1 Operations Used in Cloud Imaging Services

1.1 Client to Cloud Imaging Server Operations

1.1.1 Basic Client Service Operations

The common Basic operations are listed in Table 1; they are concerned with creating and controlling Jobs and Documents within Jobs in the Cloud Imaging Service. Although in most cases, operations affecting Jobs in the Cloud Imaging Service will be transferred to corresponding Jobs in the Imaging Devices by way of the communication between the Cloud Imaging Device Manager and the Cloud Imaging Service, these operations from the User/Client do not act on the Device Jobs directly.

The Operations include those by which a Client gets Service Elements to allow selection of Services and formulation of Job Tickets. Some of these operations do affect the state of a Job. However, none of these operations directly affect the state or configuration of the Service except to the extent that creating or canceling a Job may initiate a sequence that affects the Service.

Table 1 Basic MFD Interface Requests and Responses

Operation	Request Parameters (Notes 2)	Response Parameters (Note 3)	Note
Add<Service>HardcopyDocument	InputSource, JobId, Document Ticket(optional), ElementsNaturalLanguage(optional), LastDocument(optional), RequestingUserName	DocumentNumber, UnsupportedElements(optional)	
Cancel<Service>Document	DocumentNumber, ElementsNaturalLanguage(optional), JobId, Message (optional) RequestingUserName		
Cancel<Service>Job	ElementsNaturalLanguage(optional), JobId, Message (optional) RequestingUserName		
CancelCurrent<Service>Job	ElementsNaturalLanguage(optional), JobId(optional), Message (optional) RequestingUserName		
CancelMy<Service>Jobs	JobIds (optional), Message (optional), ElementsNaturalLanguage(optional), RequestingUserName	JobIds (optional)	1
Close<Service>Job	JobId, RequestingUserName		
Create<Service>Job	ElementsNaturalLanguage(optional), Job Ticket (optional) RequestingUserName	JobId, UnsupportedElements(optional)	
GetActive<Service>Jobs	ElementsNaturalLanguageRequested(optional), Limit(optional) RequestingUserName	ElementsNaturalLanguage(optional)JobSummaries (includes JobID, JobName, JobOriginatingUserName, JobState and perhaps JobStateReasons)(optional)	
Get<Service>DocumentElements	Document Number, ElementsNaturalLanguageRequested(optional), JobId, RequestingUserName	DocumentElements(optional), ElementsNaturalLanguage(optional)	

Get<Service>Documents	ElementsNaturalLanguageRequested(optional), JobId, RequestingUserName	Documents(list of DocumentSummaries)(optional), ElementsNaturalLanguage(optional)JobID, JobName	
Get<Service>JobElements	ElementsNaturalLanguageRequested(optional), JobId, RequestedElements (JobReceipt, JobStatus, or Job Ticket.)(optional) RequestingUserName	JobElements, ElementsNaturalLanguage(optional)	
Get<Service>Job History	ElementsNaturalLanguageRequested(optional), Limit(optional) RequestingUserName	ElementsNaturalLanguage(optional)JobSummaries (includes JobID, JobName, JobOriginatingUserName, JobState and perhaps JobStateReasons	
Get<Service>ServiceElements	ElementsNaturalLanguageRequested(optional), RequestedElements (Service Capabilities, ServiceConfiguration, ServiceDescription, ServiceStatus or DefaultJob Ticket.)(optional) RequestingUserName	ElementsNaturalLanguage(optional)ServiceElements(optional)	
Resubmit<Service>Job	ElementsNaturalLanguageRequested(optional), JobId, Job Ticket (optional) RequestingUserName	JobId, UnsupportedElements(optional)	
Resume<Service>Job	ElementsNaturalLanguageRequested(optional), JobId, Message(optional)RequestingUserName		
Send<Service>Document	ElementsNaturalLanguageRequested(optional), Document Ticket (optional) JobId, LastDocument(optional), RequestingUserName, DocumentData	DocumentNumber, UnsupportedElements(optional)	
Send<Service>Uri	DocumentUri, ElementsNaturalLanguageRequested(optional), Document Ticket (optional) JobId, LastDocument(optional), RequestingUserName	DocumentNumber, UnsupportedElements(optional)	
Set<Service>DocumentElements	DocumentNumber, ElementsNaturalLanguage(optional), SocumentTicket, JobId, Message(optional), RequestingUserName	UnsupportedElements(optional)	
Set<Service>JobElements	ElementsNaturalLanguage(optional), Job Ticket, JobId, Message(optional), RequestingUserName	UnsupportedElements(optional)	
SuspendCurrent<Service>Job	ElementsNaturalLanguage(optional), JobId(optional), Message(optional), RequestingUserName		
Validate<Service>Document Ticket	ElementsNaturalLanguageRequested(optional), Document Ticket RequestingUserName	UnsupportedElements(optional)	
Validate <Service>Job Ticket	ElementsNaturalLanguage(optional), Job Ticket, RequestingUserName	UnsupportedElements(optional)	

Notes:

Note 1: Response includes identified but un-cancellable Jobs

Note 2: The RequestingUserName, is used by the Service to determine whether the requestor is an Administrator, Operator or the Job Owner and is therefore authorized to make the request. Some implementations may require further authentication of the requestor's identity. If the requestor is not determined to have access, the Service MUST reject the request.

Note 3: All responses must include correlation to request and whether request was successful or failed.

1.1.1.1 Add<Service>HardcopyDocument[ww1]

The Add<Service>HardcopyDocument operation allows a Client to prepare a Service to request a Hardcopy Document via a scanner Subunit and to add it to an identified Job. It is analogous to the Send<Service>Document and Send<Service>Uri operations except that it is applicable to Services for which input Documents are obtained by a scan of a region of a media sheet side, such as FaxOut and EmailOut.

The Service MUST reject this request and send an appropriate message if:

1. The requestor is not the owner of the identified Job, or is not an Administrator or Operator;
2. The Service has already closed inputs to the identified Job, or
3. The Job is not found.

Otherwise, provided the request is properly constructed, complete and references valid objects, the Service MUST accept the request, MUST close the Job if the LastDocument Element is asserted, MUST be prepared to add Document Data from the identified input to the identified Job, and MUST respond to the request.

1.1.1.2 Cancel<Service>Document

The **Cancel<Service>Document** operation allows a Client to cancel a specified Document in a specified Job of the specified Service any time from when the time the Document is created up to, but not including, the time that the Document is Completed, Canceled or Aborted. Because a Document might already be in Processing by the time a Cancel<Service>Document request is received, some portion of the Document processing might be completed before the Document is actually canceled.

The Cancel<Service>Document operation does not remove the Document from the Job or the Service, but does set the specified Document's Document State Document Status Element to Canceled and the Document's Document State Reasons Element to an appropriate value. If the Job containing the Document is again submitted using Resubmit<Service>Job, the canceled Document is also submitted for processing. Thus Cancel<Service>Document has the same semantics as Cancel<Service>Job which cancels only the processing of the Job but does not delete the Job object itself.

The Cancel<Service>Document operation does not affect the states of any of the other Documents in the Job. If the Job is in the Processing state and there are more Documents to be processed, the Service does continue to process the un-canceled Documents. If there are no further Documents to process, the Job is advanced to the Completed state.

The Service MUST reject the operation and return an appropriate response message if the operation requestor is not either the Job owner or a Service or System operator or administrator. Otherwise the Service MUST accept or reject the Cancel**<Service>**Document request based on the Document's current state and, if the request is accepted, the Service MUST transition the Document to the indicated new state as follows:

Once a "success" response has been sent, the implementation guarantees that the Document will eventually end up in the Canceled state. Between the time that the Cancel<Service>Document request is accepted and when the Document enters the Canceled Document-state, the DocumentStateReasons Element MUST contain a value which indicates to any later query that, although the Document might still be Processing, it will eventually end up in the Canceled state.

1.1.1.3 Cancel<Service>Job

The Cancel<Service>Job operation changes the state of the identified Job to Canceled, provided that the Job is not already in or in a mode leading directly to a termination state. (i.e., Completed, Canceled, or Aborted.) Because a Job might already be active by the time a Cancel<Service>Job is received, a portion of the Job may be done before the Job is actually terminated.

The Service MUST accept or reject the request based on the Job's current state. If the request is accepted, the Job state is transitioned to Canceled and the Service will issue a success response. See the transition diagram under Job State . If the implementation requires some significant time to cancel a Job in the Processing or ProcessingStopped states, the Service MUST set the Job's JobStateReasons to a value indicating that the Job is transitioning to a Canceled state. If the Job already has a JobStateReasons indicating that it is transitioning to a Canceled state, then the Service MUST reject a Cancel<Service>Job operation

1.1.1.4 CancelCurrent<Service>Job

The CancelCurrent<Service>Job operation allows a Client to cause the Service to terminate processing on the currently processing Job and to move that Job to the Canceled state. As with any other Basic operation directly affecting a Job, this operation is accepted by the Service only if the originator is the Owner of the affected Job(s) or is an Administrator or Operator.

There is the potential that the current Job may have changed between the time a Client requests this operation and the time the Service implements it. Therefore, if the intent is to cancel a particular Job the Client MAY include an optional JobId parameter in the request.

