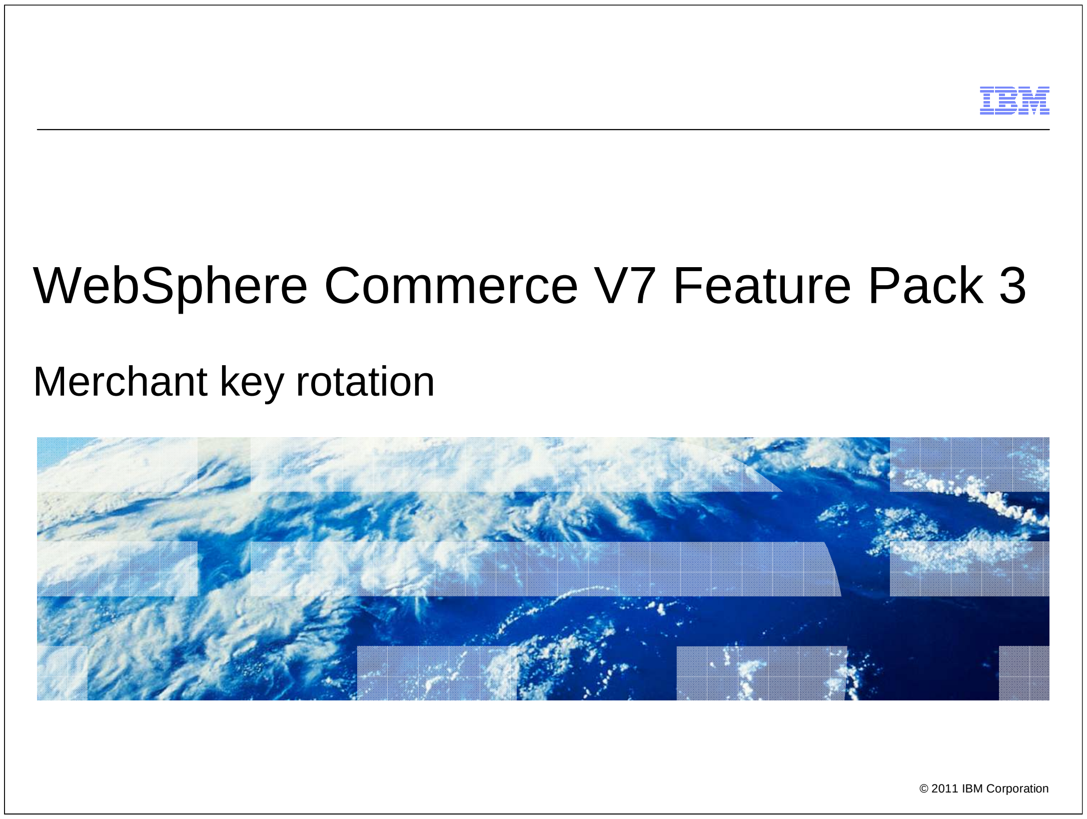
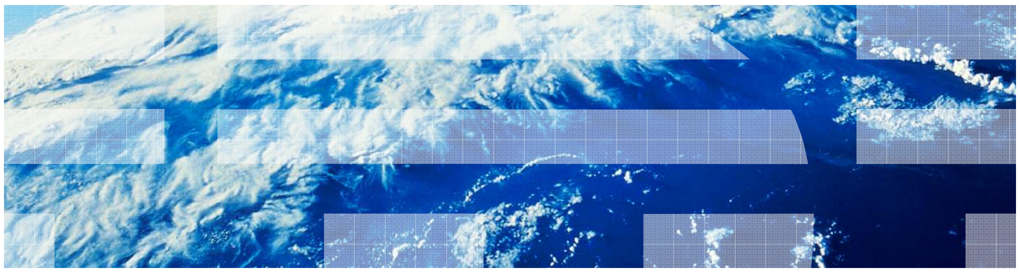


WebSphere Commerce V7 Feature Pack 3

Merchant key rotation



This presentation covers the merchant key rotation enhancements that are included in WebSphere® Commerce V7.0 Feature Pack 3.

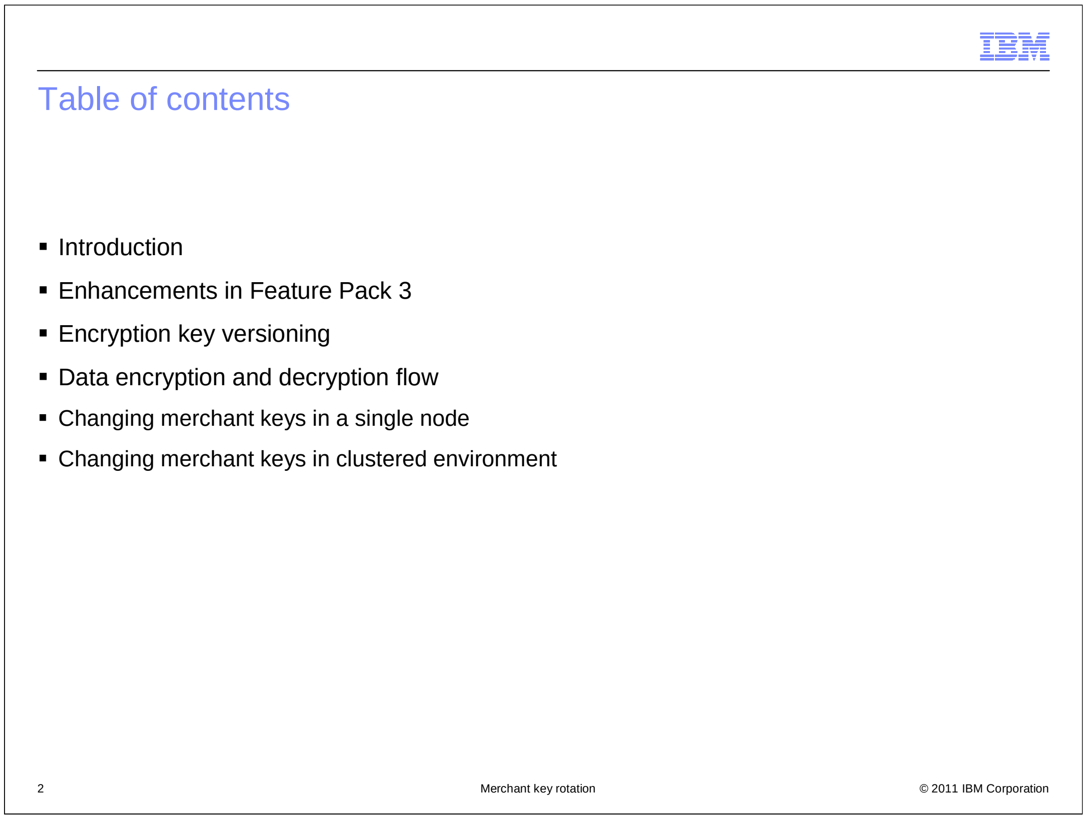
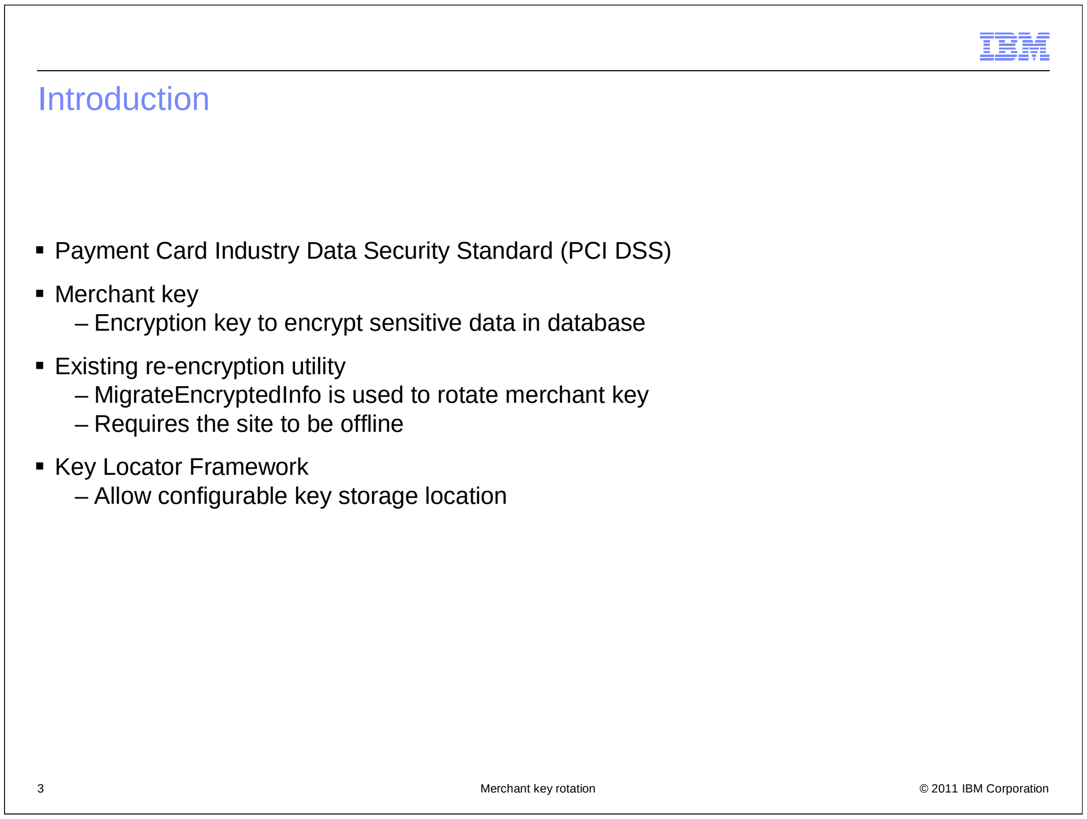


Table of contents

2 Merchant key rotation © 2011 IBM Corporation

This presentation first reviews the merchant key rotation solution provided before Feature Pack 3. It then highlights the new enhancements included in Feature Pack 3. The encryption key versioning concept will be discussed in detail. The steps for changing the merchant key in a single node environment and clustered environment are also included.



