CIOFECA FORENSICS

MOBILE APPLE NOTES

Revisiting Apple Notes (4): Gallery Objects

20 Jan 2020 · 9 mins read

TL;DR: Apple Notes has a few bespoke embedded objects which are messier than the Easy Embedded Objects previously explained. This post covers how to piece back together the final complex embedded object, the Apple Notes Gallery. I got this wrong a few times and in fixing it, learned how I was messing up other aspects of Notes.

Background

If you haven't read the earlier post about Easy Embedded Objects, please go do that now. It will explain the background assumptions of this work and show how this works on some very straight forward objects. The previous post about Embedded Tables will dive into how to examine the protobuf in greater detail and is assumed as a foundation for this post.

Below are the steps to rebuild `com.apple.notes.gallery` objects. These are specific to Notes and come about when you add an image and use the "Scan Document" feature to do so. If you still don't want to do this by hand, feel free to check out the Apple Cloud Notes Parser, which had a lot of rewrites recently to handle the Gallery object correctly.

com.apple.notes.gallery

The Type UTI of `com.apple.notes.gallery` represents a set of images that were generated from the 'Scan Documents' feature when you add an image to a Note. The idea is to take a picture of each page and put them all into one bundle to swipe through. Along the way, Notes parses the text it sees in the image.

In attempting to properly display this type of embedded object, I ended up realizing I had made a lot of mistakes in my assumptions along the way. For example, diving into here is what made me realize that a lot of objects you would not expect have thumbnails associated with them, such as URLs and PDFs (almost clear enough to read the words). In addition, because my initial test data had one image, I assumed that was always the case, but once I generated better test

data to figure out the protobuf, I ended up realizing I would have been skipping all the other images. This led to a fairly large rewrite of embedded objects in general (to find thumbnails for anything), output (to show everything that has child objects), and galleries (they show all images).

Rebuilding Tables

The first step of rebuilding a `com.apple.notes.gallery` object is the same as `com.apple.notes.table`. You will start with the Note protobuf, find the appropriate Attachment identifier for the Gallery, and pull the relevant data from the `ZICCLOUDSYNCINGOBJECT.ZMERGEABLEDATA1` field, recognizing that it is a GZipped protobuf following the same .proto file as Tables. We'll use this as an example Note:

```
root:
    2 <document> = document:
        3 Note = note:
            2 Note Text = "Gallery in depth\n\nFour images, taxes, passport, fc, mou
            5 Attribute Run = attribute_run: (1 Length = 17)
            5 Attribute Run = attribute_run:
                1 Length = 1
                12 Attachment Info = attachment_info:
                    1 Attachment Identifier = "4382F325-3332-4896-BB3E-63EB8961AA6D"
                    2 Type UTI = "com.apple.notes.gallery"
            5 Attribute Run = attribute_run: (1 Length = 40)
```

... which means this query will get our GZipped Gallery protobuf:

```
SELECT ZICCLOUDSYNCINGOBJECT.ZMERGEABLEDATA1
FROM ZICCLOUDSYNCINGOBJECT
WHERE ZICCLOUDSYNCINGOBJECT.ZIDENTIFIER="4382F325-3332-4896-BB3E-63EB8961AA6D"
```

