Getting Started (svn-Version)

Downloading and Building the Source

Before you can use Abdera, you need to download and build the source. Instructions for bulding are available in the Builder's Guide.

Parsing Atom Feed and Entry Documents

The Abdera parser is capable of handling Atom Feed and Entry documents, Atom Publishing Protocol Introspection Documents, and any other arbitrary, well-formed XML document.

```
Abdera abdera = new Abdera();
Parser parser = abdera.getParser();
// or
Parser parser = new MyCustomParser();
```

The Parser will automatically detect the kind of document being parsed and will return an appropriate org.apache.abdera.model.Document.

```
InputStream in = ...
Document<Feed> feeddoc = parser.parse(in);
Document<Entry> entrydoc = parser.parse(in);
Document<Service> servicedoc = parser.parse(in);
```

Parsing an Atom FeedURI uri = new URI("http://example.org/feed.xml");

```
InputStream in = uri.toURL().openStream();
Document doc = parser.parse(in, uri);
```

The uri parameter on the parse method establishes the Base URI for the parsed document and is used as the basis for relative URI resolution throughout the document.

Parsing an Atom EntryURI uri = new URI("http://example.org/entry.xml");

```
InputStream in = uri.toURL().openStream();
Document doc = parser.parse(in, uri);
```

Configuring the ParserURI uri = new URI("http://example.org/feed.xml");

```
InputStream in = uri.toURL().openStream();
ParserOptions options = parser.getDefaultParserOptions();
options.setCharset("utf-8");
//.. set other parser options
Document doc = parser.parse(in, uri, options);
```

Creating Atom Feed and Entry Documents

Atom Feed and Entry documents are created using instances of the org.apache.abdera.factory.Factory interface.

```
Abdera abdera = new Abdera();
Factory factory = abdera.getFactory();
// or
Factory factory = new MyCustomFactory();
```

Creating Atom Feed DocumentsFeed feed = factory.newFeed();

```
feed.setId("tag:example.org,2005:/myfeed");
feed.setTitle("My Example Feed");
// .. set other feed properties
Document doc = feed.getDocument();
doc.writeTo(System.out); // serializes the feed document to System.out
```

Creating Atom Entry DocumentsEntry entry = factory.newEntry();

```
entry.setId("tag:example.org,2005:/myentry");
entry.setTitle("My Example Entry");
// .. set other feed properties
Document doc = entry.getDocument();
doc.writeTo(System.out); // serializes the entry document to System.out
```

The Feed Object Model

The Feed Object Model (FOM) is a set of interfaces designed around the Atom Syndication Format data model. The object model provides the API by which Atom documents are read and created.

```
URI uri = ...
InputStream inputStream = ...
Document doc = parser.parse(inputStream, uri);
Feed feed = doc.getRoot();
URI id = feed.getId();
Text.Type titleType = feed.getTitleType();
String title = feed.getTitle();


List entries = feed.getEntries();
for (Entry entry : entries) {
URI entryId = entry.getId();
Text.Type entryTitleType = entry.getTitleType();
String entryTitle = entry.getTitle();
}
...
```

sing XPath

As an alternative to navigating the Feed Object Model manually, developer's may use XPath to query a parsed Document.

```
URI uri = ...
InputStream inputStream = ...
Document doc = parser.parse(inputStream, uri);


// Select the id of the document
XPath xpath = abdera.getXPath();
String id = xpath.valueOf("/a:feed/a:id", doc);


// Select all entries from the document
List entries = xpath.valueOf("//a:entry", doc);
for (Iterator i = entries.iterator(); i.hasNext() {
Entry entry = (Entry)i.next();
//...
}


// Determine if a feed contains a specific extension
boolean hasFoo = xpath.booleanValueOf("//x:foo", doc);


// The XPath support works on any element in the FOM
Entry entry = (Entry)xpath.selectSingleNode("//a:entry", doc);
String id = xpath.valueOf("a:id", entry);
```

sing Extensions

The Feed Object Model is designed to fully and dynamically support extensions to the Atom Feed format.

```
Feed feed = factory.newFeed();
// ... set other feed properties
feed.addSimpleExtension(
new QName("urn:foo", "myExtension", "a"),
"This is an extension"
);


Link link = feed.addLink("http://example.org");
link.setAttributeValue(
new QName("urn:foo", "myAttribute", "a"),
"My Attribute");
```

This results in the following Atom feed:

...
This is an extension

As an alternative to using the dynamic extensions support built into the Feed Object Model, developers may configure static extensions using the org. apache.abdera.factory.ExtensionFactory mechanism. Extension Factories are covered in detail in the Developer's Guide.

Signing and Encrypting Atom Documents

Atom Feed and Entry documents may be digitally signed and/or encrypted by using the *optional* Abdera Security module. The security module currently depends on the Apache Xerces, Apache XML Security Projects and the Bouncy Castle Java cryptography implementation.

Digitally Signing Atom DocumentsAbdera abdera = new Abdera();

```
AbderaSecurity absec = new AbderaSecurity(abdera);
Factory factory = abdera.getFactory();
Feed feed = factory.newFeed();
PrivateKey myPrivateKey = ...
X509Certificate myX509Cert = ...
Signature sig = absec.getSignature();
SignatureOptions options = sig.getDefaultSignatureOptions();
options.setSigningKey(myPrivateKey);
options.setCertificate(myX509Cert);
feed = sig.sign(feed, options);
//any modifications to the feed after this point will break the signature
```

Encrypting Atom DocumentsAbdera abdera = new Abdera();

```
AbderaSecurity absec = new AbderaSecurity(abdera);
Feed feed = factory.newFeed();
Key kek = ... // Key encryption key
Key dek = ... // Data encryption key
Encryption enc = absec.getEncryption();
EncryptionOptions options = enc.getDefaultEncryptionOptions();
options.setKeyEncryptionKey(kek);
options.setDataEncryptionKey(dek);
options.setIncludeKeyInfo(true);
Document doc = enc.encrypt(feed.getDocument(), options);
doc.writeTo(System.out); // outs the encrypted XML
```