Introduction

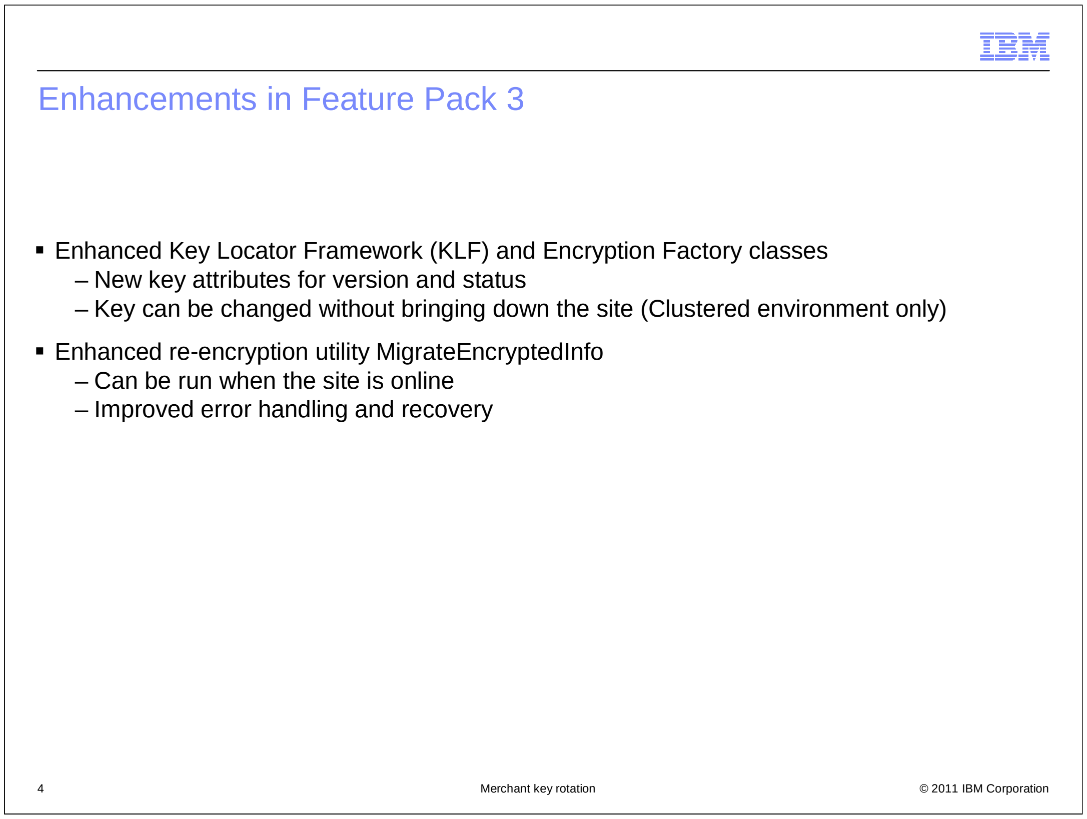
- Payment Card Industry Data Security Standard (PCI DSS)
- Merchant key
  – Encryption key to encrypt sensitive data in database
- Existing re-encryption utility
  – MigrateEncryptedInfo is used to rotate merchant key
  – Requires the site to be offline
- Key Locator Framework
  – Allow configurable key storage location

3 Merchant key rotation © 2011 IBM Corporation

The Payment Card Industry Data Security Standard mandates that sensitive information, for example credit card numbers, must be stored in an encrypted format and the encryption key must be changed regularly.

The merchant key is an encryption key that WebSphere Commerce uses to encrypt sensitive data stored in the database, such as passwords and credit card numbers. WebSphere Commerce provides the MigrateEncryptedInfo utility to help you to change the merchant key. This utility uses the current merchant key to decrypt the data, then uses the new key specified in the key configuration file to encrypt the data. With previous releases, you have to bring your site down before you can run this utility.

The Key Locator Framework (KLF) allows an encryption key to be stored and retrieved from a configurable location such as an external device.



Enhancements in Feature Pack 3

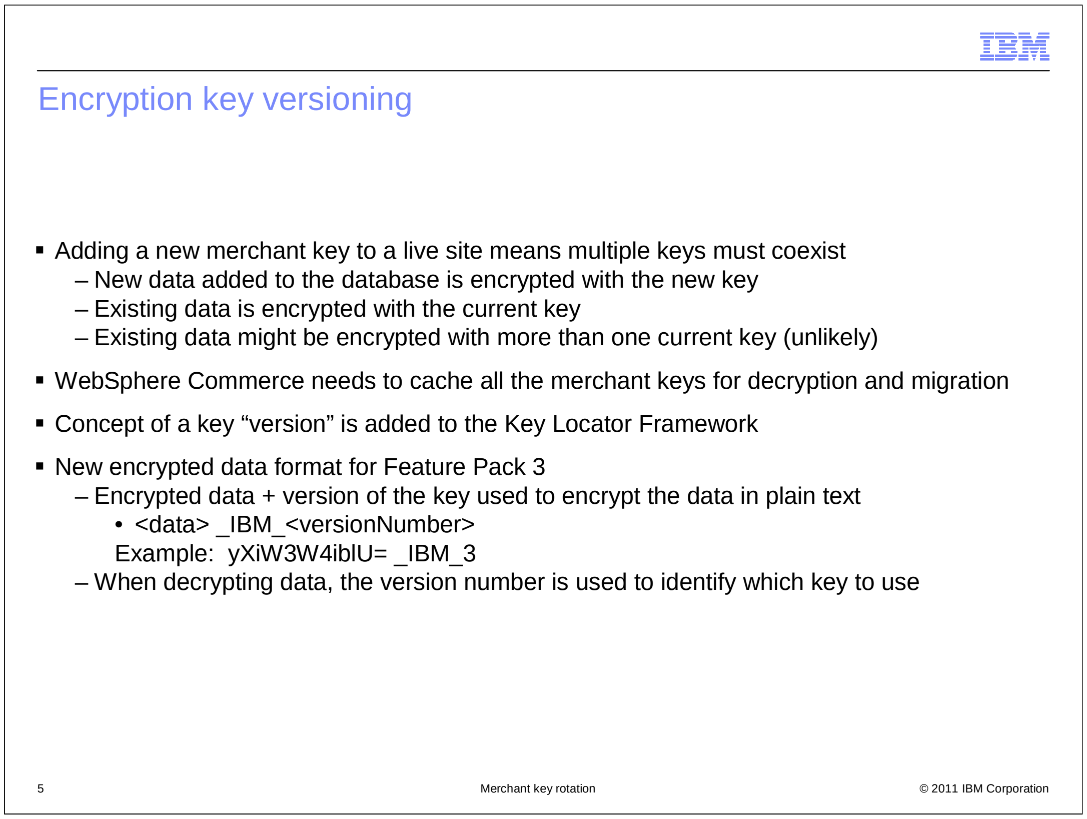
- Enhanced Key Locator Framework (KLF) and Encryption Factory classes
  - New key attributes for version and status
  - Key can be changed without bringing down the site (Clustered environment only)
- Enhanced re-encryption utility MigrateEncryptedInfo
  - Can be run when the site is online
  - Improved error handling and recovery

4 Merchant key rotation © 2011 IBM Corporation

The main goal of the Feature Pack 3 enhancements is to allow you to change the merchant key and run the MigrateEncryptedInfo utility in clustered environment without bringing your site down. In order to achieve this goal, some changes have been made to the existing implementation.

The Key Locator Framework allows multiple encryption keys to be stored and retrieved from different locations. In Feature Pack 3, the Key Locator Framework adds support for two new key attributes, version and status. These attributes are discussed later in the presentation. The Encryption Factory classes have been enhanced so that you can change the merchant key without bringing down your site.

The MigrateEncryptedInfo utility has been enhanced such that it can be run while your site is running. The error handling and recovery for MigrateEncryptedInfo has been improved. A new log file is added from which you can easily locate the data that caused the error.



Encryption key versioning

- Adding a new merchant key to a live site means multiple keys must coexist
  - New data added to the database is encrypted with the new key
  - Existing data is encrypted with the current key
  - Existing data might be encrypted with more than one current key (unlikely)
- WebSphere Commerce needs to cache all the merchant keys for decryption and migration
- Concept of a key "version" is added to the Key Locator Framework
- New encrypted data format for Feature Pack 3
  - Encrypted data + version of the key used to encrypt the data in plain text
    - <data> _IBM_<versionNumber>

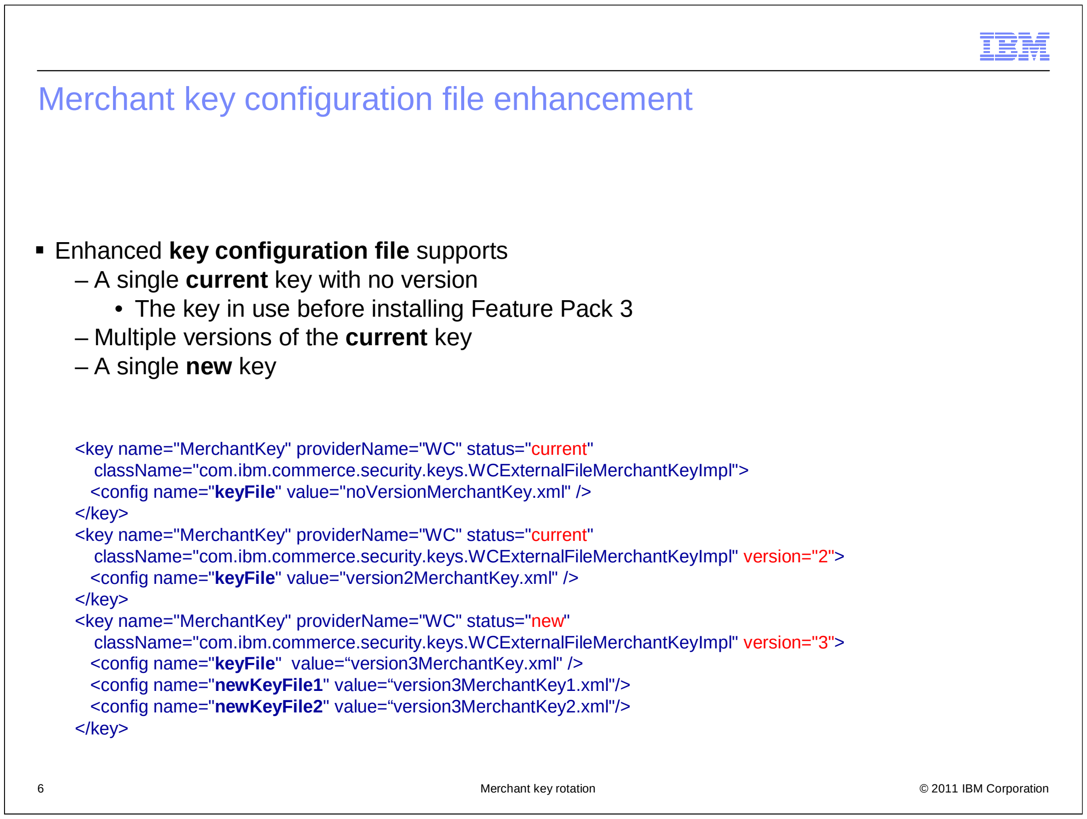
      Example: yXiW3W4iblU= _IBM_3
  - When decrypting data, the version number is used to identify which key to use

5 Merchant key rotation © 2011 IBM Corporation

Before Feature Pack 3, all data in the database is encrypted using only one key – the current key. When you need to change your encryption key to a new key, you shutdown your site and then run the MigrateEncryptedInfo utility to migrate the data in the database from the current key to your new key. Once you restart your server, any new data is encrypted with your new key.

