

DTS-E2E ICD

Version 0.8 (12/12/2011)

Authors: I.Barg, M.Fitzpatrick,

D.Scott, R.Seaman

Table of Contents

Purpose 3

Scope 3

Overview 3

The current NOAO E2EV16 system: 3

DTS Scope and High-Level Requirements: 4

E2EV17 Phased DTS integration: 4

SDM Operations Requirements: 4

Interface Documentation Template 5

Interfaces 5

1. At the Dome: 6

2. At the Mountain Cache: 8

3. At a SDM Archive Site: 10

4. At NCSA: 12

Operational DTS-E2E Interfaces 13

Delivery Application Interface 14

Definitions 15

References 16

Appendix A: iSTB DTS requirements 17

# Purpose

This document specifies interfaces and requirements for the Data Transport System (DTS) to interact with the NOAO End to End Science Archive System (hereafter in this document the E2E SYSTEM). The purpose of this document is to serve as a guide to designers and developers who are responsible for incorporating the DTS within the current E2E SYSTEM.

# Scope

This document contains a complete description of the interfaces between the current E2E system and the DTS as it relates to Dark Energy Camera (DECam) source data only. It describes the communication (inputs and outputs) and monitoring requirements between the DTS and E2E system as it relates to DECam data transport and ingest onto the NOAO data stores and searchable Archive. Specifically, it addresses the interfaces between the DTS and existing E2E SYSTEM components internal to NOAO.

This document will not discuss the data content and format requirements addressed in DECam Community Pipeline - E2E Interface Control Document (DCP-E2E-ICD) [1].

# Overview

## The current NOAO E2EV16 system:

The current NOAO E2E system as of Version 1.6 (hereafter in this document referred to as E2EV16) is a collection of physical systems and distributed services that collect, store and move astronomical data between NOAO data sources (mountain tops, pipelines) and data stores (physical and logical). Both data sources and stores are geographically distributed. Data enter the system through Save-the-Bits (iSTB) either at the mountain caches or pipeline caches in Tucson. iSTB adds some additional keywords to the FITS headers to ensure that data origin and ownership are recorded in the raw (source) data FITS headers. Once this metadata has been added, iSTB passes the file to the Integrated Data Cache Initiative (iDCI). The iDCI ingests metadata about each file (file size, md5sum and physical location) into a local database. The raw data are then copied to 2 data centers (La Serena, Chile and Tucson, Arizona) and one tape copy at the National Center for Supercomputing Applications (NCSA) in Urbana, Illinois. iSTB and iDCI together provide a fully automated E2E data transport and file repository management system.

## DTS Scope and High-Level Requirements:

The DTS as described in the DECam DES/SISPI ICD [3], is primarily a delivery mechanism for DES components that receive and transport data from the DECam instrument. The DTS as described in [3] will replace the transport portion of the current E2EV16 system. In order to incorporate the DTS into the current E2E system, there two major design criteria for the DTS:

1. The DTS design shall maintain the overall functionality of the existing transport portion of E2E system without imposing major changes to existing E2E code.
2. The DTS shall comply with the same requirements placed on the transport portion of the current E2E system in production [6]. These are summarized below:
   1. The DTS shall provide an interface for external E2E boundary objects.
   2. The DTS will guarantee the reliable and immutable transfer of data between all start and end points controlled by the DTS.
   3. The DTS will operate in such a way as to maximize its use of available bandwidth for bulk data transfer without interfering (in any significant way) with normal network traffic.
   4. The DTS will persist the state of pending data transfers across network outages, system failures and unexpected crashes of the software, recovering automatically once local or remote services become available.
   5. The DTS shall be configurable as to provide flexible routing of data to alternate sites.
   6. The DTS should provide a means to monitor and change the state of the system by operations staff.
   7. The DTS should provide a choice of transfer protocols to be used, allowing the operator to choose a protocol suitable for a particular network environment or for the type of data to be moved.

## E2EV17 Phased DTS integration:

The DTS will be used for delivery of DECam data, both DES source (raw) data and community data. Source data from other NOAO instruments will continue to use the existing mechanism found in the current E2EV16 system.