1. If the JobId is included in the request and that Job is currently in the Processing or ProcessingStopped state and the operation requestor has access rights to that Job, the Service MUST accept the request and cancel the Job.
2. If no JobId is included in the request and the operation requestor has access rights to the Job currently in the Processing or ProcessingStopped state, the Service MUST accept the request and cancel that Job.
3. If more than one Job is in the Processing or ProcessingStopped state, all currently processing Jobs to which the request originator has access MUST be canceled unless the operation included the optional JobId, in which case only the identified Job is canceled.
4. If the JobId is included in the request and that Job is not currently in the Processing or ProcessingStopped state; or if the requestor does not have access rights to the identified Job, the Service MUST reject the request and return the appropriate error code.
5. If there is no Job currently in the Processing or ProcessingStopped state or if the requestor does not have access rights to any Job that is in the Processing or ProcessingStopped state, the Service MUST reject the request and return the appropriate error code.

1.1.1.5 CancelMy<Service>Jobs

The CancelMy<Service>Jobs operation permits a user to cancel all of their own identified non-Terminated Jobs or, if no specific Jobs are identified in the request, to cancel all of their own non-Terminated Jobs in the Service. This operation works like the Cancel-Job operation except that the operation can apply to multiple Jobs. The Client specifies the set of candidate Jobs to be canceled by supplying and/or omitting the JobIds. The Service MUST check the access rights of the requesting user against *all* of the candidate Jobs. If *any* of the candidate Jobs are not owned by the requesting user, the Service MUST NOT cancel any Jobs and MUST return the appropriate error status code along with the list of any JobIds that were specifically identified in the operation request but to which the User is not authorized access.

If this check succeeds, then (and only then) the Service MUST accept or reject the request based on the current state of each of the candidate Jobs and must transition each Job to the indicated new state as shown for the antecedent Cancel-My-Jobs operation in the Standard for Internet Printing Protocol (IPP): Job and Printer Extensions Set 2 [PWG5100.11]. If any of the candidate Jobs that were not already in a Terminating state cannot be canceled, the Service MUST NOT cancel any Jobs and MUST return the appropriate error status code along with the list of JobIds for those Jobs which were specifically identified in the operation request but could not be canceled. If the requested Jobs include some Jobs that are already in a terminating state, this circumstance in itself MUST NOT interfere with the canceling of non-terminated candidate Jobs, but SHOULD result in the return of a warning message identifying the specifically identified Jobs that already were in a Terminating state.

1.1.1.6 Close<Service>Job

The Close<Service>Job operation allows a Client to close Job inputs to those Services accepting Documents, even when the last Document input operation for the Job (Send<Service>Document, Send<Service>URI or Add<Service>Document) did not include the LastDocument Element with a 'true' value. This Close<Service>Job operation supersedes and, if supported by the Service, is preferable to the practice of using a Send<Service>Document with no Document Data but with a LastDocument Element containing a 'true' value to close inputs.

The Service MUST reject this operation request if the target Job is not found or if the requestor is not the Job Owner or an Administrator. Otherwise, the Service MUST accept this operation request even if the target Job is already closed and regardless of JobState. Closing the Job MUST cause the Service to reject any subsequent Document input operation for the target Job, but MUST NOT affect the execution of any previously accepted Document input operation.

1.1.1.7 Create<Service>Job

The Create<Service>Job operation allows a Client to request creation of a Job in the Service. Upon creation, the Job is in Pending state and available for scheduling unless a Job Processing instruction prevents this. (e.g., JobHoldUntil puts it in PendingHeld state) The Create<Service>Job operation MUST fail if the Service’s IsAcceptingJobs Element value is ‘false’.

Job Processing is done on one or more Documents. Unlike the antecedent IPP Print-Job operation, the MFD Create<Service>Job may involve more than one Document. Depending upon the type of Service, the input may be a Hardcopy Document or a Digital Document. In either case, the source(s) of the input Document(s) as well as the destination(s) of the output Document(s) are identified in the Job Ticket submitted in the Create<Service>Job Request,

Once a Job is created, Documents may be input as part of that Job by Send<Service>Document, Send<Service>URI or, for Services that accept hardcopy input, Add<Service>Document operations. In Service implementations that do not accept multiple Documents (i.e., MultipleDocumentJobsSupported = False), Document input is closed after one Document is accepted. In Service implementations that do accept multiple Documents (i.e., Multiple Document Jobs Supported = True), there may be multiple Send<Service>Document, Send<Service>URI or Add<Service>Document operations. There are two methods of indicating when all Documents have been sent:

1. issuing a Close<Service>Document request

2. issuing a Send<Service>Document, Send<Service>URI or,
3. Add<Service>Document request with the LastDocument Element = True

To avoid a possible hang condition, Service implementations supporting multiple Document Jobs must also support the Multiple Operation Time Out Element that indicates the minimum number of seconds the Service will wait for the next Send or Add operation before taking some recovery action. If, for some reason, there is a longer period between Create<Service>Job and valid Send or Add operations, or between sequential Send or Add operations, the Client MUST send Send or Add requests, even if they are empty, to reset the timeout. If there is a multiple operation timeout, the Service will take remedial action according to the value that Service has indicated in its Multiple Operation Timeout Action Element.

1.1.1.8 Get<Service>DocumentElements

The Get<Service>DocumentElements operation allows a Client to obtain detailed information about the specified Document within the specified Job. This operation is parallel to the Get<Service>Job-Elements operation, but with the target and response Elements relating to a Document rather than a Job.

The Client requests specific groups of Elements (complex Elements) contained within the Document. The Document Data is not part of the Document and cannot be retrieved using this operation. However the location of the Document Data is available. The allowed values for Requested Elements are Document Receipt, Document Status and Document Ticket. Vendors may extend the allowed values.

The Service MUST return the Document Description Element values that a Client supplied in the Document Creation operation (Create<Service>Job, Send<Service>Document or Send<Service>URI) or provided in Set<Service>DocumentElements operation a plus any additional Document Description Elements that the Service has generated, such as Document State. The Service MUST NOT return any Job level Elements that the Document inherits from the Job level but MUST return Document Elements specified at the Document level. It is NOT REQUIRED that a specific Document include all Elements belonging to a group (since some Elements are optional). However, it is REQUIRED that the Service support all these group names for the Document object.

1.1.1.9 Get<Service>Documents

The Get<Service>Documents operation allows a Client to retrieve the list of Documents belonging to the identified Job. A Document summary containing a group of Document Element names with their values will be returned for each Document in the Job.

This operation is similar to the Get<Service> and Get<Service> operations except that it returns Elements from Documents rather than identified Jobs. As with the Get<Service>DocumentElements operation, the Service MUST return only those Elements that are in the Document Ticket.

1.1.1.10 Get<Service>JobElements

The Get<Service>JobElements operation allows a Client to obtain detailed information on the specified Job. Unlike the antecedent IPP Get-Job-Attributes operation, the Get<Service>JobElements request may not specify individual Elements. Rather, the Client requests specific groups of Elements contained within the Job. The allowed values for

RequestedElements are Job Receipt, Job Status, or Job Ticket. Vendors may extend the allowed values.

The Service MUST reject this request if the requestor is not authorized access to the identified Job,

1.1.1.11 Get<Service>Jobs

The Get<Service>Jobs operation provides summary information on all Jobs that have reached a terminating state (i.e., Completed, Canceled Aborted). As such, it is similar to the antecedent Get-Jobs operation with the which-Jobs Element set to ‘completed’. Unlike Get-Jobs, Get<Service>Jobs may not include a Requested Elements argument; rather, it always returns a Job Summary for each terminated Job including JobId, JobName, JobOriginatingUserName, JobState and perhaps JobStateReasons and other Service specific information.

When the operation is exercised by a User that is not an Administrator, the Job summary may not include all of the summary information, depending upon site security policy.

1.1.1.12 Get<Service>ServiceElements

The Get<Service>ServiceElements operation allows a Client to obtain detailed information on the Elements and their values supported by the Service. Unlike the antecedent IPP Get-Printer-Attributes operation, the Get<Service>ServiceElements request may not specify individual Elements. Rather, the Client requests information on one or more specific group of Elements. The allowed values for Requested Elements are Service Capabilities, Service Configuration, Service Description, Service Status or DefaultJob Ticket. Vendors may extend the allowed values.

Some Services may accept an additional argument in a Get<Service>ServiceElements request to further filter the response, much as the antecedent IPP Get-Printer-Attributes operation accepted the Document-Format Element. The individual Service specifications identify such arguments if any, their effect and whether support is mandatory.

In addition to the status message, the Service response includes the set of requested Element names and their values for all supported Elements. The response need not contain the requested Element names for any Elements not supported by the Service.

1.1.1.13 GetActive<Service>Jobs

The GetActive<Service>Jobs operation provides summary information on all Jobs in the Pending or Processing state. As such, it is equivalent to the antecedent Get-Jobs operation with the which-Jobs Element set to ‘not-completed’. Unlike the antecedent Get-Jobs operation, GetActive<Service>Jobs may not include a RequestedElements argument; rather, it always returns a JobSummary for each Active Job with the summary including JobId, JobName, JobOriginatingUserName, JobState and perhaps JobStateReasons and other Service specific information.