Feature Pack 3 allows you to keep your site running when you change your key. Once the new key is available to your server, new data added to the database is encrypted with your new key. Before you run the MigrateEncryptedInfo utility, your database might contain data encrypted with two keys: the current key and the new key. It is also possible that the database might contain data encrypted with more than one current key. If this situation happens, the data in the database might be encrypted with multiple current keys and a new key. In order to migrate encrypted data to the new key, the MigrateEncryptedInfo utility needs to know which key was used to encrypt the data. It only migrates data that was encrypted with the current key, not the new key. To keep track of the various keys, a new concept, encryption key versioning, is introduced.

When you change to a new key, you need to associate a version number to the key. The version of the key used to encrypt the data is appended to the encrypted data with a plain text suffix, such as _IBM_3. When decrypting the data, the version number is retrieved to identify which key to use, as each key has a unique version.



Merchant key configuration file enhancement

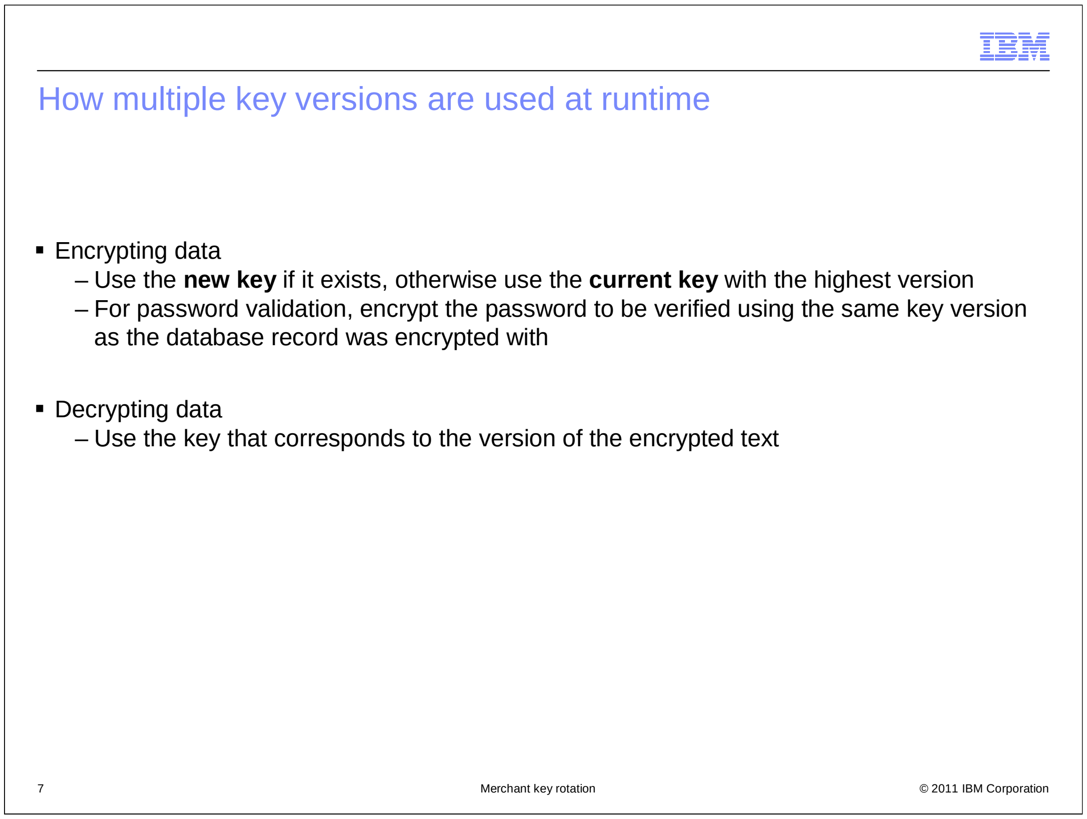
- Enhanced **key configuration file** supports
  - A single **current** key with no version
    - The key in use before installing Feature Pack 3
  - Multiple versions of the **current** key
  - A single **new** key

```
<key name="MerchantKey" providerName="WC" status="current"
    className="com.ibm.commerce.security.keys.WCExternalFileMerchantKeyImpl">
   <config name="keyFile" value="noVersionMerchantKey.xml" />
</key>
<key name="MerchantKey" providerName="WC" status="current"
    className="com.ibm.commerce.security.keys.WCExternalFileMerchantKeyImpl" version="2">
   <config name="keyFile" value="version2MerchantKey.xml" />
</key>
<key name="MerchantKey" providerName="WC" status="new"
    className="com.ibm.commerce.security.keys.WCExternalFileMerchantKeyImpl" version="3">
   <config name="keyFile"  value="version3MerchantKey.xml" />
   <config name="newKeyFile1" value="version3MerchantKey1.xml"/>
   <config name="newKeyFile2" value="version3MerchantKey2.xml"/>
</key>
```

6 Merchant key rotation © 2011 IBM Corporation

In the Key Locator Framework, keys are defined in a key configuration file. The location of the key configuration is specified in the server configuration file.

Feature Pack 3 has enhanced the key configuration file. Instead of having a single key, you can now define multiple current keys and a maximum of one “new” key. Each key must have a version number associated with it in the key configuration file with one exception. When you first install Feature Pack 3, the encrypted data in the database does not have a version suffix so its encryption key should be defined without a version number.



How multiple key versions are used at runtime

- Encrypting data
  - Use the **new key** if it exists, otherwise use the **current key** with the highest version
  - For password validation, encrypt the password to be verified using the same key version as the database record was encrypted with
- Decrypting data
  - Use the key that corresponds to the version of the encrypted text

7 Merchant key rotation © 2011 IBM Corporation

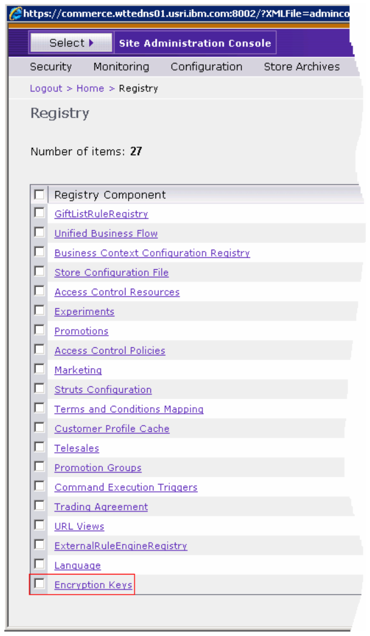
When there is a need to encrypt new data such as a new shopper's password or new credit card number, if a new key exists, it is used for the encryption. Otherwise the current key is used.

To verify a shopper's password for login, the password provided by the shopper needs to be encrypted. The encryption uses the same version of the key that was used to encrypt the password in the database. The password verification is done by comparing the encrypted version of the password supplied by the shopper with the encrypted password in the database.

When there is a need to decrypt data, for example an encrypted credit card number, the key corresponding to the version of the encrypted text is used. For example, if the encrypted text ends with "_IBM_3", then the version 3 key is used.

WCKeyRegistry class

- Caches all the encryption keys in memory
  - All versioned Merchant keys
  - Default merchant key
  - Session key
- New implementation in Feature Pack 3
  - Implements the *com.ibm.commerce.registry.Registry* interface
  - New “Encryption Keys” registry is displayed in Site Administration Console
  - Can be refreshed without taking the server down


https://commerce.wttedns01.usri.ibm.com:8002/?XMLFile=adminco
Select
Site Administration Console
Security
Monitoring
Configuration
Store Archives
Logout > Home > Registry

Registry

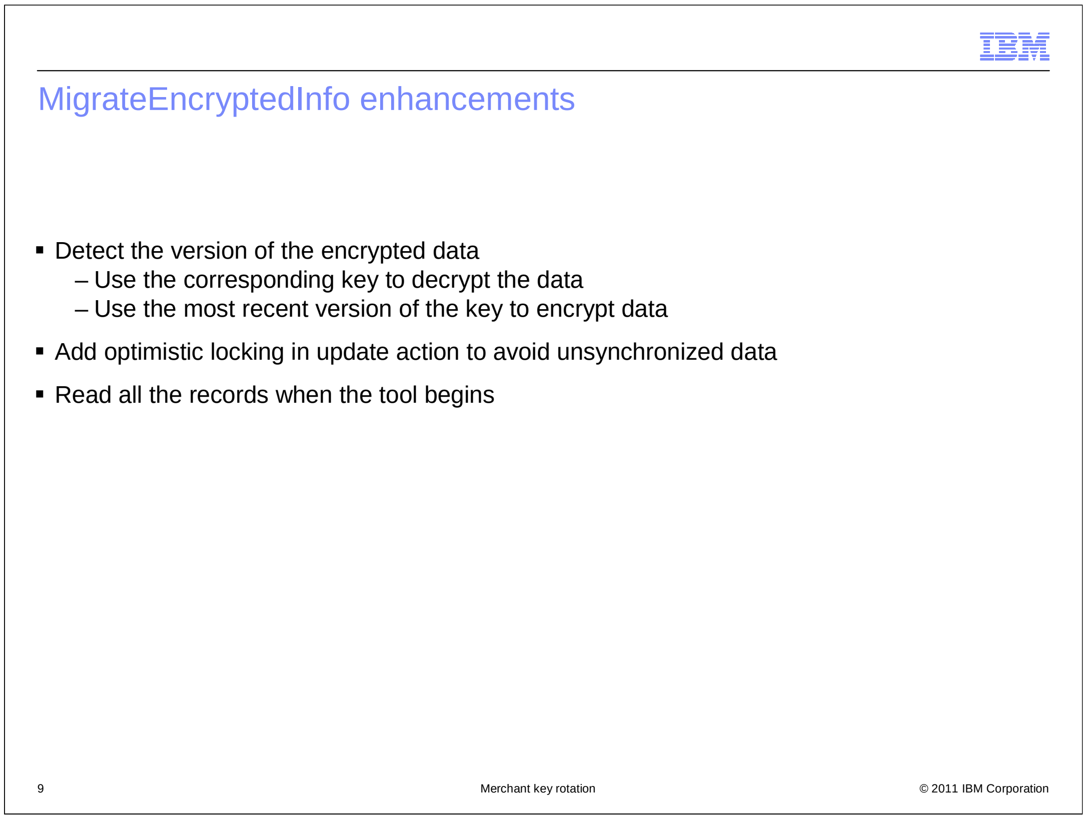
Number of items: **27**

- Registry Component
- GiftListRuleRegistry
- Unified Business Flow
- Business Context Configuration Registry
- Store Configuration File
- Access Control Resources
- Experiments
- Promotions
- Access Control Policies
- Marketing
- Struts Configuration
- Terms and Conditions Mapping
- Customer Profile Cache
- Telesales
- Promotion Groups
- Command Execution Triggers
- Trading Agreement
- URL Views
- ExternalRuleEngineRegistry
- Language
- Encryption Keys

8 Merchant key rotation © 2011 IBM Corporation

Once all the encryption keys are registered in the key configuration file, the WCKeyRegistry class is used to read this file and cache all the keys in memory. Besides the merchant keys, the WCKeyRegistry class also caches all other encryption keys in memory, such as the default merchant key and session key.