## SDM Operations Requirements:

All code to implement these interfaces (DTS, iSTB and iDCI) shall be delivered as a package that can be installed, configured and maintained by SDM Operations.

* The code shall be submitted to the SDM SVN repository;
* Documentation for each interface package should include:
  + how to install and configure the interface including required:
    - user accounts;
    - port numbers;
    - firewall rules;
    - list of pre-requisite middle-ware;
  + trouble-shooting guide;
  + user’s guide.

# Interface Documentation Template

A standard organization will be used to document each interface in this document. It consists of the following sections.

1. Interface Identity – The name of the interface.
2. Resources – The set of resources provided to its actors. Resources can be operations (such as methods, procedures, and functions), but can be more general things, such as data streams, shared data, and messaging. For each resource you should describe:
   1. *Syntax* – The signature, includes the name of the resource, names and data types of arguments, return values, etc..
   2. *Semantics* – What is the result of using this resource?
   3. *Error Handling* – Describe error conditions and exceptions that can be raised by this resource.
   4. *Requirements* – Requirements placed on either input/output interfaces.

# Interfaces

This ICD describes the standard approach to transporting DECam source (raw) data from the instrument to the CTIO Mountain cache, adding the required iSTB metadata to the headers before transporting copies downstream to NOAO archive in La Serena, Chile, the Dark Energy Survey (DES) at NCSA, Urbana, Illinois, and terminating at the NOAO archive in Tucson, Arizona.

The boundary objects are resource elements from four different sub-systems at different locations along the transport path:

1. **At the Dome**: Telescope instrument data acquisition system (DHS)
2. **At the Mountain Cache**: Save-the-Bits (iSTB), Data Transport System (DTS).
3. **At a SDM Archive Site**: Integrative Data Cache Initiative (iDCI) at La Serena and Tucson
4. **At NCSA**: End point delivery for DES survey data

This document will describe the interfaces to these four sub-systems.

# 1. At the Dome:

The DECam telescope instrument data acquisition system is called SISPI. The SISPI/Image builder will interface with iSTB via a *postproc* call.

Resource: iSTB postproc:

Postproc is lightweight, quick, and guaranteed not to block.

*Syntax:*

The SISPI/Image builder calls *postproc* (synchronously in the foreground):

/bits/bin/postproc <pathname to file>

*Semantics*:

The *postproc* will call both *dtslogger* and *dtstracker*, capture the context metadata (including calendar date and timestamp) and call *dtsq* directly and synchronously.

*Error Handling:*

* Queuing silently fails, daily monitoring detects and corrects absence.

*Requirements:*

* SISPI:
  + The current implementation of SISPI/IB will need to be modified to call the *postproc* application.
* ISTB:
  + **super** (root) access to SISPI/IB host is required for SDM operations staff;
  + User **bits** and **cache** accounts are required on SISPI/IB host.
  + These requirements will need to be implemented on each of the seven IB machines used by DECam, although a common code base can be accessed by means of shared network disks.
  + iSTB applications may not modify the target file on the SISPI/IB systems.

Resource: DTS DTSQ:

The DTS interface in the telescope dome is the DTSQ application that submits a file to the DTS for transport.

*Syntax:*

The DTSQ is typically executed by the by a ‘postproc’ script and using the following syntax:

dtsq [- q <qname>] [-f] <param>=<value> /path/file

The ‘-q’ flag is used to specify the transport queue and is optional because a default queue can be configured for the system. The ‘-f’ flag forks a child process to move the file to the DTSD at the mountain cache. This allows the caller to regain control quickly (<< 1sec). The <param>=<value> mechanism allows the DTSQ to be called with arbitrary keyword/value information that will be propagated through the DTS control files, and can later be accessed by downstream sites in the DTS Delivery application, without requiring DTS to know the specifics of how it is used with E2E.

*Semantics*:

The DTSQ is the queuing agent that submits a file for ingestion into the DTS. It allows a quick response so it won’t block the caller and provides its own transport methods to move the file to a remote DTSD.

