



# WISE Science Data System Functional Design



## Contents

- 1 Signatures
- 2 Revision history
- 3 Document number
- 4 Scope
- 5 High-level Subsystem Design
  - 5.1 Derived High-level Design Drivers
  - 5.2 Major Subsystems
  - 5.3 WSDS Input
  - 5.4 WSDS Output
- 6 Subsystem Descriptions
  - 6.1 Executive Functions (EXEC)
    - 6.1.1 Execution Wrappers (WRAP)
    - 6.1.2 Job Administration (JOBS)
    - 6.1.3 Ingest Pipeline Executive (INGESTX)
    - 6.1.4 Multi-frame Pipeline Executive (MFRAMEX)
    - 6.1.5 Scan Pipeline Executive (SCANX)
    - 6.1.6 Single Frame Pipeline Executive (SFRAMEX)
  - 6.2 Ingest Functions (INGEST)
    - 6.2.1 Delivery Handling (DELIV)
    - 6.2.2 Decommuation and Decompression (DECOM)
    - 6.2.3 Meta-data Correlation (INMETA)
  - 6.3 Pipeline Functions (PIPELINE)
    - 6.3.1 Instrumental Calibration (ICAL)
    - 6.3.2 Photometric Calibration (PCAL)
    - 6.3.3 Source Detection and Extraction (DETEX)
    - 6.3.4 Band Merging (MERGE)
    - 6.3.5 Pointing Reconstruction (PREX)
    - 6.3.6 Solar System Object Matching (SSOFIND)
    - 6.3.7 Resampling and Coaddition (COADD)
  - 6.4 Quality Analysis Functions (QA)
  - 6.5 Final Product Generation Functions (FPG)
  - 6.6 Archive Functions (ARCHIVE)
- 7 Reference Hardware Design
- 8 Throughput Requirements
  - 8.1 Assumptions
  - 8.2 Computations
  - 8.3 Discussion
- 9 Data Layout
  - 9.1 Introduction
  - 9.2 The Use of Symlinks
    - 9.2.1 Known Symlinks
  - 9.3 File Naming
    - 9.3.1 File Name Format
      - 9.3.1.1 Known Bases
      - 9.3.1.2 Known Types
      - 9.3.1.3 Known Formats
      - 9.3.1.4 Known Varients
    - 9.3.2 Some Examples
  - 9.4 Directory Structure
    - 9.4.1 The Data Root
    - 9.4.2 Reference Data
    - 9.4.3 Calibration Data
    - 9.4.4 Pipeline Input/Output Directories
      - 9.4.4.1 Scan and Frame IDs in the Directory Structure
      - 9.4.4.2 Frame Output Directory
    - 9.4.5 Ingest Directories
    - 9.4.6 Archives
- 10 WSDC Operations Concept
- 11 Glossary

# 1 Signatures

---

Roc Cutri WSDC Manager

---

Tim Conrow WSDC Architect

## 2 Revision history

2007-11-12: Draft 1  
2007-12-05: Draft 2  
2008-01-18: Version 1

## 3 Document number

WSDC D-D001

## 4 Scope

This document describes the functional design of the WISE Science Data System (WSDS). The functional design describes in coarse detail how requirements in the Functional Requirements Document ([http://web.ipac.caltech.edu/staff/roc/wise/docs/WSDC\\_Functional\\_Requirements\\_all.pdf](http://web.ipac.caltech.edu/staff/roc/wise/docs/WSDC_Functional_Requirements_all.pdf)) (FRD, document #WSDC D-R001) will be implemented. For each subsystem a *Subsystem Design Specification* (SDS) provides greater implementation detail.

## 5 High-level Subsystem Design

The WISE WSDC functionality is organized in two levels: **subsystems**, and **implementation modules**.

Subsystems serve to group related functions together for purposes of assigning resources and aiding communication. The detailed descriptions below give each subsystem's purpose.

Implementation modules group smaller, closely related sets of functions or capabilities and reflect implementation details. Several implementation modules combine to form a subsystem. The detail sections below describe the purpose, requirements, and products of each implementation module.

### 5.1 Derived High-level Design Drivers

Most of the FRD ([http://web.ipac.caltech.edu/staff/roc/wise/docs/WSDC\\_Functional\\_Requirements\\_all.pdf](http://web.ipac.caltech.edu/staff/roc/wise/docs/WSDC_Functional_Requirements_all.pdf)) is implemented by the modules making up the major subsystems. The design linking these sub-systems and modules together is driven by the following derived high-level motivators:

#### High Data Throughput

Raw, decompressed telemetry is 50GB/day. Data flow including intermediate products exceeds 500GB/day for the frame pipeline. Since reruns and coadds occur in parallel, total throughput can exceed 1 TB/day.

#### Low Latency

Quicklook turnaround is 24 hours from receipt of the raw telemetry, scan pipeline processing turnaround is 72

hours.

**High Availability**

In addition to low latency, the WSDS will provide services which must be available most of the time. The most important high availability service is receiving raw telemetry.

**Modest Hardware Budget**

WSDS budgeted with commodity, low cost hardware.

**Rigid Mission Schedule**

Ground characterization, mission scenario testing, launch, etc. wait for no one. The data release schedule is also ambitious and rigid.

**Heterogeneous Application Development**

WSDS developers have various languages and development modes which they find most productive.

These drivers lead to the following design decisions:

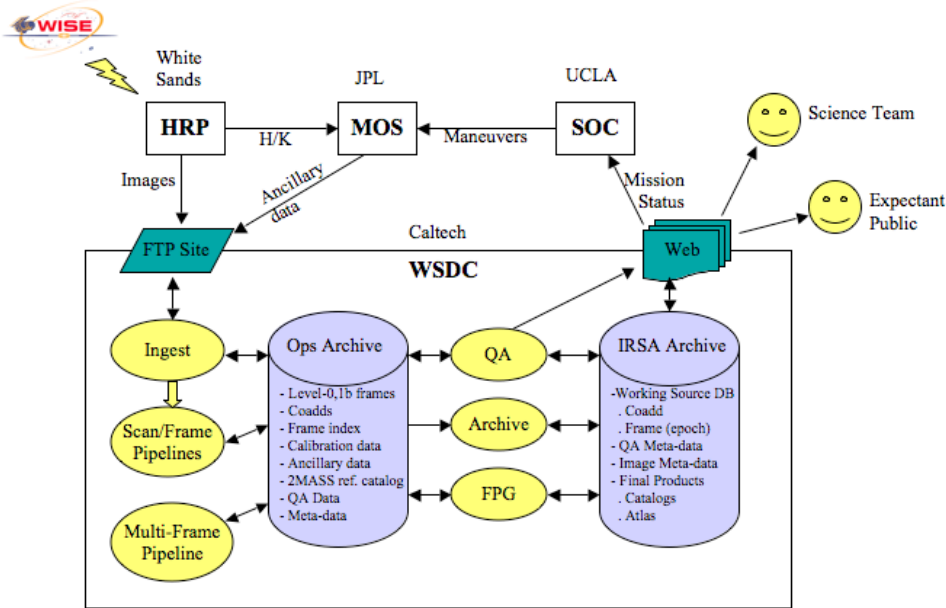
- A compute cluster based on interchangeable, inexpensive nodes linked to high-quality archival disk by a modestly scalable gigabit network. See Hardware Design and Throughput Requirements.
- Executive wrappers which provide a common external interface and services for underlying applications. See Execution Wrappers.
- Pipeline structure allowing high concurrency with frameset-level granularity. See Job Administration.
- Minimal reliance on external services, such as client/server-based RDBMS systems, for operational pipeline runs.
- Emphasize use of node-local storage to reduce network loading.
- Personnel dedicated to operations activities (as opposed to development and analysis). See Operations Concept.
- A robust disaster recovery plan (XXX TBD) including hardware architecture for backup/restore and failover. See Hardware Design.

**5.2 Major Subsystems**

The major subsystems (described in the next chapter) are:

- Executive Functions
- Ingest Functions
- Pipeline Functions
- Quality Analysis Functions
- Final Product Generation Functions
- Archive Functions

Here is a diagram relating subsystems to each other and to key external interfaces.