When the operation is exercised by a User that is not an Administrator or Operator, the Job summary may not include all of the summary information, depending upon site security policy.

1.1.1.14 Resubmit<Service>Job

The Resubmit<Service>Job operation allows a Client acting for the Job Owner or an Administrator or Operator to resubmit a previously completed Job, but with the option of

providing new Job Ticket information (other than input Document Data or input Document Data descriptive information.)

The Resubmit<Service>Job operation is applicable only to a RetainedJob. A Retained Job is one which remains in the Service after it has been completed or canceled. This may be incidentally or because it is a saved Job, which is a Completed or Canceled Job with a JobSaveDispostion Element value that indicates that the Job, including Document Data if any, should not be deleted or aged-out after the Job is completed.

If a Resubmit<Service>Job operation is accepted, the state of the retained Job is not changed; rather, a new Job is created from the identified retained Job and submitted with an implicit **CreateJob** request.

1. If the Resubmit<Service>Job request contains a processing Element that was in the retained Job but with a different value, the value supplied in the Resubmit<Service>Job operation MUST override the original value (if supported by the Service).
2. If the Resubmit<Service>Job request contains a processing Element that was not in the retained Job, the Element with the value supplied with the Resubmit<Service>Job operation MUST be applied (if supported by the Service)
3. For any processing Element in the original retained Job the value of which is not changed in the Resubmit<Service>Job request, that Element and its value MUST be applied to newly created Job except that a JobSaveDispostion Element value indicating that the Job should be saved, and certain other Service-specific Element values, MUST NOT be copied but are applied to the new Job only if they are in the Resubmit<Service>Job request.

The newly created Job is moved to the Pending or PendingHeld Job state with the same Element values as the original saved Job (except for the save Element). If any of the Documents in the saved Job were passed by reference (Send<Service>URI or Send>Service>URI), the Service MUST re-fetch the data, since the semantics of Restart<Service>Job are to repeat all Job processing. The Service MUST assign new JobUri and JobId values to the newly created Job; the JobDescription Elements that accumulate Job progress, such as JobImpressionsCompleted, JobMediaSheetsCompleted, and JobKOctetsProcessed, MUST be an accurate record for the newly created Job.

The Service MUST accept or reject the Resubmit<Service>Job Request based on the authority of the requester and the referenced Job's current state. The Requester must either be the Job owner or an operator or administrator of the Service. The target Job must be retained with a Completed or Canceled state.

1.1.1.15 Resume<Service>Job

The Resume<Service>Job operation allows a Client acting for the Job Owner or an Administrator or Operator to resume the identified Job at the point where it was suspended. Provided that no other condition exists that forces the Job to the PendingStopped state, the Service moves the Job from the ProcessingStopped state to the Pending state and removes the JobSuspended value from the Job's StateReasons Element. If the identified Job is not in the ProcessingStopped state with the JobSuspended value in the Job's StateReasons Element, the Service MUST reject the request and return an appropriate status code, since the Job was not suspended.

If a Service supports Suspend<Service>Job or SuspendCurrent<Service>Job operations, it MUST support the Resume<Service>Job operation, and vice-versa.

1.1.1.16 Send<Service>Document

The Send<Service>Document operation allows a Client acting for the Job Owner or an Administrator or Operator to input a Digital Document to a Service as part of an already created Job. In response to the Create<Service>Job, the Service will have returned the JobURI and the JobId. For each Document that the Client desires to add to this Job, the Client issues a Send<Service>Document request which includes the JobId and contains the entire stream of Document Data for one Document.

If the Service supports this operation but does not support multiple Documents per Job, Document input is closed after the first Document is accepted and the Service MUST reject subsequent Send<Service>Document requests associated with the same Job. Similarly, if the Service does support multiple Documents per Job, the Service MUST reject Send<Service>Document requests associated with a given Job after inputs to that Job have been closed either a Close<Service>Job operation or a previous Send<Service>Document with a 'true' value for the LastDocument Element. Note that the Client may send and the Service must accept a Send<Service>Document request with a 'true' value for the LastDocument Element to close input to that Job, even if that request includes no Document data.

See the Create<system>Job description for discussion of issues relating to excessive delay between multiple Send<Service>Document requests.

The Service MUST reject a Send<Service>Document request and send an appropriate message if:

1. The requestor is not the owner of the identified Job, or is not an Administrator or operator
2. The Service has already closed inputs to the identified Job,
3. The Document size, format and/or compression are not supported by the Service, or
4. The Job is not found.

Otherwise, the Service MUST accept the request, MUST close the Job if the LastDocument Element is asserted, MUST add the supplied Document Data (if any) to the identified Job, and MUST respond to the request.

1.1.1.17 Send<Service>Uri

The Send<Service>Uri operation allows a Client acting for the Job Owner or an Administrator or Operator to input a Digital Document to a Service as part of an already created Job. As such, the Send<Service>Uri operation is identical to the Send<Service>Document except that a Client supplies a URI reference (DocumentUri Element) rather than the Document Data itself. If a Service supports both operations, Client s can use both Send<Service>Uri and Send<Service>Document operations to add new Documents to an existing multi-Document Job.

As with Send<Service>Document, if the Service supports Send<Service>Uri but does not support multiple Documents per Job, the Service MUST reject subsequent Send<Service>Uri requests associated with the same Job. Similarly, if the Service does support multiple Documents per Job, the Service MUST reject Send<Service>Uri requests associated with a given Job after inputs to that Job have been closed. Job inputs can be closed either by a Close<Service>Job operation or a Send<Service>Document (NOT a Send<Service>Uri) request with a 'true' value for the LastDocument Element. Note that the Client may send and the Service must accept a Send<Service>Document request with a 'true' value for the LastDocument Element to close input to that Job even if that request includes no Document data.

The Service MUST reject this request and send an appropriate message if:

1. The requestor is not the owner of the identified Job, or is not an Administrator or operator
2. The Service has already closed inputs to the identified Job,
3. The Job is not found
4. The Document size, format and/or compression are not supported by the Service, or
5. The Service does not support the URI Scheme specified.

Otherwise, the Service MUST accept the request, MUST close the Job if the LastDocument Element is asserted, MUST add the Document Data (if any) to the identified Job, and MUST respond to the request. See the Create<system>Job description for discussion of issues relating to excessive delay between multiple Send<Service>Uri requests.

1.1.1.18 Set<Service>DocumentElements

The Set<Service>DocumentElements operation allows a Client , operating for the Job Owner or an Administrator, to set the values of identified Elements of the specified Document within the specified Job. This operation is parallel to the Set<Service>JobElements and Set<Service>ServiceElements operations and it follows the same rules for validation, but with the target and response Elements relating to a Document rather than a Job or the Service.

The Client must fully identify the Elements to be set as well as the set values. The only settable Elements are those within the Document Ticket. The Document Data is not part of the Document and cannot be changed using this operation. If a Document was originally submitted without a given settable Element that the Set<Service>DocumentElements request attempts to set, the Service adds the specified Element to the Document.

If the Client identifies a Document Element but does not specify a value for that Element, then the Service MUST remove the Element and all of its values from the Document. The semantic effect of the Client supplying the Element with no value in a Set<Service>DocumentElements operation MUST be the same as if the Element had not originally been supplied with the Document. This corresponds to the action of the out-of-band value “DeleteElement” in the antecedent IPP Set-Document-Attributes operation. Any subsequent Get<Service>DocumentElements or Get<Service>Documents request MUST NOT return any Element that has been deleted. However, a Client can re-establish such a deleted Document Element with any supported value(s) using a subsequent Set<Service>DocumentElements operation.

If the Client supplies an Element in a Set<Service>DocumentElements request with no value and that Element is not present in the Document object, the Service ignores that supplied Element in the request, does not return the Element in the Unsupported Elements group, and returns the ‘success’ status code, provided that there are no other problems with the request.

The validation of the Set<Service>DocumentElements request is performed by the Service as if the Document had been submitted originally with the new Element values (and the deleted Elements removed); i.e., all modified Document Elements and values must be supported in combination with the Document Elements not modified. If such a Document Creation operation would have been accepted, then the Set<Service>DocumentElements MUST be accepted. If such a Document Creation operation would have been rejected, then the Set<Service>DocumentElements MUST be rejected and the Document MUST be unchanged. In addition, if any of the supplied Elements are not supported, are not settable, or the values are not supported, the Service MUST reject the entire operation; the Service MUST NOT set just some

of the supplied Elements. That is, Set<Service>DocumentElements MUST be implemented as an atomic operation; after the operation, all the supplied Elements MUST be set or all of them MUST NOT be set.

The value of JobMandatoryElements supplied in the original Create<Service>Job request, if any, MUST have no effect on the behavior of the Set<Service>DocumentElements operation. Rather, the Service must consider that any Element or Element value in a Set<Service>DocumentElements operation is mandatory. The Service MUST reject any request to set a Document Element to an unsupported value or to a value that would conflict with another Document Element value.

The Service MUST respond to the Set<Service>DocumentElements operation as defined for the antecedent Set-Document-Attributes operation in the Standard for IPP Document Objects [PWG5100.5]. Although the Document's current state affects whether the Service accepts or rejects the Set<Service>DocumentElements request, the operation MUST NOT change the state of the Document object (since the Document is a passive object and the Document state is a subset of the JobState). For example, if the operation creates a request for unavailable resources, the Job (but not the Document) transitions to a new state.