*Error Handling*:

On any error the DTSQ command sets the status to 1 (one). When the ‘-f’ flag is set, a status of 0 (zero) is returned if the DTSD can be successfully contacted and is willing to accept the file. It does NOT indicate that the file was successfully moved. Without the ‘-f’ flag, the DTSQ will return a 0 (zero) upon a successful copy of the file to the DTSD.

*Requirements:*

* DTS:
  + The DTSQ binary and configuration file must be installed on the SISPI system.
    - The DTS will provide documentation of the configuration file to include directive definition and syntax.
  + The target file cannot be changed, moved or deleted until the DTSQ processing is completed.
  + Monitoring tools must exist to facilitate monitoring and fixing failed submissions. The DTSQ tools to facilitate monitoring include:
    - ‘dtsq –r’ - resubmit failed transports
    - ‘dtsq –V’ – validate submissions (print a report on what was submitted, what is left to recover).
  + The SDM Operations has existing utilities to monitor the E2E system. In order to incorporate the DTS into the existing E2E monitoring utilities, the DTS must provide two types of logs: DTS status log, and DTS transfer log. The content and format of these logs are described in section Operational DTS-E2E interfaces.
* E2E:
  + The DTS Operator is responsible for monitoring and recovering failed DTSQ submissions using the DTS monitoring tools and logs as described in Operational DTS-E2E interfaces.
  + If the dtsq is called with the ‘-f’ flag, it is the responsibility of the ~~delivery command or~~ Operator to validate submissions and/or resubmit failed transports.

# 2. At the Mountain Cache:

There are two interfaces on the mountain cache: the DTSD and the ‘ingest script’ that triggers iSTB processing.

Resource: DTSD:

The DTS Daemon is the main service daemon running at each site that is responsible for managing transport queues [3]. For DECam, the mountain cache is the first instance of a DTS Daemon in the data flow. The DTSD provides a special ‘ingest’ interface to a queue. The ingest interface is the only place where the data are allowed to be modified by the delivery application, in this case, an iSTB application to add keyword information.

*Syntax*:

The DTSD is configured, not called. Additional parameters may be added to the control file by the delivery application by creating a file called “*<queue>.par*” in the current DTS spool directory. The format of this file is “*<key> = <val>*“ pairs, one per line. The special parameter “deliveryName” may be used to specify the name of the file that is delivered to downstream sites. The first mechanism allows the delivery application to append information (e.g. results of processing, logging info, etc) that might be useful to a downstream site. The second mechanism allows a file to be delivered/identified in the application with a new name. Parameters passed to delivery applications allow any information in the control file to be accessed.

*Semantics*:

Transport of the data from the DTSQ goes directly to the DTS queue spool directory on the DTSD machine, transport is only considered successful once the file size and MD5 sum have been verified. A pre-configured delivery application is executed and upon successful completion, the transport control parameters (i.e. file size and checksums) are recalculated (only in the special case of an *ingest queue* such as the mountain cache, thereafter the file is never modified) before queue processing resumes.

*Error Handling*:

* The ‘ingest script’ shall return one of three status codes:
  + A value of zero (0) means the script executed with no errors
  + A value of one (1) means some processing error occurred but transport can continue. E2E will supply both the ingest and downstream delivery applications and can decide whether in this case additional parameters should be added to the control file for use at a downstream site.
  + A value of two (2) means a severe processing error occurred and DTS should abandon further transport of the file.

*Requirements:*

* DTS:
  + The DTSD binary and configuration file must be installed on the mountain cache.
    - The DTS will provide documentation of the configuration file to include directive definition and syntax.
  + A delivery command will call a pre-defined “*ingest script*” to operate on the DTS target file.
  + The DTS target file cannot be moved or deleted until DTSD processing is complete.
  + Sufficient storage[[1]](#footnote--1) is available to manage the DTS queue areas.
  + Monitoring tools must exist to check the status of queue processing.
  + Logging:
    - DTSD status log format is described in Operational DTS-E2E interfaces.
* E2E:
  + The DTS Operator is responsible for monitoring the DTSD.
  + The DTS Operator must be able to enable debugging, start and/or stop the DTS if necessary.
  + The DTS Operator or local daemon is responsible for monitoring the success or failure of the delivery application.
  + The ingest/delivery application must not be limited to processing of FITS files only. Unrecognized file formats should be processed to the extent possible (e.g. a copy is made, logging recorded etc) without returning an error code to the DTS.
  + The E2E *ingest script* is responsible for ensuring the validity of files that are archived (e.g. FITS file processing produces a valid file, unrecognized file are unmodified, and error during processing leaves the input file unchanged if transport continues, etc).