In order to achieve the zero-downtime objective when registering a new key, the WCKeyRegistry has been modified to implement the *Registry* interface. With this implementation, a new registry called “Encryption Keys” is displayed in the WebSphere Commerce Site Administration Console. It can be refreshed when needed without taking the server down.



MigrateEncryptedInfo enhancements

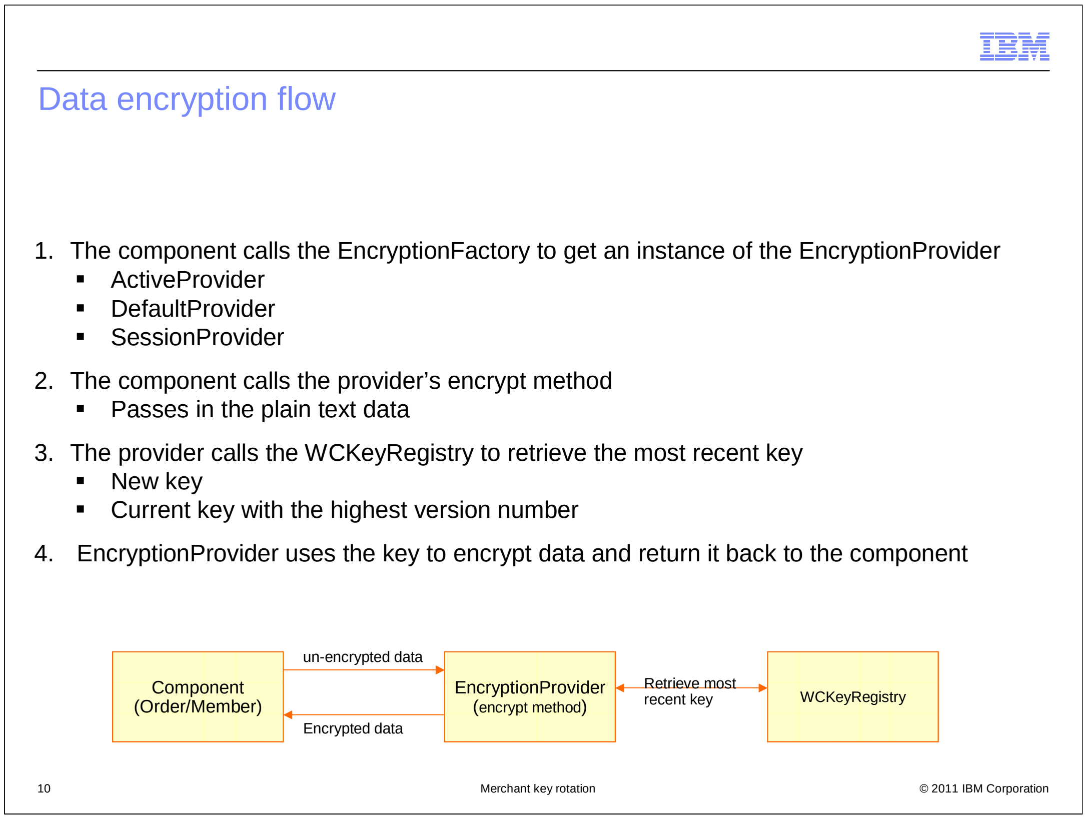
- Detect the version of the encrypted data
  – Use the corresponding key to decrypt the data
  – Use the most recent version of the key to encrypt data
- Add optimistic locking in update action to avoid unsynchronized data
- Read all the records when the tool begins

9 Merchant key rotation © 2011 IBM Corporation

In order for the MigrateEncryptedInfo utility to run while the site is online, it has been enhanced to be able to detect the version of the encrypted data. It then retrieves the corresponding version of the key to decrypt the data. To encrypt new data, it uses the most recent version of the key.

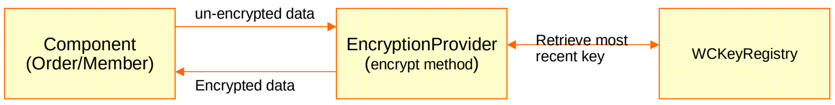
The utility has added an optimistic locking in the update action to avoid unsynchronized data. It will retry reencryption when an optimistic update failure exception occurs.

When the utility begins, it reads in all the records to be migrated. This avoids the need to keep reading data as new records are being inserted from the live site.



Data encryption flow

1. The component calls the EncryptionFactory to get an instance of the EncryptionProvider
   - ActiveProvider
   - DefaultProvider
   - SessionProvider
2. The component calls the provider's encrypt method
   - Passes in the plain text data
3. The provider calls the WCKeyRegistry to retrieve the most recent key
   - New key
   - Current key with the highest version number
4. EncryptionProvider uses the key to encrypt data and return it back to the component

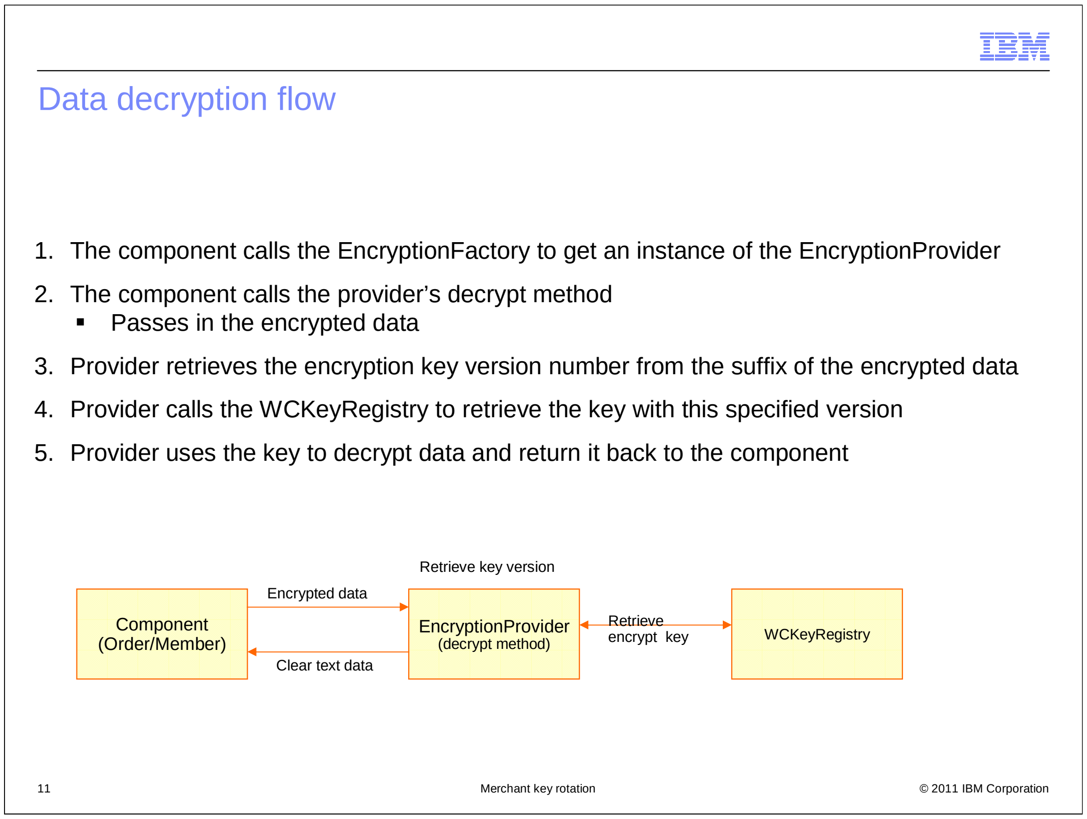


10 Merchant key rotation © 2011 IBM Corporation

The diagram shown here depicts the data encryption flow for a typical WebSphere Commerce component.

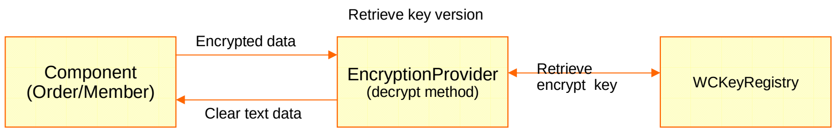
The component first calls the EncryptionFactory to get an instance of the required EncryptionProvider. There are three providers used by WebSphere Commerce. ActiveProvider is a provider that uses the active merchant key to encrypt data. DefaultProvider uses the default merchant key to encrypt data. SessionProvider is a provider that uses the active session key to encrypt data that is exposed to customers.

The component then calls the provider's encrypt method and passes in the plain text data to be encrypted. The provider first calls WCKeyRegistry to retrieve the most recent key. The most recent key can be a key with status equal to new, if it exists, or the current key with the highest version number. The EncryptionProvider uses the most recent key to encrypt the plain text data. The encrypted data is then returned back to the component.



Data decryption flow

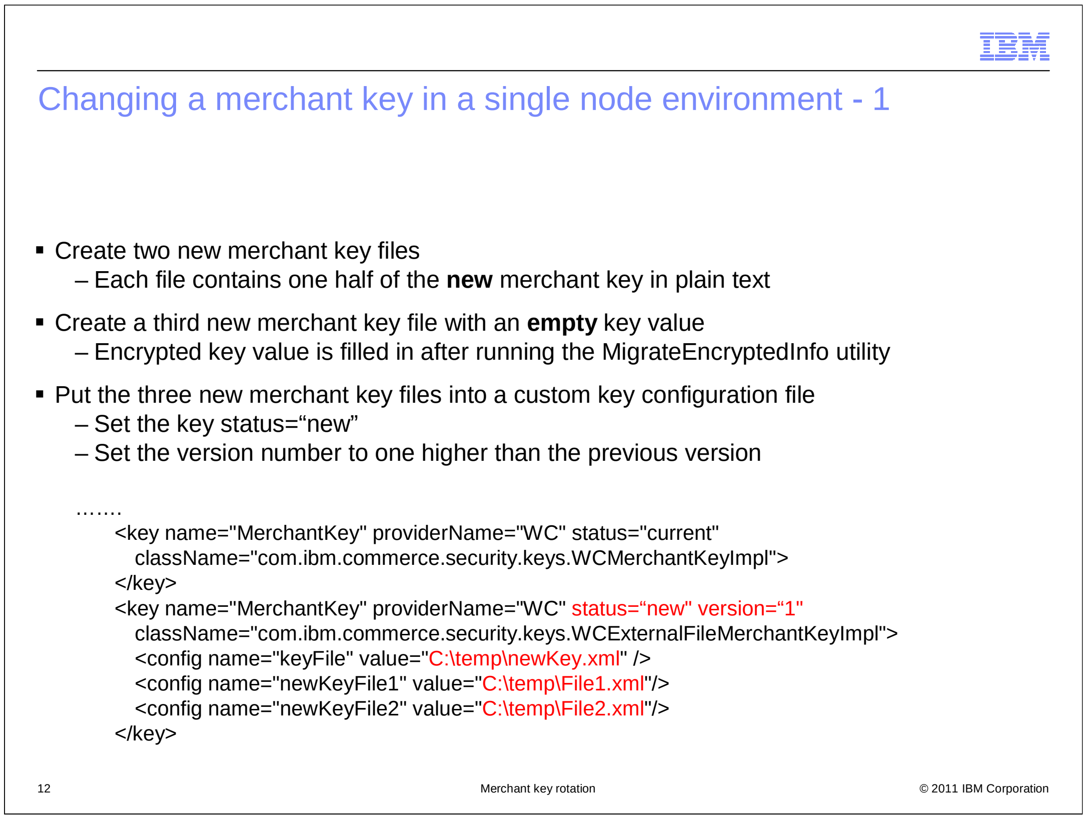
1. The component calls the EncryptionFactory to get an instance of the EncryptionProvider
2. The component calls the provider's decrypt method
   - Passes in the encrypted data
3. Provider retrieves the encryption key version number from the suffix of the encrypted data
4. Provider calls the WCKeyRegistry to retrieve the key with this specified version
5. Provider uses the key to decrypt data and return it back to the component



11 Merchant key rotation © 2011 IBM Corporation

The diagram shown here depicts the decryption flow for a component.

