10 Scan Service Theory of Operation

The Scan Service operates autonomously through three phases: Initial, Online, and Offline.

On creation, the Scan Service enters its Initial phase during which all its service attributes and connected subunits are initialized. This phase may include test of the Scanner Subunit(s) and self-testing of the Scan Service. After successful initialization, the Scan Service performs a start-up operation that brings the service Online, authenticates and registers its service with a service directory or announces its service to the network domain in which it resides. The Scan Service is then in the “Idle” state, is ready for service discovery and can accept service requests from Scan Clients.

The Scan Service generally accepts new requests as long as it is in one of the three Online phase states: Idle, Processing or Stopped. However, performing an administrative Disable() operation while in any state will stop the Scan Service from accepting a CreateScanJob() request to start a new job. Performing an Enable() operation in any state while the Scan Service is disabled will allow new jobs to be accepted again.

A user submits a Scan Job through a local (via the MFD user interface) or remote (via local network or Internet) Scan Client to a selected target Scan Service that has the desired scan capabilities. While the service is in the Online phase, it will accept all implemented Scan Service operations specified in Section 11, except that it will not act on a CreateScanJob if the Scan Service is Disabled. While the service is enabled, the Scan Client may use the CreateScanJob operation to submit a Scan Job on behalf of a user.

The Scan Service places all submitted jobs in the ActiveJobs queue where the Jobs are in the Pending State until they are advanced to the Processing State. A Pending Job may be held to delay scheduling for processing by a JobHoldUntilTime attribute in the Scan Job’s Ticket to allow time for user to walk up to the scanner and place the Hardcopy original on the scanner; a Pending Job may also be held by an administrator HoldScanJob() operation. Held jobs are released by a JobHoldUntilTime timeout or a administrator ReleaseJob() operation. Unheld jobs are scheduled for processing immediately or when a StartJob event is signaled, based on job priority relative to other jobs in the Active Jobs queue.

When a Scan Job is released for scheduling and reaches the top of ActiveJobs queue, it enters the JobProcessing state. If the Scan Service is not already in the ServiceProcessing state, it will now enter it. When job processing is complete, the Job is in the Completed state and be entered into the JobHistory queue. The Scan Service either returns to Idle or continues in the ServiceProcessing state with a subsequent Job.

During job processing, an administrative “PauseScanService()”or “PauseScanServiceAfterCurrentJob” operation will cause the Service to enter the “Stopped” state. This prevents further scheduling of jobs. Depending on implementation, the Service transition to Stopped may be delayed to allow the processing of the current job to be completed or suspended in a controlled way. Jobs that are in the JobProcessing state when the Scan Service is Stopped remain in the Active Jobs Queue but go into the ProcessingStopped job state. A

“PauseScanService()” operation allows a user to interrupt the scheduled jobs to submit and process an urgent Scan Job or a job for another service. A ScanServiceResume() operation returns the ScanService to the Processing state and allows job scheduling and processing to resume. Note that, if the Service Pause/Resume operations are to be used to allow a job processing to be interrupted to allow another job to be processed by the same service, then the inserted job must have higher scheduling priority than a ProcessingStopped job and the Service must readjust its schedule on receiving the Resume command.

When there are critical conditions impacting Scan Serviceability during “Idle” or “Processing” state, either a C.Critical event is generated or an Administrative PauseScanService() is performed to bring the service to the Stopped state. From there the condition can be fixed by user’s intervention. Then either the Scan Service generates a ~C.Critical event (removal of critical condition) or an administrator performs a Resume() operation to bring the Scan Service back to the “Idle” or “Processing” state. If the Scan Service requires a ShutdownScanService() operation before service can be resumed or for testing, a RestartScanService() must be issued to bring the Service to the On-Line phase again.

Any job in the ActiveJobs queue can be canceled via a CancelScanJob() operation by an authorized user, a. The Job is then in the Cancelled state and entered into the JobHistory queue. Jobs that are terminated because of a Service fault are in the Aborted state and are also entered into the JobHistory queue.