ICD-SW-15
Subsystem : camera

Contents

1.0 SCOPE

This Interface Control Document (ICD) specifies the interface(s) between OCS and other subsystems. Upon formal approval by the line Manager responsible for each participating system, this ICD shall be incorporated into the requirements baseline for each system.

1.1 System Identification

1.1.1 OCS

1.1.2 camera

1.2 Document Overview

The purpose of the ICD is to specify interface requirements to be met by the participating systems. It describes the concept of operations for the interface, defines the message structure and protocols which govern the interchange of data, and identifies the communication paths along which the data is expected to flow.

1.3 Glossary

BEE - Back-End Electronic
CALSYS - Camera Calibration System
CCS - Camera Control System
DDS - Data Distribution Service
FCS - Filter Controller Subsystem
FEE - Front End Electronics
FPA - Focal Plane Array Actuation
GAS - Guider data acquisition system
GS - Guider system
L2U - L2 Controller Unit
LASERCAL - Camera metrology calibration System
LSST - Large Synoptic Survey Telescope
OMG - Object Management Group
PWR - Camera Power supply System
QA - Camera Quality Assurance measurements
QoS - Quality of Service
RAS - Raft Alignment Subsystem
RTI - Real Time Innovation
SAL - Software Abstraction Layer
SAS - Science Array System
SCU - Shutter Controller Unit
SDS - Science Data Acquisition System
TC - Thermal Control
TCM - Camera Timing control
VCS - Vacuum Control Subsystem
WDS - Wavefront data acquisition system
WFS - Wave-front Sensing System
WTCM - Camera wavefront sensors Timing control<

1.4 Applicable Documents

Datastream Definitions Document - Datastream Prototypes 1.1 (Document-1887)
Definition of subsystems - LSST Project WBS Dictionary (Document-985)
Documentation standards - LSST DM UML Modeling Conventions (Document-469)
Messaging standards - OMG DDS 1.1 (Document-1835)
Software Abstraction Layer API - Middleware Software Abstration Layer (Document-3692)
Software coding standards LSST C++ Programming Style Guidelines (Document-3046)
Vendor documentation - NDDS manual (Document-2241)

2.0 CONCEPT OF OPERATIONS

2.1 System Overviews

Subsystem description : OCS

The Observatory Control System (OCS) fulfill the prime role to efficiently support the acquisition of science data from the telescope at all times of the day and night, by operating as the overall master control system that schedules, coordinates and monitors the observatory. The OCS orchestrates and controls all aspects of the observatory for all operation modes, including science observations, calibration, and engineering monitoring and maintenance. The OCS is responsible for high-level observatory operations including user interfaces, scheduling, resource allocation and system safety. Through the OCS, the LSST system can be started, monitored, adjusted, and stopped, both locally and remotely. In essence, the Observatory Control System (OCS) is the glue that coordinates and synchronizes the principal subsystems in the LSST: the Telescope Control System (TCS), the Camera Control System (CCS), the Data Management System (DMS), and the Calibration/Auxiliary Control System (CACS).

In addition to its control functions, the OCS captures, organizes, and stores system-wide state and telemetty information, and makes it available for monitoring, evaluation, and calibration processes. This information is analyzed daily in order to perform preventive maintenance, to optimize the system, and to measure performance trends over long time periods. There is a regular daily cadence to the operations of the observatory in terms of data acquisition, engineering, calibration, and science. The OCS must support all of these activities and the handover between them. An operator taking responsibility for the telescope at night must be able to see at a glance what the state the control system is, just as an engineer on the morning shift needs to know how the telescope and instrument have been left by the nighttime users.

The OCS makes use of a communications layer that provides an efficient and reliable way of exchanging messages between the different systems and components of the LSST complex. The communications layer will be built on industry standard communications middleware. Successful development of distributed application systems relies on middleware that separates application-specific functionality from the logic complexities inherent in a distributed infrastructure.

Subsystem description : camera

The camera is one of the four primary LSST technical subsystems (along with the Telescope/Site Subsystem, the Data Management Subsystem,and the Auxillary Calibration telescope) . It contains a 3.2 Gigapixel focal plane array, comprised of roughly 200 4K x 4K CCD sensors, with 10 micron pixels.

The sensors are deep depleted, back illuminated devices with a highly segmented architecture that enables the entire array to be read out in 2 s or less. These detectors are grouped into 3 x 3 arrays called rafts. Each raft contains its own dedicated front end and back end electronics boards, which fit within the footprint of its sensors, thus serving as a 144 Megapixel camera on its own.

All of the rafts, with their associated electronics, are mounted on a silicon carbide grid inside a vacuum cryostat, with an intricate thermal control system that maintains the CCDs at an operating temperature of roughly minus 90 degrees centigrade.

The grid also contains sets of guide sensors and wavefront sensors at the edge of the field. The entire grid, with the sensors, is actuated at a rate ~ 30 Hz in a fast guiding mode to maintain a very narrow psf (0.7 arcseconds median), which is limited mainly by seeing fluctuations in the overlying atmosphere.