The component first calls the EncryptionFactory to get an instance of the EncryptionProvider. The component then calls the provider's decrypt method and passes in the encrypted data. The provider first retrieves the version of the key used to encrypt the data. The version number is appended as the suffix of the encrypted data. The provider then calls the WCKeyRegistry to retrieve the key with this specified version. This key is used to decrypt the data to retrieve the plain text data. The plain text data then is returned back to the component.



Changing a merchant key in a single node environment - 1

- Create two new merchant key files
  - Each file contains one half of the **new** merchant key in plain text
- Create a third new merchant key file with an **empty** key value
  - Encrypted key value is filled in after running the MigrateEncryptedInfo utility
- Put the three new merchant key files into a custom key configuration file
  - Set the key status="new"
  - Set the version number to one higher than the previous version

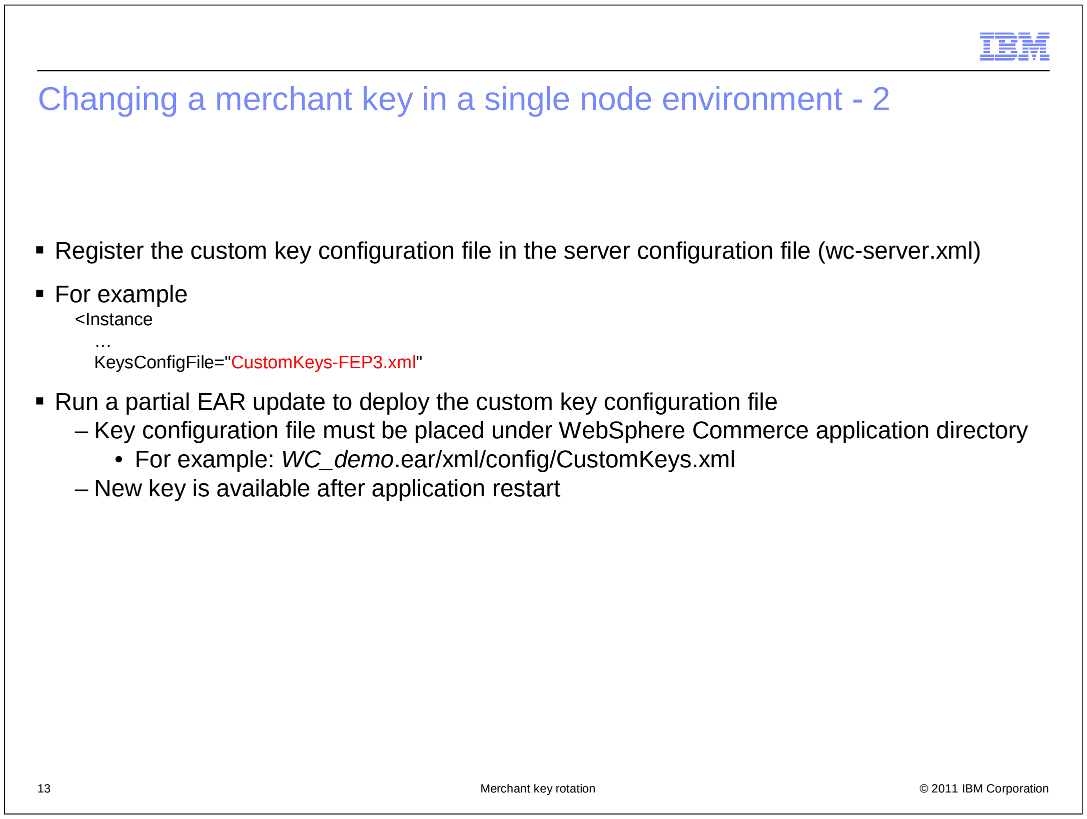
```
…….
    <key name="MerchantKey" providerName="WC" status="current"
      className="com.ibm.commerce.security.keys.WCMerchantKeyImpl">
    </key>
    <key name="MerchantKey" providerName="WC" status="new" version="1"
      className="com.ibm.commerce.security.keys.WCExternalFileMerchantKeyImpl">
      <config name="keyFile" value="C:\temp\newKey.xml" />
      <config name="newKeyFile1" value="C:\temp\File1.xml"/>
      <config name="newKeyFile2" value="C:\temp\File2.xml"/>
    </key>
```

12 Merchant key rotation © 2011 IBM Corporation

If you want to continue to use the the steps provided in the Information Center before fix pack 3 to change the merchant key, that process is still available after you upgrade to fix pack 3. You must bring your site down before you can change your merchant key using the older process.

The steps shown here are for fix pack 3, which allows you to change the merchant key to a new value while your server is online. These steps are for WebSphere Commerce in a single node environment.

As before, you first need to create two new merchant key files where each file contains one half of the the new key in plain text. Next, you need to create another merchant key file which will hold the encrypted key. You then create a custom key configuration file to define which keys are the current keys and which one is the new key. The difference is that when you define a new key, you must assign a version number to it. The new key's version does not have to start at any specific number. Any version number is fine as long as the new key version number is higher than the current key.



Changing a merchant key in a single node environment - 2

- Register the custom key configuration file in the server configuration file (wc-server.xml)
- For example

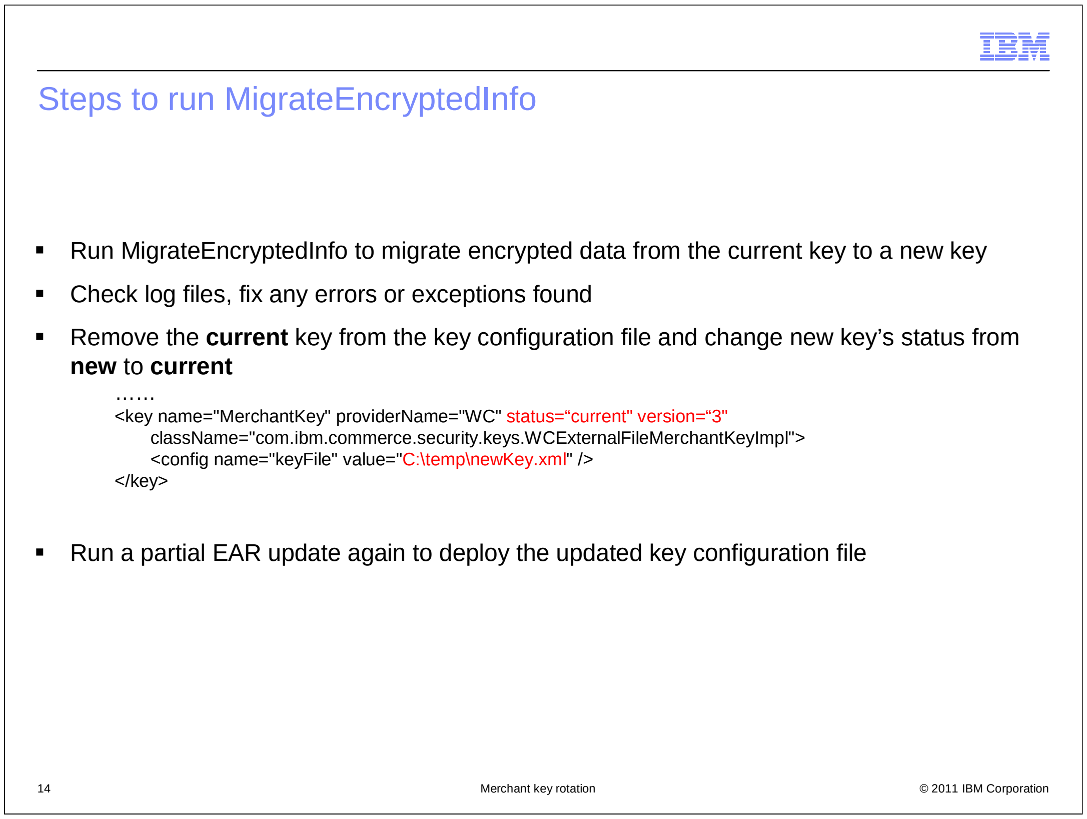
```
<Instance
  …
  KeysConfigFile="CustomKeys-FEP3.xml"
```

- Run a partial EAR update to deploy the custom key configuration file
  - Key configuration file must be placed under WebSphere Commerce application directory
    - For example: *WC_demo*.ear/xml/config/CustomKeys.xml
  - New key is available after application restart

13 Merchant key rotation © 2011 IBM Corporation

After you create the key configuration file, you need to register it in the server configuration file. Next, run a WebSphere Application Server partial EAR update to deploy the custom key configuration file to the runtime. The destination key configuration file must be in the WebSphere Commerce application directory.

During the EAR file update, the WebSphere Commerce application is stopped and restarted. This time is much shorter than restarting the application server. After the partial update completes, the new merchant key is available in the Encryption Keys registry for the new data encryption.



