implementing the policies and key management.

The location of these files can be found in zonelist.xml, but we refer to signconf.xml if we speak about the general signer configuration file. The actual location of these files can be found in zonelist.xml.

Date/time durations are as specified [here](#).

On this Page

- [Signer Configuration File](#)
 - [Preamble](#)
 - [Configuration per zone](#)
 - [Signatures](#)
 - [Authenticated Denial of Existence](#)
 - [Key Information](#)
 - [Source of Authority](#)

Signer Configuration File

Preamble

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: signconf.xml.in 3061 2010-03-16 19:23:51Z jakob $ -->
```

Each XML file starts with a standard element "<?xml...". As with any XML file, comments are included between the delimiters "<!--" and "-->".

Configuration per zone

```
<SignerConfiguration>
```

The enclosing element of the XML file is the element <SignerConfiguration?> which, with the closing element </SignerConfiguration>, brackets one signer configuration.

```
...
<Zone name="opendnssec.org">
```

Each zone is included in the <Zone>...</Zone> elements. Each zone has a "name" attribute giving the name of the zone.

Signatures

The next section of the file is the Signatures section, which lists the parameters for the signatures created using the policy. This is directly copied from the [KASP policy file](#).

```
...
<Signatures>
  <Resign>PT2H</Resign>
  <Refresh>P3D</Refresh>
  <Validity>
    <Default>P7D</Default>
    <Denial>P7D</Denial>
  </Validity>
  <Jitter>PT12H</Jitter>
  <InceptionOffset>PT300S</InceptionOffset>
</Signatures>
```

For more information on the meaning of the elements and their elements, take a look at [KASP policy file](#).

Authenticated Denial of Existence

Authenticated denial of existence - proving that domain names do not exist in the zone - is handled by the <Denial> section, as shown below:

<Denial> includes one element, either <NSEC3> (as shown above) or <NSEC>.

NSEC3

- <NSEC3> tells the signer to implement NSEC3 scheme for authenticated denial of existence (described in [RFC 5155](#)). The elements are:
- <OptOut/> - if present, enable "opt out". This is an optimisation that means that NSEC3 records are only created for authoritative data or for secure delegations; insecure delegations have no NSEC3 records. For zones where a majority of the entries are delegations that are not signed - typically TLDs during the take-up phase of DNSSEC - this reduces the number of DNSSEC records in the zone.
- <Resalt> is the interval between generating new salt values for the hashing algorithm.
- <Algorithm>, <Iterations> and <Salt> are parameters to the hash algorithm, described in [RFC 5155](#).

NSEC

Should, instead, NSEC be used as the authenticated denial of existence scheme, the <Denial> element will contain the single element <NSEC/> - there are no other parameters.

Key Information

Parameters relating to keys can be found in the <Keys> section.

Common Parameters

The section starts with a common parameter, TTL:

```
...
<TTL>PT3600S</TTL>
```

<TTL> is the time-to-live value for the DNSKEY resource records.

Keys

Each key has a number of elements so the signer knows how the keyset should be used and published.

Those are held in the <Key> section:

```
...
<Key>
<Flags>257</Flags>
<Algorithm>5</Algorithm>
<Locator>DFE7265B783F418685380AA784C2F31D</Locator>
<Publish/>
<KSK/>
<ZSK/>
</Key>
```

- <Flags> tells the signer what value it should set on this key when publishing the corresponding DNSKEY resource record.
- <Algorithm> determines the algorithm used for the key (the numbers reserved for each algorithm can be found in the appropriate [IANA registry](#)).
- <Locator> stores the CKA_ID of the key, e.g. the location of the physical key.
- <KSK/> - if present, use key as KSK. If a key is used as KSK, it must sign the DNSKEY RRset. It should not sign other RRsets, unless the <ZSK/> element is also present.
- <ZSK/> - if present, use key as ZSK. If a key is used as ZSK, it must sign all other RRsets. It should not sign the DNSKEY RRset, unless the <KSK/> element is also present.
- <Publish/> - if present, publish the corresponding DNSKEY resource record in the zone. The Public Key RDATA should be retrieved from the HSM using the CKA_ID (<Locator>).

The keyset is closed with the tag:

```
...
</Keys>
```

Source of Authority

When automating DNSSEC, the SOA record needs to be adjusted from time to time. The necessary parameters can be found in the <SOA> section:

```
...
<SOA>
<TTL>PT3600S</TTL>
<Minimum>PT3600S</Minimum>
<Serial>unixtime</Serial>
</SOA>
```




These values will override values set for the SOA record in the input zone file.

The values are:

- <TTL> - TTL of the SOA record.
- <Minimum> - value for the SOA's MINIMUM RDATA element.
- <Serial> - the format of the serial number in the signed zone. This is one of:
 - counter - use an increasing counter (but use the serial from the unsigned zone if possible)
 - datecounter - use increasing counter in YYYYMMDDxx format (xx is incremented within each day)
 - unixtime - the serial number is set to the "Unix time" (seconds since 00:00 on 1 January 1970 (UTC)) at which the signer is run.
 - keep - keep the serial from the unsigned zone (do not resign unless it has been incremented)

Auditing

The zone will be audited before it is written to the signed directory, if the following tag is included.

```
...
<Audit />
```

This is the last section of the signconf specification, so the next element is the zone closing tag:

```
...
</Zone>
```

and the file is ended by closing the <SignerConfiguration> tag:

```
</SignerConfiguration>
```

Date Time durations

This section explains how timing is described in all of the configuration files.

All date/time durations in the configuration files are specified as defined by [ISO 8601](#).



Durations are represented by the format P[n]Y[n]M[n]DT[n]H[n]M[n]S. In these representations, the [n] is replaced by the value for each of the date and time elements that follow the [n]. Leading zeros are not required. The capital letters 'P', 'Y', 'M', 'W', 'D', 'T', 'H', 'M', and 'S' are designators for each of the date and time elements and are not replaced.

- P is the duration designator (historically called "period") placed at the start of the duration representation.
- Y is the year designator that follows the value for the number of years.
- M is the month designator that follows the value for the number of months.
- W is the week designator that follows the value for the number of weeks.
- D is the day designator that follows the value for the number of days.
- T is the time designator that precedes the time components of the representation.
- H is the hour designator that follows the value for the number of hours.
- M is the minute designator that follows the value for the number of minutes.
- S is the second designator that follows the value for the number of seconds.

For example, "P3Y6M4DT12H30M5S" represents a duration of "three years, six months, four days, twelve hours, thirty minutes, and five seconds". Date and time elements including their designator may be omitted if their value is zero, and lower order elements may also be omitted for reduced precision. For example, "P23DT23H" and "P4Y" are both acceptable duration representations.



Exception: A year or month vary in duration depending on the current date. For OpenDNSSEC, we assume fixed values

- One month is assumed to be 31 days.
- One year is assumed to be 365 days.

Zone content

OpenDNSSEC can handle various formatting of the zone file, including different directives and Resource Records (RRs).

On this Page

- [Formatting](#)
- [Supported Directives](#)
- [RR types](#)
 - [Handling of unknown RR types](#)

Formatting

The zone file can be formatted in many ways including multi-lined RR, comments, etc.

Supported Directives

As defined in RFC 1035 the following directives are supported by OpenDNSSEC:

\$ORIGIN <code>example.com.</code>	What origin to use.
\$TTL <code>1h3m</code>	The default TTL to use. Treated as seconds, if no suffix is used: s, m, h, d, w, S, M, H, D, W
\$INCLUDE <code><path></code>	Include a file from a given path



Although OpenDNSSEC makes a copy of the unsigned zone file, it does not copy files included in the zone file. It is therefore recommended that you use absolute paths for included files and, if you make use of the auditor, that you don't edit them when a sign operation is going on.

RR types

OpenDNSSEC support all of the RR specified by [IANA](#), with some exceptions:

Not supported	ATMA, APL, EID, NIMLOC, HIP, SINK, NINFO, RKEY, TA
Obsoleted	MD, MF, WKS, GPOS, SIG, KEY, NXT, A6, and NSAP-PTR
Not allowed in master	NULL, OPT, TKEY, TSIG, IXFR, AXFR, MAILB, MAILA, *

Handling of unknown RR types

But OpenDNSSEC does handle unknown RR types in accordance with [RFC3597](#) e.g:

