## Configuration/data files parsing support

Most of the configuration data files parsing support resides in the *sshd-common* artfiact:

<dependency>  
 <groupId>org.apache.sshd</groupId>  
 <artifactId>sshd-common</artifactId>  
 <version>...same version as the rest of the artifacts...</version>  
 </dependency>

The code contains support for parsing the [*authorized\_keys*](http://man.openbsd.org/sshd.8#AUTHORIZED_KEYS_FILE_FORMAT), [*known\_hosts*](http://www.manpagez.com/man/8/sshd/), [*ssh\_config*, *sshd\_config*](https://www.freebsd.org/cgi/man.cgi?query=ssh_config&sektion=5), and [*~/config*](http://www.gsp.com/cgi-bin/man.cgi?topic=ssh_config) files. The code resides in the *sshd-common* artifact - specifically the KeyUtils#getPublicKeyEntryDecoder, AuthorizedKeyEntry#readAuthorizedKeys, KnownHostEntry#readKnownHostEntries and HostConfigEntry#readHostConfigEntries.

### PEM/OpenSSH

The common code contains built-in support for parsing PEM and/or *OpenSSH* formatted key files and using them for authentication purposes. As mentioned previously, it can leverage *Bouncycastle* if available, but can do most of the work without it as well. For *ed25519* support, one must provide the *eddsa* artifact dependency.

### [PUTTY](https://www.putty.org/)

The code contains built-in support for parsing PUTTY key files (usually *.ppk*) and using them same as SSH ones as key-pair providers for autentication purposes. The PUTTY key file(s) readers are contained in the org.apache.sshd.common.config.keys.loader.putty package (specifically PuttyKeyUtils#DEFAULT\_INSTANCE KeyPairResourceParser) of the *sshd-putty* artifact. **Note:** the artifact should be included as an extra dependency:

<dependency>  
 <groupId>org.apache.sshd</groupId>  
 <artifactId>sshd-putty</artifactId>  
 <version>...same version as the rest of the artifacts...</version>  
 </dependency>

### [OpenPGP](https://www.openpgp.org/)

The code contains the *sshd-openpgp* module that enables using *OpenPGP* private key files as identity providers.

<dependency>  
 <groupId>org.apache.sshd</groupId>  
 <artifactId>sshd-openpgp</artifactId>  
 <version>...same version as the rest of the artifacts...</version>  
 </dependency>

The [support](https://issues.apache.org/jira/browse/SSHD-757) for it is currently still in its infancy, and therefore this feature should be considered **experimental** for the time being. However, within its limitations it supports

* RSA keys
* DSA keys
* ECDSA keys

(\*) For now ed25519 keys are not supported by this module.

The code reads **all** the available key pairs in the key file without any distinction between encryption, decryption, authentication or signature ones.

This code relies on the [jpgpj](https://github.com/justinludwig/jpgpj) support module

<dependency>  
 <groupId>org.c02e.jpgpj</groupId>  
 <artifactId>jpgpj</artifactId>  
 <version>...</version>  
 </dependency>

(which in turn automatically uses *Bouncycastle* - so if one does not want *Bouncycastle* one cannot use this module).

#### Using OpenPGP authorized keys entries

In order to be able to read authorized\_keys files that may contain *OpenPGP* keys references, one needs to register the relevant PublicKeyEntryDataResolver-s. This is done by calling PGPPublicKeyEntryDataResolver#registerDefaultKeyEntryDataResolvers once during the *main* code setup. This will enable the code to safely read authorized keys entries having the format specified in the [OpenSSH PGP configuration](https://www.red-bean.com/~nemo/openssh-gpg/):

pgp-sign-dss 87C36E60187451050A4F26B134824FC95C781A18 with-comment  
 pgp-sign-rsa 87C36E60187451050A4F26B134824FC95C781A18

Where the key data following the key type specification is the fingerprint value of the referenced key. In order to use a "mixed mode" file (i.e., one that has both SSH and *OpenPGP* keys) one needs to replace the default AuthorizedKeysAuthenticator instance with one that is derived from it and overrides the createDelegateAuthenticator method in a manner similar as shown below:

// Using PGPAuthorizedEntriesTracker  
public class MyAuthorizedKeysAuthenticatorWithBothPGPAndSsh extends AuthorizedKeysAuthenticator {  
 ... constructor(s) ...  
  
 @Override  
 protected PublickeyAuthenticator createDelegateAuthenticator(  
 String username, ServerSession session, Path path,  
 Collection<AuthorizedKeyEntry> entries, PublicKeyEntryResolver fallbackResolver)  
 throws IOException, GeneralSecurityException {  
 PGPAuthorizedEntriesTracker tracker = ... obtain an instance ...  
 // Note: need to catch the PGPException and transform it into either an IOException or a GeneralSecurityException  
 Collection<PublicKey> keys = tracker.resolveAuthorizedEntries(session, entries, fallbackResolver);  
 if (GenericUtils.isEmpty(keys)) {  
 return RejectAllPublickeyAuthenticator.INSTANCE;  
 } else {  
 return new KeySetPublickeyAuthenticator(id, keys);  
 }  
 }  
}  
  
// Using PGPPublicRingWatcher  
public class MyAuthorizedKeysAuthenticatorWithBothPGPAndSsh extends AuthorizedKeysAuthenticator {  
 ... constructor(s) ...  
  
 @Override  
 protected PublickeyAuthenticator createDelegateAuthenticator(  
 String username, ServerSession session, Path path,  
 Collection<AuthorizedKeyEntry> entries, PublicKeyEntryResolver fallbackResolver)  
 throws IOException, GeneralSecurityException {  
 PGPPublicRingWatcher watcher = ... obtain an instance ...  
 // Note: need to catch the PGPException and transform it into either an IOException or a GeneralSecurityException  
 Collection<PublicKey> keys = watcher.resolveAuthorizedEntries(session, entries, fallbackResolver);  
 if (GenericUtils.isEmpty(keys)) {  
 return RejectAllPublickeyAuthenticator.INSTANCE;  
 } else {  
 return new KeySetPublickeyAuthenticator(id, keys);  
 }  
 }  
}

**Note:** in order to support GPG v2 .kbx files one requires up-to-date [Bouncycastle](https://mvnrepository.com/artifact/org.bouncycastle/bcpg-jdk15on/1.61) and [jpgpj](https://mvnrepository.com/artifact/org.c02e.jpgpj/jpgpj/0.6.1) versions.