1.1.1.19 Set<Service>JobElements

The Set<Service>JobElements operation allows a Client operating for the Job Owner or an Administrator, to set the values of identified Elements of the specified Job. The Client must fully identify the Elements to be set as well as the set values. In the response, the Service returns success or rejects the entire request with indications of which Element or Elements could not be set to the specified values.

This operation is parallel to the Set<Service>DocumentElements and Set<Service>ServiceElements operations and it follows the same rules for validation, but with the target and response Elements relating to a Job rather than a Document or the Service

If the Client identifies a Job Element but does not specify a value for that Element,, then the Service MUST remove the Element and all of its values from the Job. The semantic effect of the Client supplying the Element with no value in a Set<Service>JobElements operation MUST be the same as if the Element had not originally been supplied with the Job. This corresponds to the action of the out-of-band value “DeleteElement” in the antecedent IPP Set-Job-Attributes operation. Any subsequent Get<Service>JobElements or Get<Service>Jobs request MUST NOT return any Element that has been deleted. However, a Client can re-establish such a deleted Job Element with any supported value(s) using a subsequent Set<Service>JobElements operation.

If the Client supplies an Element in a Set<Service>JobElements request with the DeleteElement value and that Element is not present on the Job object, the Service ignores that supplied Element in the request, does not return the Element in the Unsupported Elements group, and returns the ‘success’ status code, provided that there are no other problems with the request.

The validation of the Set<Service>JobElements request is performed by the Service as if the Job had been submitted originally with the new Element values (and the deleted Elements removed); i.e., all modified Job Elements and values must be supported in combination with the Job Elements not modified. If such a Job Creation operation would have been accepted, then the Set<Service>JobElements request MUST be accepted. If such a Creation operation would have been rejected, then the Set<Service>JobElements MUST be rejected and the Job MUST be

unchanged. In addition, if any of the supplied Elements are not supported, are not settable, or the values are not supported, the Service MUST reject the entire operation; the Service MUST NOT partially set some of the supplied Elements. In other words, after the operation, all the supplied Elements MUST be set or none of them MUST be set, thus making the Set<Service>JobElements an atomic operation.

The value of JobMandatoryElements supplied in the original Create<Service>Job request, if any, MUST have no effect on the behavior of the Set<Service>JobElements operation. Rather, the Service must consider that any Element or Element value in a Set<Service>JobElements operation is mandatory. The Service MUST reject any request to set a Job Element to an unsupported value or to a value that would conflict with another Job Element value.

The Service MUST accept or reject the Set<Service>JobElements operation according to the rules defined for the antecedent Set-Job-Attributes operation in Internet Printing Protocol (IPP):Job and Printer Set Operations [RFC3380].

1.1.1.20 SuspendCurrent<Service>Job

The SuspendCurrent<Service>Job operation allows a Client operating for the Job Owner or an Administrator, to suspend a Job by setting a condition in a Job that is currently in the Processing or ProcessingStopped state. This condition, reflected by the JobSuspended value in that Job's JobStateReasons Element, causes that Job to be in the ProcessingStopped state. The Service is able to processes other Jobs normally, provided that no other inhibiting conditions exist. Note that a Job may be ProcessingStopped state for other reasons and that, once it has been suspended, the Job will remain in the ProcessingStopped state even after the other conditions have been removed.

There is the potential that the current Job may have changed between the time a Client requests this operation and the time the Service implements it. Therefore, if the intent is to suspend a particular Job, the Client can include an optional JobId parameter in the request.

The target Job is the Job identified by the JobId, if included in the request

If the JobId is not included in the request, any Jobs in the Processing or ProcessingStopped state to which the requestor has access rights.

The Service MUST reject the request and send an appropriate message if:

a. There is no target Job in the Processing or ProcessingStopped state to which the requestor has access rights.
b. The target Job or all potential target Jobs have already been suspended.

The Service MUST accept the request, cancel any target Job(s) that have not been previously suspended, and return an appropriate message if:

1. The target JobId is included in the request and that Job is currently in the Processing or ProcessingStopped state (but is not suspended), and the requestor has access rights,
2. If no JobId is included and the requestor has access rights to the Job that is currently in the Processing or ProcessingStopped state (but is not suspended), the Service MUST accept the request and suspend that Job.
3. If more than one Job is in the Processing or ProcessingStopped state (but are not suspended), all such Jobs MUST be suspended unless the operation request included the optional JobId, in which case only the identified target Job MUST be suspended.

4. If the JobId is included in the request and that Job is not currently in the Processing or ProcessingStopped state; or if the JobId is not included and there is no Job currently in the Processing or ProcessingStopped state, the Service MUST reject the request and return the appropriate error code.
5. If the JobId is included in the request and that Job has been suspended; or if no JobId is included and is currently in the Processing or ProcessingStopped state, the Service MUST reject the request and return the appropriate error code.

The Resume<Service>Job operation causes a suspended Job to be released. If a Service supports SuspendCurrent<Service>Job operation, it MUST support the Resume<Service>Job operation, and vice-versa.

1.1.2 Administrative Service Specific Operations

Administrative Service operations directly affect the specific Services within Cloud Imaging Service and/or affect the Jobs of multiple Job Owners. Access is reserved for Administrators or Operators. The Administrative Service Operations are listed in Table 2 and are described below. Note that these operations are accessible to only Users with proper administrative access rights to the Cloud Imaging Service. These operations do not directly affect the Cloud Imaging Device Manager(s) which connect to the Cloud Imaging Service, or the Devices with which these Cloud Imaging Device Managers interface.

Table 2 Imaging Service Specific Administrative Operations

Operation	Request Parameters (Note 2)	Response Parameters (Note 3)	Note
Cancel<Service>Jobs	ElementsNaturalLanguage(optional), JobIds(optional), Message (optional) RequestingUserName	JobIds (optional)	1
Disable<Service>Service[ww2]	ElementsNaturalLanguage(optional) Message (optional), RequestingUserName		
Enable<Service>Service	ElementsNaturalLanguage(optional), Message (optional) RequestingUserName	-	
HoldNew<Service>Jobs	ElementsNaturalLanguageRequested (optional), JobHoldUntil | JobHoldUntilTime, Message(optional), RequestingUserName		
Pause<Service>Service	ElementsNaturalLanguageRequested (optional), Message(optional), RequestingUserName		
Pause<Service>ServiceAfterCurrentJob	ElementsNaturalLanguageRequested (optional), Message(optional), RequestingUserName		
Promote<Service>Job	ElementsNaturalLanguageRequested (optional), JobId, Message(optional), PredecessorJobID(optional), RequestingUserName		
ReleaseNew<Service>Jobs	ElementsNaturalLanguageRequested (optional), Message(optional), RequestingUserName		
Restart<Service>Service	ElementsNaturalLanguageRequested (optional), IsAcceptingJobs| IsAcceptingResources (optional), Message(optional), RequestingUserName		
Resume<Service>Job	ElementsNaturalLanguageRequested(optional), JobId, Message(optional), RequestingUserName		
Resume<Service>Service	ElementsNaturalLanguageRequested(optional), Message(optional), RequestingUserName		
Set<Service>ServiceElements	DefaultJob Ticket(optional), RequestingUserName ElementsNaturalLanguageRequested (optional), Capabilities(optional), CapabilitiesReady(optional), Description(optional), Message(optional),	Unsupported Elements(optional)	
Shutdown<Service>Service	ElementsNaturalLanguageRequested(optional), Message(optional), RequestingUserName		4

Note 1: Cancel<Service>Jobs response includes identified but un-cancellable Jobs

Note 2: The RequestingUserName, is used by the Service to determine whether the requestor is an Administrator, Operator or the Job Owner and is therefore authorized to make the request. Some implementations may require further authentication of the requestor's identity. If the requestor is not determined to have access, the Service MUST reject the request.

Note 3: All responses must correlate to request and indicate whether request was successful or failed.

Note 4: Forcing Service Shutdown may also force the state of any active Jobs to Aborted.

1.1.2.1 Cancel<Service>Jobs

The Cancel<Service>Jobs operation allows the Operator or Administrator of the Service to cancel all identified non-Terminated Jobs or, if no specific Jobs are identified in the request, to cancel all non-Terminated Jobs in the Service. It differs from the Cancel<Service>Job operation in that it works on a number of Jobs at once. If, following the legal Job state Transitions in Table, the Service cannot successfully cancel all explicitly or implicitly requested Jobs that are not already in the terminated state it MUST NOT cancel any Jobs but MUST return an error code. In this case, the Service MUST also return the list of JobIds for those Jobs that were explicitly identified in the request but could not be canceled.

The set of candidate Jobs to be canceled is specified by the supplied JobIds. If no JobIds are supplied, it is implicit that all Jobs that are not in a Terminating state are to be canceled. As with all Administrative operations, the Service MUST check the access rights of the requesting user. Provided that the requester has access rights, the Service MUST check the current state of each of the candidate Jobs. If any of the candidate Jobs cannot be canceled, the Service MUST NOT cancel any Jobs and MUST return the indicated error status code along with the list of offending JobId values. If there are no Jobs that cannot be canceled, the Service MUST transition each identified Job to the indicated new state as defined for the antecedent Cancel-Jobs operation in paragraph 6.1 of Standard for Internet Printing Protocol (IPP):9 Job and Printer Extensions Set 2 [PWG5100.11].