**5.3 WSDS Input**

The major required input data for the WSDC is

- Raw images from the four detectors as compressed pixel data in CCSDS source packets and a manifest describing the files in each delivery (c.f. MOS GDS ICD ([http://web.ipac.caltech.edu/staff/roc/wise/reviews/cdr/presentations/WISE\\_MOS\\_GDS\\_ICD\\_08\\_14\\_07.pdf](http://web.ipac.caltech.edu/staff/roc/wise/reviews/cdr/presentations/WISE_MOS_GDS_ICD_08_14_07.pdf))). Provided by the HRP at White Sands.
- Housekeeping telemetry as comma-separated value files. Telemetry values in world (engineering) units (c.f. MOS GDS ICD ([http://web.ipac.caltech.edu/staff/roc/wise/reviews/cdr/presentations/WISE\\_MOS\\_GDS\\_ICD\\_08\\_14\\_07.pdf](http://web.ipac.caltech.edu/staff/roc/wise/reviews/cdr/presentations/WISE_MOS_GDS_ICD_08_14_07.pdf))). Provided by MOS at JPL.
- Ancillary orbit event meta-data data as a Predicted Event File (PEF) (c.f. MOS GDS ICD ([http://web.ipac.caltech.edu/staff/roc/wise/reviews/cdr/presentations/WISE\\_MOS\\_GDS\\_ICD\\_08\\_14\\_07.pdf](http://web.ipac.caltech.edu/staff/roc/wise/reviews/cdr/presentations/WISE_MOS_GDS_ICD_08_14_07.pdf))). Provided by MOS at JPL.
- Ancillary S/C attitude, position and time information as NAIF (<http://naif.jpl.nasa.gov/naif/toolkit.html>) C-kernels, SP-kernels, and Clock-kernels (c.f. MOS GDS ICD ([http://web.ipac.caltech.edu/staff/roc/wise/reviews/cdr/presentations/WISE\\_MOS\\_GDS\\_ICD\\_08\\_14\\_07.pdf](http://web.ipac.caltech.edu/staff/roc/wise/reviews/cdr/presentations/WISE_MOS_GDS_ICD_08_14_07.pdf))). Provided by MOS at JPL.
- The mission plan as expected scan directions and frame counts. Provided by the SOC at UCLA.
- Ground and on-orbit calibration values, tables, and images for flats, darks, sky offsets, pixel non-linearity, spatial distortion, etc. Provided by JPL and SDL.

## 5.4 WSDS Output

The ultimate output of the WSDS is the released data products: the WISE Source Catalog, Image Atlas, and a list of solar system objects matching WISE level-1b source extractions. The WSDC will also be the archive for the level-0 and level-1b frames. In addition many intermediate products will be generated: the frame and coadd working source databases, transient intermediate images, meta-data, instrument telemetry, QA data and evaluations, etc.

An Explanatory Supplement will be distributed. The Supplement will describe each released data product.

# 6 Subsystem Descriptions

## 6.1 Executive Functions (EXEC)

### Purpose

Provide high-level services for controlling and coordinating processing jobs and the information they need as input and produced as output.

### 6.1.1 Execution Wrappers (WRAP)

#### Purpose

Command line parameter handling, text output control, file transformations.

#### Requires

Parameter and I/O file specifications

#### Provides

Uniform interface for job execution

#### Description

Command execution wrappers are small executables which themselves call the application which performs the desired function. Fundamentally, the job of the wrapper is to move high-level administrative tasks out of the underlying application so it can be more focussed and easier to write. The wrapper provides a uniform interface to the outside world through use of a single parameter passing API and a common way of handling textual output from the underlying application. Wrappers are also responsible for checking for successful completion of the executed application. Wrappers may well provide other services, such as: moving, renaming, massaging or otherwise preparing input or output files; checking for completeness and integrity of prerequisites; setting parameter defaults by examination of context; employing job administration functions to execute many instances of the application concurrently; other services necessary to prepare for execution of the application and cleaning up after it.

### 6.1.2 Job Administration (JOBS)

#### Purpose

Control submission, monitoring, termination, and disposition of processing jobs.

#### Requires

Job specifications

#### Provides

Interface for job control

#### Description

The jobs administration functions provide convenient access to users and programmers to the non-interactive processing jobs which implement the WSDS system. Job administration functions include:

- Controlling the submission of jobs, including control of the concurrent execution of many related jobs
- Deleting (terminating) running jobs, possibly in large numbers at one go
- Monitoring their progress and key output

- Providing information about the them (job ID, name, command line, output location, status, errors or warnings, resources used, etc.)
- Determining their final disposition (success or failure)
- Locate output data for a particular frame or coadd after the job has completed

These capabilities will be provided through APIs, command line and GUI interfaces. The ability to perform these functions on jobs scattered across many machines is crucial.

### 6.1.3 Ingest Pipeline Executive (INGESTX)

#### Purpose

Organize INGEST functional elements into a single executable to produce level-0 frames and associated meta-data, and to archive the raw telemetry.

#### Requires

HRDP telemetry, telemetry manifest, ancillary data, parameters

#### Provides

Level-0 frames as FITS images and associated meta-data in FITS headers

#### Description

The ingest executive manages the following ingest functions for each delivery from MOS or the HRDP:

- Detection of incoming telemetry from the HRDP and the completion of the transfer
- Detection and reading of the telemetry manifest associated with the telemetry transfer
- Deconvolution and decompression (if necessary) of science frames from the telemetry and reading and storing of related meta-data from the source packet headers (e.g. the VTC)
- Updating the frame index and generation of a delivery report
- Determination from the frame index of which frames are needed and archiving those frames
- Receiving, indexing and archiving ancillary data delivered from MOS
- Correlation of the ancillary data to the level-0 frames and adding to the frame meta-data from the ancillary data (e.g. ADCS-generated frame position and orientation, exposure start/stop times, temperatures, etc.)
- Starting the quicklook pipeline (a variant of the scan pipeline) on a selection of frames from the delivery
- Determining when a complete set of frames for a scan are available and starting the scan pipeline on these frames

### 6.1.4 Multi-frame Pipeline Executive (MFRAMEX)

#### Purpose

Organize PIPELINE functional elements into a single executable to produce coadds and associated catalogs and meta-data.

#### Requires

Output from FRAMEPIPE, coadd specification, parameters

#### Provides

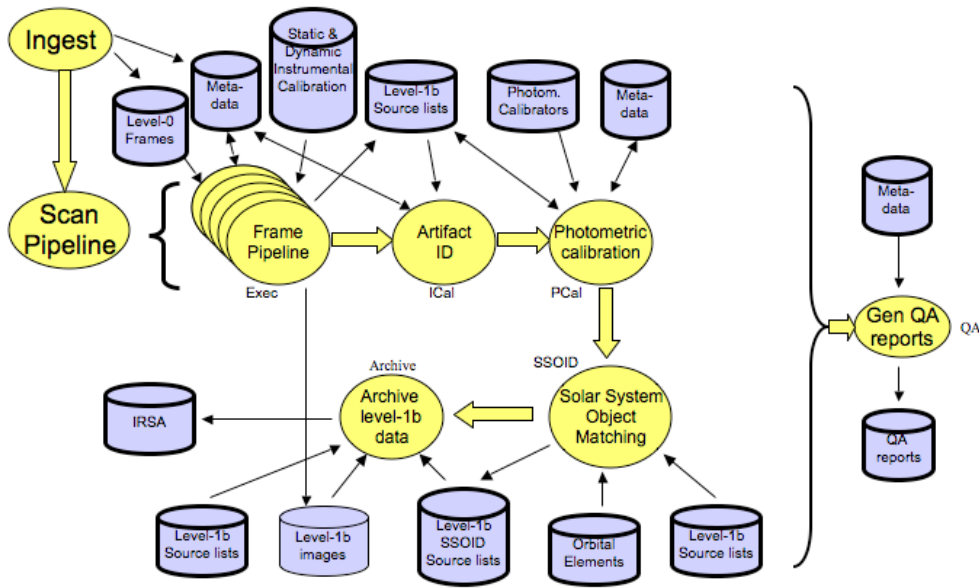
A single coadd and associated catalogs, meta-data, and QA output.