Resource: iSTB ingest script:

The DTSD has transferred source file to a pre-configured *delivery directory*, checks the md5sum then triggers iSTB ingest using the *istbcmd* command. (In the case of an ingest queue on the mountain, the delivery directory is the DTS spool directory itself.)

*Syntax:*

/noaosw/zdts/bin/istb-proc $D $MD5 $S

* Inputs follow the macro substitutions defined by the DTS Delivery Application Interface:
* $D – path to delivered file (delivered filename)
* $MD5 – MD5 checksum
* $S – object size (bytes)

*Semantics:*

*istbcmd* will call a modified version of iSTB that will read from the DTS spool directory, blocking that thread and all other ingest script instances on other threads. iSTB will write to the same hierarchy as currently, (NOTE: a separate partition than the DES 3-day store). After iSTB successfully creates the first archival copy, it will:

* execute the *DciArchT* script as it currently does, to add the file to the iDCI registration queue;
* update the DTS “*queue.par”* with “DeliveryName”.

*Error Handling:*

*Istbcmd* will return to the calling program:

* 0 (zero) - file successfully processed;
* 1 (one) - some error with DTS target file in TBD format.
* (see above for required error codes)

*Requirements:*

DTS:

* On 'delivery', the DTS must write to a pre-configured DTS spool directory.
* After 'delivery' the DTS must verify the 'md5sum' on local disk.
* The DTS target file must be readable by **cache** user account.
* All intervening directories in the DTS pathname must be readable and executable by user **cache.**

iSTB:

* The modified iSTB code must co-exist with existing E2E processing of other NOAO data steams. Additional accounts and/or iSTB working directories are TBD.

## 

# 3. At a SDM Archive Site:

Resource: DTSD:

The SDM archive site (La Serena, Chile or Tucson, Arizona) can be either a DTS ‘transfer’ or ‘endpoint’ node in the DTS queue data path. Typically the only difference will be a site-dependent delivery application. This application will execute on the delivered file and is expected to interface with the E2E system to move the file to permanent mass-storage and possibly register with the NOAO searchable Archive.

*Syntax*:

The DTSD is configured, not called. Additional parameters may be added to the control file by the delivery application by creating a file called “*<queue>.par*” in the current DTS spool directory. The format of this file is “*<key> = <val>*“ pairs, one per line. The special parameter “deliveryName” may be used to specify the name of the file that is delivered to downstream sites.

*Semantics*:

Transport between two DTSD machines is directly to the DTS queue spool directory, delivery happens by copying the file from the spool area to the configuration-specified *delivery directory*. Once the the transport has been verified (i.e. MD5 checked) and the delivery copy made, a pre-configured delivery application is executed and upon successful completion, the transport control parameters are recalculated before queue processing resumes.

*Error Handling*:

* TBD

*Requirements:*

* DTS:
  + The DTSD binary and configuration file must be installed on the Tucson/La Serena cache.
    - The DTS will provide documentation of the configuration file to include directive definition and syntax.
  + A delivery command will call a pre-defined “*DciArchT*” to operate on the DTS target file.
  + The DTS target file cannot be moved or deleted until DTSD processing is complete.
  + Sufficient storage[[2]](#footnote-0) is available to manage the DTS queue areas.
  + Upon successful delivery to the ‘endpoint’ or ‘transfer’ node, the DTS target file is removed by the DTSD and the status log is updated.
  + Monitoring tools must exist to check the status of queue processing.
  + Logging:
    - DTSD status log and DTSD transfer log as described in the Operational DTS-E2E Interfaces.
* E2E:
  + The DTS Operator is responsible for monitoring the DTSD.
  + The DTS Operator must be able to enable debugging, start and/or stop the DTS if necessary.
  + The DTS Operator or local daemon is responsible for monitoring the success or failure of the delivery application.
  + The ingest/delivery application must not be limited to processing of FITS files only. Unrecognized file formats should be processed to the extent possible (e.g. a copy is made, logging recorded etc) without returning an error code to the DTS.
  + The E2E *ingest script* is responsible for ensuring the validity of files that are archived (e.g. FITS file processing produces a valid file, unrecognized file are unmodified, and error during processing leaves the input file unchanged if transport continues, etc).

Resource: idci-proc:

The DTSD has transferred a source file to a pre-configured *delivery directory*, and has verified the md5sum. The DTS then triggers iDCI registration using the ‘delivery application’ named *idci-proc*:

*Syntax:*

/noaosw/zdts/bin/idci-proc $D $MD5 $S

* Inputs follow the macro substitutions defined by the DTS Delivery Application Interface:
* $D – path to delivered file (delivered filename)
* $MD5 – MD5 checksum
* $S – object size (bytes)

*Semantics:*

The *idci-proc* will process the file as follows:

* Copy the DTS target file to a pre-defined iDCI incoming directory;
* Validate the md5sum and file size
* Add the file to the iDCI registration queue.

Once the file has been added to the iDCI registration queue, it will be responsibility of the Integrated Cache Initiative.

*Error Handling:*

*Idci-proc* will return to the calling program:

* 0 (zero) - file successfully processed;
* 1 (one) - some error occurred, possible error status:
  + 100 – some system error occurred, return system message;
  + 200 – some iDCI error occurred, return iDCI message;
  + 300 - some UNKNOWN error occurred.

*Requirements:*

* DTS:
  + If an
* iDCI:
  + Requires that the DciTrackD is running on pre-configured port.
  + The DciTrackD is configured to accept connections from a list of authorized clients.

# 4. At NCSA:

The delivery end point for the DES survey data.

Resource: DTSD:

*Syntax:*

*Semantics:*

*Error Handling:*

*Requirements:*

Resource: des-proc:

# Operational DTS-E2E Interfaces

The SDM Operations has existing utilities to monitor the E2E system. It requires two types of logs:

* *DTS status log* – logs all events using the syslog message format:

message = date time hostname source,num: severity message\_text

In this specification:

* *message* is the syslog message. The maximum length of message is 1024 bytes.
* *date* is the date the message was generated.
* *time* is the time the message was generated.
* *hostname* is the host name of the system that generated the message.
* *source* is the DTS method that generated the message.
* *num* is the instance number that generated the message.
* *severity* is the message severity as described in the DTS documentation (DEBUG, INFO, WARNING, ERROR, FATAL).
* *message\_text* is the message text, which depends on the type of message.

Example:

Aug 21 12:10:02 des-dhs dtsq,2735: INFO: Submitted /path/file to des queue (2735)

* *DTS transfer log* – SDM Operations has existing utilities to collect file transfer statistics. The format of this log is similar to the syslog format above, with the following difference:
* *message\_text* is the message text, which requires the following format:

yyyy-mm-dd hh:mm:ss UTC,/path/filename,md5sum,filesize,tot\_seconds,MB/s,Mb/s

# Delivery Application Interface

A “Delivery Application” can be executed by the DTSD at each node in the data path. This application expects at least one argument, the full path to the file being delivered, however additional parameters may be specified (see below). The command is specified in the DTS configuration file and is of the form:

/path/dtsapp $F [<params>…] # use original filename

/path/dtsapp $D [<params>…] # use delivered filename

Except for the initial “ingest” node, the delivery application executes on a copy of the file and is free to do whatever it likes with the file. The delivery application is responsible for deleting this file after processing if necessary. In the case of an ingest node, the file may be modified; the file may also be renamed provided that a parameter file be produced containing a ‘deliveryName’ parameter containing the new name of the file.

