

DTS-E2E ICD

Version 0.1 (04/12/2011)

Authors: I.Barg, D.Scott, R.Seaman

Table of Contents

Purpose 3

Scope 3

Overview 3

The current NOAO E2EV16 system: 3

Incorporating the DTS within the NOAO E2EV16 system: 4

Proposed hybrid system for E2EV17: 5

ICD: Copy Remote DECam Data using DTS 6

Preconditions 6

Summary of data flow 7

Interface Documentation Template 7

Interfaces 8

DTSD 8

DTSQ 9

iSTB 9

DciArchT 11

DciTrackD 13

iclient 16

DsQueue 17

SDM Operations Requirements 18

References 19

Definitions 19

APPENDIX A: DciTrackD Configuration 21

# 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, developers and testers who are responsible for incorporating the DTS within the current E2E SYSTEM. Specifically, it addresses the interfaces between the DTS and existing E2E SYSTEM components internal to NOAO.

# 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) 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.

This document will not discuss the data content and format requirements which have been 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 (hearafter 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 relies on the BSD UNIX line printer daemon (lpd) to provide queued network data transfers from instruments in the various NOAO and affiliated telescopes and pipelines located in Tucson. iSTB adds some additional keywords to the FITS headers to insure 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 (filesize, md5sum and physical location) into a local database. The 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. The iDCI is a Perl wrapper around iRODS [2] icommands and is used for transport and file repository management in La Serena and Tucson, and off-site storage at NCSA. iSTB and iDCI together provide a fully automated data transport and file repository management system.

## Incorporating the DTS within the NOAO E2EV16 system:

The DTS as described in the DECam DES/SISPI ICD [3], was designed to handle interfaces to the E2E system and external projects. The DTS as described in [3] is primarily a delivery mechanism for DES components that receive and transport data from the DECam instrument. This ‘initial focus’ is described in the SDM Program Plan 2009 [4]:

“*The initial focus will be for use with the DECam instrument data transfers, both for the Dark Energy Survey (DES) and DECam Community data. The expected data volume and constraints on timely delivery require the implementation of a replacement for the current DTS system to make better use of the available bandwidth. Later phases of the project have the potential to replace the existing DTS used in other aspects of the SDM E2E system*.”

The ‘*current DTS*’ is referring to the transport portion of the current E2EV16 system. The DTS as described in [3] cannot easily be ‘dropped into’ the current E2EV16 system for the following reasons:

1. The DTS is a collection of tasks designed primarily to provide fast efficient data transport and monitoring. The transport method in E2EV16 is a simple iRODS [2] ‘iget’.
2. The DTS messaging system is based on XML-RPC. The E2EV16 message system based on the TCP server daemon.
3. With the DTS *queued* transport, data are placed in a named data queue that defines the route of the data through the DTS system, files are delivered to each of the sites in the named queue that are "*downstream*" of the DTS submission site only (i.e. the data flows in one direction). The E2EV16 uses two queues to accomplish the “downstream” delivery of data:
   1. Data are queued for transport from the instrument to the iSTB cache using LPR;
   2. Data are queued for transport from the iSTB cache the nearest NOAO data center cache using a perl wrapper around iRODS (*iget*).
4. The DTS queued transport allows multi-site delivery in one queue. The E2EV16 uses a Point2Point (P2P) messaging system, each message has only one consumer.
5. The DTS definition of PUSH and PULL is different than E2EV16. In E2EV16 all data are PULLED from the source host to a destination host.
6. The iDCI is a collection of Perl wrapper scripts around IRODS icommands. The current E2EV16 does most of what the DTS proposes and more:
   1. manages the physical resources at each E2EV16 cache;
   2. provides a virtual file system for API to interface to the physical data store;
   3. provides parallel and threaded data transport for efficiency;
   4. bundles small files into larger tar files to optimize transfer;
   5. insures data integrity by monitoring data flow and reports missing or corrupted files downstream from the original source.
   6. post file transfer rates to a central web site for Operations monitoring.

To fully incorporate the DTS into the current E2E as deployed in E2EV16 would take a major re-write of the iDCI or perhaps a new design altogether.