#### Description

The coadd pipeline executive manages the following pipeline functions:

- Locate level-0 frames covering a given coadd geometric specification
- Read or regenerate level-1b frame data
- Identify outliers using temporal stacking in interpolated space
- Match backgrounds of level-1b frames
- Project, interpolate, and resample level-1b pixels into coadd image
- Detect and characterize sources and archive extractions to the coadd working database
- Write and archive QA data and meta-data

### 6.1.5 Scan Pipeline Executive (SCANX)



**Purpose**

Execute the frame pipelines on all available frames for one scan in a highly concurrent manner, performing any necessary dynamic calibrations; apply relative photometric calibration to frame source lists and image headers; perform any multi-frame detector artifact identification required.

**Requires**

Frame list, frame index, parameters

**Provides**

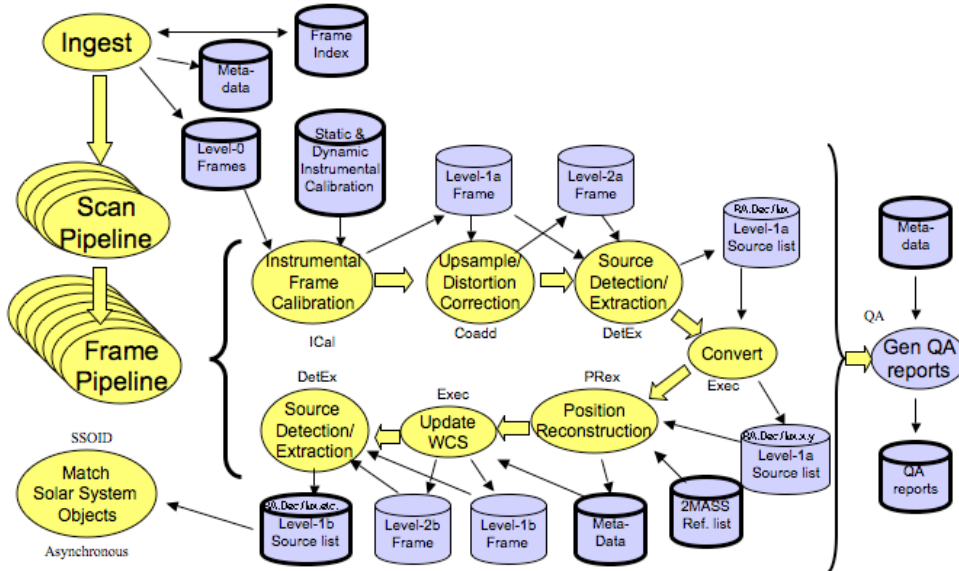
Frame pipeline output

**Description**

The scan pipeline is a wrapper around the frame pipeline which manages the concurrent execution of a given set of frames, usually the frames comprising a complete scan. Internally executed frame pipelines are monitored for success or failure and this and other information is made available to operators or other users through JOBS functions. Dynamic calibrations may be performed by preliminary runs of partial frame pipelines.

**6.1.6 Single Frame Pipeline Executive (SFRAMEX)**

Here is a diagram showing the major subsystems/modules and data interfaces involved in the frame pipeline.

**Purpose**

Organize pipeline functional elements into a single executable which produces level-1 pixel data, and associated catalogs and meta-data

**Requires**

Raw frame and ancillary housekeeping data, calibration, frame manifest, parameter files

**Provides**

Meta-data for reproducing level-1 pixel data, level-1 frames, source catalogs, QA output

**Description**

The frame pipeline executive manages the pipeline functions for a level-0 frame:

- Determine available resources, especially disk space, and remove level-1 frames as needed (oldest first) to make room for processing
- Perform relative instrumental calibration of the frame to level-1a
- Detect and characterize (extract) sources for position reconstruction
- Perform position reconstruction, i.e. refining the frame's position and orientation, and determine any scale corrections and record this information as meta-data
- Build level-1b frames by adding refined astrometric information to FITS headers
- Build a level-2 frame with distortion-corrected pixels
- Characterize from the level-1b frame sources detected on the level-2b frames
- Archive key frame meta-data
- Asynchronously match frame (epoch) working database sources to asteroids

Frame meta-data, source catalogs, level-1 frames, etc. will be kept in a directory which is unique to each processing attempt on each frame. Data may later be archived in IRSA-accessible tables by the Archive sub-system. Intermediate images will eventually be deleted.

**6.2 Ingest Functions (INGEST)****6.2.1 Delivery Handling (DELIV)****Purpose**

Detect incoming HRDP data and frame manifest, ancillary data delivery, detect delivery completion

**Requires**

Incoming data directory location, frame manifest

**Provides**

Telemetry or other transferred in a staging area

**Description**

One or more servers will be dedicated to ingest of HRDP image data and MOS-generated ancillary data. These servers will have read/write access to a receiving area where external institutions (White Sands, JPL, UCLA, others?) will place data according to TBD protocol. DELIV will detect transfer completion and move the data to remote staging areas where further processing will commence. DELIV may initiate the downstream

processing itself or it may begin asynchronously.

### 6.2.2 Decommutation and Decompression (DECOM)

#### Purpose

Read pixel data from source packets, write level-0 frames as standard FITS files with meta-data from packet headers, especially the vehicle time code (VTC).

#### Requires

Raw source packets from HRDP

#### Provides

Level-0 frames with meta-data, frame completeness report, updated frame index

#### Description

DECOM reads raw HRDP output and constructs raw images (compressed or uncompressed) from the source packets and gather meta-data from the packet headers (e.g. the VTC). If necessary the raw images are decompressed. The images are stored as BITPIX=32 FITS files with meta-data in the headers. Band 4 images are upsampled by a factor of 2 through simple pixel replication. The resulting frames constitute the archival level-0 frame data. The level-0 FITS files may be stored compressed. In constructing the level-0 frames, incomplete frames or frames that could not be decompressed are not saved. A frame index database is updated recording the frame's VTC, and delivery date and time, as well as the name and location of the resulting FITS output file. A frame completeness report (format contents TBD) will be generated for each delivery.

### 6.2.3 Meta-data Correlation (INMETA)

#### Purpose

Correlate level-0 frames to ancillary housekeeping data (SPICE kernel, etc.)

#### Requires

Level-0 frames with VTC in meta-data, ancillary housekeeping telemetry for time covering frame exposure

#### Provides

Selected ancillary meta-data in FITS frame image headers, updated frame index, meta-data file

#### Description

Newly delivered level-0 frames will be correlated by time (the VTC) to the ancillary data derived asynchronously from payload telemetry delivered by MOS. If no ancillary data, especially the SPICE kernel, can be matched to the level-0 frames, INMETA will wait for a predetermined amount of time for the data to be ingested. If no ancillary data is matched even after the wait period is over, the level-0 frames that cannot be matched cannot be processed. The headers of all matched level-0 frames are updated to include ADCS position and orientation information from the decoded SPICE kernel, as well as TBD other ancillary data. The frame index is updated to include key meta-data for each matched frame. In addition, a meta-data file, which may, for example, be a FITS table file, be written in the same location as the matching level-0 frame to hold all pertinent meta-data.

## 6.3 Pipeline Functions (PIPELINE)

### 6.3.1 Instrumental Calibration (ICAL)

#### Purpose

Produce level-1 frames from level-0 frames by applying the static and computing and applying the dynamic instrumental calibration

#### Requires

Level-0 frames and meta-data, static instrumental calibration

#### Provides

Level-1a frames, QA data

#### Description

The following paragraphs describe some of the various calibration functions. For an in-depth overview of the instrumental calibration steps and the input calibrations needed, see Instrumental Image Calibration Steps ([http://web.ipac.caltech.edu/staff/fmasci/home/wise/SingleOrbit\\_Cal.html](http://web.ipac.caltech.edu/staff/fmasci/home/wise/SingleOrbit_Cal.html)) and Scan Pipeline Processing Instrumental Calibration ([http://web.ipac.caltech.edu/staff/fmasci/home/wise/ScanPL\\_instrumental\\_cal.pdf](http://web.ipac.caltech.edu/staff/fmasci/home/wise/ScanPL_instrumental_cal.pdf)) .

- Electronic Bias/Offset Correction

Use the reference (aka border, aka "over-scan") pixels to apply a pixel offset correction, as: subtract output-by-output reference pixel column offsets, then compute a running average of 7 reference pixel rows and subtract that prior to computing noise/dark current/etc.