```
example.com.      IN              TYPE1              # 4 0A000001
```

Migrating to OpenDNSSEC

It is possible to migrate a DNSSEC signed zone over to OpenDNSSEC. How to migrate your DNSSEC signed zone over to OpenDNSSEC really depends on how your current solution looks like.

The zone data is no problem. Just place a copy of the unsigned zone in the directory for unsigned zones. But the trick is to maintain the private and public keys.

When moving from one system to another, you need to exchange public keys between them in order to always have a valid DNSSEC state. There are three possible solutions:

- Export the keys
- Prepublish DNSKEY record
- Start fresh

On this Page

- [Export the keys](#)
 - [On disc](#)
 - [On a smartcard with no PKCS#11 interface](#)
 - [On an HSM](#)
 - [Add the keys to OpenDNSSEC](#)
- [Prepublish DNSKEY record](#)
- [Start fresh](#)

Export the keys

One solution is to move the key pairs and make them accessible by OpenDNSSEC. The goal is to have the key pairs available to the system using PKCS#11.

The key pairs can e.g. be stored:

- on disc (e.g. BIND .private-key format)
- on a smartcard with no PKCS#11 interface
- in an HSM

On disc

When the key pairs are stored on disc, it means that you have access to files containing the key pairs. The key pairs can be imported into your new HSM using the PKCS#11 API or any tool available from your HSM vendor.

The BIND .private-key file can be converted into the PKCS#8 file format using the tool available with SoftHSM. If you have another file format, then OpenSSL probably can help you to convert it into the PKCS#8 file format.

```
softhsm-keyconv --topkcs8 --in Kexample.com.+005+42952.private --out key.pem
```

- **--topkcs8**, To indicate that you want to convert from BIND .private-key format to PKCS#8.
- **--in <path>**, The path to the BIND .private-key file.
- **--out <path>**, A path to the temporary PKCS#8 file.

The PKCS#8 file can then be imported into the SoftHSM token (if you are using SoftHSM as your HSM).

```
softhsm --import key.pem --slot 1 --pin 123456 --label A2 --id A2
```

- **--import <path>**, The path to the PKCS#8 file that you want to import. This should point to the temporary file that you created in the previous step.
- **--slot <number>**, The key should be imported to a token. Indicate which slot it is connected to.
- **--pin <PIN>**, Provide the PIN so that we can login to the token.
- **--label <text>**, Choose an arbitrary text string. Not used by OpenDNSSEC.
- **--id <hex>**, Choose an ID of the new key pair. The ID is in hexadecimal with a variable length. It must not collide with an existing key pair.

On a smartcard with no PKCS#11 interface

Just connect a smartcard reader to your system and insert your smartcard. Then use **opencsc** and **pcscd** to give it a PKCS#11 interface. Remember to protect the location where you have your smartcard reader, since the smartcard needs to be online.

On an HSM

You can either move the HSM to the new server and install it there. Or some vendors may have some functionality to export/transfer the key pairs.

Add the keys to OpenDNSSEC

Once you have the key pairs available on the system via PKCS#11, then you must add them to OpenDNSSEC. Give this command before

you start OpenDNSSEC. Also make sure that the zone is properly configured with OpenDNSSEC.

```
ods-ksmutil key import --cka_id <CKA_ID> --repository <repository> --zone <zone> --bits <size>
--algorithm <algorithm> --keystate <state> --keytype <type> --time <time>
```

- **--cka_id <CKA_ID>**, Each key object in the HSM has an ID, the CKA_ID attribute. The private and public key object must have the same ID in order for OpenDNSSEC to find them. The CKA_ID of the key pair to import is indicated, in hexadecimal, by using this option. E.g. *A2*
- **--repository <repository>**, The name of the repository, from conf.xml. E.g. *softHSM1*
- **--zone <zone>**, The name of the zone. E.g. *example.com*
- **--bits <size>**, The key length, E.g. *1024*
- **--algorithm <algorithm>**, The algorithm. E.g. *5* or *7*
- **--keystate <state>**, The key state. E.g. *active* or *ready*
- **--keytype <type>**, The key type. *KSK* or *ZSK*
- **--time <time>**, The time stamp when the key entered the given state. So that OpenDNSSEC know when to change the state. E.g. *200910301000*
- **--retire <retire>**, Optional. If you set the state to active, then you may set the time when the key should be retired. E.g. *201001010000*. Otherwise will OpenDNSSEC use the key lifetime from the KASP.

The difference between active and ready is:

- **active**: Will make the key active, thus used for signing. If there already is an active key, then you will have two of them. If this is not desired, then make sure to give this command right after setup and before you start the system.
- **ready**: The key will only be published in the zone. It will become active in a future rollover, if the key parameters match the policy.

Prepublish DNSKEY record

When moving from one system to another, you need to exchange public keys between them in order to always have a valid DNSSEC state.

The steps below will perform a manual Double-DS KSK rollover and a manual Pre-Publication ZSK rollover. There must be a period of time between each step and the system rollover; otherwise there will not be sufficient time for the information to propagate out on the Internet. The exact time depends on your setup, but it is typically between two and four weeks. Read more in the DNSSEC Key Timing draft from IETF.

1. Before the system rollover you need to:
 - Extract the DS corresponding to the KSK in the new system and publish it in the parent zone.
 - Publish the new ZSK in the old system.
 - Publish the old ZSK in the new system.
2. System rollover:
 - Re-delegate the zone in the parent zone.
3. After the system rollover you need to:
 - Remove the old DS from the parent zone.
 - Remove the new ZSK from the old system.
 - Remove the old ZSK from the new system.

Start fresh

A third solution is to start fresh. Remove any DS records from the parent zone. Stop signing your zone when the DS records are removed from the DNS caches. It is safe to remove the public keys from your zone when the signatures are not present in any DNS caches. Then transfer the zone over to OpenDNSSEC. And let OpenDNSSEC start signing it again.



Your zone will not be secured by DNSSEC during this transfer.

Running OpenDNSSEC

This section describes how to start OpenDNSSEC and the operations used to manage and monitor the system.

The details of the command utilities shown below can be found [here](#).

On this Page

- [Before starting OpenDNSSEC for the first time](#)
- [Starting / Stopping the system](#)
- [Uploading a Trust Anchor](#)
- [Key Management](#)
- [Zone Management](#)
- [Updating the KASP policy](#)
- [Logging](#)



All directories are prepared by the build script and are set to be owned by root, so all commands in the default configuration must also be run by root. To change this, alter the configuration or privileges on the files and directories.

Before starting OpenDNSSEC for the first time

Before you run the system for the first time you must import your policy and zone list into the database using the following command:

```
ods-ksmutil setup
```

After running this the first time, you will be ready to run OpenDNSSEC with an empty set of zones. On the other hand, if this command is run on an existing database, then will all meta-information about the zones be lost. The keys would then still exist in HSMs, so you should not forget to clean them up.

Starting / Stopping the system

OpenDNSSEC consist of two daemons, *ods-signerd* and *ods-enforcerd*. To start and stop them use the following commands:

```
ods-control start
```

A proper-looking response to this commands is:

```
Starting enforcer...
OpenDNSSEC ods-enforcerd started (version 1.2.0b1), pid 11424
Starting signer engine...
OpenDNSSEC signer engine version 1.2.0b1
```

At any time, you can stop OpenDNSSEC's daemons orderly with:

```
ods-control stop
```

After this, your logs should contain messages like:

```
Stopping enforcer...
Stopping signer engine..
Engine shut down.
```

Uploading a Trust Anchor

Your zone will be signed, once you have setup the system and started it. When you have verified that the zone is operational and working, it is time to upload the trust anchor to the parent zone. The Enforcer is waiting for zone to be connected to the trust chain before considering the KSK to be active.

```
ods-ksmutil key list --verbose
```

```
Keys:
Zone:                               Keytype:   State:    Date of next transition:  CKA_ID:
Repository:                          Keytag:
example.com                           ZSK        active    2010-10-15 06:59:28
92abca348b96aaef42b5bb62c8daffb0     softHSM2   28743
example.com                           KSK        ready     waiting for ds-seen
9621ca39306ce050e8dd94c5ab837001     softHSM1   22499
```

1. Export the public key either as DNSKEY or DS, depending on what format your parent zone wants it in. See the section below, *Export the public keys*, on how to get the key information.
2. Notify the Enforcer when you can see the DS RR in your parent zone. You usually give the keytag to the Enforcer, but if there are KSKs with the same keytag then use the CKA_ID.

```
ods-ksmutil key ds-seen -z example.com -x 22499
```

or

```
ods-ksmutil key ds-seen -z example.com -k 9621ca39306ce050e8dd94c5ab837001
```

```
Result:
Found key with CKA_ID 9621ca39306ce050e8dd94c5ab837001
Key 9621ca39306ce050e8dd94c5ab837001 made active
```

And you will see that your KSK is now active:

```
ods-ksmutil key list

Keys:
Zone:                               Keytype:   State:    Date of next transition:
example.com                           ZSK        active    2010-10-15 07:20:53
example.com                           KSK        active    2010-10-15 07:31:03
```

Key Management

The details of common key management activities are described on the [Key Management](#) page - these include:

- Configuring the system to only make keys active once they have been backed up.
- Exporting public keys.
- Performing manual key rollovers.

Zone Management

The details of common zone management activities are described on the [Zone Management](#) page - these include:

- Adding / Removing zones
- Updating an unsigned zone

Updating the KASP policy

When you make changes to a policy or add a new policy in kasp.xml you must update the changes to the database.

```
ods-ksmutil update kasp
```

Logging

Details of logs produced by the system can be found on the [Logging](#) page.

Key Management

This sections describes common key management activities of OpenDNSSEC.

The details of the command utilities shown below can be found [here](#).

On this Page

- [Marking keys as backed up](#)
- [Export the public keys](#)
- [Key rollovers](#)
 - [First step](#)
 - [Second step for KSKs](#)
 - [Key rollovers on exact dates](#)

Marking keys as backed up

You can configure the system to only make keys active once they have been backed up. This is done by editing the `conf.xml` file. The user must do backups and then notify OpenDNSSEC about this, so that the key rollover process can continue. The keys must be backed up regularly, because OpenDNSSEC is generating new keys prior to a rollover.

1. First prepare the backup by telling the Enforcer that you want to do backup of the keys. This is so that keys generated after you have done your backup won't accidentally be marked as backed up.

For all of the repositories:

```
ods-ksmutil backup prepare
```

or a single repository:

```
ods-ksmutil backup prepare --repository <repository>
```

2. Then you can safely do your backups. **Please read the documentation of your HSM for instructions on how to do backups.** When you are done, then notify the Enforcer about this:

For all of the repositories:

```
ods-ksmutil backup commit
```

or a single repository:

```
ods-ksmutil backup commit --repository <repository>
```



The command `ods-ksmutil backup done` will mark your keys as backed up in one step. This means that keys may have been generated between you doing the backup and giving the command. Thus accidentally marking them as backed up. This command is deprecated and should not be used, unless you make sure to stop the Enforcer when doing your backup.



Backups are specified by way of a repository option in `conf.xml`:

```
<RequireBackup/>
```

If you decide you want to change this facility, you should edit `conf.xml` accordingly, and run:

```
ods-ksmutil update conf
```

It will report something along the lines of:

```
RequireBackup set.
```

Export the public keys

You need to publish your key to the parent or to interested parties. If you are doing a key rollover, then only the ready KSK should be exported. The following command will extract the trust anchors:

```
ods-ksmutil key export --zone example.com [--keystate READY]
ods-ksmutil key export --zone example.com --ds [--keystate READY]
```

What you get in return is the DNSKEY or DS in BIND-format.

Key rollovers

First step



The rollovers are done automatically according to the policy of the zone. But a **manual** keyrollover may be desired in cases of emergency, such as having lost a private key.

This can be done using the `ods-ksmutil` command like this:

```
ods-ksmutil key rollover --zone example.com --keytype KSK
```

This will roll the KSK key in a timely manner following the policy used for the zone `example.com`. If you want to roll the Zone Signing Key use `--keytype ZSK` instead.

You can also roll all the keys for zones which have a certain policy. This can be useful if you want to move all keys from one key store to another.