# Proposed hybrid system for E2EV17:

The DTS is scheduled to be used by the E2EV17 system for delivery of DECam data, both DES and Community source (raw) data. Source data from other NOAO instruments will continue to use the existing mechanism found in the current E2EV16 system. A diagram of this hybrid system is shown in Figure 1 DTS-E2E Data Flow

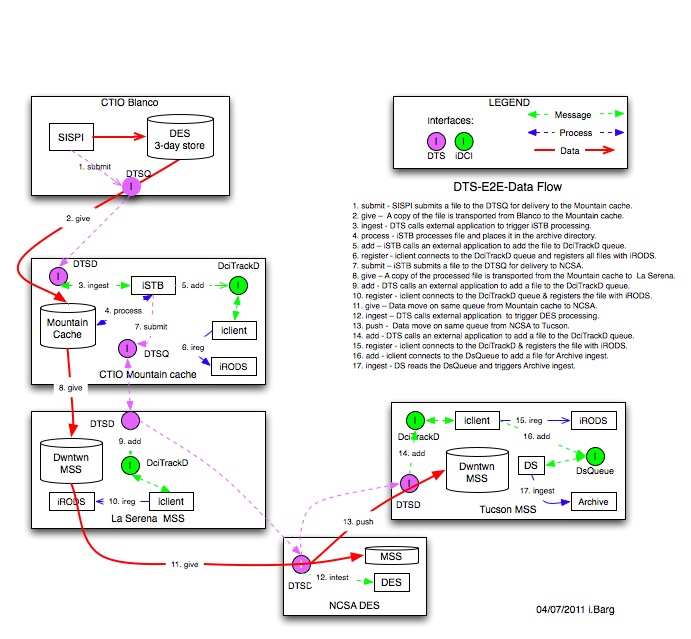


Figure DTS-E2E Data Flow

# ICD: Copy Remote DECam Data using DTS

## Preconditions

* DECam source data have been captured by SISPI.
* Source data shall be in FITS tile-compressed format.
* The DTSQ is installed and configured on the DES DHS 3-Day store.
* The NOAO Mountain Cache has sufficient disk space to store and process 3 nights of DES data.

## Summary of data flow

1. submit - SISPI submits a file to the DTSQ for delivery to the Mountain cache.

2. give – A copy of the file is transported from Blanco to the Mountain cache.

3. ingest - DTS calls external application to trigger iSTB processing.

4. process - iSTB processes file and places it in the archive directory.

5. add – iSTB calls an external application to add the file to DciTrackD queue.

6. register - iclient connects to the DciTrackD queue and registers all files with iRODS.

7. submit – iSTB submits a file to the DTSQ for delivery to NCSA.

8. give – A copy of the processed file is transported from the Mountain cache to La Serena.

9. add - DTS calls an external application to add a file to the DciTrackD queue.

10. register - iclient connects to the DciTrackD queue & registers the file with iRODS.

11. give – Data move on same queue from Mountain cache to NCSA.

12. ingest – DTS calls external application to trigger DES processing.

13. push - Data move on same queue from NCSA to Tucson.

14. add - DTS calls an external application to add a file to the DciTrackD queue.

15. register - iclient connects to the DciTrackD & registers the file with iRODS.

16. add - iclient connects to the DsQueue to add a file for Archive ingest.

17. ingest - DS reads the DsQueue and triggers Archive ingest.

# 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.
3. Data Types and Constants – Define any new data types or constants used by your resource.
4. Error Handling – Describe any error-handling behavior that is *common* across resources here. Error handling specific to a resource should described within the resource section (2.a above).
5. Variability – Describe *configuration parameters* and how they affect the semantics of the interaction of the interface.
6. Quality-Attribute Characteristics – Document what quality attribute characteristics, such as performance or reliability, the interface makes known to its users.
7. Rationale and Design Issues – Record the reasons behind the design of an element’s interface.
8. Usage Guide – Some clear examples of the most common ways the interface might be used.

This template is just a guideline. You may wish to modify sections, or leave some sections blank or TBD, but you must provide enough information for a user to determine how to use your resource.

