From: Mark Zbikowski
Sent: Thursday, January 02, 1997 4:48 PM
To: David Vaskevitch; Steve Madigan; Jim Allchin (Exchange)
Subject: RE: Storage Unification / Webification Memo

From: Jim Allchin (Exchange)
Subject: FW: Storage Unification / Webification Memo

I would be interested in your views on this.

Well... I worked with Darryl on this for a while... many of his conclusions parallel what we were discussing in ToddW's collaboration meeting (basically, get bidirectional IStorage/ADO ⇔ existing protocol communications).

I've (belatedly) come to the conclusion that NTFS.SYS (or OFS.SYS) which is the "core file system" cannot be all things to all people. Product requirements, architectural restrictions, and development logistics largely prevent this. You can, however, get 100% of the feature set by broadening the notion of "File System" from NTFS.SYS to encompass the services that Darryl identifies in Figure 1 (section 5.1). Rather than calling this collection a "File System" as he does, which is guaranteed to confuse people, call it "Storage Services".

As far as I can tell, the storage services comprise these parts:

1. Stream storage.
2. Structured record storage
3. Content index
4. Transaction management

These pieces, as long as they are bound tightly enough, should provide the facilities needed by the clients as well as provide enough logical separation that we could actually develop and ship them. The pieces would be:

1. Ntfs
2. Query/Ci
3. Sql
4. OLE

There would need to be glue code in Ntfs/Query/Sql to make sure that there would be a 1-1 correspondence between the "File Storage Table" and the stream storage in Ntfs. The relationship between properties in a document and the fields in the corresponding row need to be tightly managed.

There's some transaction isolation problems in the directory implementation we have today. These would need to be worked out.

Finally, there are some issues about getting the plumbing (as in protocol binding) to work efficiently and reliably. I'd hate to shuttle stream data through an ODBC session or retrieve tabular, structured data through SMB's.

Back to Darryl...

His discussion of the data model (section 5.2) is largely the original Cairo plan. There are some problems about guaranteeing uniqueness of GUIDs on documents created on Win95++.

Darryl goes further and discusses browser architecture (sure sounds like the old Cairo shell) (section 5.0). He (and apparently Bill) still has a bee in his bonnet about "outline view" of a document that exposes the structure to the browsing public. SteveM's structure-factorization memo addresses some of this at a rational level, that of having applications factor out common structure *presentation* and make it a separate component. Darryl (Bill?) still want to do an "outline view" in a completely document-independent way, which I believe is impossible to do in a way that the user could understand.

Worse, this breaks encapsulation bigtime, for some small benefit. I'd rather the apps get factored into structure-viewers and let them deal with this issue.

I really don't like the idea, (5.8.2), of "links are objects". He makes them first-class objects in both the programming model (that's OK) and in storage (not OK, encapsulation violation). Finally, he spends quite a bit of time on link architecture that seems to be more of a user model than a architecture or implementation. The model itself seems to be beyond what most users would tolerate from a complexity standpoint.

The desire to make links robust seems to fall into a similar trap that the OLE group is in. The problem of locating a document after inter-machine moves is tossed to a "crawler" who indexes objects based on their GUID. When mending a

broken link and simple efforts (looking on the same server and using Query/CI there) fail, you then consult the master-crawler index. Given the crawling (and by design, *out-of-date*) nature, it will likely never have the true location of a document.

Annotation-by-links/private-web seems to be a separate application architecture that allows links to be kept in a database "on the side"... Of course, there are plenty of garbage collection issues here.

Finally, Darryl doesn't go into the degree of transactioning that is supported. This was the primary downfall of structured storage on OFS and it is a significant design/architecture item.

All in all, it is nice to see another person come up with the same data model as Cairo...