```
ods-ksmutil key rollover --policy default --keytype KSK
```



This step is not needed for a scheduled rollover.

Second step for KSKs

Unlike ZSKs, a KSK rollover requires a second step involving manual intervention. This intervention is a multi-stage process. First, the DNSKEY record for the new key is added to the zone. Then, after a suitable interval, the new DS record is submitted to the parent; at this point the old DS record can be removed from the parent.

The stages are:

1. Extract the DNSKEY record for the new key and publish it in the parent zone. (The new record replace any existing records for the zone being signed.) When it is time for this to happen a message with log-level "info" will be sent to syslog looking something like:


```

Mar 16 11:39:05 sion ods-enforcerd: DS Record set has changed, the current set looks like:
Mar 16 11:39:05 sion ods-enforcerd: example.com. 3600 IN DNSKEY 257 3 7
AwEAAbcTSmphJUMKvegvDggGspRM8IHlKZqoU5pkPaTtRLkioxGyZ5iIh4bNnvqmx1zWIttuJ6erGUMoatMm3SXxiTr90La
;{id = 51994 (ksk), size = 2048b}
Mar 16 11:39:05 sion ods-enforcerd: Once the new DS records are seen in DNS please issue the
ds-seen command for zone example.com with the following cka_ids,
04260cd6eac67280cd2dea94c6e38cb7

```

The DNSKEY or DS RR can also be retrieved by using the commands in the section *Export the public keys*.

This step can be automated or semi-automated by placing a command in the <DelegationSignerSubmitCommand> tag. This should point to a binary which will accept the required key(s) on STDIN.

- When the records indicated have been seen in DNS then this can be communicated to OpenDNSSEC with the ds-seen command as indicated:

```
ods-ksmutil key ds-seen --zone example.com --cka_id 04260cd6eac67280cd2dea94c6e38cb7
```

- If the DS records were not swapped, i.e. the old DS was left in the parent when the new one was added, then the --no-retire flag can be added to the ds-seen command. Then, at some later time, the old key can be retired with the command:

```
ods-ksmutil key ksk-retire --zone example.com ---cka_id 87f1385b114f9f9b299e6b551d728bfb
```

or

```
ods-ksmutil key ksk-retire --zone example.com
```

The former command will retire the specific key (provided the key is active, and the action will not leave the zone without any active keys). The latter command will retire the oldest active key on the zone, again provided it will not leave the zone without any active keys.



If you wish to run like this and use the DelegationSignerSubmitCommand hook then you will need to add the current key back into the set yourself.

Key rollovers on exact dates

Some users want to have more control over their key rollovers and roll keys on exact dates, for example the first day of each month. To do this you need to specify that you want manual key rollovers in the kasp.xml configuration. Add the <ManualRollover/> tag to the type and key you want to roll manually.

When this is done you can add the rollover commands to a cron job, with a command like this:

```
ods-ksmutil key rollover --zone example.com --keytype ZSK
```

(The need to manually time intervals is a limitation of version 1.2 of the software. Future versions of OpenDNSSEC will prompt with reminders at the appropriate times as well as offering alternative KSK rollover methods.)

Zone Management

This section describes common Zone Management activities in OpenDNSSEC.

The details of the command utilities shown below can be found [here](#).

On this Page

- [Adding / Removing zones](#)
- [Updating an unsigned zone](#)

Adding / Removing zones

Zones can be added and removed at will. If the optional parameters are not given, then it will default to the policy *default* and the */var/opendnssec/* subdirectories.

```
ods-ksmutil zone add --zone example.com [--policy <policy> --signerconf <signerconf.xml> --input <input> --output <output>]
ods-ksmutil zone delete --zone example.com
```

This command will report positively with a message like:

```
zonelist filename set to /etc/opendnssec/zonelist.xml.
SQLite database set to: /var/opendnssec/kasp.db
Imported zone: example.com
```



Using this command thousands of times might be slow since it also writes to *zonelist.xml*. Use *--no-xml* to stop this behavior. Then export the zonelist when you are finished:

```
ods-ksmutil zonelist export > zonelist.xml
```

Alternatively, you could manually edit the *zonelist.xml* and then give the command:

```
ods-ksmutil update zonelist
```

After zones are added, they will show up in your logs as follows:

```
ods-enforcerd: Zone example.com found.
ods-enforcerd: Policy for example.com set to default.
ods-enforcerd: Config will be output to /var/opendnssec/signconf/example.com.xml.
```

If you opened the latter file, you would find the settings that were applied to the zone at the time this file was added.

Updating an unsigned zone

When you update the content of an unsigned zone you must tell the signer engine to re-read the unsigned zone file using the *ods-signer* command like this:

```
ods-signer sign example.com
```

This also have the effect that you schedule the zone for immediate resigning.

Logging

Currently all logging is handled by syslog. Other logging methods may come later.

There is a lot of output from the daemons of which a lot of things right now are for debugging. We might reduce the amount of logging for later versions of OpenDNSSEC.

The signer outputs some statistics into the logs:

```
[STATS].opendnssec.org RR[count=32 time=1(sec)] NSEC[count=32 time=1(sec)] RRSIG[new=1 reused=31
time=1(sec) avg=1(sig/sec)] AUDIT[time=2(sec)] TOTAL[time=5(sec)]
```

The RR[...] block says something about reading the unsigned zone file. The line above tells us that there have been 32 RRs read and it took about one second. The count will be 0 on lines where only re-signing occurred.

The NSEC[...] block says something about the number of NSEC or NSEC3 records created on the latest run. This count will also be 0 on lines where only re-signing occurred.

The RRSIG[...] block says something about the number of created signatures. In the above example, the last time there was only created one new signature, and 31 other signatures were reused. The re-signing was finished in about a second and thus the average number of signatures created per second is in this example 1.

The AUDIT[...] block tells us how long the auditor needed to check upon the zone.

The TOTAL[...] block tells us how long the re-signing took, including reading the unsigned zone and adding NSEC(3) records, if applicable.

Command Utilities

This section details the various command utilities that are available with OpenDNSSEC.

On this Page

- [ods-control](#)
- [ods-ksmutil](#)
- [ods-signer](#)
- [ods-hsmutil](#)
- [ods-auditor](#)
- [ods-hsmspeed](#)
- [ods-kaspcheck](#)
- [hsmbully](#)
- [Daemons](#)
 - [ods-signerd](#)
 - [ods-enforcerd](#)

ods-control

Is a wrapper around the commands below.