Steps to run MigrateEncryptedInfo

- Run MigrateEncryptedInfo to migrate encrypted data from the current key to a new key
- Check log files, fix any errors or exceptions found
- Remove the **current** key from the key configuration file and change new key's status from **new** to **current**

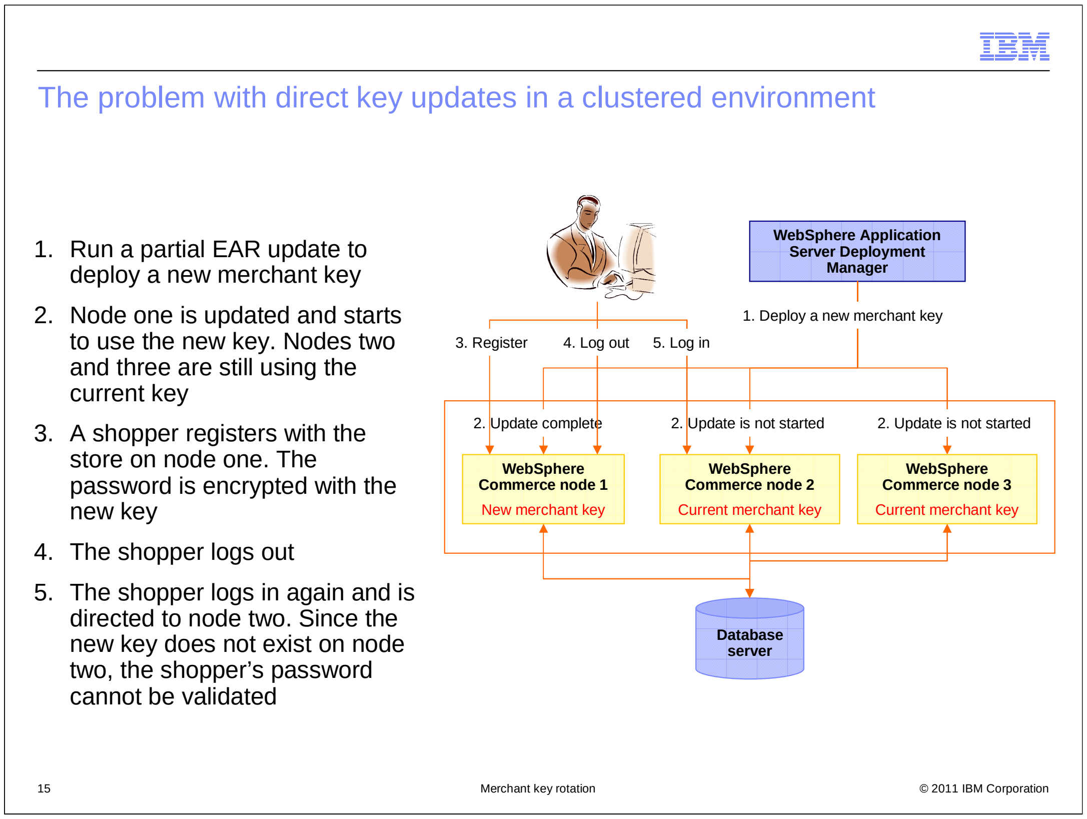
```
……
<key name="MerchantKey" providerName="WC" status="current" version="3"
     className="com.ibm.commerce.security.keys.WCExternalFileMerchantKeyImpl">
     <config name="keyFile" value="C:\temp\newKey.xml" />
</key>
```

- Run a partial EAR update again to deploy the updated key configuration file

14 Merchant key rotation © 2011 IBM Corporation

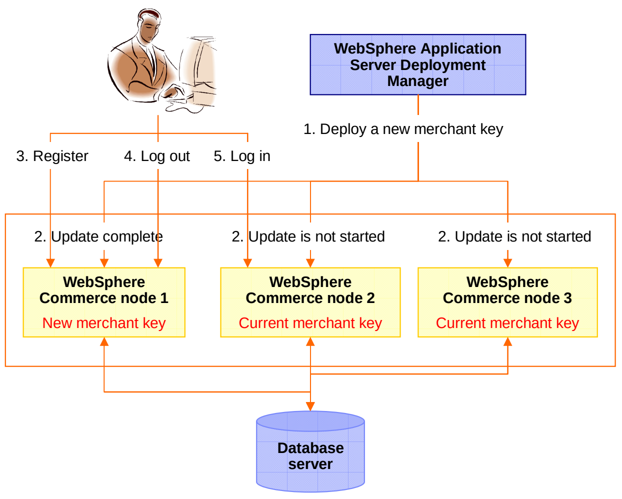
The steps to run the MigrateEncryptedInfo utility to migrate the encrypted data from the current key to a new key are shown here. The utility should be run after you make your new key available in the runtime.

After running MigrateEncryptedInfo, you need to check the log files for errors. After you ensure all data has been migrated successfully, you can remove the current key from the key configuration file and change the status of the "new" key to "current". You also need to run another partial EAR update to republish the key configuration file.



The problem with direct key updates in a clustered environment

1. Run a partial EAR update to deploy a new merchant key
2. Node one is updated and starts to use the new key. Nodes two and three are still using the current key
3. A shopper registers with the store on node one. The password is encrypted with the new key
4. The shopper logs out
5. The shopper logs in again and is directed to node two. Since the new key does not exist on node two, the shopper's password cannot be validated



15 Merchant key rotation © 2011 IBM Corporation

You cannot apply the single node steps to change the merchant key in clustered environment. Consider following this scenario:

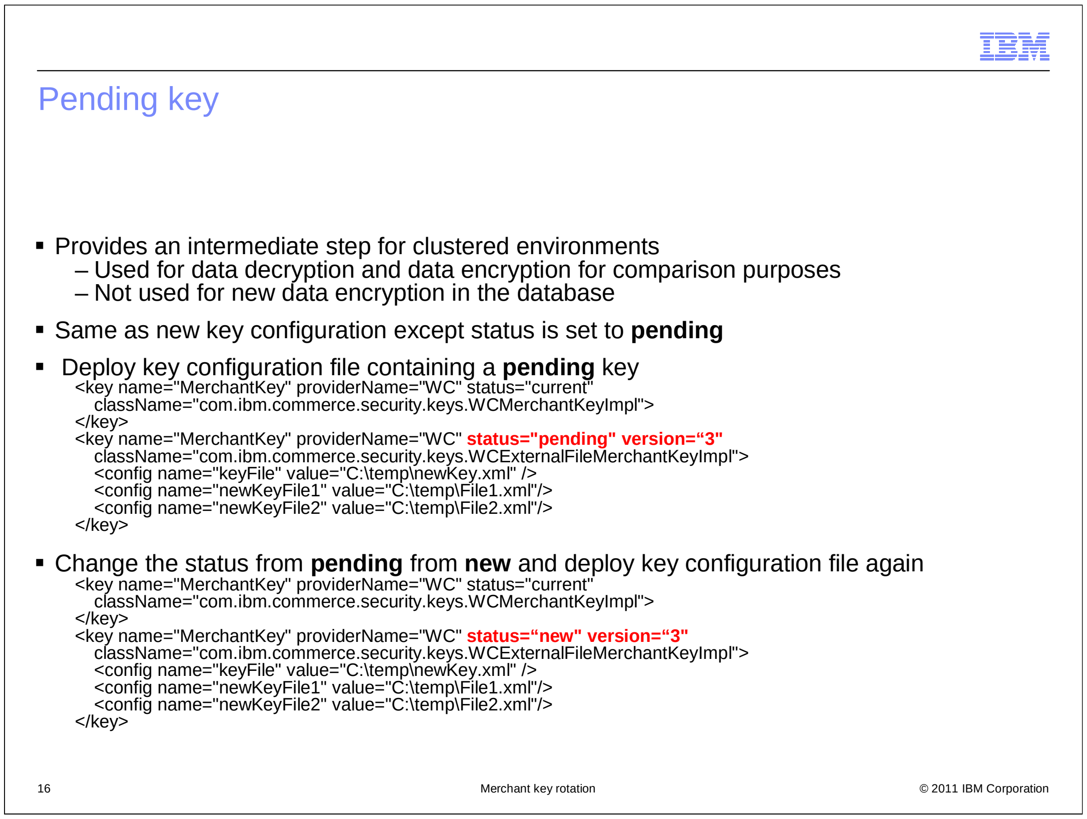
In step one, you run the partial EAR update to deploy your key configuration file. In a clustered environment, deployment is performed node by node. During the EAR update, the WebSphere Commerce application on node one is restarted.

In step two, node one starts to use the new merchant key, but the key update on nodes two and three has not been completed.

In step three, a shopper registers a new account on node one. Since the new merchant key is already available on node one, the shopper's password is encrypted with the new key and the encrypted password is saved in the database.

In step four the shopper logs out.

In step five, the shopper logs back in again right away. This time, the shopper's session is directed to node two on which the key update has not been completed yet. When the WebSphere Commerce application on node two tries to validate the password, it cannot find the same version of the key that was used to encrypt the password in its key configuration file. The shopper is not able to log on from node two.



Pending key

- Provides an intermediate step for clustered environments
  – Used for data decryption and data encryption for comparison purposes
  – Not used for new data encryption in the database
- Same as new key configuration except status is set to **pending**
- Deploy key configuration file containing a **pending** key

```
<key name="MerchantKey" providerName="WC" status="current"
   className="com.ibm.commerce.security.keys.WCMerchantKeyImpl">
</key>
<key name="MerchantKey" providerName="WC" status="pending" version="3"
   className="com.ibm.commerce.security.keys.WCExternalFileMerchantKeyImpl">
   <config name="keyFile" value="C:\temp\newKey.xml" />
   <config name="newKeyFile1" value="C:\temp\File1.xml"/>
   <config name="newKeyFile2" value="C:\temp\File2.xml"/>
</key>
```

- Change the status from **pending** from **new** and deploy key configuration file again

