Encryption 2

Encrypting a Field

Secure fields may be stored in the database in encrypted form. An OJB converter will be created to decrypt/encrypt going from and to the database. This converter must be specified in the OJB field-descriptor tag for the field, like so:

org.kuali.module.financial.OJB-repository-financial.xml

```
<descriptor-repository version="1.0">
    ...
    <class-descriptor class="org.kuali.module.financial.bo.Payee"
table="FP_DV_PAYEE_T">
        ...
        <field-descriptor name="taxIdNumber" column="DV_TAX_ID_NBR"
jdbc-type="VARCHAR" index="true"
   conversion="org.kuali.core.util.OjbKualiEncryptDecryptFieldConversion" />
        ...
    <class-descriptor>
</descriptor-repository>
```

The Encryption Service

All encryption is done through a core service. The various points in the frameworks will call this central service for the actual encrypting/decrypting. Institutions wishing for different encryption algorithms can simply override the `encryptionService` Spring bean. For example, say you want to turn off encryption for development purposes, you can just add the following to your override Spring beans file....

edu.sampleu.kuali.SampleuSpringBeans.xml

```
<bean id="encryptionService"
class="org.kuali.core.service.impl.NoEncryptionEncryptionServiceImpl" />
```

The out of the box encryption service implementation will also implement the Demonstration interface. Warnings will be displayed on each page of the application until a new service implementation is put in place or the provided implementation is changed to not implement the Demonstration i nterface.

As stated, KFS comes with two implementations of `EncryptionService`: `NoEncryptionEncryptionServiceImpl` and `DemonstrationGradeEncryptionServiceImpl`. Neither are advisable for use in environments with actual sensitive data. However, the Java SE comes with a number of encryption algorithms which can be used in an institution specific implementation of `EncryptionService`. Interested parties should look at this document on Cryptography in Java 1.5 to find choices that come prepackaged with Java. Very interested parties should look at Bruce Schneier's excellent books, including the classic Applied Cryptography. Most institutions likely also have a security officer or task force who would be very interested in the institution specific implementation of the `EncryptionService` interface.

Naturally, changing the encryption algorithm will mean that the encryption key will need to be a valid key for the algorithm, and that the size of encrypted values will be different from algorithm to algorithm. If a stronger encryption algorithm is implemented, then the database field widths of encrypted values may have to be made much longer.

Again, the point here is that implementing institutions should not, under any circumstance, use the KFS pre-packaged `EncryptionService` implementations to encrypt secure or sensitive data entered within KFS.

The Encryption Key

Of primary importance to practically all encryption algorithms is the encryption key. The key needs to be held in a location where KFS can access

it but where non-administrative users have no access to it. The `DemonstrationGradeEncryptionServiceImpl` uses the `encryption.key` property in the `security.directory`/security.properties file (database connection properties also exist in this file). The `encryption.key`, and `security.directory` configuration properties may be overridden. Again, protecting the encryption key is crucial to the integrity of the encryption system.

Encryption Warning Message

If Kuali implements the demonstration version of the encryption framework then a warning message is to be displayed throughout the system. The motivation for that is to supply a fully operational system but at the same time make implementers aware that proper encryption of sensitive data is their responsibility.

The check is in KualiAction which places a message (KeyConstants.MESSAGE_DEMONSTRACTION_WARNING) into the GlobalVariable MessageList based on meeting both (&&) conditions:

a) If the system parameter "demonstrationEncryptionCheck_FLAG" (SYSTEM Security Group) is set to true.

b) EncryptionService is implementing the Demonstraction interface. This is per the Marker interface pattern.

As part of this modification the KualiLookup.jsp and KualiInquiry.jsp had to be modified to display messages. This should not cause any problems since they were already displaying error message.

Mass Encrypting Data

Chances are that for most Kuali implementing institutions, there's going to be a process of loading data that already exists into the KFS database. What happens to sensitive data at that point?

Thankfully, there is an Ant task for kuali_project called "post-load-encrypt". This task depends on a file which holds the names of business objects with fields that need to be encrypted, plus the fields that should be encrypted for that business object. Here's an example:

```
org.kuali.module.chart.bo.Account=chartOfAccountsCode,accountNumber
org.kuali.moudle.chart.bo.ObjectCode=universityFiscalYear,chartOfAccountsCode,nextYear
FinancialObjectCode
```

The location of the file can be set using the configuration property `encrypt.attributes.properties.file`. Running the ant task will go through every table and field specified in the `encrypt.attributes.properties.file` and encrypt those fields.

Please see here for a more complete introduction to the related material.

Unable to render {include} The included page could not be found.