```
usage: ods-control ksm|hsm|signer|start|stop
```

- The first three options pipe commands to *ods-ksmutil*, *ods-hsmutil*, and *ods-signer*.
- The last two options start and stop the two daemons of OpenDNSSEC, *ods-enforcerd* and *ods-signerd*.

ods-ksmutil

You need a way to interact to the KASP Enforcer, for example to add and remove zones that are handled by OpenDNSSEC. The *ods-ksmutil* utility provides a number of commands to make this easier, all commands are invoked on the unix command line.

- **You must run the *setup* option before you ever run any sub-system in OpenDNSSEC.** This reads the configuration *kasp.xml* and imports these settings into the KASP Enforcer database.



The *setup* command deletes the current content of the database! (Including information on keys; such that existing keys will become unusable and new keys will need to be generated.)

- If you make any changes to *kasp.xml* these changes must be imported into the database. Use the *update* command to do this without losing any other data.

- To add a zone to be handled by OpenDNSSEC, use the `zone add` command. This command needs a parameter to specify the zone, and optional parameters for which policy to use and which paths to use for input and output. An example of use:

```
ods-ksmutil zone add -z example.com -p default -i /var/example.com -o /var/example.com.signed
```

- The command `zone delete` is simpler and needs no further parameters but the name of the zone.

A complete list of commands can be found by running:

```
ods-ksmutil -h
```

or they are shown in detail here: [ods-ksmutil commands](#)

ods-signer

The `ods-signer` provides a Command Line Interface to the `ods-signerd`. There are a number of commands you give to `ods-signer`. If you start the CLI without any command line parameters you enter a shell where you can issue commands:

```
ods-signer
cmd> help
Commands:
zones          show the currently known zones
sign <zone>   read zone and schedule zone for immediate (re-)signing
sign --all    read all zones and schedule all for immediate (re-)signing.
clear <zone>  delete the internal storage of this zone.
               All signatures will be regenerated on the next re-sign.
queue         show the current task queue.
flush        execute all scheduled tasks immediately.
update <zone> update this zone signer configurations.
update [--all] update zone list and all signer configurations.
start        start the engine.
reload       reload the engine.
stop         stop the engine.
verbosity <nr> set verbosity.
```

The same commands can be passed as command line arguments in your unix shell.

ods-hsmutil

The `ods-hsmutil` utility is designed to interact directly with your HSM and can be used to manually list, create or delete keys. It can also be used to perform a set of basics HSM tests.



Be careful before create or deleting keys using `ods-hsmutil`, as the changes are **not** synced with the KASP Enforcer.

ods-auditor

The Auditor (`ods-auditor`) can do an audit of the zones in the system to see if the signer complies to what the policy mandates. It is run automatically (unless disabled) after each resigning of a zone and will stop the signed zone from being distributed if it finds any issues. Any errors found by the `ods-auditor` will be logged to the configured syslog utility. This should be checked for debug if you have issues.

You can also run the Auditor yourself, to get feedback on the current status, to loop through all zones run:

```
ods-auditor
```

or, to audit just one zone, run:

```
ods-auditor -z <zone>
```

It is possible to override the audit type specified in the `kasp.xml` Policy for the zone. To run a full audit, use the `--full` flag, and use `--partial` to force a partial audit of the zone.



If you are using the partial auditor to audit your very large zone, you may wish to run an occasional off-line full audit. To do this, take a copy of your signed and unsigned zone files, and run :

```
ods-auditor -z <zone> --full --signed <path/to/signed/file> --unsigned <path/to/unsigned/file>
```

ods-hsmspeed

The tool `ods-hsmspeed` does performance testing on your HSM. This is also useful to find out at what speed you can get from SoftHSM on your CPU.

ods-kaspcheck

This tool is provided to check that the configuration files (`conf.xml` and `kasp.xml`) are semantically sane and contain no inconsistencies.



It is advisable to use this tool to check your configuration before starting to use OpenDNSSEC.

```
ods-kaspcheck -h
Usage: ods-kaspcheck options
Specific options:
  -c, --conf PATH_TO_CONF_FILE Path to OpenDNSSEC configuration file
      (defaults to the default conf.xml file)
  -k, --kasp PATH_TO_KASP_FILE Path to KASP policy file
      (defaults to the path given in the configuration file)
Common options:
  -h, -?, --help          Show this message
```

hsmbully

The `hsmbully` tool may be used to test your HSM for compliance with PKCS#11. This tool is not part of OpenDNSSEC, but can be found in the SVN repository:

```
svn co http://trac.opendnssec.org/browser/trunk/hsmbully hsmbully
```

Daemons

You can also run the two OpenDNSSEC daemons `ods-signerd` and `ods-enforcerd` from the command line, they are installed into the `sbin` directory.

ods-signerd

This is the component that performs all of the signing. It first reads `zonelist.xml` and then goes through all zones to sign them if needed. Start the daemon by running:

```
ods-signer start
```

or if you want to use specific command line options:

```
ods-signerd -h
Usage: ods-signerd [OPTIONS]
Start the OpenDNSSEC signer engine daemon.

Supported options:
-c | --config <cfgfile> Read configuration from file.
-d | --no-daemon        Do not daemonize the signer engine.
-l | --single-run       Run once, then exit.
-h | --help             Show this help and exit.
-i | --info             Print configuration and exit.
-v | --verbose          Increase verbosity.
-V | --version          Show version and exit.

BSD licensed, see LICENSE in source package for details.
Version 1.1.0-trunk. Report bugs to <http://trac.opendnssec.org/newticket>.
```

ods-enforcerd

The Enforcer daemon creates keys if needed (and configured to); it also maintains the states of the keys according to the appropriate policy. As the states of keys change, it communicates these changes to the signer via the configuration files that the signer uses when signing the zones. To run, call:

```
ods-enforcerd
```

ods-kmutil

This is a utility that allows several different actions to be performed (relatively) easily:

On this Page

- Global Options
- Command: setup
- Command: update
- Command: zone add
- Command: zone delete
- Command: zone list
- Command: repository list
- Command: policy export
- Command: policy list
- Command: key list
- Command: key export
- Command: key import
- Command: key rollover
- Command: key purge
- Command: key generate
- Command: key ds-seen
- Command: key ksk-retire
- Command: backup done
- Command: backup list
- Command: database backup
- Command: rollover list
- Allowed values

Global Options

```
--config <config>      aka -c
```

Change the conf.xml file that is used, from the default.

Command: setup

```
ods-ksmutil setup
```

Import conf.xml, kasp.xml and zonelist.xml into a database (deletes current contents, including any keys).

Command: update

```
ods-ksmutil update kasp
ods-ksmutil update zonelist
ods-ksmutil update conf
ods-ksmutil update all
```

Update database from config_dir (like above, but existing contents are kept)

Command: zone add

```
ods-ksmutil zone add
```

Add a zone to both zonelist.xml and the database (both locations read from conf.xml).

Options

```
--zone <zone>                aka -z
[--policy <policy>]          aka -p
[--signerconf <signerconf.xml>] aka -s
[--input <input>]            aka -i
[--output <output>]          aka -o
```

Defaults are provided for all options but zone name.

Command: zone delete

```
ods-ksmutil zone delete
```

Delete a zone to both zonelist.xml and the database (both locations read from conf.xml).

Options

```
--zone <zone> | --all        aka -z / -a
```

Command: zone list

```
ods-ksmutil zone list
```

List zones from the zonelist.xml

Command: repository list

```
ods-ksmutil repository list
```

List repositories from the database

Command: policy export

```
ods-ksmutil policy export
```

Export a policy from the database in kasp.xml format.

Options

```
--policy <policy> | --all      aka -p / -a
```

Command: policy list

```
ods-ksmutil policy list
```

List policies available.

Command: key list

```
ods-ksmutil key list
```

List information about keys in zone.

Options

```
[--verbose]
--zone <zone> | --all      aka -z / -a
(will appear soon:
[--keystate <state>]      aka -e
[--keytype <type>]       aka -t
[--ds]                    aka -d   )
```

Command: key export