1.1.2.2 Disable<Service>Service

The Disable<Service>Service operation prevents the Service from creating any new Jobs by negating the IsAcceptingJobs Element. This operation has no effect upon the Service State and the Service is still able to process operations other than Create<Service>Job. All previously created or submitted Jobs and all Jobs currently processing continue unaffected.

If the requestor is determined to have proper access, the Service MUST accept this request and MUST negate the IsAcceptingJobs Element.

The IsAcceptingJobs Element value is reaffirmed by the Enable<Service>Service operation. If an implementation supports Disable<Service>Service it must also support Enable<Service>Service and vice-versa.

1.1.2.3 Enable<Service>Service

The Enable<Service>Service operation asserts the IsAcceptingJobs Element to allow the Service to accept new Create<Service>Job requests. The operation has no effect upon the Service State or any other operation requests the Service may receive.

If the requestor is determined to have proper access, the Service MUST accept this request and MUST assert the IsAcceptingJobs Element. The Service MUST then be able to accept and implement Create<Service>Job requests, provided that no other inhibiting condition exists.

If a Service implementation supports the Disable<Service>Service operation, then it must also support Enable<Service>Service operation and vice-versa.

1.1.2.4 HoldNew<Service>Jobs

The HoldNew<Service>Jobs operation allows a client to prevent any new Jobs from being eligible for scheduling by forcing all newly-created Jobs to the PendingHeld state with a JobHoldUntil or JobHoldUntilTime Job Processing Element added, depending upon the Element supplied with the HoldNew<Service>Jobs operation request. The operation has the same effect as a Hold<Service>Jobs operation except that any Jobs in the Pending or Processing state when the HoldNew<Service>Jobs request is accepted are allowed to go to completion, provided that no other conditions or operations prevent this.

The JobHoldUntil parameter allows a client to specify holding new Jobs indefinitely or until a specified named time period. The JobHoldUntilTime parameter allows a client to hold new Jobs until a specified time. Provided that the requestor is authorized and the operation and requested parameters are supported, a Service MUST accept a HoldNew<Service>Jobs request and MUST add the supplied 'JobHoldUntil' or JobHoldUntilTime Element to the Jobs. This HoldNew<Service>Job condition may be cleared by a ReleaseNew<Service>Jobs operation.

If the HoldNewJobs operation is supported, then the ReleaseNew<Service>Jobs operation MUST be supported, and vice-versa

1.1.2.5 Pause<Service>Service

The Pause<Service>Service operation allows a client to send the Service to the Stopped state. In this Service state, the Service MUST NOT advance any Job to Job Processing state. Depending on implementation, the Pause<Service>Service operation MAY also stop the Service from continuing to process any current Job, sending the Job to the ProcessingStopped state. That is, depending upon implementation, any Job that is currently in the Processing state may be sent to the ProcessingStopped state as soon as the implementation permits; or the Job may continue to a termination state as determined by other conditions. The Service MUST still accept CreateJob operations to create new Jobs, provided that there are no other conditions preventing it.

If the Pause<Service>Service operation is supported, then the Resume operation MUST also be supported, and vice-versa.

Service State transitions resulting from a Pause<Service>Service operation are the same as defined for the antecedent Pause-Printer operation in paragraph 3.2.7 of IPP/1.1: Model and Semantics [RFC29110. The Pause<Service>Service action should be done as soon as the possible after the request is accepted. If the implementation will take more than negligible time to stop processing (perhaps to finish processing the current Job), the Service may remain in the ‘Processing’ state but MUST add the 'MovingToPaused' value to the Service’s StateReasons Element. When the Service transitions to the 'Stopped' state, it removes the 'MovingToPaused' value and adds the 'Paused' value to the Service’s StateReasons Element. If the implementation permits the current Job to stop in mid processing, the Service transitions directly to the ‘Stopped’ state with the Service’s StateReasons Element set to the 'Paused' value and the current Job transitions to the 'ProcessingStopped' state with the JobStateReasons Element set to the 'Stopped' value.

For any Jobs in the 'Pending' or 'PendingHeld' state, the ‘Stopped' value of the Jobs' JobStateReasons Element also applies. However, the Service need not update those Jobs'

JobStateReasons Element and need only return the 'Stopped' value when those Jobs are queried (so-called lazy evaluation).

Provided that the requestor is authorized, the Service MUST accept the Pause<Service>Service request in any Service state and act as defined for the antecedent Pause-Printer operation in paragraph 3.2.7 of IPP/1.1: Model and Semantics [RFC29110].

1.1.2.6 Pause<Service>ServiceAfterCurrentJob

The Pause<Service>ServiceAfterCurrentJob operation allows a client to stop the Service from processing any Jobs once any Jobs currently in Processing are completed. This operation has no effect on the current Jobs and the Service MUST complete the processing of the current Jobs, provided that no other condition or operations preclude it. The Service MUST still accept **CreateJob** operations to create new Jobs, but MUST not cause any Jobs to enter 'Processing'. If the Pause<Service>ServiceAfterCurrentJob operation is supported, then the Resume<Service>Service operation MUST also be supported.

Service State transitions resulting from a Pause<Service>ServiceAfterCurrentJob operation are as identified for the antecedent Pause-Printer-After-Current-Job operation in IPP: Job and Printer Operations [RFC3998]. Note that, in implementations where the Service implementation is not able to pause Jobs currently in the Processing state, the response to the Pause<Service>ServiceAfterCurrentJob request and the Pause<Service>Service request are exactly the same.

If the implementation will take more than negligible time to finish processing the current Jobs, the Service will remain in the Processing state and must add the 'MovingToPaused' value to the Service's StateReasons Element. When the Service transitions to the 'Stopped' state, it removes the 'MovingToPaused' value and adds the 'Paused' value to the Service's StateReasons Element.

For any Jobs in the 'Pending' or 'PendingHeld' state, their state is unchanged but the JobStateReasons Element must be set to the 'Stopped' value. However, the Service need not update those Jobs' JobStateReasons Element and only need return the 'Stopped' value when those Jobs are queried (so-called lazy evaluation).

Provided that the requestor is authorized, the Service MUST accept the request in any Service state and MUST transition the Service to the indicated new State as follows before returning the operation response as defined for the antecedent Pause-Printer-After-Current-Job operation in IPP: Job and Printer Operations [RFC3998].

1.1.2.7 Promote<Service>Job

The Promote<Service>Job operation schedules the identified Job to be processed next, after the currently processing Jobs or, if the request includes the predecessor JobId, immediately after the identified predecessor Job. The Promote<Service>Job operation is a combination of the IPP Promote-Job and Schedule-Job-After operations. If the predecessor Job is not specified, it acts in the same way as the antecedent IPP Promote-Job operation. If the predecessor Job is specified, it acts the same way as the antecedent IPP Schedule-Job-After operation.

The identified target Job must be in the 'Pending' state. If the identified target Job is not in the 'Pending' state or if the predecessor Job is identified and it is not in the 'Pending', 'Processing' or 'ProcessingStopped' state, the Service MUST reject the request and return an appropriate status code. If the Promote<Service>Job request is accepted, the target Job MUST be processed

immediately after the current Jobs or identified predecessor Job reaches a Termination state (Canceled, Completed or Aborted)

Note that the action of this operation is consistent even if a previous Promote<Service>Job Request has caused some other Job to be scheduled after the current or predecessor Job; that is, within the rescheduling time limitations of the Service, the Job identified in the last Promote<Service>Job Request accepted will be processed next.

1.1.2.8 ReleaseNew<Service>Jobs

The ReleaseNew<Service>Jobs operation allows a client to remove the condition initiated by HoldNew<Service>Jobs and to release all Jobs previously forced to a PendingHeld state by the HoldNew<Service>Jobs initiated condition so that these Jobs are eligible for scheduling. This is done by removing the 'JobHoldUntilSpecified' and ‘JobHeldByService’ values from the Job's JobStateReasons Element and changing the Jobs’ states to ‘Pending’.

Provided that the requestor is authorized, the Service MUST accept this request in any Service state and the Service MUST remove the 'JobHoldUntilSpecified' value from the Job's JobStateReasons Element for any Job previously forced to a PendingHeld state by the HoldNew<Service>Jobs initiated condition.

If the ReleaseNew<Service>Jobs operation is supported, then the HoldNew<Service>Jobs operation MUST be supported, and vice-versa.

1.1.2.9 Restart<Service>Service

The Restart<Service>Service operation causes a Service in any state, even a previously shut down instance of a Service, to be initialized and set to the Idle state, provided that no errors occur or conditions exist that would prevent normal operation. The handling of Jobs that were in the Processing, Pending, PendingHeld, and ProcessingHeld states state prior to Restart is implementation dependent, but a Service Restart MUST be performed as gracefully as possible and in a way preserving the content and integrity of any non-terminated Jobs. Job history data, if supported, SHOULD also be preserved; a particular Service may make this mandatory.

