Improved Credential and SSL Configuration for EE 7

1. Introduction:

SSL, trust stores, keystores and credential repositories are generally difficult areas to configure for Java EE environments. The configuration and management interfaces are vendor specific and non-standard.

Java EE 7's goal for providing PaaS functionality for application developers makes this even more important. In order to make application archives portable from one cloud provider to another, we need to unify and simplify the approach to this configuration and provisioning.

Standardizing the ability to provision keystores and metadata (as necessary) by bundling them with an application at deployment time will provide portability of this configuration across PaaS providers.

It will also require that platform implementations differentiate these key entries and certificates by application when they are provided with the application. The method of differentiating this configuration is an implementation specific detail. It may result in application level configuration, the leveraging of server instances that are dedicated to an application or any number of other means for platforms to ensure application level semantics for the SSL configuration.

The addition of this provisioning facility is supplemental to the existing functionality and platform specific mechanisms available today. Backward compatibility is expected for applications that require this configuration to be done but do not provide it with the application itself.

The SSL related networking components of a EE 7 PaaS platform implementation will utilize the provisioned certificates and keys as required to meet the requirements for secure interaction. The SSL termination point in any given implementation is specific to that offering and not dictated by the platform specification.

Client specific SSL configuration will be leveraged by a platform implementation also through vendor specific means. JAX-WS Dispatch objects, for instance, will be able to leverage the provisioned certificates through configuration specific to the Dispatch implementation and/or the consumer's instantiation and initialization properties.

1.1. Standardized Aspects:

- Configuration by Convention
  - well-known aliases for SSL related key pairs and passphrases
  - well-known filenames for the stores themselves
  - differentiated by application
- Alias Mapping
  - mapping of the well-known aliases to actual aliases within the repositories
  - to address inability to set the actual alias
  - to point multiple well-known aliases to the same key material
- Secure provisioning of keys, certs and passwords with applications
  - it is assumed that the platform implementation will likely provide alias management

capabilities in addition to support for these bundled stores

1.1.1. Configuration by Convention

By specifying the filenames of the repositories and aliases expected within them in the EE platform spec, configuration will not be necessary for deployment machinery and/or runtime code to resolve the key pairs and related passphrases that are needed for establishing SSL connections.

The following optional repositories are expected to be found within an application's META-INF directory.

Listing 1. Store Names and Location
```
META-INF/
      passwdstore.p12 // password aliases and passwords
      truststore.p12  // trusted certs
      keystore.p12    // keys and certs
```

Listing 2. Well-known Aliases within keystore.p12
```
ssl-server // server identity to connecting clients
ssl-client // server identity when acting as a client to another server
```

Listing 3. Well-known Aliases within passwdstore.p12
```
ssl-server // password alias for accessing the ssl-server cert in keystore.p12
ssl-client // password alias for accessing the ssl-client cert keystore.p12
keystore-phrase // passphrase for the keystore.p12
```

The password required to access the aliases within the passwdstore.p12 is required to be provided by the deployer of the application into the deployment interface for the platform implementation. It is then used to retrieve the passphrase for accessing the keystore.p12 in order to retrieve the key pairs for SSL connections.
There may be a need to support an optional alias mapping facility within the archive as well. This would enable stores that do contain key material for the well-known aliases to be used by mapping the expected names to the actual aliases within the store. (See Open Issues – 2.2. Alias Mapping)

1.1.2. Alias Mapping

In addition to the well-known aliases there may be a need to provide a mapping from the well-known aliases to the actual aliases within the keystore.

1.1.2.1. Mapping to Actual Aliases

In some instances, the alias used within the keystore itself may not be the well-known alias expected by this proposal. This facility would enable the packager to map the well-known alias to that within the actual store and allow the alias to be resolved to the appropriate key material.

For instance, when using openssl to create a self-signed cert and key, there is no way to indicate what the alias in the resulting store should be.

The default alias is “1”.

1.1.2.2. Server Identity as Client Identity

In other cases, it may be desired that the same certificate be used whether the server is acting as a server or a client. By providing a mapping from the ssl-client alias to the ssl-server alias the certificate chain is only stored once in the repository but both aliases are resolved.

1.1.2.4. Mapping Syntax

Without the ability to specify the alias it may make sense to provide a mapping syntax to indicate which of the aliases within the actual store represent the well-known usage aliases.

Something like:

Listing 4. alias-mapping

```
<alias-mapping>
      <alias>ssl-server</alias>
      <keystore-alias>1</keystore-alias>
</alias-mapping>
<alias-mapping>
      <alias>ssl-client</alias>
      <keystore-alias>1</keystore-alias>
</alias-mapping>
```

In listing 4 above, we have a mapping from the well-known ssl-server alias to an actual alias within the keystore.p12 called '1'. We also have a mapping for the well-known alias ssl-client to the same cert as the server.

1.1.3. Secure Provisioning of Keys, Certs and Passwords with Applications

In order to facilitate the provisioning of key material, certificates, passwords and aliases at deployment time in a cross platform manner, we need to introduce an archive for bundling them securely.

Platform vendors may use the deployed archives directly as the credential stores for resolving password aliases or extract the contents of the artifacts and populate preferred repositories at deployment time for use by the platform runtime.