```
ods-ksmutil key export
```

Export key information in a suitable format for putting into a zonefile

Options

```
--zone <zone> | --all      aka -z
[--keystate <state>]      aka -e
[--keytype <type>]       aka -t
[--ds]                    aka -d
```

Command: key import

```
ods-ksmutil key import
```

Add a key which was created outside of the OpenDNSSEC code into the database

Options


```

--cka_id <CKA_ID>          aka -k
--repository <repository>  aka -r
--zone <zone>              aka -z
--bits <size>              aka -b
--algorithm <algorithm>    aka -g
--keystate <state>         aka -e
--keytype <type>           aka -t
--time <time>              aka -w
[--retire <retire>]        aka -y

```

Command: key rollover

```
ods-ksmutil key rollover
```

Rollover active keys on a zone or policy

Options

```

--zone <zone> / --policy <policy>
[--keytype <type>]

```

"keytype" specifies the type of key to roll (both are rolled if nothing is specified) After running, the enforcer will be woken up so that the signer can be sent the new information

If the policy that the zone is on specifies that keys are shared then all zones on that policy will be rolled. A backup of the sqlite DB file is made (if appropriate).

Command: key purge

```
ods-ksmutil key purge
```

Remove keys that are in the "Dead" state from the repository and from the enforcer DB

Options

```

--zone <zone> / --policy <policy>      aka -z / -p

```

Command: key generate

```
ods-ksmutil key generate
```

Create enough keys for the given policy to last for the period of time given by interval.

Options

```

--policy <policy>          aka -p
--interval <interval>     aka -n

```

Intervals are specified in the format used in the configuration files, see [Configuration](#).

Command: key ds-seen

```
ods-ksmutil key ds-seen
```

Indicate that a submitted DS record has appeared in the parent zone (this triggers the completion of a KSK rollover, or the provisioning of a standby KSK).

Options

```
[--zone <zone>                aka -z]
--keytag <keytag>             aka -x
--cka_id <CKA_ID>             aka -k
[--no-retire]
```

Specifying a zone will speed up the search of keys by narrowing the field but is not mandatory; `cka_id` can be used to resolve a keytag clash. By default the command will simultaneously move the current key into the retired state. If you wish to delay this step then add the `--no-retire` flag and use the `ksk-retire` command when needed.

Command: key ksk-retire

```
ods-ksmutil key ksk-retire
```

Move a key from active to retired (if a replacement key is already active).

Options

```
--zone <zone>                aka -z
--keytag <keytag>           aka -x
--cka_id <CKA_ID>          aka -k
```

Specifying a zone alone will retire the oldest key in the zone; if the `cka_id` or `keytag` are specified then that key will be retired. The specified key must be in the active state, and there must be 2 or more active keys on the zone for this command to work.

Command: backup done

```
ods-ksmutil backup done
```

Indicate that a backup of the given repository has been done, all non-backed up keys will now be marked as backed up.

This is especially important if the repository used has the **RequireBackup** flag set.

NOTE: Keys generated between a backup being made and the backup done command being run will be erroneously marked as having been backed up. To avoid this, either choose a backup schedule that doesn't run while the enforcer might be generating keys, or shutdown the enforcer while a backup is performed.

Options

```
--repository <repository>   aka -r
```

(If no options are given then all repositories are marked as backed up.) Include this call in a HSM backup process to avoid warnings or errors about using non-backed up keys.

Command: backup list

```
ods-ksmutil backup list
```

List the backups that have been made on the given repository.

Options

```
--repository <repository>      aka -r
```

Command: database backup

```
ods-ksmutil database backup
```

Make a copy of the enforcer database (if using sqlite). It makes sure that the database is in a consistent state by taking a lock out first.

Options

```
[--output <output>]           aka -o
```

If `--output` is omitted then the usual `enforcer.db.backup` is used.

Command: rollover list

```
ods-ksmutil rollover list
```

List the expected dates and times of upcoming rollovers.

Allowed values

When specifying a keystate the following keywords are recognised:

```
GENERATED|PUBLISHED|READY|ACTIVE|RETIRED|REVOKED|DEAD
```

When specifying a keytype the following keywords are recognised:

```
KSK|ZSK
```

When specifying a time (for **key import**) the following formats can be used:

```
YYYYMMDD[HH[MM[SS]]]          (all numeric)

or D-MMM-YYYY[:| ]HH[:MM[:SS]] (alphabetic month)
or DD-MMM-YYYY[:| ]HH[:MM[:SS]] (alphabetic month)
or YYYY-MMM-DD[:| ]HH[:MM[:SS]] (alphabetic month)