- Dark Subtraction

Subtract dark frames to eliminate dark current. Dark frames will only be available from ground calibration.

- Droop Correction

The droop correction is a global offset correction based on the total illumination on the detector. For the purposes of droop computation, the exact pre-calibration pixel values will be used. The contribution of saturated pixels will need to be estimated for a robust determination of droop.



- Pixel Response Correction

Divide frames by a response map to correct for variations in pixel-to-pixel response. Response maps will be made during ground calibration and updated in flight using "super-flats".

- Non-linearity Correction

Make a linearity correction to pixel values. This correction depends on the pixel value (as a fraction of full well). Non-linearity corrections will be determined on the ground and can be updated in flight.

- Saturation Flagging

Raw pixel values will be examined for one of the pixel values indicating SUR value was saturated. The presence of saturation will cause a pixel to be flagged in a mask (though the value is preserved). No calibration information is required for saturation flagging.

- Detector Artifact Handling

Known and anticipated artifacts at this time are:

- Pixel persistence, AKA latent images
- Diffraction spikes
- Internal reflections
- Out of field scattered light

- Compute sky offset images dynamically for a scan of frames
- Compute flats dynamically from data following an anneal
- Provide tools for producing static calibration data from ground and non-orbit data

### 6.3.2 Photometric Calibration (PCAL)

#### Purpose

Take level-1,2 or 3 frames pr source lists derived from them and produce a photometric calibration relative to standard stars

#### Requires

Level-0 frames and meta-data, source lists, data from standard stars

#### Provides

Magnitude offsets to produce relatively calibrated photometric magnitudes

#### Description

Identify photometric standard stars in level-1b,2b frames or level-3 coadds, derive relative photometric zero-points

### 6.3.3 Source Detection and Extraction (DETEX)

#### Purpose

Detect and extract sources and source characteristics from a frame or coadd image

#### Requires

COADD output: Level-2 frames or level-3 coadds and associated products

#### Provides

Source catalog and meta-data, QA data

#### Description

DETEX performs detection ("segmentation" in image analysis terms) and extraction (source characterization), including astrometry and photometry, in a number of different contexts. Each context may conceivably use a different detection or characterization engine from other contexts, if necessary.

The different contexts identified thus far are:

- From level-2a (upsampled frames with distortion-corrected pixels prior to WCS refinement) frames for PREX, optimizing for speed and positional accuracy; the level-2 frames have
- From level-2b frames (level-2a frames with updated WCS) for solar system object matching and QA trending and other epoch-dependent science
- From coadds for addition to the coadd working database and in the final catalog

The complete list of required source characteristics is TBD, but includes at least

- position in sky (possibly more than one coordinate system) and image coordinates,
- flux, possibly in a few different states of calibration,
- position and flux errors,
- source shape information (e.g. PSF FWHM estimate, 2nd moments),
- background and noise metrics,
- and track back information on the source frames.

Detection (segmentation) and extraction will occur in different applications. The detection step is called MDET and is a multi-band detector. The photometry/astrometry characterization step is called MPHOT. MPHOT is in turn divided into two applications, one doing aperture photometry/astrometry, and one doing profile fitting

photometry/astrometry.

### 6.3.4 Band Merging (MERGE)

#### Purpose

To positionally associate extractions for a given image from multiple bands to provide enhanced extraction data

#### Requires

DETEX output for a level-2 frame, or a coadd, from one or more bands

#### Provides

A new band merged source list, with additional or updated characterization data, QA data

#### Description

Extractions for all bands present for a given frame or coadd are read and positionally associated. For each source, color information as well as enhanced position estimates and reliability are generated and recorded, along with the original single-band data, in a new band merge source file. For uniformity's sake, MERGE needs to be able to generate a band merge source file even if there's only one band.

**This module's output may be partially or fully accomplished in the DETEX module since DETEX will do multi-band detection.**

### 6.3.5 Pointing Reconstruction (PREX)

#### Purpose

Determine refined position, orientation, and distortion corrections too produce level-1b frames from level-1a frames

#### Requires

Level-1a frames for one or more bands including at least bands 1 or 2, DETEX output, point source astrometric reference catalog (from the 2MASS PSC)

#### Provides

Level-1b frames, with corrections for frame position and orientation and distortion

#### Description

Using matches between all extractions from level-1a frames for all available bands--at least including bands TBD--to the astrometric reference catalog, refine knowledge of the frame's position and orientation. Write the needed corrections as meta-data, and apply the frame WCS position center and orientation corrections to the level-1a frames to create level-1b frames.

### 6.3.6 Solar System Object Matching (SSOFIND)

#### Purpose

Match level-1b frame extracted sources to known solar system objects (SSO's)

#### Requires

Level-1b frame working source database and epoch, known SSO ephemerides and/or orbital elements, S/C ephemerides

#### Provides

Known SSO predictions and match source list

#### Description

Read the level-1b frame extracted source lists to get candidate source positions and observation epochs. Match these to known solar system objects.

### 6.3.7 Resampling and Coaddition (COADD)

#### Purpose

Produce coadded images from level-1 pixels

#### Requires

Level-0 images and other frame pipeline products

#### Provides

Coadd images and related prerequisites for DETEX.

#### Description

COADD encompasses these functions:

- Locating frames matching spatial and temporal requirements using the frame index
- Resample/interpolate level-1b pixels onto a common grid to identify outliers using a temporal stacking method
- Gain/throughput matching of level-1b frames by rescaling pixel values to a common photometric zero-point. This will be the photometric zero point for the output coadd
- Background (offset) matching of level-1b frames to produce seamless coadds
- Reproject, resample and combine level-1b pixels into a level-3 coadd or level-2 frame
- Generate accompanying coverage and uncertainty (sigma) coadds
- Provide prerequisites needed by DETEX

## 6.4 Quality Analysis Functions (QA)

#### Purpose

Provide Quality Analysis reports on level-1 frames, level-3 coadds, catalogs, dynamic calibration products,

source lists, ingest results, and other WSDS output

#### Requires

Frames or coadds, catalogs, pipeline QA output

#### Provides

QA reports

#### Description

The QA subsystem performs these functions:

- PSF and Scan-sync Analysis
- Survey Coverage Analysis
- Source Count and Confusion Analysis
- Artifact Analysis
- N/M Analysis
- Performance estimates for completeness, reliability, photometry, astrometry, sensitivity
- QA Report Display and Distribution

## 6.5 Final Product Generation Functions (FPG)

#### Purpose

Create exportable products from coadd images and working source databases (WSDB)

#### Requires

Coadds, coadd WSDBs, other products from the archive and coadd pipeline

#### Provides

Exportable coadd images (atlas images), exportable catalog

#### Description

The FPG subsystem is a collection of utilities for aiding in the preparation of the releasable products; the image atlas and the source catalog. Much of the work of product preparation will be manual--such as composing a query to select the rows of the working database to release as the catalog product--but some activities can be aided by support utilities. Here are some examples:

##### Catalog Generation

- Interfaces to IRSA functions for easy command line source selection
- Summary statistics generation
- Generating compact but readable source lists

##### Image Atlas Generation

- Image filtering tools?
- Convenient mosaic generation

## 6.6 Archive Functions (ARCHIVE)

#### Purpose

Provide tools and APIs to access the varied archival data generated by the WSDS

#### Requires

Raw or processed archival data, indices, parameters, required data specs, DB schemata, etc.

#### Provides

Requested data location or other meta-data information, and/or the actual data, in toto or in part

#### Description

WSDS archival data can be split into two general types, depending on whether it's loaded into IRSA servers or not.

- IRSA databases

Much WSDS data, final or intermediate, will be loaded into IRSA DBMS tables. Existing IRSA access tools and APIs can be used to read/write this data. New tools/APIs may also need to be written.

- Other databases

Databases exist distributed in a predetermined directory structure on data servers. ARCHIVE will provide tools and APIs to find, write and access this data, however it is distributed.

## 7 Reference Hardware Design

Here I describe one possible hardware implementation which could meet the throughput requirements (c.f. Throughput Reuirements) for the frame pipeline (FRAMEX).

