Packaging section 2

Directory format and meta data.

1. Scope

Packaging of documents and related data is done in one file. Several (XML) streams representing data for the office applications will be combined into one package. This document will talk about the naming and positioning of files inside the package. Section one gives good arguments about the file format the whole should be compressed with, and is effectively out of scope of this document. Section one can be found at: http://xml.openoffice.org/package.html

2. Definitions

For clarity a defintion of a few terms for use within this document:

package	A package is one file on the filesystem that combines a mimimum of 1 document. The package is what contains all that we talk about in this document
document	A document comprises everything a user considers to part of his text, spreadsheet or presentation. In general this will include several images or OLE objects in addition to the main content. Each one of these is a file. If images are linked into the document, they are not considered part of it. Whether an image is linked or embedded into the document is determined by the user. Due to limitations in the current OLE handling, OLE objects must always be embedded.
file	A file is one xml stream or one picture inside the package

3. Introduction

Any office document needs to store its data (a text for example) in the package. Any package contains one or more files. Each application will decide for itself which files it will create and which names and content these files provide. To make the exchange of data between a broad set of office applications possible the packager should follow a limited set of rules on the naming of the directories and the files inside the package.

Any application of file manager opening the application should quickly be able to find information like:

- Which document type(s) are stored in this package
- Who is the author of the document.
- Is the document encrypted

Storing the content and any extra data for our document inside the zip requires first of all to identify the main document. On the openoffice mailing list, it was first proposed to be a document with a specific name, later quickly overruled to allow any name, but to register the name of the document in a meta-data file also contained in the package.

The discussion focussed on the format of the meta data document and the result is a document standard that will allow total flexibility in naming and archiving inside the archive.

The meta data file will have a specific and unique name inside the zip to allow software using this standard to access the info. The file will be an xml-stream because that allows any 3th party to add extra information without problems. We also suggest using namespaces to avoid naming conflicts when expanding the file format.

4. The requirements

- An office package will always have exactly one main file, this file is the one that is handed over to the application, the meta data file should be able to identify that file.
- The file-type (or mime-type) of the main document can be anything the author wants, discussion on the exact content are outside the scope of this document.
- The package should store any number of 'help files' that belong to the document, an quick example is an embedded picture in the document.
- One file in the package has a default name, it will always be present. This file contains meta data of the package with information on how the main document can be found and opened.
- Nesting documents; the practice which allows documents to be present inside other documents. This is only usefull if the application that opens the nested document does not know it opened the document inside another application. In other words; the nesting has to be done in such a way that a nested application can receive a URL from the package as a main document and use that exactly like it is the only main document in the package.

5. Meta data file basic, storing the description for every file in the package

It is crucial that the main document in the package can be identified so the application knows which file to open. Another wish is that further files in the package can be identified without opening them. In all modern Operating Systems and file managers the concept of mime types is used, or can be used to identify files.

The major problem with mime types is that they have to be officially registered to allow a claim of that specific mime-type for an application. The registration process is something most open source office applications will most likely not be able to do.

While it is not a problem to allow mime types which are not standardized to be used the usage can pose problems in the future.

A second method of identifying files can be done via namespaces. This is slightly different from the xml namespaces but provides the same concepts. The `namespace` tags as seen below the `main` tag in the following example specify what kind of information the xml stream provides.

```
<package xmlns="http://openoffice.org/package/v1">
    <document mime_type="application/x-kword" xmlns:ooo="http://openoffice.org">
        <main href="maindoc.xml" mime_type="application/xml">
            <namespace>http://www.kde.org/kword</namespace>
            <namespace>http://www.kde.org/kword/annotation</namespace>
        </main>
        <entity href="styles.xml" mime_type="application/xml" ooo:type="styles">
            <namespace>http://www.kde.org/kword</namespace>
        </entity>
        <entity href="pictures/pic.gif" mime_type="image/gif"/>
        <document mime_type="application/x-kspread">
            <main href="part0/file1" mime_type="application/xml">
                <namespace>http://www.kde.org/kspread</namespace>
            </main>
            <entity href="part0/pictures/pic.gif" mime_type="image/gif"/>
        </document>
    </document>
</package>
```

The `document` tag contains a `mime_`type attribute. This mime type is used to identify the application that can load the document. If a file browser wants to find out what it should do on opening this package; it can stop there. (see also 'Magic' below).