... resulting in this, once gunzipped and parsed (as always, this is trimmed for readability):

```
mergeabledata1_proto:
    2 Mergeable Data Object = mergeable_data_object:
        3 Mergeable Data Object Data = mergeable_data_object_data:
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                5 List = list:
                    1 List Entry = list_entry: (2 Object ID = object_id(6 Object Inc
                    1 List Entry = list_entry: (2 Object ID = object_id(6 Object Inc
                    1 List Entry = list_entry: (2 Object ID = object_id(6 Object Inc
                    1 List Entry = list_entry: (2 Object ID = object_id(6 Object Inc
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                13 Object Map = mergeable_data_object_custom_map:
                    1 Type = 1
                    3 Map Entry = map_entry:
                        1 Key = 0
```

```
                2 Value = object_id:
                    4 String Value = "B6D010C3-4E14-4E98-8E3B-60C45E6A4E8A"
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                9 <chunk> = message:
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                13 Object Map = mergeable_data_object_custom_map:
                    1 Type = 1
                    3 Map Entry = map_entry:
                        1 Key = 0
                        2 Value = object_id:
                            4 String Value = "5A60D806-A02D-4B2D-B1A4-69948CB8E1FE"
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                9 <chunk> = message:
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                13 Object Map = mergeable_data_object_custom_map:
                    1 Type = 1
                    3 Map Entry = map_entry:
                        1 Key = 0
                        2 Value = object_id:
                            4 String Value = "93A07935-3855-4BF2-9E4A-07BADD859FC6"
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                9 <chunk> = message:
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                13 Object Map = mergeable_data_object_custom_map:
                    1 Type = 1
                    3 Map Entry = map_entry:
                        1 Key = 0
                        2 Value = object_id:
                            4 String Value = "E7A542B9-C005-423D-822A-A2E655BE30E1"
            3 Mergeable Data Object Entry = mergeable_data_object_entry:
                9 <chunk> = message:
            4 Mergeable Data Object Key Item = "self"
            5 Mergeable Data Object Type Item = "com.apple.CRDT.NSNumber"
            5 Mergeable Data Object Type Item = "com.apple.CRDT.NSString"
```

This format is simpler than the table in that a gallery is one-dimensional, you only need to know which embedded pictures go in it. You can apply the same methodology we learned for Tables to identify that all of these have the same “Key Item” of “self” and use an “Object ID” to identify the `ZICCLOUDSYNCINGOBJECT.ZIDENTIFIER` as a `String Value` for each of the images contained therein. By looping over each of these `Mergeable Data Object Entry` items that have a `Custom Map`, you’ll find all of the images that make up this gallery.

When you query `ZICCLOUDSYNCINGOBJECT` for those `ZIDENTIFIER`s, you’ll find that each is of type `public.jpeg` and has OCR applied to the images. See the first embedded objects post for more about both handling `public.jpeg` and OCR.

```
SELECT ZICCLOUDSYNCINGOBJECT.Z_PK, ZICCLOUDSYNCINGOBJECT.ZTYPEUTI,
       ZICCLOUDSYNCINGOBJECT.ZIDENTIFIER, ZICCLOUDSYNCINGOBJECT.ZOCRSUMMARY
FROM ZICCLOUDSYNCINGOBJECT
WHERE ZICCLOUDSYNCINGOBJECT.ZIDENTIFIER IN ("B6D010C3-4E14-4E98-8E3B-60C45E6A4E8A",
```

```
"5A60D806-A02D-4B2D-B1A4-69948CB8E1FE",
"93A07935-3855-4BF2-9E4A-07BADD859FC6",
"E7A542B9-C005-423D-822A-A2E655BE30E1")
```

Z_PK	ZTYPEUTI	ZIDENTIFIER	ZOCRSUMMARY
206	public.jpeg	5A60D806-A02D-4B2D-B1A4-69948CB8E1FE	150 TEARD 150 TEARS
205	public.jpeg	93A07935-3855-4BF2-9E4A-07BADD859FC6	PASSPORT United States of America
200	public.jpeg	B6D010C3-4E14-4E98-8E3B-60C45E6A4E8A	
204	public.jpeg	E7A542B9-C005-423D-822A-A2E655BE30E1	B100 Full-size Corded Mouse logitech logitech logitech® logitech*

Conclusion

I was glad to tackle this type of embedded object last because it helped me firm up my understanding of all the previous types, along with thumbnails. Because the Gallery object data format could hold just about anything and is new in iOS 13, I wouldn't be surprised to see additional uses of it in future releases, beyond just scanning documents. There is no really good reason for users not to be able to organize their pictures into Notes using a Gallery. Understanding these bespoke Notes types will likely only get more important the longer Apple develops the application.

Previous
‹ Revisiting Apple Notes (3):...

Next
Proper Preparation Prevents... ›