The WSDC hardware system will have five major components:

#### A compute cluster

Composed of up to about 30 nodes, each of which has dual quad-core x64 processors (i.e. 8 cores), >=500GB internal disk storage, and >=8GB RAM. Some nodes may be super-sized for use in coadding, which may require more memory. Notionally, 20 of these nodes are devoted to scan/frame processing.

#### A single gigabit network

Interconnects all compute hardware and disk and backup storage. Possibly with multiple subnets for scaling access to the compute cluster or isolating parts of the network (ingest, backup). Trunking may be employed between backend RAID arrays (SAN?) and servers to increase bandwidth.

#### Dedicated servers

These are more robust than the compute nodes (dual power supplies, backplanes, mirrored disk, etc.). I know such servers are required for development and ingest.

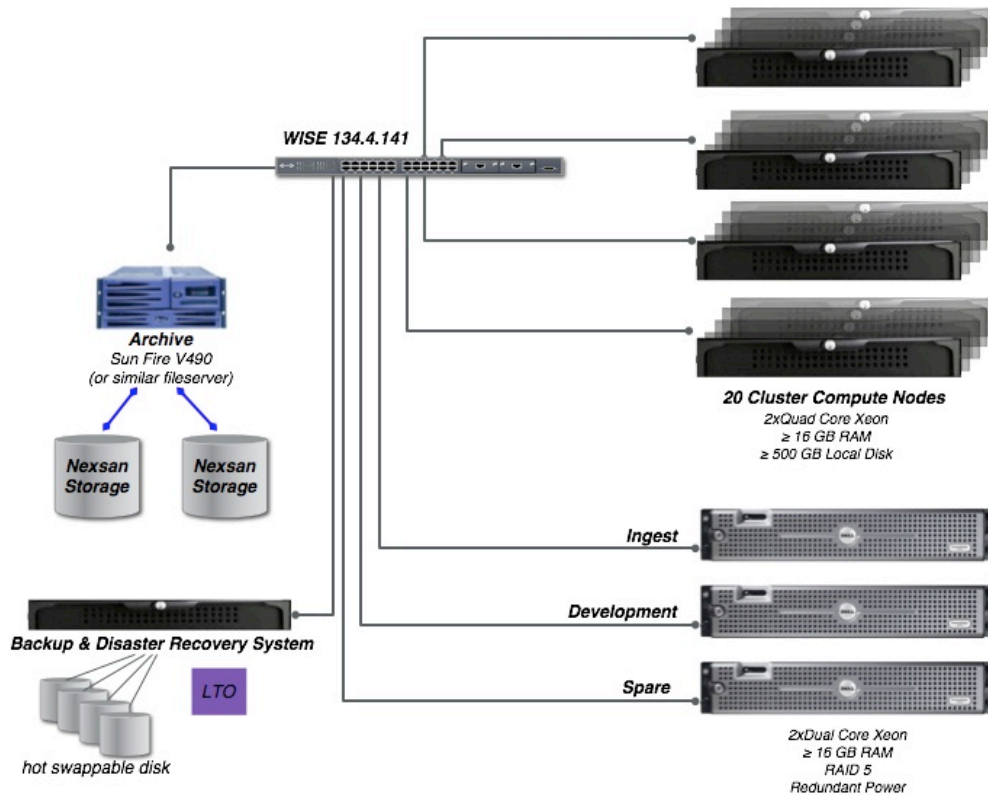
#### Archival disk

50-100 TB of RAID-5 (or RAID-Z?) disk and associated servers, probably using multiple devices.

#### A backup and offline archive system

Both tape- and removable disk-based.

Here is a diagram laying out the main features of one possible implementation of the above features.



## 8 Throughput Requirements

### 8.1 Assumptions

- Frame processing through pixel resampling (but not coaddition)
- Same amount of work per pixel as 2MASS
- Clock cycles mean about the same thing on 2MASS h/w as on WISE h/w
- CPU unconstrained by memory (no paging)
- A "frame" below refers to one exposure from a single band

### 8.2 Computations

- 2MASS run time per scan is 4047 secs

```
total   pixcal posfrm pixphot mapcor bmrgr pospts + gal + ohd
4047 =   787 + 167 + 1485 + 106 + 36 + 33 + 1397 37
```

- 2MASS pixels: 273 frames/scan \* 256x256 pixels/frame = ~16Mpix/scan
- 2MASS CPU: single ~300MHz SPARC
- Benchmarks (from <http://serv.apphy.fukui-u.ac.jp/~tajima/bench/result.html>):

```
matvec   : 48 Mflop/s (Mega-floating point ops per sec)
intgl4   : 134 Mflop/s
runge    : 192 Mflop/s
```

- Use 100Mflop/s as an average. Implies about 3 clock cycles per flop.
- $4000s * 100Mflop/s / 16 Mpix = 25kflop/pix$
- WISE core: 2.7GHz
- Assume same  $\sim 3$  clock cycles per flop  $\Rightarrow \sim 1Gflops$
- $25kflop/pix * 1Mpix/bandframe = 25Gflop/bandframe$
- $25Gflop/bandframe / 1Gflop/s = 25s/bandframe * 4 bands = \sim 2 min.s/frameset/core$

### 8.3 Discussion

If we make the totally unjustified assumption that we'll run at about 50% CPU time (i.e. 50% time doing I/O and paging, etc.), then we get about 4 minutes per frameset. **Note that the prototype pipeline is currently running at 3 minutes per frame per core, but without full-up source detection and photometry.**

I've previously computed that we can afford about 30 dual quad-core (8 core) machines reasonably configured (500GB disk, 8GB RAM). Assume we use 20 of them for scan/frame processing. With 160 cores we can process (assuming 1 frameset per core) 160 frames in  $\sim 240$  seconds, or about 1.5 s/framesets. Add in extra time for more complex coaddition, analysis, and 100% pad, and we have roughly 4 s/framesets.

At roughly 7000 framesets/day, this will require  $\sim 28000$  seconds, or about 8 hours. This leaves time for one rerun in parallel, and 8 hours of downtime for backup, maintenance, and other activities.

## 9 Data Layout

### 9.1 Introduction

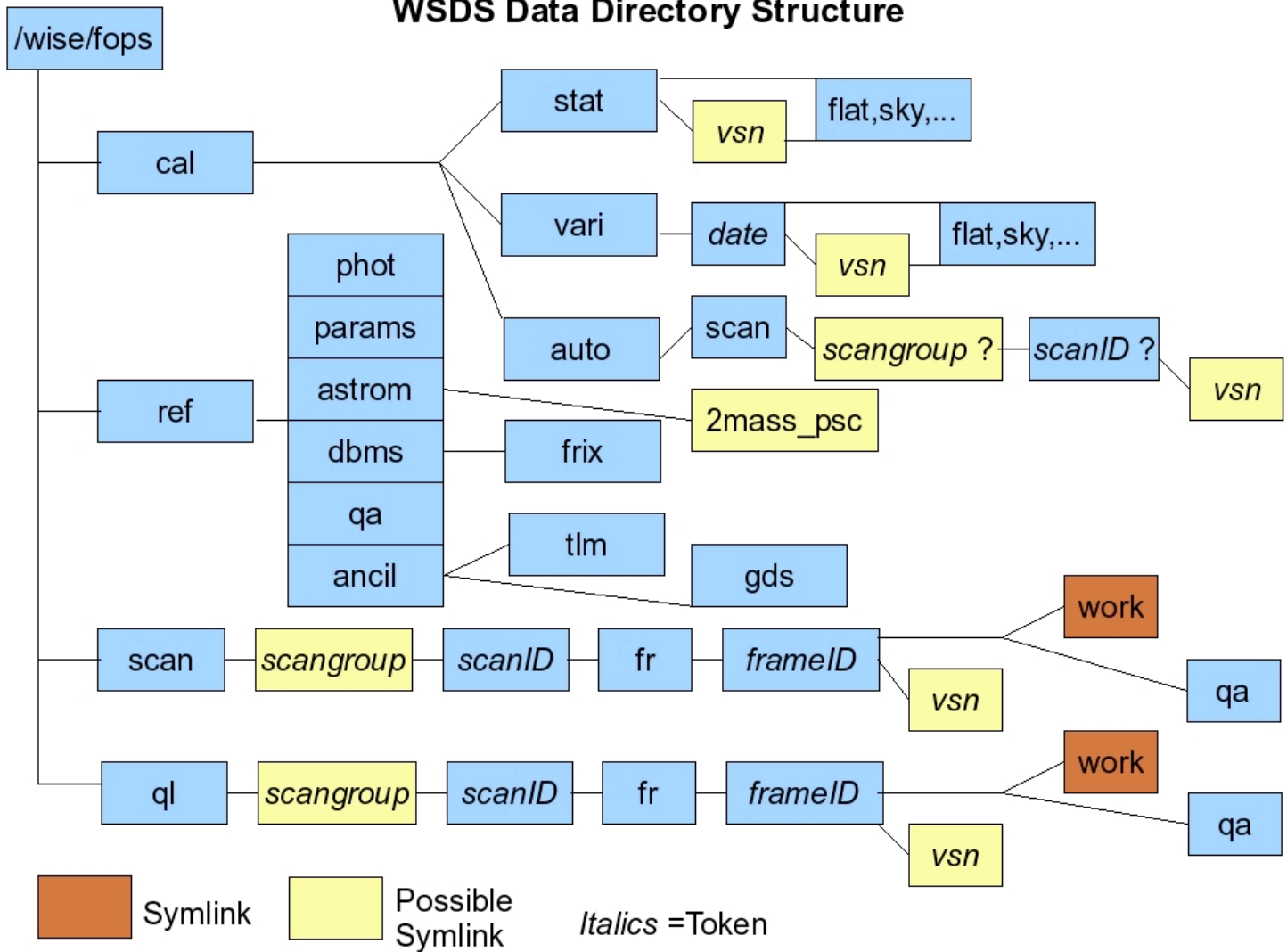
This section describes the location on WSDS networked storage of the various data files needed by the WSDS processing system.