Provided that the requestor is authorized, the Service MUST accept the request Restart<Service>Service regardless of its current state. Providing that no conditions exist that would normally prevent these actions, the Service MUST reinitialize its State to Idle, clear the StateReasons Element and set the IsAcceptingJobs Element to true.

1.1.2.10 Resume<Service>Service

The Resume<Service>Service operation allows a client to cause the Service to resume scheduling Jobs after scheduling has been paused. Provided that the requestor is authorized and the Service supports this operation, a Service MUST accept a Resume<Service>Service request regardless of the current Service state, corresponding to the actions defined for the antecedent Resume-Printer operation in Internet Printing Protocol/1.1: Model and Semantics [RFC2911]. If there are no other reasons why the Service is in the Stopped state, this operation returns the Service from the Stopped state to the Idle or Processing state from which it was paused, and removes the 'Paused' value to the Service’s StateReasons Element.

If the Resume<Service>Service operation is supported, then the Pause<Service>Service operation MUST be supported, and vice-versa.

1.1.2.11 Set<Service>ServiceElements

The Set<Service>ServiceElements operation allows a Client to set the values of identified Elements in the Service, provided that they are settable. Settable Elements may be in Service Capabilities, Service Configuration, Service Description and DefaultJob Ticket but not in Service Status.

The Service MUST reject the entire request with indications of which Element or Elements could not be set if a client request attempts to:

1. Set a non-settable Element (including an Element not in the Service Capabilities, Service Configuration, Service Description or DefaultJob Ticket groups, a read-only Element, and an Element not supported or not supported as a writable Element in the specific Service implementation)
2. Set a settable Element to an invalid value or to a value that conflicts with the values of other Service Elements, including Elements being set in the same request.
3. Set a greater number of Elements in one operation than are supported by the Service implementation (a Service implementation need not support set of more than one Element at a time).

A Set<Service>ServiceElements operation that specifies an Element but provides no value for that Element is not an error but rather a request to eliminate that Element and whatever value it has.

If there is no reason to reject setting all of the specified Elements to the specified values or elimination of the Element, the Service MUST accept this operation request when it is in the Idle or Stopped state, and SHOULD accept the request when it is in the Processing state.

If the Service accepts the request, only those Elements specified in the request are changed unless the definition of one or more of the set Elements explicitly specifies an effect upon some other Element.

1.1.2.12 Shutdown<Service>Service

The Shutdown<Service>Service operation forces the Service to the 'Down' state from any state that it is in, in an orderly manner. That is, the Service MUST stop accepting any further client requests, and MUST stop scheduling Jobs for processing as soon as the implementation allows, although it SHOULD complete the processing of any currently processing Jobs. Once down, the Service will no longer respond to any Client requests other than Restart<Service>Service request. As with the antecedent IPP Shutdown-Printer operation all Jobs MUST be preserved. As with Restart<Service>Service, Service shutdown must be performed as gracefully as possible and in a way in preserving the content and integrity of any non-terminated Jobs. Job history data, if supported, SHOULD also be preserved.

Once shut down, a Service can be roused from its Down state by a Restart<Service>Service operation. If a Service implementation supports Shutdown<Service>Service it must also support Restart<Service>Service and vice-versa. In the down state, the only operation request that a Service will respond to is a Restart<Service>Service operation.

Provided that the requestor is authorized, the Service MUST accept this operation and following an orderly progression, transition to the Down state regardless of the current state of the Service.

1.1.3 Administrative Cloud Imaging Service Operations

These Administrative Service operations directly affect the Cloud Imaging Service and/or affect the Jobs of multiple Job Owners. Access is reserved for Administrators or Operators. The Administrative Service Operations are listed in Table 3 and are described below. These operations are available only in Cloud Imaging Services that include a System Control Service.. These operations are accessible to only Users with proper administrative access rights to the Cloud Imaging Service. These operations do not directly affect the Cloud Imaging Device Manager(s) which connect to the Cloud Imaging Service, or the Devices with which these Cloud Imaging Device Managers interface.

Note that some operations parallel the <service> specific administrative operations described above. It is understood that an operations of this type but without the specific imaging Services specified in the operation, applies to the System Control Service.

Table 3 Administrative Cloud Imaging Service Operations

Operation	Request Parameters (Note 2)	Response Parameters	Note
DisableAllServices [1]	*ElementsNaturalLanguage, Message,* **RequestingUserName**		
EnableAllServices [1]	*ElementsNaturalLanguage, Message*, **RequestingUserName**		
GetSystemElements	*ElementsNaturalLanguageRequested*, **RequestedElements, RequestingUserName**	*ElementsNaturalLanguage, System Elements,*	
ListAllServices	*ElementsNaturalLanguageRequested*, , **RequestingUserName**	*ElementsNaturalLanguage*, **List of Service summary**,	
PauseAllServices [2]	*ElementsNaturalLanguage, Message*, **RequestingUserName**		
RestartAllServices [1, 5, 6, 7]	*ElementsNaturalLanguage, IsAcceptingJobs* | *IsAcceptingResources, Message*, **RequestingUserName,** *StartServicePaused*		
RestartService [3, 4, 5, 6, 7]	ElementsNaturalLanguage, **Id**, *IsAcceptingJobs* | *IsAcceptingResources, Message,* **RequestingUserName**, *ServiceType, StartServicePaused*		
ResumeAllServices [2]	*ElementsNaturalLanguage, Message,* **RequestingUserName**		
ShutdownAllServices [1]	*ElementsNaturalLanguage, Message,* **RequestingUserName**		
ShutdownService [1, 8]	*ElementsNaturalLanguage*, **Id**, *Message*, **RequestingUserName ServiceType**		
StartupAllServices [1, 5, 6, 7]	*ElementsNaturalLanguage, IsAcceptingJobs, Message*, **RequestingUserName**, *StartSystemPaused*		
StartupService [1, 5, 6, 7]	*ElementsNaturalLanguage*, *IsAcceptingJobs, Message,* **RequestingUserName ServiceType,** *StartServicePaused*	**Id**	
DeleteService [1, 8]	**Id**, *ElementsNaturalLanguage*, *Message*, **RequestingUserName**, **ServiceType**		
PauseAllServicesAfterCurrentJob [2]	*ElementsNaturalLanguage, Message*, **RequestingUserName**		

Operation	Request Parameters (Note 2)	Response Parameters	Note
SetSystemElements	*ElementsNaturalLanguage, Message,* **OperationMode**, **RequestingUserName SystemElements**	*UnsupportedElements*	

- 1 The operations do not apply to the SystemControlService.
- 2 The operation only applies to Job based Services (e.g., CopyService, FaxOutService, FaxInService, PrintService, ScanService, and TransformService),
- 3 When the target Service is the SystemControlService the implementation MUST restart the SystemControlService and MAY restart the other Services as well.
- 4 When the target Service is the SystemControlService the implementation of the restart may be soft (i.e., affects software only) or hard (i.e., hardware and software reinitialized).
- 5 When the Service startup is complete the Service state is 'Idle' (See note 6). The Service will then follow the Service state model as defined in section 7.2.1 of [PWG5108.01]
- 6 When the operation contains the "StartServicePaused" parameter and it is set to 'true', the resulting Service state is 'Stopped' (i.e., transitions from 'Down' to 'Idle' then immediately to 'Stopped'). The Service will then follow the Service state model as defined in section 7.2.1 of [PWG5108.01]
- 7 When the operation contains the "IsAcceptingJobs" or "IsAcceptingResources" parameter and it is set to 'false', the Service state is 'Idle' (See note 6). The Service will then follow the behaviors as defined in section 7.3.2.2 of [PWG5108.01] or section 8.2.1 of [PWG5108.03] respectively.
- 8 This operations results in an error when applied to the SystemControlService.

1.1.3.1 DeleteService

The DeleteService operation removes an instance of a Service. The result is that all data associated with the identified Service is deleted and that Service can no longer be restarted. It is an error to specify a Service that is not shutdown or to specify the SystemControlService itself.

1.1.3.2 DisableAllServices

The DisableAllServices operation is consistent with the operation Disable<Service>Service specified in [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and MUST set the IsAcceptingJobs/ IsAcceptingResources Element to 'false' for all hosted Services. This operation does not affect the SystemControlService itself. This operation has no effect upon the Services' State elements.

1.1.3.3 EnableAllServices

The EnableAllServices operation is consistent with the operation Enable<Service>Service specified in [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and MUST set the IsAcceptingJobs/ IsAcceptingResources Element to 'true' for all hosted Services. This operation has no effect upon the Services' State elements. This operation does not affect the SystemControlService itself.

1.1.3.4 GetSystemElements

Unlike the Get<Service>ServiceElements [PWG5108.01] operation that allows access to only the elements of the specified <Service>, the GetSystemElements operation allows a SystemControl Client to obtain detailed information about the System Object as well as the SystemControlService.

For the SystemControlService, this operation can request the elements directly below the SystemControlService element (e.g., ServiceDescription, ServiceStatus). This operation MUST NOT query information from any other Service.

For the Cloud Imaging Service, this operation can request the elements directly below the System element (e.g., SystemConfiguration, SystemDescription, and SystemStatus).