D-MM-YYYY[:| ]HH[:MM[:SS]]     (numeric month)
DD-MM-YYYY[:| ]HH[:MM[:SS]]    (numeric month)
or YYYY-MM-DD[:| ]HH[:MM[:SS]] (numeric month)
```

... and the distinction between them is given by the location of the hyphens.

Troubleshooting

There are a number of common issues that are straightforward to diagnose and fix... If OpenDNSSEC is not behaving as expected then the first place to look is in the logs. Where these will be depends on your system and your configuration.

The following are log messages which you may see, and what to do about them (if anything).

On this Page

- [Enforcer](#)
- [Signer](#)

Enforcer

ods-enforcerd: ERROR: Trying to make non-backed up ZSK active when RequireBackup flag is set

This is not an error as such. It means that in conf.xml you have indicated that keys should not be used unless they are backed up. However, the enforcer has determined that if it continues then a non backed up key will be made active.

The solution Take a backup of your keys (how this is done will depend on your key storage).

Once this has been done then run **ods-ksmutil backup done** to mark all keys as having been backed up.

ods-enforcerd: WARNING: Making non-backed up KSK active, PLEASE make sure that you know the potential problems of using keys which are not recoverable

This is the same as above, but without **RequireBackup** being set in conf.xml

ods-enforcerd: WARNING: key rollover not completed as there are no keys in the ready state: ods-enforcerd will try again when it runs next

This is seen when a rollover is happening but there is no replacement key ready (because one has not been published for long enough). It indicates that the rollover will be delayed until the replacement key is ready, the time that this will happen depends on the policy.

ods-enforcerd: Could not call signer engine

If the enforcer makes a change to a zones signer configuration (say it adds a new key) it calls the signer to get it to resign that zone. This message indicates that the signer is not running, although it has been seen on a system where everything is working fine. (See `KNOWN_ISSUES`.)

ods-enforcerd: Not enough keys to satisfy zsk policy for zone or ods-enforcerd: Not enough keys to satisfy ksk policy for zone

One of these messages will be seen if the enforcer does not have enough unallocated keys to provide for the zone specified. If the **ManualKeyGeneration** tag is set in conf.xml then you will need to create new keys using **ods-ksmutil key generate**, otherwise new keys will be created when the enforcer runs next. (Don't forget to backup any new keys.)

ods-enforcerd: Rollover of KSK expected at <DATE TIME> for <ZONE>

This is not an error, but a notification of an upcoming (scheduled) rollover. This will appear in your logs at a time prior to the rollover as configured in conf.xml (the **Enforcer/RolloverNotification** tag).

ods-enforcerd: WARNING: KSK Retirement reached; please submit the new DS for <ZONE> and use ods-ksmutil key ksk-roll to roll the key.

Rolling a KSK requires the DS record of the replacement key to be published in the parent of the zone. This message indicates that your KSK has reached the end of its life (as specified by your policy), and that it is time to submit the DS record to the parent.

ods-enforcerd: Error: database in config file <path_to_conf.xml> does not match libksm

This indicates that either you have libksm built for sqlite, but have specified a MySQL database in conf.xml, or *vice versa*.

The solution is to either rebuild libksm or to change conf.xml

ods-enforcerd: Error reading config

This usually means that conf.xml is either absent, not readable by the user, or badly formed. There should be a line above this one which gives a more specific error message.

ods-enforcerd: Error getting db lock

When using sqlite any process using the database tries to get an exclusive write lock on a file in the same directory as the kasp.db. If this directory is not writeable by the user then this message will be seen, again a more specific error message should have been issued.

ods-enforcerd: Repository <NAME> is full, cannot create more <KSKs|ZSKs> for policy <POLICY>

In conf.xml a capacity can be specified for a repository. When this is reached then no more keys will be generated in that repository.

The solution is to either run **ods-ksmutil key purge** to remove dead keys, or to raise this capacity and run **ods-ksmutil update conf** to push this change into the database. If the repository is really at capacity, and purge does not free up any space, then a new repository will be needed.

ods-enforcerd: Repository <NAME> is nearly full, will create X <KSKs|ZSKs> for policy <POLICY> (reduced from Y)

Y keys were needed to satisfy the policy, but the repository only has room for X more. This warning might precede the error detailed above, and the solution is the same.

ods-enforcerd: NOTE: keys generated in repository <NAME> will not become active until they have been backed up

This is not an error, but a reminder that a backup needs to be done (as new keys have just been generated).

ods-enforcerd: Signconf not written for <ZONE>

Some error has happened and the enforcer will not overwrite the existing signconf file, so the old one will be left in place. There should be a more specific message indicating the root cause just prior to this line. (*E.g. attempting to use a non backed up key.*)

ods-enforcerd: There are no <KSKs|ZSKs> in the generate state; please use "ods-ksmutil key generate" to make some ManualKeyGeneration has been set (in conf.xml) and the system has run out of keys.

The solution is to run the **ods-ksmutil key generate** command, back up the keys, and the system will recover when it runs next.

Signer

These messages might show up in the logs if there is a parse or semantic error in one of the configuration files.

```
ods-signerd: error: unable to read cfgfile <file>
ods-signerd: error: unable to parse cfgfile <file>
ods-signerd: error: unable to read conf rng file <file>
ods-signerd: error: unable to create XML RelaxNGs parser context
ods-signerd: error: unable to parse a schema definition resource
ods-signerd: error: unable to create RelaxNGs validation context
ods-signerd: error: configuration file validation failed
ods-signerd: error: unable to create new XPath context for cfgfile <file>
ods-signerd: error: unable to evaluate required element <element> in cfgfile <file>
ods-signerd: error: cfgfile <file> has errors
ods-signerd: error: unable to evaluate xpath expression <expr>
ods-signerd: error: unable to open zone list file <file>
ods-signerd: error: unable to extract zone name from zonelist
ods-signerd: error: unable to read zone <dname>; skipping
ods-signerd: error: unable to add zone <zone> to zone list
ods-signerd: error: error parsing zone list file <file>
ods-signerd: error: invalid salt <salt>
ods-signerd: error: unable to parse signconf file <file>
ods-signerd: error: unable to read signconf file <file>
ods-signerd: error: signconf-check: no signature resign interval found
ods-signerd: error: signconf-check: no signature resign interval found
ods-signerd: error: signconf-check: no signature default validity found
ods-signerd: error: signconf-check: no signature denial validity found
ods-signerd: error: signconf-check: no signature jitter found
ods-signerd: error: signconf-check: no signature inception offset found
ods-signerd: error: signconf-check: no nsec3 algorithm found
ods-signerd: error: signconf-check: wrong nsec type <rrtype>
ods-signerd: error: signconf-check: no keys found
ods-signerd: error: signconf-check: no dnskey ttl found
ods-signerd: error: signconf-check: no soa ttl found
ods-signerd: error: signconf-check: no soa minimum found
ods-signerd: error: signconf-check: wrong soa serial type <string>
```

These messages might show up in the logs if the signer engine daemon was unable to start up. All of them are provided with a specific message indicating the cause.

```
ods-signerd: error: unable to create command handler, <reason>
ods-signerd: error: cannot connect to command handler: <reason>
ods-signerd: error: setup failed: chdir to <directory> failed: <reason>
ods-signerd: error: setup failed: unable to drop privileges
ods-signerd: error: setup failed: unable to fork daemon: <reason>
ods-signerd: error: setup failed: unable to setsid daemon: <reason>
ods-signerd: error: setup failed: unable to write pid file
ods-signerd: error: setup failed: unable to start command handler
ods-signerd: error: setup failed: unable to start command handler
ods-signerd: error: setup failed: error initializing libhsm (errno <no>)
ods-signerd: error: signer setup failed
ods-signerd: error: failed to fork zone fetcher: <reason>
ods-signerd: error: failed to setsid zone fetcher: <reason>
ods-signerd: error: cannot stop zone fetcher: <reason>
ods-signerd: error: cannot start zone fetcher
```

These messages might show up in the logs if the signer was unable to sign the zone

```
ods-signerd: error: task [read zone <dname>] failed
File not found or readable, parse error, ...
ods-signerd: error: task [add dnskeys to zone <dname>] failed
HSM re-initialized, no privileges for accessing HSM, ...
ods-signerd: error: task [update zone <dname>] failed
DNS related errors in zone, for example other RRs next to a CNAME
ods-signerd: error: task [nsecify zone <dname>] failed
ods-signerd: error: task [sign zone <dname>] failed
No privileges for accessing HSM, ...
```

```
ods-signerd: error: task [audit zone <dtype>] failed
Auditor found problems
ods-signerd: error: task [write zone <dtype>] failed
Output directory not writable
```

These messages might show up in the logs if a zone update failed