The layout of many kinds of data on our distributed processing system must meet these requirements:

- Data must be in predictable locations derivable from more fundamental data (e.g. a frame ID).
- Sometimes data must be viewable across the network, other times it need only be visible to the currently-running pipeline.
- Our storage must be extensible without having to move large amounts of data around.
- Some data, e.g. calibration data, may need to be segregated by date/time (or, equivalently, scan ID and/or frame ID)
- Frame data (archival and otherwise), coadd data, and their derived products will be spread out over a combination of network-accessible disk archives and local storage.
- Reference data (quasi-static calibration data, reference catalogs, etc.) must be findable by all processes.
- We must clearly segregate ground data, flight data, and sim data at a high level.
- Data granularity must be such that a pipeline (or subsystem) will generally find all it needs in one or a few places, but a given directory must not have so much data in it that the file count becomes excessive.

Here is a diagram summarizing most of what is described.

## WSDS Data Directory Structure



### 9.2 The Use of Symlinks

Symlinks are a handy way to overlay a single, regular directory structure on top of data with may be physically scattered among many different physical devices. If one chooses the right granularity for one's directories (providing sufficient opportunity for indirection) one can also expand available space without displacing other data. For these reasons much of our data access will be through symlinks. In principle, one could move a great deal of data to a completely different location, and as long as the over-lying symlinks were adjusted, nothing would appear to have changed for programs or code needing access to the data.

But symlinks have a dark side too. The very fact they obscure the underlying physical layout of the data can sometimes engender confusion (you can easily go down through symlinks, but going back up ain't so easy). Also, some standard utilities (such as 'find') will not work so easily through symlinks. Also, if one wishes to record for posterity where something is/was located, special care might be required to record not just the overlying symlink-based path, but the true path as well. Symlinks are allowed to point to non-existent targets, which is a great and useful thing, but can also be confusing and sometimes frustrating.

**In the sections below, it is possible that any given path element may actually be a symlink**, though most symlinks will be limited to those documented in the next section. To a coder this means one has to be careful using relative paths. E.g. if one navigates to directory `/a/b/c/d` (i.e. it is your CWD) and plans on getting to directory `/a/b/c/e` by cd'ing to `../e`, you will fail if `d` is in fact a symlink pointing to a disjoint directory. We will try to arrange things so that one is unlikely to fall into such a trap.

Overall, symlinks are too handy not to use, but we will try to limit their use in ways that will mitigate problems.

#### 9.2.1 Known Symlinks

## work

The **work** directories in the frame pipeline output directories will be symlinks to node-local storage. All volatile images created as part of the frame pipeline will be stored here to avoid using the network. Note that this scheme has complicating repercussions:

- Since the work directory must be local to the node running the frame pipeline, subsequent reruns of the same version must delete the old work directory and create a new one on the currently-running node.
- All nodes must see all other nodes' local storage using the same name everywhere. (This appears to be tricky due to surprising interactions with NFS and the automounter.)

## latest version

The latest, highest numbered **v1, v2, ...** subdirectories (labeled **vsn** in the diagram) of the frame pipeline output directories will be a symlink to '.'. The other version directories will be real subdirectories of the output directory. Various calibration directories will also utilize this scheme.

## scangroup

In order to be able to load balance frame pipeline output across disks and servers, we may break the scans into groups which will be pointed to by symlinks. This is why the **scangroup** path element exists.

## 9.3 File Naming

The overall goal is to have file names that are reasonably compact but will be unique in most contexts--i.e. they do not rely on being in a given directory to maintain uniqueness--while maintaining a degree of readability. Even whether a directory full of various products sorts in a pleasing way when listed is considered.

In time, API's for composing and decomposing file names in C and fortran will be written. One exists now for use in Perl wrappers.

### 9.3.1 File Name Format

All WISE products will follow a common file name structure. Each file name is comprised of a series of fields separated by dashes (-). Fields may not contain a dash, and, to avoid various forms of confusion with either the shell or parameter handling, should also not include any of !\$&\*(){}[]";'< >/?= .

*base*[-band]-*type*[-*variant*].*format*>[.gz]

The fields are:

#### *base*

The fundamental identifier of the product. The thing that distinguishes this data from all others of the same band and type (see below). E.g. a frame ID, or a coadd ID. Speculatively, we may find reason to extend base names to handle more complex situations requiring additional underscore-separated subfields (see examples).

#### *band*

One of w1, w2, w3, or w4. This field is **optional** and should be omitted for products which are not specific to a band. On the other hand, products which are relevant to all bands, such as band-merged products, should use 'wx' (for now; more discussion needed. See below.).

#### *type*

The type of product; i.e. an identifier describing what the data are.

#### *variant* (aka variety) or version

An indicator of an available supply of variants this is; i.e. a data level indicator. This field is **optional**.

#### *format*

The format of the data.

#### *gz*

The usual tag indicating the file is compressed. This is **optional**.

#### 9.3.1.1 Known Bases

##### frameset ID

An alphanumeric tag uniquely identifying a frameset. The current definition of a frameID is <scan\_id><frame\_num>. <scan\_id> will be a 5 digit number plus a letter and counts scan over the whole mission, and <frame\_num> will be a 3 digit number and refers to the frame number for that scan.

##### coadd ID

a unique identifier for coadd products. As yet the coadd ID is undefined.

#### 9.3.1.2 Known Types

Here are some file types:

- 'int' for data images (i.e. for 'intensity')
- 'flg' for flag images
- 'msk' for mask images
- 'cov' for coverage images
- 'unc' for noise (uncertainty) images
- 'meta' for a meta-data table
- 'sdex' for single-frame detections, before photometry and astrometry (i.e. output from Ken's sdex code)

- 'scat' for a single-frame catalog after photometry and astrometry (derived from level 1+?), i.e. after PRex and Tom's photometer have had their say
- 'dex' and 'cat' are the versions of the above from coadds
- 'smcat' and 'mcat' are the single-frame and coadd band-merged detection and catalog products

### 9.3.1.3 Known Formats

- 'fits' for FITS files (images and tables)
- 'tbl' for IPAC table files
- 'param' for standard (Perl wrapper) parameter files (ASCII text)
- 'log' for a standard wrapper/pipeline log file (ASCII text)
- 'pd' for a Perl data dump file (from Data::Dumper)
- 'txt' for free-format text files
- 'rec' for record-oriented binary files
- 'bin' for arbitrary, undefined binary data
- jpg, gif, ps, etc. are the usual image products.

