Further restrictions can be imposed by using the `-noprint`, `-nocopy`, `-noedit` and `-noannotations` options, which disable printing, copying text, editing in Adobe Acrobat and making annotations, respectively.

3. Usage (embedded)

When FOP is embedded in another Java application you need to set an options map on the renderer. These are the supported options:

Option	Description	Values	Default
ownerPassword	The owner password	String	
userPassword	The user password	String	
allowPrint	Allows/disallows printing of the PDF	"TRUE" or "FALSE"	"TRUE"
allowCopyContent	Allows/disallows copy/paste of content	"TRUE" or "FALSE"	"TRUE"
allowEditContent	Allows/disallows editing of content	"TRUE" or "FALSE"	"TRUE"
allowEditAnnotations	Allows/disallows editing of annotations	"TRUE" or "FALSE"	"TRUE"

Note:
Encryption is enabled as soon as one of these options is set.

An example to enable PDF encryption in Java code:

```
Driver driver = new Driver();
driver.setRenderer(Driver.RENDER_PDF);
Map rendererOptions = new java.util.HashMap();
rendererOptions.put("ownerPassword", "mypassword");
rendererOptions.put("allowCopyContent", "FALSE");
rendererOptions.put("allowEditContent", "FALSE");
rendererOptions.put("allowPrint", "FALSE");
driver.getRenderer().setOptions(rendererOptions);
driver.setOutputStream(...
```

4. Environment

In order to use PDF encryption, FOP has to be compiled with cryptography support. Currently, only JCE is supported. JCE is part of JDK 1.4. For earlier JDKs, it can be installed