# 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 La Serena, NCSA DES and terminating at Tucson.

The boundary objects are resource elements from three different sub-systems:

1. Data Transport System (DTS)
2. Save-the-Bits (iSTB)
3. Integrative Data Cache Initiative (iDCI)

This document will describe only the boundary objects illustrated in Figure 1 DTS-E2E Data Flow**Error! Reference source not found.**

## DTSD

Interface Identity:

The DTS Daemon is the main service daemon running at each site [3].

Resources:

* Syntax
* Semantics
* Error Handling

Data Types and Constants:

Error Handling:

Variability:

Quality-Attribute Characteristics:

Rationale and Design Issues:

Usage Guide:

Requirements:

## DTSQ

Interface Identity:

The DTS Queuing agent that submits data for ingestion into the DTS system [3].

Resources:

* Syntax
* Semantics
* Error Handling

Data Types and Constants:

Error Handling:

Variability:

Quality-Attribute Characteristics:

Rationale and Design Issues:

Usage Guide:

Requirements:

## iSTB

Interface Identity:

The iSTB client interfaces with the several types of cameras operating on NOAO and partner telescopes and with SDM-operated pipelines. Supporting the wide range of cameras and pipelines requires flexibility in configuration. The best way to achieve this is to simplify the boundary interface as much as possible. The minimum required information is the full pathname to each newly created data file, synchronously in turn.

Resource

*Syntax:*

The DTS has transferred source file to a pre-defined *Delivery Directory*, checks the md5sum then triggers iSTB ingest using the following command:

/path/bits/bin/istbcmd fullpath md5sum

*Semantics:*

Inputs:

* fullpath – the full unix pathname to new FITS file
* md5sum – MD5 (128-bit) checksum

Outputs:

* first archival copy written to proper directory on cache host computer
* md5 checksum of the archival file
* filesize (in bytes) of archival file
* trigger to execute next command(s) downstream
  + add the file to DciTrackD queue using ‘DciArchT’ (see iDCI section)
  + submits a file to the DTSQ for delivery to NCSA (see DTSQ section)

*Error Handling:*

* 0 (zero) - file successfully received (file exists and md5sum checks?)
* 1 (one) - some error with file causes rejection

Data Types and Constants

Error Handling

Variability

Quality-Attribute Characteristics

Rationale and Design Issues

Usage Guide

Requirements

* On 'delivery', the DTS must write to a pre-defined *incoming* directory retaining the original file path hierarchy (relative to the incoming root). Example:
  + <site A src > ct4m:/cp/20111201/image001.fits
  + <site B dest> dtsct:/decam/incoming/cp/20111201/image001.fits
* After 'delivery' the DTS must verify the 'md5sum' and local disk.
* iSTB only supports FITS files
* input file must be readable by "cache" account
* all intervening directories in the pathname must be readable by cache
* postproc command (script) must execute synchronously with data acquisition or pipeline
  + i.e., postproc must execute in the foreground

- actual network queuing occurs in a background queue layered on LPRng, so

- LPRng is required on data acquisition or pipeline client host

- cache account write access to spool directory required on client host

- cache account write access to /bits/track log directory required

- root (super) access to client host is required for SDM operations staff

- "bits" and "cache" accounts required on clients

## DciArchT

Interface Identity:

The command line interface to the DciTrackD is called the list manager client interface DciArchT.

Resources:

The DciArchT is the list manager client command line interface. It is the ‘external application’ that iSTB and DTS will use to interface with DciTrackD message queues.

*Parameters:*

The *DciArchT* accepts the following parameters:

-a 'path\_to\_file md5sum filesize' - add 'file' to the list of files to be processed, include the file md5 and filesize.

-p <port> - the DciTrackD port to connect to [DEFAULT=lowest configured port].

-e - shutdown the DciTrackD server.

*Preconditions:*

* Requires that the DciTrackD is running on pre-configured port.
* The DciTrackD is configured to accept connections from a list of clients authorized to connect.

*Postconditions:*

* The DciArchT returns immediately, so not to block the calling program. Any returned values from the DciTrackD are logged in:

/tmp/yyyymmdd\_DciArchT.log

* If DciArchT cannot connect to the DciTrackD daemon, it re-tries every 10 seconds until a successful connection is made.