Return Values:

A zero status means the command executed successfully, a non-zero value indicates an error occurred. There is no way to determine exactly what the error is without specifying a dictionary of return codes.

Allowed Macro Substitutions:

$F path to delivered file (original filename)

$D path to delivered file (delivered filename)

$S object size (bytes)

$Q name of delivery queue

$E UT epoch at the time the object was queued

$SUM32 32-bit SYSV checksum

$CRC32 32-bit CRC checksum

$MD5 MD5 checksum

$FULL full path to originating file

$ON original filename (minus path)

$OP original path to filename

$DN delivery filename (minus path)

$DP delivery path

$SP path to filename on source host

$OH originating host name

$<param> replaced with value of named parameter (or ‘ ‘)

# Definitions

Data Store – Temporary or permanent physical data storage. Maybe spinning disks or tape.

DES 3-Day Store – the storage system at CTIO into which SISPI puts the final FITS image.

Delivery Application – The application that is executed by the DTS system at a site that does whatever processing is required to ‘*deliver’* a file.

Delivery Directory – The director outside the DTS working area into which a copy of a transported file is placed. The directory must be specified for each site that receives data.

DciTrackD – the iDCI message daemon.

DTS – Data Transport System as described in [3].

DTSQ - a DTS queueing agent on the DECam controlled computer at the CTIO Blanco telescope.

DTSD - a DTS daemon is the main service daemon running at each site.

Ingest Application – The application that is executed at the DTS submission site to process a file before movement within the DTS. This application typically modifies a file so it is suitable for archiving by the NOAO E2E system.

iDCI – Intergrated Cache Initiative, the SDM file repository management system.

Image Builder – The component of the SISPI system that assembles the FITS image and submits a *postproc* request for data tranport.

iSTB - a instance of the iSTB sub-system on the CTIO E2E Mountain cache.

NOAO E2E – The NOAO End-to-End system of software services for acquisition, pipeline, archive and data management and distribution.

Mountain cache – The central storage area at CTIO that collects data for archiving from each of the telescope domes.

MSS – Mass Storage System.

SDM – Science Data Management program at NOAO.

SISPI – The DECam data acquisition system.

# 

# References

[1] STR-2010-02: *DECam Community Pipeline E2D Interface Control Document* - <http://chive.tuc.noao.edu:8080/DPPDOCS/software-technical-reports/STR-2009-02_E2E_ICD_v1.02.pdf>

[2] iRODS: Integrated Rule-Oriented Data System developed by the by the DICE Center at UNC at Chapel Hill, NC (<http://www.irods.org>).

[3] DECam DES/SISPI - DTS Interface Control Document (ICD).

[4] SDM Program Plan 2009 – For the period FY2009-FY2010, Version 1.5 (06/19/09).

[5] Seaman, R. L. 2000, in ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, eds. N. Manset, C. Veillet, D. Crabtree (San Francisco: ASP), 133

[6] iDCI: Integrated Data Cache Initiative - High Level Requirements and Scope. <http://chive.tuc.noao.edu:8080/DPPDOCS/operations-documentation/software-system/application-components/noao-e2e/e2ev1.5/iDCI_project_definition.pdf>

# Appendix A: iSTB DTS requirements

Client-side (SISPI/Image builder):

1) SISPI calls (synchronously in the foreground):

/bits/bin/postproc <pathname to file> [perhaps other arguments]

Postproc is lightweight, quick, and guaranteed not to block.

2) Postproc calls dtslogger and captures reliable (if arbitrarily delayed from shutter close) timestamp and calendar date. Pathname and other metadata are passed downstream to local front-end queue (however implemented).

3) Front-end queue (a FIFO that does indeed block as each entry pathname is processed) runs in the background asynchronously from the data acquisition/image builder workflow(s). If there are multiple workflows and/or multiple SISPI/IB hosts we sort out what this means to sequencing later. Various activities may be initiated from the queue, but basically it calls a script.