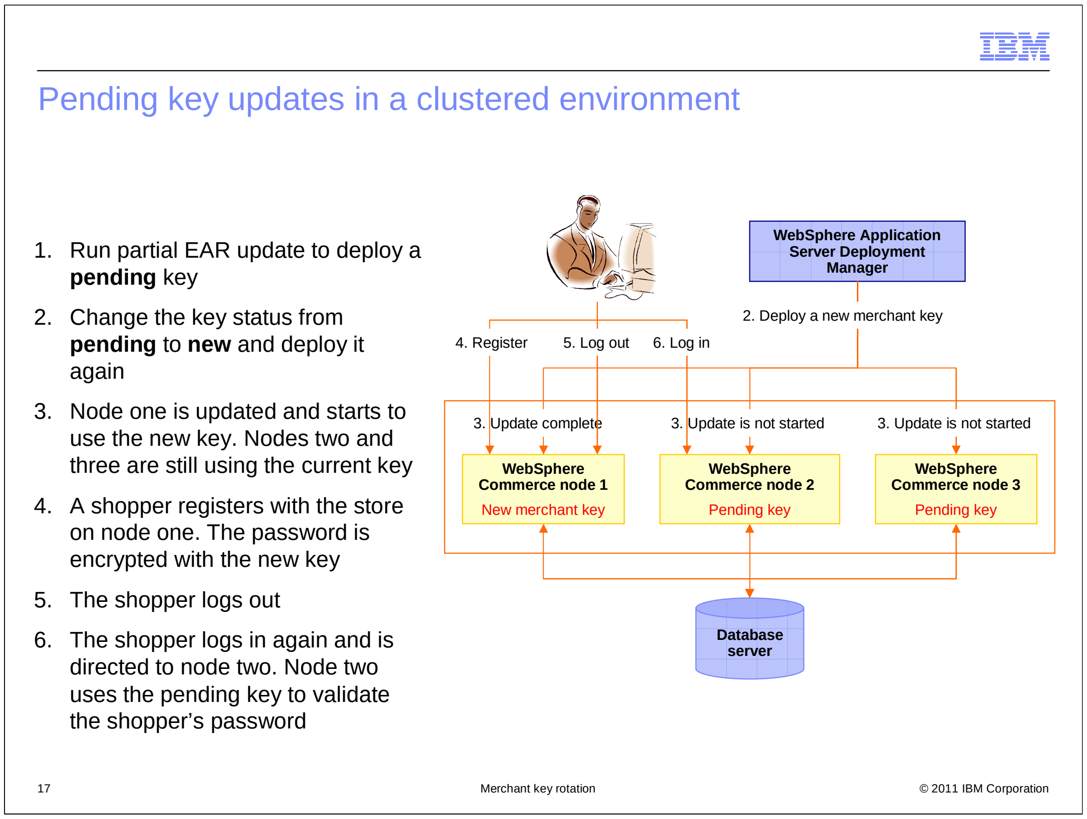
```
<key name="MerchantKey" providerName="WC" status="current"
   className="com.ibm.commerce.security.keys.WCMerchantKeyImpl">
</key>
<key name="MerchantKey" providerName="WC" status="new" version="3"
   className="com.ibm.commerce.security.keys.WCExternalFileMerchantKeyImpl">
   <config name="keyFile" value="C:\temp\newKey.xml" />
   <config name="newKeyFile1" value="C:\temp\File1.xml"/>
   <config name="newKeyFile2" value="C:\temp\File2.xml"/>
</key>
```

16 Merchant key rotation © 2011 IBM Corporation

To address the issue shown on the previous slide, Feature Pack 3 introduces another new type of key, a “pending” key.

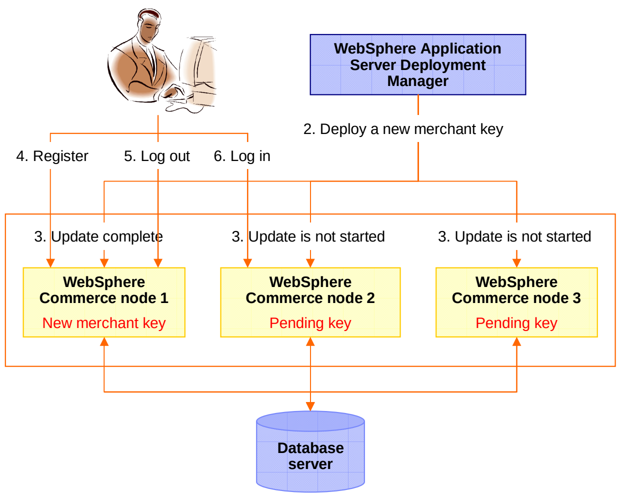
The “pending” status means the key is not active yet and it cannot be used for new data encryption. It can only be used for data decryption and data encryption for comparison with previously encrypted data such as password validation. A pending key is never used to encrypt data for writing back to the database.

Before you deploy a new key into a clustered environment, you put your new key into the key configuration file, but mark its status to be “pending”. You then deploy the key configuration file to the cluster. After the pending key is updated to all the nodes in the cluster, change the key’s status from “pending” to “new” and deploy the key configuration file again.



Pending key updates in a clustered environment

1. Run partial EAR update to deploy a **pending** key
2. Change the key status from **pending** to **new** and deploy it again
3. Node one is updated and starts to use the new key. Nodes two and three are still using the current key
4. A shopper registers with the store on node one. The password is encrypted with the new key
5. The shopper logs out
6. The shopper logs in again and is directed to node two. Node two uses the pending key to validate the shopper's password



WebSphere Application Server Deployment Manager
2. Deploy a new merchant key
4. Register
5. Log out
6. Log in
3. Update complete
3. Update is not started
3. Update is not started
WebSphere Commerce node 1
New merchant key
WebSphere Commerce node 2
Pending key
WebSphere Commerce node 3
Pending key
Database server

17 Merchant key rotation © 2011 IBM Corporation

Shown here is how the "pending" key can solve the problem shown earlier.

Step one is not shown in the diagram. In this step, you first deploy a pending key. Since the pending key is only used for decryption, no records in the database are encrypted with the pending key.

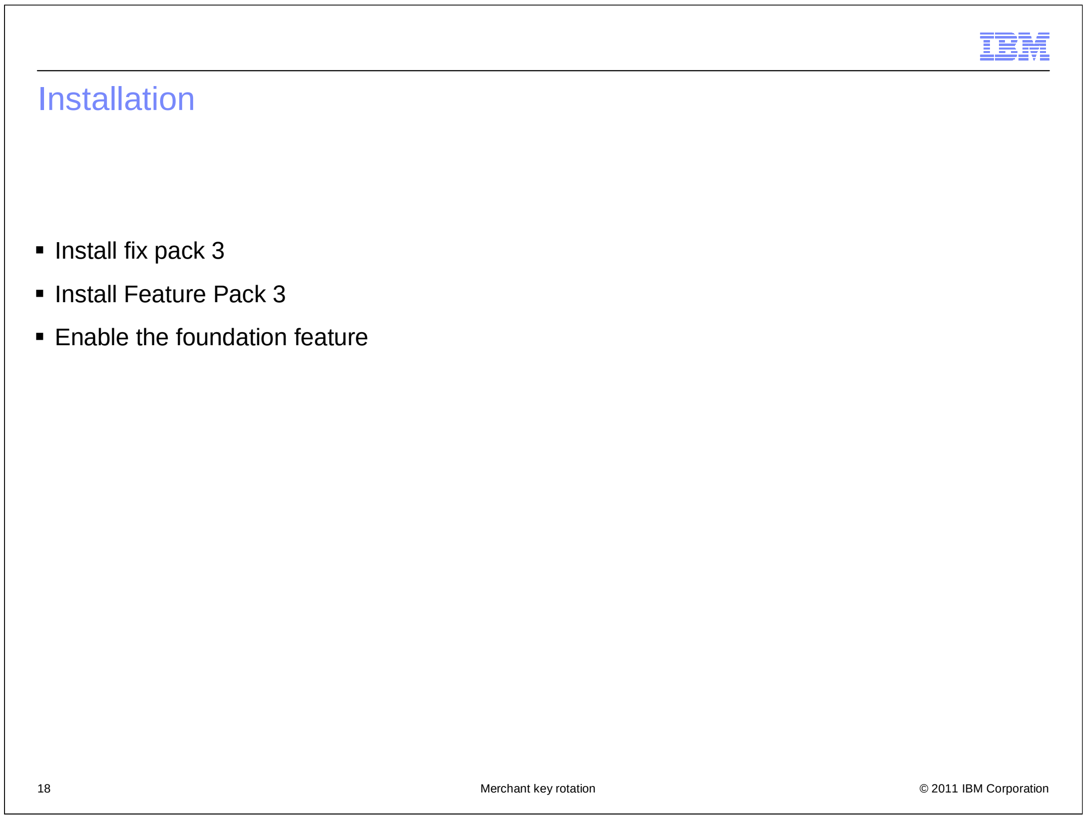
In step two you deploy the new key.

In step three, the key update has been completed on node one, but not on nodes two and three.

In step four, a shopper registers a new account on node one. The shopper's password is encrypted using the new key.

In step five, the shopper logs out from node one.

In step six, the shopper logs on again right away and is directed to node two. Since node two has the pending key whose version is the same as the new key which was used for password encryption, node two uses the pending key to encrypt the logon password. The encrypted password is then compared with the encrypted password in the database. The shopper is able to log on from node two if the password they provided is correct.

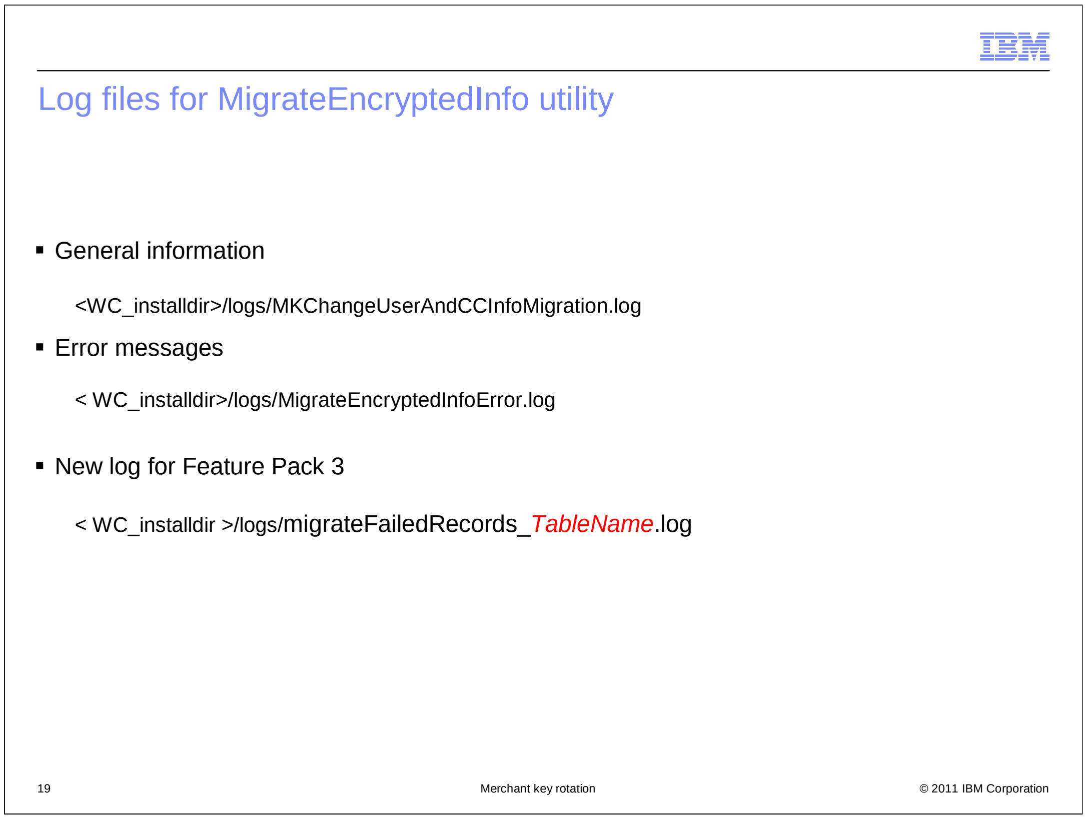


Installation

- Install fix pack 3
- Install Feature Pack 3
- Enable the foundation feature

18 Merchant key rotation © 2011 IBM Corporation

To get this new merchant key rotation feature, you need to install WebSphere Commerce fix pack 3, Feature Pack 3 and enable the foundation feature.