Data Types and Constants:

* string '*track':* a space separated quoted string containing 'path\_to\_file md5sum filesize'.
* string *path\_to\_file*: name of file to process including pathname. The pathname may be a absolute physical path or a logical iRODS path:
  + To queue a local file to be transferred or registered with IRODS, you must use absolute path:

/<dataroot>/<source>/<yyyymmdd>/<tele>/<propid>/unique.fits.fz

* + To queue a file for Archive ingest you must use iRODS logical path:

/<irods-zone>/>/<source>/<yyyymmdd>/<tele>/<propid>/unique.fits.fz

* string *md5sum* - MD5 (128-bit) checksum
* integer *filesize* - in bytes

Error Handling:

* 0 (zero) - file successfully received (file exists and md5sum checks?)
* 1 (one) - some error with file causes rejection

Variability:

The DciArchT can be called by external clients like iSTB or DTS.

The DciArchT can be used from the command line to shutdown the DciTrackD server, with the ‘-p port’ and ‘-e’ parameter alone.

Quality-Attribute Characteristics:

The DciArchT forks a child process and returns immediately as to not block the calling client. The disadvantage to this approach is if the DciTrackD server is not running and listening on the configured port, this will leave orphaned DciArchT processes running.

Rationale and Design Issues:

The DciArchT was created primarily for iSTB, which did not want to call the DciTrackD directly because it did not want to ‘wait’ for the DciTrackD to connect.

Usage Guide:

Usage: DciArchT [-D] [-a trackname] [-e] [-p port]

Where: -a <track> - to add a track to queue

-e to shutdown queue (admin user)

-p use port number other than default

-D don't fork child process.

Either 'a' or 'e' must be specified.

The DciArchT can be invoked from the command line. The following example adds a track to the DciTrackD on the default port:

$DCIDATA/bin/DciArchT -a '/noaocache/mtn/20110227/ct13m/smarts/ct2981004.fits.fz 1ee028f13eb91b10e8f154c01562240c 1247040'

The following example uses the DciArchT to shutdown the DciTrackD daemon on same port 2135:

$DCIDATA/bin/DciArchT –e -p 2135

## DciTrackD

Interface Identity:

The DciTrackD is a file list manager. It is a Perl implementation of the TCP server daemon. It manages a list of tracks (filename with metadata) that need to be processed (transferred, registered or ingest) by the iDCI.

Resources:

The DciTrackD daemon currently runs as a background job on a iDCI cache. It is typically invoked from a crontab entry looks like this:

0,10,20,30,40,50 \* \* \* \* $DCIDATA/bin/DciTrackD –c \

$DCIDATA/conf/clients –f \ $DCIDATA/var/locks/DciTrackD-2335.queue –p 2335 \

-v 1 >>$DCIDATA/var/logs/DciTrackD-2335.log 2>&1

*Parameters:*

The DciTrackD daemon listens to a non-privileged port number, and responds to one of three requests:

* add [track] - add new track to the queue of tracks which needs to be transferred
* remove [track] - remove a track from the queue. Usually invoked by a iDCI client (iclient.pl, iBundle, iUnBundle) with permissions to ‘remove’ a track from the queue.
* tracks - send a list of files in the queue to the client. Usually invoked by an iDCI client when it wants to get a list of tracks which need to be processed (transferred, registered with the ICAT or ingested by the Archive).
* exit - shutdown the server.

In addition, the DciTrackD will accept the following command line options:

* -c [file] - Use 'file' containing IP addresses and /or hostname (one per line) which are a list of clients authorized to connect. Requests from clients which are not authorized are ignored. Defaults to allowing connections from localhost. The format of the file looks like this:

localhost 2135=aefrt,2335=aefrt,2435=aefrt

dtsct.ctio.noao.edu 2135=aefrt,2335=aefrt,2435=aefrt

dtsct 2135=aefrt,2335=aefrt,2435=aefrt

dsas3.ctio.noao.edu 2135=t,2335=t,2435=rt

dsas3 2135=t,2335=t,2435=rt

Where the hostname is followed by a list of authorized actions:

a - add track

e - shutdown the queue daemon

r - remove track

t - send a list of tracks in queue

* -f [file] - Use 'file' for writing the updated queue. Defaults to queue.DciTrackD in CWD.
* -p [port] - Listen on port number 'port'. Defaults to 1335.
* -v [level] - The verbosity level. Default=0, means no messages, while higher means more.

