CCDB-B2B-WS Migration Changes

Dependency Descriptor File

Since we pulled ccdb-common.jar out to it's own project, there is a now a requirement to declare a dependency on it. We have also needed to pull in dependencies for other JBoss provided libraries as part of the migration to JBoss's new modular classloader. Here is the deployment structure file:

```
<jboss-deployment-structure xmlns="urn:jboss:deployment-structure:1.2">
	<deployment>
		<dependencies>
			<module name="javax.wsdl4j.api"/>
		</dependencies>
	</deployment>
	<sub-deployment name="ccdb-b2b-ws-ejb.jar">
		<dependencies>
			<module name="com.advancestores.crm.ccdb.common" annotations="true" export="true"/>
		</dependencies>
	</sub-deployment>
</jboss-deployment-structure>
```

Other Changes

- CustomerLookupService.java
  - Updated @SecurityDomain from org.jboss.annotation.security.SecurityDomain to org.jboss.ejb3.annotation.SecurityDomain for EAP6 compatibility.
  - Added @RolesAllowed (javax.annotation.security.RolesAllowed) as this is needed by the updated LoginModule in order to authenticate a user.
  - Refactored some error handling to get rid of some dead code. (approx. lines 73 - 90)
- Regenerated all WSDL types code using JBOSS_HOME/bin/wsconsume.sh
  - Refactored DTO and DataMapper code accordingly
  - One thing to note is the way CXF generates Soap Exception types
    - When CXF sees an exception type define in WSDL/Schema, it generates two classes to capture this:
    - The first is the class that maps to the XML schema type. It names this according to the given name in the WSDL.
    - The second is the actual Throwable Exception object. This is named by adding a '_Exception' to the name of the above WSDL Type (i.e. WSException and WSException_Exception)
    - The Throwable object is a wrapper around the WSDL Type.
    - So, to return a Soap Error containing the proper xml structure you must use something like:
      - throw new WSException_Exception(new WSException(errorcode, message))
- WS-Security
  - This web service implements the WS-Security for protecting the identity and integrity of SOAP messages.
  - For information about how WS-Security works, see the overview of the Apache CXF Documentation.
  - Our ws-security policy information is as follows:
    - Sign the Body of every message (request and response) using sender's private key. This signature will be validated by the recipient using senders public key.
    - Encrypt the Body of every message (request and response) using recipient's public key. This will be decrypted by the recipient using recipient's private key.
    - Signature happens before encryption
  - We enforce this Policy by creating a ws-policy on the server.
    - Normally this would be part of the wsdl, but it was not in the old 4.2.3 version of the service and since we were promising minimal impact to other teams, we had to find a way around this
    - Our policy was created in a separate file: ccdb-b2b-ws-ejb/src/main/resources/META-INF/wssPolicy.xml
    - We activated this policy for our web service using the @Policies and @Policy annotation in com.advancestores.ccdb.b2b.service.CustomerLookupService
  - Additionally, we need to tell Apache CXF where to find our keys
    - We do this by referencing an endpoint config in the webservices subsystem of standalone-full.xml (@EndpointConfig(configName = "CRM-B2B-Endpoint")). See below for standalone config
    - The properties files and keystores referenced by the endpoint config live in JBOSS_SERVER_CONFIG_DIR/ws-security/
    - The properties "ws-security.*.username" properties refer to keys in our keystore, as it is a convention to name keys according to the 'user' or service sending the message (i.e. crmb2b_server, ecomm_b2b_client, ccdbtax_client, etc.)
    - NOTE: Notice the property "ws-security.encryption.username" property has a value of "useReqSigCert". This is a special term in CXF to instruct the server to use whichever public key was using in the signature validation to encrypt the response. This allows us to accomodate multiple Private keys, and therefore serve multiple clients, as long as the client's public key exists in our keystore.

Endpoint Config in standalone-full.xml

```
            <endpoint-config name="CRM-B2B-Endpoint">
                <property name="ws-security.signature.properties" value="file://${jboss.server.config.dir}/ws-security/crmb2b_server.properties"/>
                <property name="ws-security.encryption.properties" value="file://${jboss.server.config.dir}/ws-security/crmb2b_client.properties"/>
                <property name="ws-security.signature.username" value="crmb2b_server"/>
                <property name="ws-security.encryption.username" value="useReqSigCert"/>
                <property name="ws-security.callback-handler" value="com.advancestores.ccdb.b2b.policy.KeystorePasswordCallback"/>
            </endpoint-config>
```