4) The front-end script calls dtstracker (and might translate into FITS, compress, tee the data, etc, for other instruments). Dtstracker accepts the calendar date argument to tie it reliably to the log produced by dtslogger (since the front-end queue may run minutes, hours or days later than postproc). I suspect the multiple IB nodes will mount common partitions and the log files can be squirreled away in common.

5) The front-end script then calls dtsq with various metadata arguments (host, account, calendar date, timestamp, etc).

6) Dtsq does its business. If the front-end queue relies on dtsq, perhaps this is initiated earlier in some fashion. I believe dtsq is lightweight, meaning I don't believe it copies the input file to a local spool directory (unlike lpr). Since DECam has a local three-day store, this should not be a frequent issue, but will be with other instruments. Observers often move, rename, delete, or modify-in-place the original camera files, sometimes automatically as data are taken. Our goal is to capture the raw data, not a random observer's poorly conceived attempt at efficiency in bookkeeping.

7) If the SISPI/IB architecture permits the initial postproc capture to be heavier weight (even for community data sets), we can perhaps forgo the front end queue. In that case postproc will call both dtslogger and dtstracker, capture the context metadata (including calendar date and timestamp) and call dtsq directly and synchronously. This is likely plan A until we learn otherwise about the rotating pool of IB nodes.

a) SISPI/IB will still call postproc in the foreground (waiting for completion)

b) we will still have to handle the multiple IB instances

c) the issues in #6 will still apply - as it would even if dtsq is called directly - presumably this is mitigated (for DECam only) by the delivery receipt handshake with downstream server(s)

8) Dtsq will arrange for data to appear on the mountain cache in the appropriate DTS spool directory. It will trigger the ingest command (a script I gather?)

Server-side (mountain cache):

9) The DTS ingest script will call a modified version of the iSTB daemon. Since concurrency protection is not supplied by the environment (unlike lpd), I will need to implement some semaphore or lock file. The original tape-based STB bitmon program (similar to lpc) had a lock file to prevent simultaneous attempts to swap tapes, etc. ICE also has a lock file. Similar will probably suffice here.

10) The new iSTB [filter|command|task] will need to be modified to accept arguments as provided by DTS. We'll need to know how that works.

11) iSTB will read from the DTS spool directory, blocking that thread and all other ingest script instances on other threads. iSTB will write to the same hierarchy as currently. Too much Ops and archive functionality relies on this.

12) After iSTB successfully creates the new file (the first archival copy), it will run a script as currently to perform any utility chores (eg, compression for non-DECam instruments). This will likely be a no-op for DECam.

13) The final action of iSTB will be to delete the original input spool file and replace it by a symlink to the output file. This symlink will likely be renamed to the iSTB serial number standard we currently use, thus using the mechanism Mike was discussing.

14) iSTB will exit with the appropriate status. We'll need to work out the options. Mostly need something that says ok - proceed or not ok - discard the spool file. Perhaps also halt the queue, since this is the current paradigm, but perhaps DTS provides a more coherent way to manage the queue. The point is that iSTB will detect various error conditions (missing telescope schedule, etc) and will need to halt ingest processing while permitting the network transport to continue for subsequent files.

15) DTS will continue with whatever that thread does. The next blocked ingest script will awaken - likely this requires additional thought. If the queue drains, no busy waiting should result.

16) Data will be retired from the caches using some automatic script or manual procedure that Ops will have to sort out. Presumably DTS will manage the spool directory links using its own heuristics. Perhaps Ops can simply build on this and delete the file pointed at by the symlink before deleting the symlink. I suspect we'll need to play with it to sort out the appropriate behavior.

17) One issue we haven't discussed is account management. This may well differ from the DCI's group management scheme, but Ops procedures make some basic assumptions about the cache account owning data, for instance. I also always log into the instrument computers as bits, precisely so I don't have write permission on spool directories, log files, etc. We run other commands (eg, md5checker) as opsmon, and so forth. At any rate it should not be presumed that data ownership is a free parameter and we'll need to know what DTS assumes about this.

1. A staging area large enough to stage 3 days of DES source data (1TB min). [↑](#footnote-ref--1)
2. A staging area large enough to stage 3 days of DES source data (1TB min). [↑](#footnote-ref-0)