In the document exactly one `main` tag is present. This is used by the hosting application as its main document-file. Any other file in the package will be referenced from within the main document. The mime type in this tag is for the target application and should be the officially registered mime type for the type of file used.

The `namespace` tags are optional but should be provided when more then one application is able to read the XML stream (assuming the main file or entity is an XML stream). When for example SVG or MathML based content is embedded in the XML stream it is wise to add their namespace so external applications can harvest information quickly.

Each file stored in the package for this document receives an entity line. Standardized XML streams inside the OpenOffice standard can be identified with an `ooo:type` attribute. In the above example all applications able to read the `ooo:type="styles"` file type can quickly find this kind of data in a package. The styles file type is an example type and non-existent at the time this document was writen.

As stated in the requirements, a document should be able to have sub-documents. Embedding of document is others is done by simply allowing documents to be a sub element of another document tag.

6. The reason why all meta data is in one file

Other meta data like author and the number of pages of the document can be used to search for documents (pacakages actually) more easily. We first considered adding a second document-info file for this; but as we found out that things like a history and encryption info should also be possible it started to seem silly to have a new file for each kind of meta data.

The following statement was made; "*metadata is metadata. I suggest to define a single metadata block which is* *extensible. The define a predefined set for which you can write a DTD. This is the way most extensible spec are written at W3C at the moment.*" (Daniel Veillard) We setteled on that.

In light of this the DTD of the XML example above is given here with the possebility to extend it by adding tags in a different namespace preferably at the same level as the top-most `<document>` tag.

See the DTD at the end of the document.

7. Meta data file basic enhancements (#pages) so 3th party SW can read it and display pretty data about the doc

TODO: Future enhancement; adding a number of possible blocks at the same level of the hierarchy as <document>

8. Meta data file gpg stuff, for encryption and signing docs

Same as 7 (see also http://www.w3.org/TR/xmldsig-core/)

9. Meta data file cvs stuff, for revision control and possibly locking

Same as 7

10. one subdir for every sub-doc, so the doc hierarchy is the same as the package hierarchy

TODO

11. Magic

In most mature environments the type of a file can be asked from the file itself; while we are waiting for the widespread introduction of filesystems that store the filetype we are going to store the file type elsewere. In Unix the idea of magic numbers is in wide spread use; where a number of bytes in the beginning of the file have a certain value at a certain offset for this specific filetype. The specifics are configured in a system wide 'magic' file that describes the offsets and text and couples it to a mime-type.

In OpenOffice packaging the wrapper file is a zip file, as discussed above. There is an identification for a zip file present for all known zip software. Since we do an extention on top of the zip file format we have to add another identifying mark for our documents.

Identifying a document as a document for a specific application means to associate the file with a mime-type which the application registered. In the zip file format the only way to allow a default offset (which the magic-protocol requires) is by using the first filename in the zip; since the zip file format tells us that the first filename will be described at a default offset in the zip pacakge. That offset is 30 bytes from the beginning.

To evade problems when unpacking the zip using regular tools a specific filename will be used for all documents packaged in the openoffice packaging standard. The mime type for the application will be the contents of that first file.

The filename we use is 'mimetype'

An example mime type is '`application/x-kword`'

For this specific example the magic file has to be:

```
# kword
0    string   PK\003\004                 application/x-zip
>30  string   mimetype
>>38 string   application/x-kword        application/x-kword
```

12. DTD complete with namespace etc.

The meta file containing the most basic data for managing the package:

```
<!ELEMENT package (document) >
<!ATTLIST package >
```

```
    <!ELEMENT document (main|entity*|document*) >
    <!ATTLIST document
       mime_type CDATA #REQUIRED>

    <!ELEMENT main (namespace*) >
    <!ATTLIST main
       href CDATA ID #REQUIRED
       mime_type CDATA #REQUIRED>

    <!ELEMENT entity (namespace*) >
    <!ATTLIST entity
       href CDATA ID #REQUIRED
       mime_type CDATA #IMPLIED>
```

Thomas Zander (zander@kde.org)

13. History:

5-Sept 2001 First mockup

22-Apr 2002 added file-magic.

5-May 2002 rewrite of lots of stuff which was unclear. Put it in KWord, added DTD for file types.