```
ods-signerd: error: cannot keep SOA SERIAL from input zone (<serial>): output SOA SERIAL is <serial>
<SOA><Serial> is set to keep in kasp policy file, but SOA SERIAL in unsigned zone file was not increased
ods-signerd: error: occluded (non-glue non-DS) data at <dtype> NS
Found unallowed RRs at the delegation
ods-signerd: error: occluded data at <dtype> (below <dtype> DNAME)
Found RRs below DNAME
ods-signerd: error: occluded (non-glue) data at <dtype> (below <dtype> NS)
Found non-glue RRs below delegation
ods-signerd: error: other data next to <dtype> CNAME
Found unallowed RRs next to CNAME
ods-signerd: error: multiple records for singleton type at <dtype> <rrtype>
Found multiple RRs of a singleton RRtype (CNAME or DNAME) at the same owner name
ods-signerd: error: update zone <dtype> failed: zone data contains errors
```

These messages might show up in the logs if one of the backup files was corrupted

```
ods-signerd: error: error creating DNSKEY for key <locator>
ods-signerd: error: error adding DNSKEY[<keytag>] for key <locator>
ods-signerd: error: error creating NSEC3 parameters for zone <dtype>
ods-signerd: error: error adding NSEC3PARAMS record to zone <dtype>
ods-signerd: error: error adding DNSKEYs to zone <dtype>
ods-signerd: error: error adding NSEC3PARAMS RR to zone <dtype>
ods-signerd: error: cannot backup zone: cannot open file <file> for writing
ods-signerd: error: error adding key from backup file <file> to key list
ods-signerd: error: error recovering DNSKEY[<keytag>] rr
ods-signerd: error: error recovering nsec3 parameters from file <file>
ods-signerd: error: error recovering NSEC3PARAMS rr
ods-signerd: error: error reading key credentials from backup
ods-signerd: error: error reading RRSIG from backup
ods-signerd: error: expecting RRtype RRSIG from backup
ods-signerd: error: error reading domain from backup file
ods-signerd: error: error adding domain from backup file
ods-signerd: error: error reading NSEC(3) RR from backup file
ods-signerd: error: error adding NSEC(3) RR from backup file
ods-signerd: error: unable to recover zone state from file <file>: <reason>
ods-signerd: error: unable to recover denial of existence from file <file>: <reason>
ods-signerd: error: unable to recover unsorted zone from file <file>: <reason>
ods-signerd: error: unable to recover dnskeys from file <file>: <reason>
ods-signerd: error: unable to recover rrsigs from file <file>: <reason>
ods-signerd: error: domain part in backup file is corrupted
ods-signerd: error: unable to recover RR to domain: failed to add RRset
ods-signerd: error: ods-signerd: error: unable to recover RRSIG to domain: no NSEC RRset
ods-signerd: error: unable to recover RRSIG to domain: no NSEC3 RRset
ods-signerd: error: unable to recover RRSIG to domain: no such RRset
ods-signerd: error: nsec3params part in backup file is corrupted
ods-signerd: error: key part in backup file is corrupted
ods-signerd: error: unable to recover signconf backup file <file>: corrupt
```



Finally, note that it might take a while if you want to shut down the signer engine daemon. There might be a worker busy that will first finish his job. Especially if a worker is currently busy replacing signatures in a huge zone, this might take several minutes.

Frequently Asked Questions

On this Page

- [The Signer Engine creates files in the tmp-directory, but nothing is written to the signed-directory. What went wrong?](#)
- [I get this message when doing a manual rollover: "WARNING: key rollover not completed as there are no keys in the 'ready' state; ods-enforcerd will try again when it runs next"](#)
- [I am using a Sun Crypto Accelerator \(SCA\) 6000 under Linux and all HSM operations fail with CKR_HOST_MEMORY](#)
- [How does OpenDNSSEC provide the DS record to the parent zone?](#)
- [How can I load the signed zone into my name server?](#)
- [SoftHSM doesn't work with OpenSC on MacOSX 10.6](#)

The Signer Engine creates files in the tmp-directory, but nothing is written to the signed-directory. What went wrong?

Is auditing enabled for this zone and is there a finalized file in the tmp-directory?

If yes, then the Auditor does not allow to distribute this zone. Could be that unsupported RR were used, Signer Engine and the Auditor disagree on how to parse the information, or that OpenDNSSEC is not following the policy. Check the log to see what it complained about, or run:

```
ods-auditor -s <path to the "zone".finalized file> -z "zone"
```

If no, then something went wrong in the signing process. Please check the logs and report to the OpenDNSSEC team.

I get this message when doing a manual rollover: "WARNING: key rollover not completed as there are no keys in the 'ready' state; ods-enforcerd will try again when it runs next"

OpenDNSSEC makes sure that the zone is secure during the rollover process. This message comes when there is no key that has been published long enough. You probably have no standby keys in your policy. When you initiate the rollover, then OpenDNSSEC first needs to publish the key and after a moment make it active. So do not worry, the rollover process will be finished in a moment.

I am using a Sun Crypto Accelerator (SCA) 6000 under Linux and all HSM operations fail with CKR_HOST_MEMORY

You need to make sure the user running OpenDNSSEC is a member of the opencryptoki group (usually "pkcs11"), or it cannot access the shared memory region used by openCryptoki.

How does OpenDNSSEC provide the DS record to the parent zone?

Currently, manual intervention is required for a KSK rollover. This intervention is a three-stage process that is described on the page [Using OpenDNSSEC](#) in the Key rollovers section.

For future versions, we are automating this process as much as possible, including integration points for interfacing with a parent registry.

How can I load the signed zone into my name server?

The configuration file conf.xml holds a specific Signer configuration section. In there, you can configure *NotifyCommand* to be called by the signer after the zone has been successfully signed. You can put %zone and %zonefile in here, which will expand to the name of the zone that was signed and the filename of the signed zone.

For example:

```
<NotifyCommand>nscd reload<\NotifyCommand>
```

or

```
<NotifyCommand>rndc reload %zone<\NotifyCommand>
```

SoftHSM doesn't work with OpenSC on MacOSX 10.6

If you're building SoftHSM in 64-bit mode (which is the default on 10.6), you need a 64-bit version of OpenSC as well – e.g. the latest [OpenSC SCA](#) snapshot.

Note: pkcs11-tool from OpenSC is typically used for low-level PKCS#11 debugging and is not required by OpenDNSSEC.

Reporting bugs

If you encounter **anything strange** in either the documentation, the code or in the commands you are using, feel free to post a message to the `opendnssec-user` mailing list. You can subscribe to it here:

<http://lists.opendnssec.org/mailman/listinfo/opendnssec-user>

If you have a **bug report**, please enter it into our `trac`-system:

<http://trac.opendnssec.org/newticket>

Be sure to include all the error messages you get, the versions of the other libraries you are using and what actions you did to trigger the error message.

Reference Material

Versions of the documentation for download are available.

Useful [reference material](#) can be found below:

- Training:
 - [Training Videos and Study Material](#)
- Technical references
 - [OpenSSL](#)
 - [PKCS#11](#)
- HSMs
 - [HSM Buyer's Guide](#)
 - [HSM Vendors](#)

Downloads

Latest Versions of Downloadable Documentation

We supply documentation for the latest version of of the documentation in PDF and HTML format:

- [OpenDNSSEC Documentation_v1.3.pdf](#)
- [OpenDNSSEC Documenation_v1.3.html.zip](#)