*Preconditions:*

* The clients file must exist, and must be configured authorized hostnames.
* The port must be a non-privileged port and accessible by remote clients.
* The logfile directory must exist and the DciTrackD daemon user must have write permission.

*Postconditions:*

* On successful startup, the DciTrackD writes the authorized clients to the DciTrackD log.
* If the server is already running, an attempt to start a new daemon fails since only one application can listen to a given port at a time (see, e.g., The Perl Cookbook, Section 17.2 Writing a TCP Server).

Data Types and Constants:

* string *track:* is a space separated quoted string containing 'filename md5sum filesize'.
* string *filename*: name of file to process including pathname. The pathname may be a absolute physical path or a logical iRODS path:
  + To queue a local file to be transferred or registered with IRODS, you must use absolute path:

/noaocache/mtn/20100215/soar/2010A-0318/cp353484.fits.fz

* + To queue a file for Archive ingest you must use iRODS logical path:

/noao-tuc-z1/mtn/20100215/soar/2010A-0318/cp353484.fits.fz

* string *md5sum* - MD5 (128-bit) checksum
* integer *filesize* - in bytes

Error Handling:

* Possible error codes:
  + 100 - add NOT permitted
  + 101 - add nonexistant track
  + 102 - add empty track
  + 120 - tracks/remove NOT permitted
  + 121 - remove nonexistant track
  + 130 - exit NOT permitted
  + 200 - connect NOT permitted

Variability:

The DciTrackD can be called directly from a Perl program, or from command line. This document will assume that all external calls to the DciTrackD will use the command line interface. Therefore, the Perl API resource will not be described.

Quality-Attribute Characteristics:

The DciTrackD is not thread enabled, and can only accept on connection at a time.

Rationale and Design Issues:

To preserve the track queue between server stops and restarts, the DciTrackD server dumps the contents of the queue to a file every time a client successfully updates it. When the DciTrackD server is restarted, it reads this DciTrackD queue file during initialization. The file can be specified with the '-f' line option as shown above.

Usage Guide:

Examples of calling the DciTrackD via the Perl API are not covered in this document. Example of calling the DciTrackD via a command line interface is described in the resource DciArchT.

## iclient

Interface Identity:

The *iclient* is a iDCI client that interfaces with IRODS and the Archive. The *iclient* is required at each iDCI cache to register *metadata* about each files that is stored on the local cache. The *iclient* is the interfaces that triggers ingest into the Archive.

Resources:

* Syntax
* Semantics
* Error Handling

Data Types and Constants:

Error Handling:

Variability:

Quality-Attribute Characteristics:

Rationale and Design Issues:

Usage Guide:

Usage: iclient.pl -a action -p port [qsvDI]

Where: action = 'iget' || 'ireg'.

port = port to listen to.

Requirements:

## DsQueue

Interface Identity:

The DsQueue is Data Service message queue provided by the Archive. The DsQueue, a Perl script that immediately returns to the caller. Internally, the script forks a child process which contacts the Data Entry Service to begin Archive ingest.

Resources:

*Syntax*:

DsQueue -a "<file://$DFNAME> $MD5VAL $FILSIZ <newline> <file://$HDRNAME> $HDRMD5 $HDRSIZ"

*Semantics:*

-a *string – add a file to be ingested*

Inputs:

* A double-quoted string containing:
  + the physical path of the file,
  + the MD4VAL (md5sum)
  + the FILSIZ (filesize in bytes)
  + optionally followed by a <newline> plus the physical path of the files ASCII FITS header.
  + The double-quoted string contains

*Error Handling:*

Data Types and Constants:

Error Handling:

Variability:

Quality-Attribute Characteristics:

Rationale and Design Issues:

Usage Guide:

Requirements:

# SDM Operations Requirements

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

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

# 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

# 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.

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.

DciTrackD - iDCI message queue.

DsQueue - Data Service message queue.

iclient - iDCI client that interfaces with IRODS and Archive.

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.

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

SISPI – The DECam data acquisition system.

# APPENDIX A: DciTrackD Configuration

TBD: (I.Barg)