1.1.3.5 ListAllServices

This operation provides summary information on all Cloud Imaging Service hosted Services including the SystemControlService. The response returns a ServiceSummary for each Service that includes Id, ServiceName, ServiceState, ServiceStateReasons for the Service's endpoint and other general information.

1.1.3.6 PauseAllServices

The PauseAllServices operation is consistent with the operation Pause<Service>Service specified in [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and transition all the currently active job based Services (e.g., CopyService, FaxOutService, FaxInService, PrintService, ScanService, TransformService) to the Stopped state. During the transition each <Service>ServiceStateReasons MUST contain the reason 'MovingToPaused'. This operation does not affect the SystemControlService.

1.1.3.7 PauseAllServicesAfterCurrentJob

The PauseAllServicesAfterCurrentJob operation is consistent with the operation Pause<Service>ServiceAfterCurrentJob specified in [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and transition all the currently active job based Services (e.g., CopyService, FaxOutService, FaxInService, PrintService, ScanService, TransformService) to the Stopped state in an orderly manner. During the transition each <Service>ServiceStateReasons MUST contain the reason 'MovingToPaused'. No pending jobs may be scheduled and all processing jobs will complete. This operation does not affect the SystemControlService.

1.1.3.8 RestartAllServices

The RestartAllServices operation is consistent with the operation Restart<Service>Service specified in [PWG5108.01]. This operation does not affect the SystemControlService. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and MUST reinitialize all hosted Services, except the SystemControlService. This includes setting the State to 'Idle', clearing the StateReasons Element and setting the IsAcceptingJobs/IsAcceptingResources Element to 'true' if applicable. Note that parameters control subsequent Service behavior (See the last paragraph in this section). When the Service startup is complete the Service state is 'Idle' (See below). The Service will then follow the Service state model as defined in section 7.2.1 of [PWG5108.01].

When the operation contains the "StartServicePaused" parameter and it is set to 'true', the resulting Service state is 'Stopped' (i.e., transitions from 'Down' to 'Idle' then immediately to 'Stopped'). The Service will then follow the Service state model as defined in section 7.2.1 of [PWG5108.01]. When the operation contains the "IsAcceptingJobs" or "IsAcceptingResources"

parameter and it is set to ‘false’, the Service state is ‘Idle’ (See note 6). The Service will then follow the behaviors as defined in section 7.3.2.2 of [PWG5108.01] or section 8.2.1 of [PWG5108.03] respectively. Parameters that do not apply to target Service are silently ignored.

1.1.3.9 RestartService

The RestartService operation is consistent with operation Restart<Service>Service specified in [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and MUST reinitialize the specified Service State to ‘Idle’, clear the StateReasons Element and set the IsAcceptingJobs/IsAcceptingResources Element to ‘true’ if applicable. Note that parameters control subsequent Service behavior (See the last paragraph in this section). When the Service startup is complete the Service state is ‘Idle’ (See below). The Service will then follow the Service state model as defined in section 7.2.1 of [PWG5108.01]. This operation can specify any Service including the SystemControlService.

When the SystemControlService is the target of this operation the system behavior is implementation specific. The implementation may reinitialize the existing Service or shutdown and instantiate the SystemControlService. It is also implementation specific whether or not restarting the SystemControlService also causes a restart of all the other hosted Services.

When the operation contains the “StartServicePaused” parameter and it is set to ‘true’, the resulting Service state is ‘Stopped’ (i.e., transitions from ‘Down’ to ‘Idle’ then immediately to ‘Stopped’). The Service will then follow the Service state model as defined in section 7.2.1 of [PWG5108.01]. When the operation contains the “IsAcceptingJobs” or “IsAcceptingResources” parameter and it is set to ‘false’, the Service state is ‘Idle’ (See note 6). The Service will then follow the behaviors as defined in section 7.3.2.2 of [PWG5108.01] or section 8.2.1 of [PWG5108.03] respectively. Parameters that do not apply to target Service are silently ignored.

1.1.3.10 ResumeAllServices

The ResumeAllServices operation is consistent with the operation Resume<Service>Service specified in [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and transition every job based Service (e.g., CopyService, FaxOutService, FaxInService, PrintService, ScanService, and TransformService) to the ‘Idle’ state. The Service will then follow the Service state model as defined in section 7.2.1 of [PWG5108.01]. This operation does not affect the SystemControlService.

1.1.3.11 ShutdownAllServices

The ShutdownAllServices operation is consistent with the operation Shutdown<Service>Service specified in [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and forces each of the Services, except the SystemControlService, to the ‘Down’ state from any state that it is in, in an orderly manner. This operation does not affect the SystemControlService itself.

1.1.3.12 ShutdownService

The ShutdownService operation is consistent with the operation Shutdown<Service>Service specified in [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and force the specified Service to the ‘Down’

state from any state that it is in, in an orderly manner. It is an error to specify the SystemControlService itself.

1.1.3.13 StartupAllServices

The StartupAllServices operation initializes all the shutdown Services and takes them through the ‘Down’ state to ‘Idle’, assuming that there are no inhibiting conditions See sections 7.2.1 of [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request and initializes each of the Services, except the SystemControlService. This operation does not affect the SystemControlService itself.

1.1.3.14 StartupService

The StartupService operation creates a new instance of the specified Service type and takes it through the ‘Down’ state to ‘Idle’, assuming that there are no inhibiting conditions See section 7.2.1 of [PWG5108.01]. If the requestor is determined to have proper access, the SystemControlService MUST accept this request, create a new instance and initialize the Service of the specified type. It is an error to specify the SystemControlService type.

1.1.3.15 SetSystemElements

Unlike the Set<Service>ServiceElements [PWG5108.01] operation, the SetSystemElements operation allows a SystemControl Client to modify information about the System Object as well as the SystemControlService Elements.

For the SystemControlService, this operation can set the SystemControlService’s settable elements (i.e., elements in ServiceDescription and none in ServiceStatus). This operation MUST NOT set elements from any other Service (i.e. any other Service under the Services element at the System root).

For the System Object, this operation can set settable elements the elements directly below the System element (i.e., elements in SystemConfiguration and SystemDescription but not in SystemStatus). The SystemConfiguration element in a SetSystemElements operation has additional rules and an alternative syntax.

The alternative syntax for SystemConfiguration permits schema enforcement of setting only a few elements within a SystemConfiguration element. Although an element may be mandatory in the model, the SetSystemElements operation need not contain a mandatory element unless it is the element being set.

An alternative syntax is also used when an element is a reference to another element instead of a contained element (e.g., InputChannelInterface in System.SystemConfiguration.InputChannels.InputChannel). For simplicity and convenience of the GetServiceElements operation, when accessing an element with a referenced association to another element, the entire referenced element is replicated in place. Thus there is no need to use the reference identifier and make another query to obtain the information. However when the SetSystemElements operation acts upon an element with a referenced association, it action is to modify the reference identification and not the referenced element. Therefore when a SetSystemElements operation modifies an element with a referenced association, the element value will be an integer that corresponds to the identifier of the referenced element. To modify

the referenced element itself, the elements themselves (e.g., Interface in System.SystemConfiguration.Interfaces) are modified using the SetSystemElements operation.

1.2 Cloud Imaging Device Manager to Cloud Imaging Service Operations

In a traditional imaging model, operations are initiated by the agent forwarding the Imaging request; i.e., the Client sends requests to a Service and the Service may send requests to a subordinate Service, such as one in a Device. However, in this Cloud Imaging Model, it is likely that a Cloud Imaging Service is isolated from the Cloud Imaging Device Manager by a firewall and cannot initiate requests. Therefore, the following operations are used by the Cloud Imaging Device Manager to get Imaging Job information from and provide Device and Job status to the Cloud Imaging Service.

The following characteristics of the model must be observed in understanding these operation descriptions.

- All Operations are in a request/response form with the request sent by the Cloud Imaging Device Manager and the response sent by the Cloud Imaging Service. The protocol used must assure correlation of request to response. The content of requests and responses will typically be reversed compared to analogous operations in a traditional Imaging model.
- A Cloud Imaging Device Manager can interface with multiple Cloud Imaging Services.
- A Cloud Imaging Service can interface with no more than one Cloud Imaging Device Manager.
- The protocols used by the Cloud Imaging Device Manager in initiating requests to the Cloud Imaging Service must provide for the identification and authentication of the Cloud Imaging Device Manager, as well supporting security requirements appropriate to the use of the Cloud Imaging facility.
- Some Cloud Imaging Device Managers can front-end multiple Imaging Devices. The Cloud Imaging Device Manager may reports capabilities and status values for each Device individually; or it can report capabilities and status values which are an intersection or union of capabilities and status of the devices it represents. In the former case, specific devices may be selected by the User through the Cloud Imaging Serice. In the latter case, the Cloud Imaging Service has no knowledge of the individual devices and it is up to the Cloud Imaging Device Manager to schedule Jobs and map Jobs to Imaging Devices

890

891 **Table 3 - Cloud Imaging Device Manager to Cloud Imaging Service Operations**

Operation	Request Parameters	Response Parameters	Note
GetFetchable<Service>Jobs			
Fetch<Service>Job			
Acknowledge<Service>Job			
Fetch<Service>Document			
Acknowledge<Service>Document			
Put<Service>JobDocumentData			
Update<Service>ServiceState			
Update<Service>JobState			
Update<Service>DocumentState			
UpdateFetchable<Service>Jobs			

892

893 ### 1.2.1 GetFetchable<Service>Jobs.