Log files for MigrateEncryptedInfo utility

- General information

  <WC_installdir>/logs/MKChangeUserAndCCInfoMigration.log
- Error messages

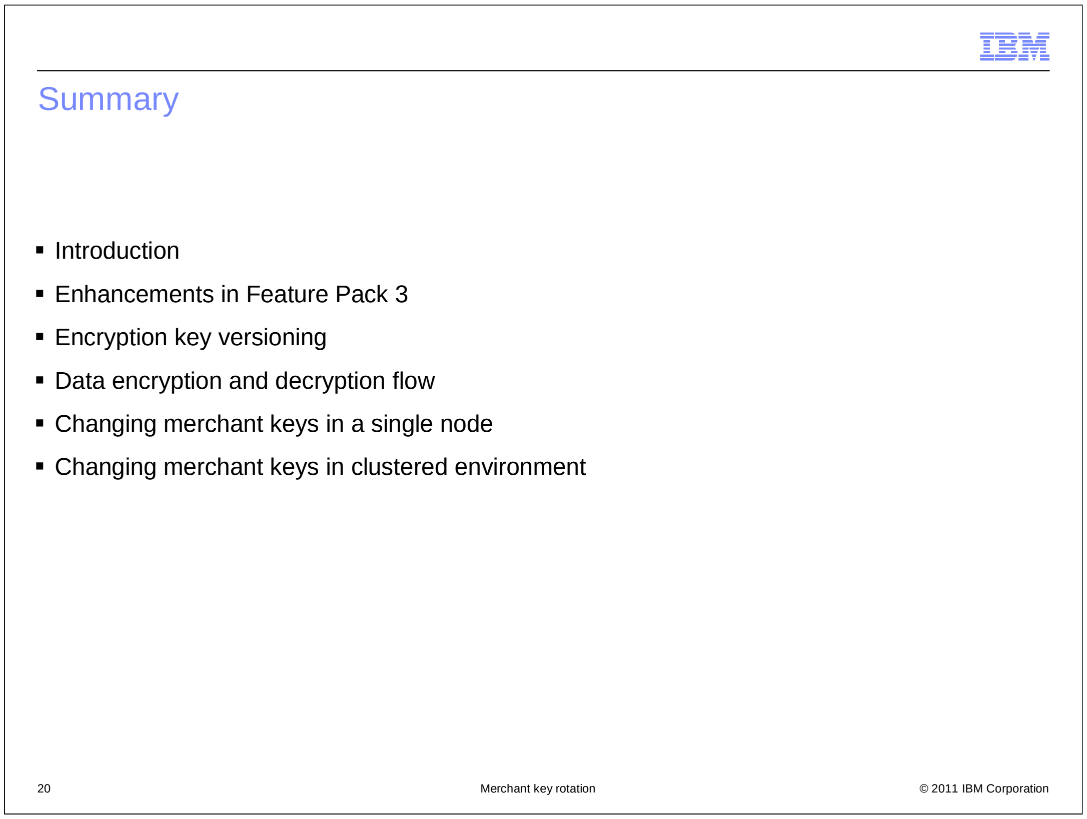
  < WC_installdir>/logs/MigrateEncryptedInfoError.log
- New log for Feature Pack 3

  < WC_installdir >/logs/migrateFailedRecords_*TableName*.log

19 Merchant key rotation © 2011 IBM Corporation

All the log files for MigrateEncryptedInfo utility are listed here. The first two are existing log files and have not changed in Feature Pack 3.

The third log file is new for Feature Pack 3. It contains the records that failed migration when running the MigrateEncryptedInfo utility. The log file name contains the table name so that you can easily tell which table's data generated the log file. This log file contains necessary information, such as the table's column name, unique record ID and failed data, so that you can easily identify the data that caused a failure.

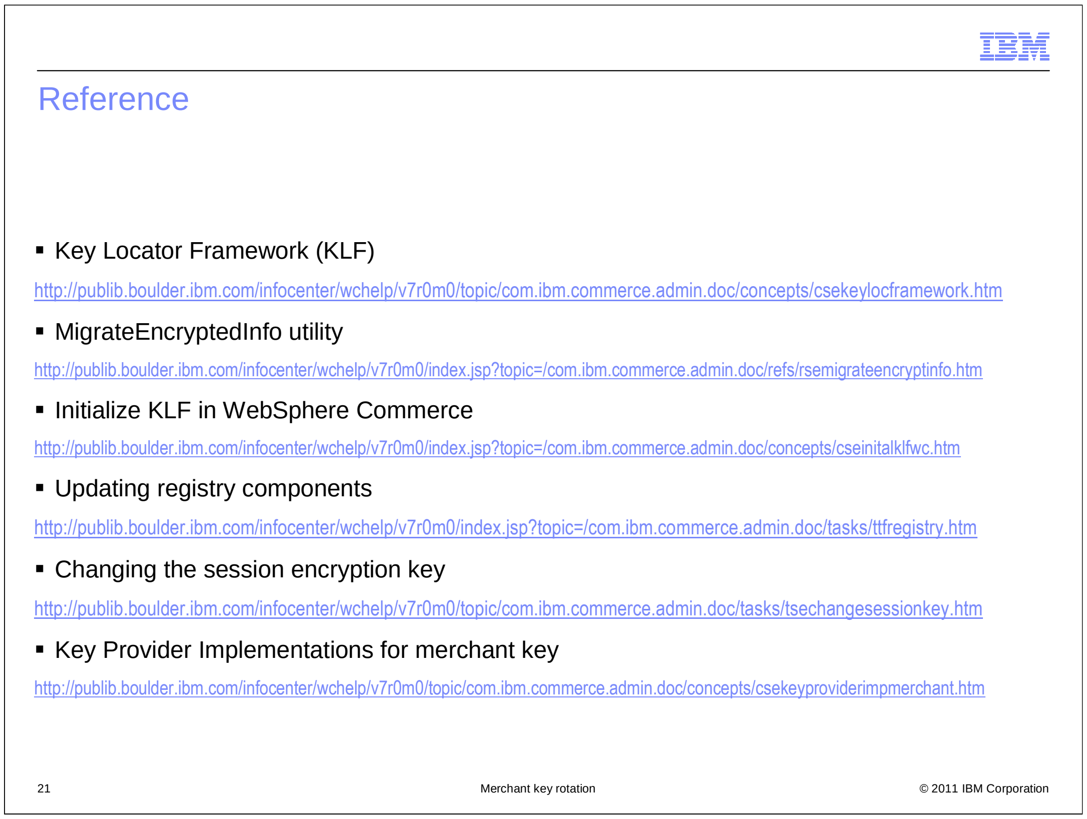


Summary

- Introduction
- Enhancements in Feature Pack 3
- Encryption key versioning
- Data encryption and decryption flow
- Changing merchant keys in a single node
- Changing merchant keys in clustered environment

20 Merchant key rotation © 2011 IBM Corporation

Feature Pack 3 enhances the merchant key solution so that you don't have to bring your site down to rotate the merchant key. To use this new solution, you need to assign a version number to each of your keys to differentiate the keys. The WebSphere Commerce runtime uses the version number to get the right key to do the data encryption and decryption. To achieve the zero-downtime merchant key rotation in a clustered environment, you must deploy a pending key first before you rotate your merchant key to a new key.



Reference

- Key Locator Framework (KLF)

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.admin.doc/concepts/csekeylocframework.htm

- MigrateEncryptedInfo utility

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.admin.doc/refs/rsemigrateencryptinfo.htm

- Initialize KLF in WebSphere Commerce

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.admin.doc/concepts/cseinitalklfwc.htm

- Updating registry components

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.admin.doc/tasks/ttfregistry.htm

- Changing the session encryption key

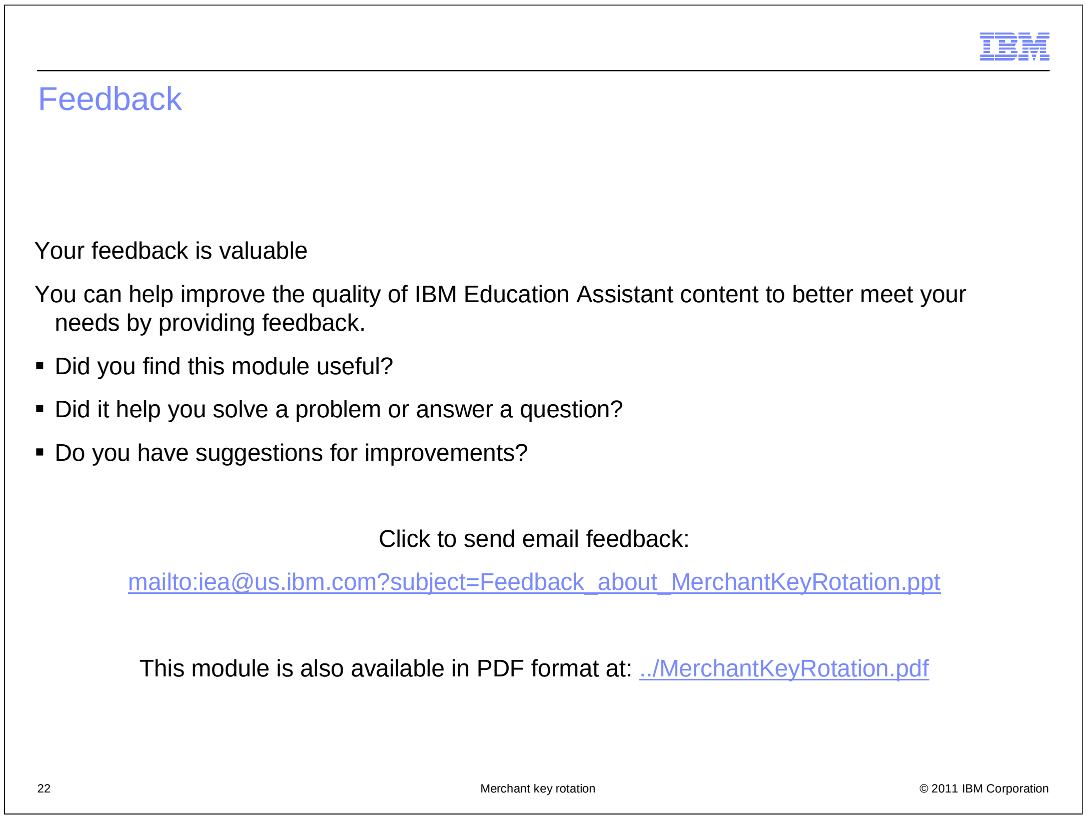
http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.admin.doc/tasks/tsechangesessionkey.htm

- Key Provider Implementations for merchant key

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.admin.doc/concepts/csekeyproviderimpmerchant.htm

21 Merchant key rotation © 2011 IBM Corporation

Here are some useful links.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_MerchantKeyRotation.ppt

This module is also available in PDF format at: ../MerchantKeyRotation.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information