Review of Cloud Imaging Model

a. ftp://ftp.pwg.org/pub/pwg/cloud/wd/wd-cloudmodel10-20140128.docx

b. Q: How does client discover cloud services? Added paragraphs on this and showed lines on diagram

A: Two ways:

Client-side account credentials point to cloud system control service, and client uses ListAllServices operation to get a list

of available services (similar to CUPS-Get-Printers and future IPP Get-Printers operation for IPP System Control

Service spec)

⁃

LDAP/DNS-SD/etc. discovery protocols for public services (e.g. hotel managed printing services)

c. Q: Does the IDS group deal with client/proxy association/registration issues? No effect

A: Yes, that will be one of the items addressed by IDS

d. Q: Do we assume proxy can register with multiple cloud services?

A: Yes Fanout instance included see (comment 'i')

e. Q: Any kind of financial elements with proxy interface?

A: No, that is external to the model (i.e. there is a pre-existing business relationship that enables registration) no effect

f. Section 4.1.2.1: changes made

Line 727: "The proxy also periodically queries ..."

Line 728: "to check for waiting jobs" (drop "notification of")

Lines 729-734: Move "A failure to ..." after following paragraph, "failure to receive a query from the Proxy".

g. Section 4.1.2.2:

owner == Local Imaging System Owner changes made

h. Q: What about conformance requirements? no effect

A: Pretty loose for the model spec, binding specs will have the

usual SHOULD, MUST, etc.

Historically, MFD Model had conformance requirements for

operations and elements, but interoperability requirements tend to

just be in the binding specs (e.g. IPP)

i. Section 4.1.4: Changes made to text and diagram

⁃device -> equipment

May have one proxy talking to multiple Cloud Imaging Services of the same or different types.

j. Figure 4:Show one Proxy talking to multiple Cloud Imaging Services -changes made

k. Q: Can multiple proxies be chained? no effect

A: Conceptually yes. Fanout allows both direct (traditional Semantic Model/IPP interface) and indirect (the Proxy interface in the Cloud Imaging Model) usage, and this can be daisy-chained as needed. Question this. Proxy can interface with service which interfaces with downstream service, and conceivably proxy can interface with service that interfaces with proxy...but what would be the purpose?

l. Q: What about poll delays/responsiveness? Isn't directly talking to printer faster? no effect

A: Bindings will likely provide long-running "get" operations - you "poll" to wait for notifications, response comes as soon as event is available

Talking directly \*is\* faster, however the purpose of this model is to enable imaging when the client is unable to create a direct connection to the service due to firewall or other network restrictions

m. Section 4.2.1.2:

Yes, we need to be able to target a device to conform to SM/IPP fan-out

May change name/terminology here to follow SM ? No effect

n. Section 4.2.2, item 5:

Q: How to represent capabilities for multiple devices?

A: No way to report separate device capabilities as a single Local service

Solution: register multiple Local services, one per device

Solution: construct constraints that prevent combinations that are not supported (e.g. color, duplex, tabloid not supported by any one device)

Talk about implementation choice WRT intersection (only the common capabilities) vs. union (all capabilities) vs. separate local services - Don't understand the problem. Proxy will provide composite of capabilities and status as though there were a single service. Jobs will be directed to appropriate local service by proxy. User and Cloud service do not know nor care which specific local service is being used.

o. Section 4.2.2.3:

Q: What about race conditions, e.g. two proxies fetching the same job?

A: While we don't talk about it here, in IPPSIX we use a "first proxy to fetch wins" approach, with the other proxy getting a "not fetchable" error . Answer is reasonable, but why would Cloud System provide same job to multiple proxies? Presumably User has selected desired endpoint.

Add paragraph, "If the Job is no longer available to be fetched, an error is returned" (the model requires bindings to handle concurrency issues). Comment added. Indeed , general error comment added to assumptions

p. Section 4.2.2.9

Don't want best effort for registration, if everything isn't OK then the response is an error with a list of elements that are not supported - It was my intent that the Cloud Service did not have to accept all the features of the Local Service that the proxy reported as available. (Just as the Local service did not need to make all of its featuires accessable)

("I can't do scan") or missing ("I need your geo-location"). Obviously, errors will be provided in response to incomplete or erronious requests, and .text modified to reflect this. But I suggest that first example is unreasonable. Proxy will contact Cloud Imaging service specified by Cloud System Control Service for each local service the Proxy has listed for a Local System. Contacting the wrong Cloud Service would seem unlikely.

OK to ignore proxy info that the cloud doesn't care about (e.g. geo-location), do we need to report it? Yes, it is dsirable so that proxy does not need to update undesired info. I believe this is covered - is it necessary to make this process clearer?

IPP has successful-ok-ignored-or-substituted-attributes status code

What about reporting mobile printer geo-location in a moving car?

Might be useful to report attributes/elements that are not required (don't tell me about your geo-location) Yes , this was intent.

q. Section 4.2.2.11:

Should not allow proxy to deregister permanently, that is something you do through the cloud-specific interface, just like the initial setup prior to register. Assume request is to change de-register to suspend registration. This, at least, is necessary.

Q: Do we even need/want the operation?

A: Yes, because one proxy can register multiple systems

r. Section 4.2.2.13

Response is just success or error (document not found, etc.)

s. Section 4.2.2.13/4.2.2.14:

Q: Should we combine UpdateJobStatus and UpdateActiveJobs, to make a single UpdateJobs operation to update 1 to N jobs in one step?

A: No, see below

UpdateActiveJobs response, in all cases, returns list of job IDs and their updated states in the Cloud Imaging Service

New "invalid" job state for jobs that are not fetched or do not exist Question if this is optimum response. In other cases, decision was " Response is just success or error " Why not just error message and list of fetched but cloud terminated jobs? Analogous to UpdateServiceElements, where response is error message with rejected or unwanted elements.

t. Section 4.2.2.15:

Eliminate GetCloudTerminatedJobs. OK, Function is now implicit in UpdateActiveJobs response.

u. GetServiceNotifications:

JobTerminated: Add list of terminated jobs and their states. OK.

v. Figure 5:

Add a second set of GetServiceNotifications requests after the responses OK

w. Figure 5:

Q: Can job states persist across re-registration?

A: Assume they can (not necessarily, but they could) and that the Proxy still does an UpdateActiveJobs to discover the fate of the old jobs. Cloud would move existing jobs to fetchable when the proxy doesn't include them in UpdateActiveJobs, as it does with any UpdateActiveJobs operation. - Eliminate note.

x. Figure 6:

Drop GetCloudTerminatedJobs OK

y. Figure 9:

Drop GetCloudTerminatedJobs OK

Add GetJobElements operation OK - understood that this is to allow optional Proxy record-keeping on Job activity and resolution.

GetServiceNotification includes No longer just a JobTerminated flag - includes job Id.