894 GetFetchable<Service>Jobs is a request for the list of jobs ready to be fetched by the Cloud
895 Imaging Device Manager. The Cloud Imaging Device Manager will use the response to this
896 request to identify the requested Jobs in its subsequent FetchImagingJob request.

897 The operation can accommodate job scheduling at either the Cloud Imaging Service or the Cloud
898 Imaging Device Manager. When the Cloud Imaging Service is handling job scheduling, the
899 Cloud Imaging Server will return a list containing at most a single Job. The Job is identified by
900 its JobUuid in the Cloud Imaging Server. If the Cloud Imaging Device Manager (or the
901 Imaginger) does job scheduling, the Cloud Imaging Service response is a list of fetchable jobs
902 including a job summary element group (i.e., job summary collection in IPP) and a minimal set
903 of information useful for scheduling (e.g., Finishings, Media, ImagingColorModeType, Sides),
904 in addition to the JobUuids.

905 When the Cloud Imaging Device Manager is registered, the registration information will
906 determine whether the Cloud Imaging Device Manager or the Cloud Imaging Service is to do
907 Job scheduling.

908 ### 1.2.2 Fetch<Service>Job.

909 Once the Cloud Imaging Device Manager has received a response to a
910 GetFetchable<Service>Jobs request indicating that there are one or more Jobs waiting, it sends a
911 Fetch<Service>Job request to the Cloud Imaging Server. This request includes the Cloud
912 Imaging Device Manager Job Uuids reported in the GetFetchable<Service>Jobs response which
913 correspond to the Jobs the Imaging Manager wishes to receive.

914 The Fetch<Service>Job response is analogous to the request portion of CreateImagingJob. This
915 response includes the operational attributes of the Job request (e.g., RequestingUserName,
916 JobPassword) as well as the Job's ImagingJobTicket information. It does not include either the
917 document data or a reference to document data; the Cloud Imaging Device Manager must issue a
918 FetchImagingDocument message to get this data.

1.2.3 Acknowledge<Service>Job.

The Acknowledge<Service>Job operation is analogous to the CreateImagingJob response in a traditional Imaging model. This operation identifies the Job by the Cloud Imaging Service Uuid and correlates this to the newly created Job's Imaginger JobUuid and Job State along with any UnsupportedElements. If the Job request is rejected, this status along with appropriate reason information is communicated.

The Cloud Imaging Service response to this message returns the state of the subject Imaging Job in the Cloud Imaging Service. This response serves to confirm that the Acknowledge Imaging Job message was received, as well as to inform the Cloud Imaging Device Manager of any externally prompted state change (e.g., a Client Job Cancel) or to inform the Cloud Imaging Device Manager of some error or inconsistency in the message (e.g., reference to a non-existent or not available job.)

1.2.4 Fetch<Service>Document.

After the Cloud Imaging Device Manager/Device has created the job, it may eventually need specific Document information. The Cloud Imaging Device Manager Fetch<Service>Document operation retrieves the Document or Document Data reference along with operational elements for Jobs using Services that require a Digital Data input; this include Print, FaxOut and EmailOut Services. The operation can also be used for Services where some aspects of Device functionality are handled by the Cloud Imaging Service, such as for FaxIn if the facsimile input and EmailIn if fetch from the EMail server is handled by teh Cloud Imaging Service. The request must include the Job and Document identification corresponding to the information received in response to the GetFetchable<Service>Jobs operation.

The Fetch<Service>Document. response is analogous to the request portion of the Send<Service>Document or Send<Service>Uri operation. This response includes the operational attributes (e.g., RequestingUserName, JobPassword) as well as the Document Data content (i.e., the Document Digital Data itself or a reference to it) for the requested Document. If supported, a DocumentTicket can also be passed.

1.2.5 Acknowledge<Service>Document.

The Acknowledge<Service>Document operation is sent by the Cloud Imaging Device Manager after the response to the Fetch<Service>Document has been received. The operation is analogous to the Send<Service>Document or Send<Service>Uri response in a traditional Imaging model. This operation identifies the newly created Device DocumentUuid and State along with any UnsupportedElements, if applicable.

The CloudImagingService response to this message returns the state of the subject Document in the Cloud Imaging Service. This response serves to confirm that the Acknowledge Imaging Document message was received, as well as to inform the Cloud Imaging Device Manager of any externally prompted state change (e.g., a Client Job Cancel) or to inform the Cloud Imaging Device Manager of some error or inconsistency in the message (e.g., reference to a non-existent or not available document.)

1.2.6 Put<Service>JobDocumentData.

The Cloud Imaging Device Manager sends a message containing the Document Data that an associated Device has obtained in executing a Job that requires the Service output Digital Document Data and that the specified destination for this data is not directly accessible to the Imaging Device Manager or the Imaging Device.. From the definition of Imaging Services, this includes Scan and EmailIn Jobs, and may include Print and FaxIn Jobs.

The message includes the Cloud Imaging Service Uuid for the Job and

1.2.7 Update<Service>ServiceState.

The Cloud Imaging Device Manager sends a message reporting its current state whenever its state changes, along with state message and reasons. The state of the Cloud Imaging Device Manager considers both its condition and the state of the Devices(s) with which it interfaces. The operation includes a sparsely populated object of the appropriate type. For example if the configuration of an interfaced Imaging Device changes in a way to affect the composite Cloud Print Manager state, then the UpdateDeviceState request would contain only the relevant portions of the composite ImagingServiceConfiguration. If media were added, removed or changed in an input tray, the InputTrays element group would be returned. The state that the Cloud Imaging Service reports to the Client will usually reflect this Cloud Imaging Device Manager state.

The Cloud Imaging Service response is primarily an acknowledgment of message receipt, but optionally may include the revised state of the Cloud Imaging Service.

1.2.8 Update<Service>JobState.

The Cloud Imaging Device Manager sends a message reporting the current state of an identified Imaging Job whenever that state of that Job changes, along with state message and reasons. The Job state in the Cloud Imaging Device Manager considers the state of the Job in the Device to which it was directed. The operation includes a sparsely populated object of the appropriate type. For example, if the Imaging Device completes a Job, the Update<Service>JobState message would contain the elements in Imaging DeviceJobStatus that have been changed and a final version of the ImagingDeviceJobReceipt.

The Cloud Imaging Service response is primarily an acknowledgment of message receipt, but optionally may include the revised state of the subject Job in the Cloud Imaging Service.

1.2.9 Update<Service>DocumentState:

The Cloud Imaging Device Manager may send a message reporting the current state of a identified Document whenever that state of that Document changes, along with state message and reasons. The Document state in the Cloud Imaging Device Manager considers the state of the corresponding Job and Document in the Imaging Device to which the Job was directed. In some cases, as when the Cloud Imaging Device Manager is doing acquisition of referenced Document Data or preprocessing, Document state may be determined by the Cloud Imaging Device Manager rather than the servicing Imaging Device. The state that the Cloud Imaging Service reports to the Client will usually reflect this Cloud Imaging Device Manager reported state.

The Cloud Imaging Service response is primarily an acknowledgment of message receipt, but optionally may include the revised state of the subject Document in the Cloud Imaging Service.

1.2.10 UpdateFetchable<Service>Jobs.

The Cloud Imaging Service relies upon the Acknowledge<Service>Job and Update<Service>JobState messages from the Cloud Imaging Device Manager to follow the state of each Job, and uses this information to synchronize its state for that Job, which is what is communicated to the Client. If the communication from the Cloud Imaging Device Manager is disrupted, or if the Cloud Imaging Device Manager (or perhaps the Device to which the Job has been directed) is reset, this synchronism is lost. The UpdateFetchable<Service>Jobs message for each Service supported must be sent by the Cloud Imaging Device Manager when it senses that communication with the Cloud Imaging Service has been restored after a disruption, after any hard reset, and after power-up initialization. The UpdateFetchable<Service>Jobs message for a specific Service must be sent by the Cloud Imaging Device Manager when any Device handling that Service has been hard reset or otherwise may have lost track of its Jobs. This message allows the Cloud Imaging Service to resynchronize itself with respect to which of the jobs it has made available have been accepted by the Cloud Imaging Device Manager and with the states of those Jobs.

The UpdateFetchable<Service>Jobs message includes a list of Jobs that the Cloud Imaging Device Manager is aware that it has fetched and has acknowledged or has intended to acknowledge, along with the current states of these jobs. The list for each Service includes all fetched Jobs up to but not including Jobs for which the Cloud Imaging Device Manager has sent and the Cloud Imaging Service has acknowledged an Update<Service>JobState message indicating that the Job is in a terminating state. Jobs are identified by their Cloud Imaging Service Uuid (i.e., the same way that they were identified in the GetFetchable<Service>Jobs response).

On receiving the message, the Cloud Imaging Service moves any jobs it believes have been fetched and not completed but are not in the list provided by the Cloud Imaging Device Manager back to the fetchable job list, and readjusts the state of all Jobs listed in the UpdateFetchable<Service>Jobs message. The response to this message is a simple message received acknowledge.