**NOTE:** All persistent products meant to be viewed/used by non-developers shall evolve to being either fits or tbl format, or a standard plot type (jpg, pdf, ps), except for the various textual QA data and reports.

### 9.3.1.4 Known Variants

- The various frame data product levels: 0, 1a-c, 2, 3.
- Can also hold a version number if multiple versions of a file must be maintained in a single run

## 9.3.2 Some Examples

- A band 1 level 0 frame from early in the mission: 00223a192-w1-int-0.fits
- A band 3, instrumentally calibrated (level 1a) frame containing intensity values: 00223a192-w3-int-1a.fits
- An uncertainty frame corresponding to the previous intensity frame: 00223a192-w3-unc-1a.fits
- A mask frame corresponding to the previous frame: 00223a192-w3-msk-1a.fits
- A PRex meta-data file for the same frame: 00223f192-meta-sfprex.tbl
- Output from wsdex (the wrapper around Ken's sdex code) for the same frame: 00223a192-w1-sdex.tbl
- A band-merged, post-photometry catalog: 00223a192-smcat.tbl
- The same thing but with a (speculative) subfield in the base name to indicate a specially massaged version of the data is in use: 00223a192\_pp1-smcat.tbl ('pp1' for pre-processing style #1).

## 9.4 Directory Structure

### 9.4.1 The Data Root

All data, products, etc. received for or generated by pipeline and associated applications will be accessed through

- /wise/fops for actual flight operations.
- /wise/tops for pre- or post- flight test operations.

While we are still far from flight, we can use 'fops'. We just have to remember to clear it out (copy to 'tops'?) when we get close to launch. When we are preserving the sanctity of 'fops' (near or after launch) we can conduct tests etc. in 'tops'.

Below I will always use 'fops'.

### 9.4.2 Reference Data

Reference data is anything generated somewhere else which is fairly static. Astrometric or photometric catalogs are examples. Obvious subdirectories might be

- /wise/fops/ref/astrom for astrometric reference catalogs, each in its own sub-directory.
- /wise/fops/ref/photom for photometric reference catalogs, each in its own sub-directory.
- /wise/fops/ref/params for parameters. There will be a 'wrap' sub-directory for standard perl wrapper parameters. Other applications/functions may require their own directories. These are high-level parameter files and will be overridden by the contents of parameters in /wise/base/deliv/[ops,dev,...]/params and by any run-specific parameters.
- /wise/fops/ref/frix for frame index DB files (i.e. related files which describe frame data locations and status).
- /wise/fops/ref/ancil for frame-related ancillary data, e.g. H/K telemetry. This is a refined set of DBMS (SQLite?) tables to be used by ingest in constructing the level-0 frames, not the raw ancillary data archive, which may be very little more than a date-tagged copy of what was sent from MOS).
- /wise/fops/ref/qa for global QA data. Placeholder. Not sure what really goes here.

### 9.4.3 Calibration Data

Calibration data is divided into sections based on how often the data are expected to be updated.



- `/wise/fops/cal/stat` for static calibration data (ground data or IOC data generated only once. Cal. data may be versioned within the static directory. The active version is always in the main directory; other versions are in subdirectories named "vd[d]", e.g. v1, v2, v2.2, etc. E.g. `/wise/fops/cal/stat` would contain the active static calibration files, and `/wise/fops/cal/stat/v3` points to a particular, inactive version.
- `/wise/fops/cal/vari` for data which will periodically be adjusted throughout the mission, but is not generated autonomously by the pipeline. Subdirectories will use dates or date ranges encoded as `yyddd` or `yyddd-yyddd`. Below the date range, the data will be versioned as above.
- `/wise/fops/cal/auto` for data generated autonomously by the pipeline. Subdirectories will scan groups and scanIDs. Within the scanID data will be versioned as above.

#### 9.4.4 Pipeline Input/Output Directories

The two main pipelines are the scan/frame pipeline, and the coadd pipeline. I'll be mainly concerned with the scan/frame pipeline.

The scan/frame pipeline is run in two different modes after ingest: the quicklook mode, and the full mode. These two modes will run in completely separate hierarchies:

- `/wise/fops/ql`: Quicklook output.
- `/wise/fops/scan`: Full scan/frame pipeline output.

##### 9.4.4.1 Scan and Frame IDs in the Directory Structure

I will just refer to `/wise/fops/scans` below; `/wise/fops/ql` parallels its structure.

The next level down segregates the frame data by scan IDs and frame IDs. But because there will be so many scans over the life of the mission (easily > 10,000), for convenience the scans should be subdivided into groups of no more than a few hundred scans per subdirectory. Accordingly, we divide the data up based on the last three digits/characters of the scan ID. Scans within one scan group directory use their full 5 digit+letter ID:

`/wise/fops/scans/42b/00142b`

By using the low order three characters, we can do elementary load balancing by spreading adjacent scans across different disks through scan group symlinks.

Below the scan, the frame ID is the sequential number within that scan, a 3 digit number, within the scan's 'fr' subdirectory:

`/wise/fops/scans/42b/00142b/fr/185`

Each frame may have multiple processing attempts, each represented by a version number of the format `vd[d]` (e.g. v1, or v2, or v2.2). The latest run will always be in the main directory. So a full path name to scan/frame pipeline output would be `/wise/fops/scans/42b/00142b/fr/185` and that for a particular old version would be `/wise/fops/scans/42b/00142b/fr/185/v3`. The current, active version is pointed to by a version symlink, i.e. if the active version is v4, the v4 symlink would point to '!'.

##### 9.4.4.2 Frame Output Directory

Each frame output directory will contain most relevant data at the top level, but will have two supplementary subdirectories, 'qa', and 'work':

`/wise/fops/scans/001/00142b/fr/185/qa` `/wise/fops/scans/001/00142b/fr/185/work`

'qa' holds qa-specific output, and 'work' will point to local disk space and the contents are to be considered volatile.

#### 9.4.5 Ingest Directories

The ingest process which reads and uncompresses the pushed HRP data from White Sands and creates level-0 data frames sets will write to its own hierarchy:

- `/wise/fops/ingest/tlm`: ingest meta-data divided by delivery IDs. Delivery IDs are TBD but will probably be a shortened version of the date/time-based file names used for the telemetry files. E.g. `YYMMDD_HHMMSS`. Since there will be upwards of 700 of these over the course of the mission, these will be divided by month. Thus each delivery will have a directory of the form `YYMM/YYMMDD_HHMMSS`. E.g. `/wise/fops/ingest/tlm/0912/091225_011256`.
- `/wise/fops/ingest/meta`: ingest meta data, such as a delivery index for frame and h/k (see below) data.

#### 9.4.6 Archives

At the moment, four WSDC archives are contemplated:

- `/wise/fops/arch/10`: Level-0 frame archive. Follows the naming scheme for scan/frame pipeline output.
- `/wise/fops/arch/ancil`: Raw ancillary data archive. Structure TBD.
- `/wise/fops/arch/fr`: Level-1b frame archive. Follows the naming scheme for scan/frame pipeline output.
- `/wise/fops/arch/coadd`: Coadd archive. This is for the pre-planned, laid out, official, proto-atlas coadds, not

the on-the-fly, on-demand coadds.

## 10 WSDC Operations Concept

Here is a brief accounting of a day-in-the-life at the WSDC during operations. This is not an exhaustive treatment of WSDC resource utilization, but shows just one example how resources for one random day might go. Exact timings represent reasonable guesses based on what we know, which is still sketchy for many processes.