The PKCS#12 standard provides a Personal Information Exchange file format. Previously, there has been a focus on providing interoperability for the exchange of certificates and private keys. However there has been no real standardization of using PKCS#12 SecretKey support and therefore the interoperability for password exchange is not pre-existing. (See Open Issues – 2.1. File Format for Standard EE Password Store)

We will need to define the usage of the PKCS#12 format for this purpose in terms of:

- how to represent the SecretKey within the PKCS#12 SecretBag
- how to associate the alias to be used to resolve the secretKey (friendly name, localKeyId)
- any metadata needed for advanced management of passwords (expiry date, etc)

Existing tooling for the management of PKCS#12 files, at least for certificates and keys, is generally available and can be leveraged for creating these stores.

The use of JKS and JCEKS has been ruled out due to their proprietary natures.

2. Example of Provisioning Processing

The following is an abstracted example of how this mechanism would be utilized during application deployment and related dependency provisioning.

NOTE: This is not intended to be a blueprint for a required implementation but as one example of a valid interpretation of a provisioning algorithm for this proposal. The details of any particular implementation's algorithm will be as unique as their PaaS offering.

2.1. General Flow Description

2.2 Application Deployed to PaaS Environment

2.3 **Dependency Discovery**

2.3.1. Detects services that need to be provisioned in order to support the application

2.4 Provisioning of Required Services is Orchestrated

2.4.1. Domain names are communicated to the DNS as required

2.4.2. Load balancers are configured to accommodate the application and its specific configuration

2.4.2.1. Utilized the well-known aliases and mapping facility as appropriate the required certificates and keys are extracted from the provided stores

2.4.2.2. Propagates the certificates and keys required for SSL

2.4.2.3. Ensures that the domain name and embedded hostname within the certificates meet any hostname verification requirements

2.4.3. Client side certificates (for outbound connections) are made available to the application for client access

2.5 Public PaaS Application Domains are be published out of band

2.5.1. Customer owned domains must have their DNS records updated to reflect the provided IP address of the PaaS provider exposed endpoint

3. Open Issues

3.1. File Format for the Standard EE Password Store

We need a standard format to securely exchange/deploy credentials to an EE platform. It is desired for this standard to be a well known industry standard that is interoperable and has sufficient support in tooling and consuming APIs.

Our primary candidate has been PKCS#12.

PKCS#12 is a “standard” format established and published by RSA Laboratories and is recognized as a defacto standard for Personal Information Exchange. Even though PKCS#12 is not a product of an actual industry standards body, RSA makes no patent claims on the general constructions within it.

There are a couple interesting issues with using PKCS#12 for password exchange.

3.1.1. Interoperability

Given the open nature of much of the PKCS#12 specification there are numerous ways to represent the standard content and related metadata within a PKCS#12 file.

Interoperability would need to be addressed by clearly specifying the scheme to use in encoding the aliases, passwords and required metadata within PKCS#12.

2.1.1.1. QUESTION: Can we define the encoding scheme for storing passwords within PKCS#12?

It would also require enforcement within the TCK to ensure that the expected file format is consumed properly by the platform implementation.

3.1.2. Tooling

Currently there is a lack of generally available tooling for the management of passwords within the generic SecretBag constructs of PKCS#12.

JDK 7 support for PKCS#12 keystores is limited to private keys and certificate chains.

Support for secret keys and trusted certs will be added in JDK 8.

There is also planned enhancements to Keytool for management of secret keys within the JDK 8 timeframe.

Vendors can provide their own tooling that complies with the scheme defined within the standard use as described in the final specification or utilize tooling that is provided as part of the RI.

3.1.2.1. QUESTION: Are these tooling constraints acceptable given the envisioned support from keytool, and the PKCS12Keystore in the JDK 8 timeframe and the RI?

3.1.3. Existing Providers

A number of providers have only implemented those portions of the PKCS#12 specification that is pertinent to existing PKI requirements and potentially entirely skip SecretBag entries within the stores while loading them.

Existing provider implementations will need to be aware of the specifics of the scheme used to represent EE password aliases and metadata in order to add support for them.

We would need to make a provider that is capable of consuming this particular usage pattern of PKCS#12 generally available to platform implementations and tooling providers.

2.1.3.1. QUESTION: Is the requirement to support passwords through PKCS#12 SecretBag on providers acceptable?

2.1.3.2. QUESTION: Is the requirement to support passwords through PKCS#12 SecretBag on providers necessary given a generally available provider through the RI?

3.2. Keystore and Password Store Consolidation

It is likely that the credentials stored within the passwdstore.p12 described in this document could be consolidated into the keystore.p12. This document initially describes them separately in order to allow them to evolve separately into the best approach to meet all requirements. Expert group deliberations can decide to consolidate them or leave them separate.

3.2.1. QUESTION: Should we consolidate the passwdstore.p12 and keystore.p12 initially and only separate them if need be?

3.3. Multiple Domains for a Given Application

In the event that multiple domains are to be requested for a given application, we would need to accommodate this through the alias mapping mechanism.

The details of how to map from the ssl-server for one domain to the actual alias in the keystore.p12 as well as another domain name and its alias needs to be determined.

One possibility would be to include the domain name as an attribute on the alias-mapping element – as seen in Listing 5.

Listing 5. alias-mapping with domain names

```
<alias-mapping domain=”mydomain”>
      <alias>ssl-server</alias>
      <keystore-alias>1</keystore-alias>
</alias-mapping>
<alias-mapping domain=”yourdomain”>
      <alias>ssl-server</alias>
      <keystore-alias>1</keystore-alias>
</alias-mapping>
<alias-mapping domain=”mydomain”>
      <alias>ssl-client</alias>
      <keystore-alias>1</keystore-alias>
</alias-mapping>
```

3.3.1. QUESTION: Are multiple domains for an application required?

3.3.2. QUESTION: Would the additional element in the mapping suffice?