### Assumptions

- 4 deliveries a day, 5 hours apart;
- 8 scans completed per delivery
- 3 FTE QA analysts
- 2 FTE operations personnel
- 5/40 staffing
- A delivery of science data requires 3.5 hours from arrival at the HRP to complete transmission to the WSDC
- Ancillary data arrives 1 hour before the science data and takes 10 minute to transfer and 20 minutes to archive
- Ingest of science data requires 2 hours
- Quicklook requires 30 minutes to complete
- ScanPipe requires 2 hours to complete for the ~8 new scans completed in a single delivery
- Operations staff require 30 minutes of ops personnel time per delivery
  - 15 minutes to log and archive to disk and tape newly ingested telemetry (doesn't count actual recording time)
  - 15 minutes to monitor, examine and report on a ScanPipe run
- Additionally, operations staff must devote
  - 2 hours to recent coadd processing, either running (starting) or monitoring
  - 2 hours to system-level WSDS backups
- QA analysis requires about 1 hour per scan
  - 5 minutes for a normal scan, and 30 minutes for problematic scans
  - 1 scan per delivery (1/8) will be troubled
- Additionally, QA analysts must devote
  - 3 hours to coadd QA
  - 3 hours to medium/longterm trending analysis
- It is a Monday, so 8 weekend deliveries must be handled
- 2 overnight deliveries must be handled

**WSDC Operations for a Sample Monday**

Time	Ops Personnel Activity		QA Personnel Activity			Automated Activity
	1	2	1	2	3	
00:00	-	-	-	-	-	Receipt of ancillary data delivery 1 begins.
00:10	-	-	-	-	-	Ancillary data delivery 1 ends. Data are archiving begins.
00:30	-	-	-	-	-	Ancillary data delivery 1 archiving complete.
01:00	-	-	-	-	-	Receipt of science data delivery 1 begins.
04:30	-	-	-	-	-	Delivery 1 receipt complete. Ingest begins.
05:00	-	-	-	-	-	Receipt of ancillary data delivery 2 begins.
05:10	-	-	-	-	-	Ancillary data delivery 2 ends. Data are archiving begins.
05:30	-	-	-	-	-	Ancillary data delivery 2 archiving complete.
06:00	-	-	-	-	-	Receipt of science data delivery 2 begins.
06:30	-	-	-	-	-	Delivery 1 ingest complete. Quicklook initiated on selected frames. ScanPipe initiated on newly completed scans.
07:00	-	-	-	-	-	Delivery 1 Quicklook completed.
07:30	-	-	-	-	-	
08:00	-	-	-	-	-	
08:30	-	-	-	-	-	Delivery 1 new scan ScanPipe processing completed.
09:00	Weekend runs	Overnight runs	Weekend runs	Weekend runs	Overnight runs	
09:30	"	"	"	"	"	Delivery 2 receipt complete. Ingest begins.

10:00	"	-	"	"	"	Receipt of ancillary data delivery 3 begins.
10:10	"	-	"	"	"	Ancillary data delivery 3 ends. Data are archiving begins.
10:30	"	Delivery 1	"	"	"	Ancillary data delivery 3 archiving complete.
11:00	Status Review	Status Review	Status Review	Status Review	Status Review	Receipt of science data delivery 3 begins.
11:30	Weekend runs	-	-	-	Delivery 1	Delivery 2 ingest complete. Quicklook initiated on selected frames. ScanPipe initiated on newly completed scans.
12:00	-	-	-	-	"	Delivery 2 Quicklook completed.
12:30	-	-	-	-	-	
13:00	"	-	Coadds	Trending		
13:30	"	Delivery 2	"	"	Delivery 2	Delivery 2 new scan ScanPipe processing completed.
14:30	"	-	"	"	"	Delivery 3 receipt complete. Ingest begins.
15:00	Run Coadds	Backups	"	"	-	Receipt of ancillary data delivery 4 begins.
15:10	Monitor Coadds	"	"	"	-	Ancillary data delivery 4 ends. Data are archiving begins.
15:30	"	"	"	"	-	Ancillary data delivery 4 archiving complete.
16:00	"	"	-	-	-	Receipt of science data delivery 4 begins.
16:30	"	"	-	-	-	Delivery 3 ingest complete. Quicklook initiated on selected frames. ScanPipe initiated on newly completed scans.
17:00	Status Review	Status Review	Status Review	Status Review	Status Review	Delivery 3 Quicklook completed.
18:00	-	-	-	-	-	
18:30	-	-	-	-	-	Delivery 3 new scan ScanPipe processing completed.
19:30	-	-	-	-	-	Delivery 4 receipt complete. Ingest begins.
21:30	-	-	-	-	-	Delivery 4 ingest complete. Quicklook initiated on selected frames. ScanPipe initiated on newly completed scans.
22:00	-	-	-	-	-	Delivery 4 Quicklook completed.
23:30	-	-	-	-	-	Delivery 4 new scan ScanPipe processing completed.

## 11 Glossary

### Ancillary data

Meta-data related to processing image data derived from payload housekeeping and ancillary telemetry.  
Examples: the VTC, SPICE kernels (derived from ADCS telemetry), instrument temperatures, etc.

### API

Application Programming Interface. The means by which an application accesses the functionality of a library or module.

### Atlas

A collection of specially generated coadds which comprise the releasable WISE image product.

### Bandframe

Image for a single band for one exposure.

### Catalog

A selection of entries and column values (possibly refined) from a working source database constructed to meet certain completeness, reliability, astrometric and photometric requirements.

### CCSDS

The Consultative Committee for Space Data Systems. Refers to the standard defining the packet structure used in WISE telemetry, in particular the source packet structure encoding WISE science images delivered by the HRDP at White Sands to the WSDC.

### Coadd

An image generated from the coadded pixels from one or more level-1 frames covering a defined rectangular region of sky. See also "Image, Level-3"

### Frame

Detector data from a single exposure. Might refer to a single band, or to all available bands, depending on context.

### Frame index

A database storing meta-data on the characteristics, location and processing status of all framesets received at

the WSDC.

Frame, Level-0 frame

Same as a raw frame but with 4-byte floating point pixels (BITPIX=-32) and band 4 replicatively upsampled to 1024x1024. This is the archival frame format.

Frame, Level-1a frame

A level-0 frame with relative photometric calibrations applied.

Frame, Level-1b frame

A level-1a frame with refined frame position and orientation WCS information

Frame, Level-1c frame

Old terminology for a level-2b frame

Frame, Level-2a frame

A level-1a frame which has been resampled, and possibly upsampled, to correct for distortion

Frame, Level-2b frame

A level-1b frame which has been resampled, and possibly upsampled, to correct for distortion

Frame, Raw

Detector data exactly as assembled from science source packets. Pixels are unsigned 2-byte integers. Bands 1-3 frames are 1024x1024, band 4 is 512x512.

Frameset

Images in all available bands for one exposure.

HRDP, HRP

High Rate Data Processor. The server and associated storage at White Sands which autonomously receives and processes WISE telemetry and pushes the reordered, decommutated source packets either to the WSDC or the JPL MOS.

ICD

Interface Control Document

Image, Level-3

A coadd

Internal Calibration

See Relative Calibration

IRSA

Infrared Science Archive. The group/system at IPAC which provides the archive and delivery service for WISE data products and other internal databases.

Job

A process or logical collection of processes and threads performing a computational task.

Latent image

A false "ghost" image in frame data caused by a persistent excess current and/or a change in the response of pixels illuminated by a bright source in a prior frame. May be either a positive or negative deviation from the background.

Module

The level of functional grouping below sub-system.

MOS

Mission Operations System. The group/system at JPL responsible for operating the WISE satellite, including commanding and managing telemetry.

Relative Calibration

Instrumental frame calibrations which result in all pixels having flattened, linear, zero-offset response. Pixel fluxes are not yet tied to an absolute photometric reference.

Scan

A period of observation running from one ecliptic pole crossing to the next. I.e. about half an orbit.

Source Packet

A unit of telemetry in a standard format which contains primary telemetry data, such as image data or housekeeping engineering data.

S/C

Spacecraft

SOC

Science Operations Center. The group/system at UCLA responsible for generating the survey plan and the ADCS commands necessary to implement it.

Sub-system

The highest-level division, or grouping, of functions within the WSDC. See also "module."

VTC

Vehicle Time Code. The time stamp marking frame data and ancillary data generated on-board the S/C. Converted to UTCS in processing.

Working source database (WSDB)

A database of raw source extractions from frames (level-1) or coadds.

WSDC

WISE Science Data Center. The group at IPAC responsible for processing of WISE science data and releasing data products.

WSDS

WISE Science Data System. The collection of software, hardware, and facilities which performs all data processing at the WSDC.

Retrieved from "[http://wise.ipac.caltech.edu/wiki/index.php/Project\\_Docs:FDD](http://wise.ipac.caltech.edu/wiki/index.php/Project_Docs:FDD)"