The entrance window to the cryostat is the third of the three refractive lenses. The other two lenses are mounted in an optics structure at the front of the camera body. The camera body also contains a mechanical shutter, and a carrousel assembly that holds five large optical filters, any of which can be inserted into the camera field of view for a given exposure. A sixth optical filter will also be fabricated which can replace any of the five via an automated procedure accomplished during daylight hours.

The camera system consists of multiple subsystems that include utilities, the camera body vessel and mechanisms for shuttering and optical filtering, the imaging sensors, optical lenses and filters, a computerized data acquisition and control system, the cryostat holding the detector array, readout and control electronics, wavefront sensors, and guide sensors.

2.1.1 Interface Overview

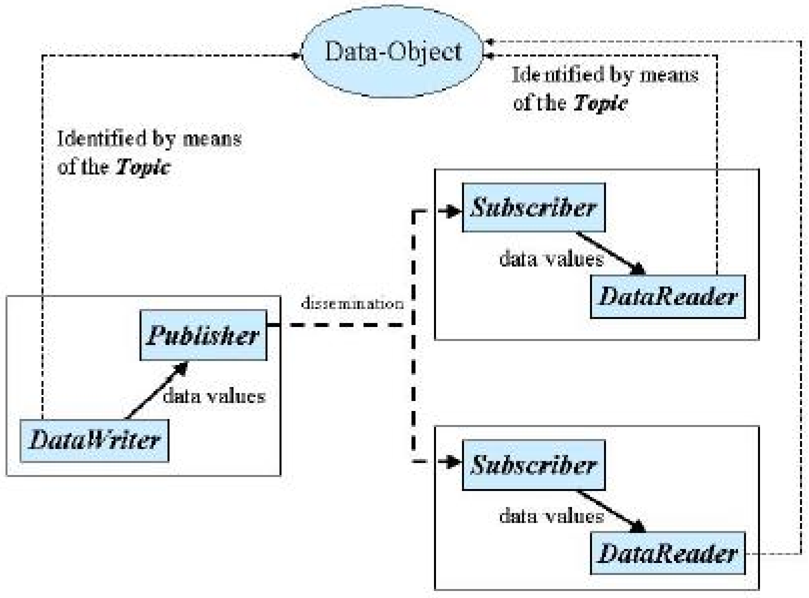
Hardware - Network diagram

----------ADD NETWORK DIAGRAM HERE----------------------

Software

The publish-subscribe communications model provides a more efficient model for broad data distribution over a network than point-to-point, client-server, and distributed object models. Rather than each node directly addressing other nodes to exchange data, publish-subscribe provides a communications layer that delivers data transparently from the nodes publishing the data to the nodes subscribing to the data. Publishers send events to the communications layer, which in turn, passes the events to subscribers. In this way, a single event published by a publisher can be sent to multiple subscribers. Events are categorized by topic, and subscribers specify the topics they are interested in. Only events that match a subscribers topic are sent to that subscriber. The service permits selection of a number of quality-of-service criteria to allow applications to choose the appropriate trade-off between reliability and performance.

The combination of Client-Server and Publish-Subscribe models leads to the concept of Command/Action/Response model, in that the transmission of commands is decoupled from the action that executes that command. A command will return immediately; the action begins in a separate thread. Figure 3 illustrate this model by means of a simplified sequence diagram. When an application receives a command, it validates the attributes associated with that command and immediately accepts or rejects the command. If the command is accepted, the application then initiates an independent internal action to meet the conditions imposed by the command. Once those conditions have been met, an event is posted signifying the successful completion of the action (or the unsuccessful completion if the condition not be met). In this figure, callbacks are implemented using the event features of the publish-subscribe model.



Information flows with the aid of the following constructs : Publisher and DataWriter on the sending side, Subscriber, and DataReader on the receiving side.

A Publisher is an object responsible for data distribution. It may publish data of different data types. A DataWriter acts as a typed4 accessor to a publisher. The DataWriter is the object the application must use to communicate to a publisher the existence and value of data-objects of a given type. When data-object values have been communicated to the publisher through the appropriate data-writer, it is the publisher's responsibility to perform the distribution (the publisher will do this according to its own QoS, or the QoS attached to the corresponding data-writer). A publication is defined by the association of a data-writer to a publisher. This association expresses the intent of the application to publish the data described by the data-writer in the context provided by the publisher.

A Subscriber is an object responsible for receiving published data and making it available (according to the Subscribers QoS) to the receiving application. It may receive and dispatch data of different specified types. To access the received data, the application must use a typed DataReader attached to the subscriber. Thus, a subscription is defined by the association of a data-reader with a subscriber. This association expresses the intent of the application to subscribe to the data described by the data-reader in the context provided by the subscriber.

Topic objects conceptually fit between publications and subscriptions. Publications must be known in such a way that subscriptions can refer to them unambiguously. A Topic is meant to fulfill that purpose: it associates a name (unique in the domain), a data-type, and QoS related to the data itself. In addition to the topic QoS, the QoS of the DataWriter associated with that Topic and the QoS of the Publisher associated to the DataWriter control the behavior on the publisher's side, while the corresponding Topic, DataReader, and Subscriber QoS control the behavior on the subscribers side.

When an application wishes to publish data of a given type, it must create a Publisher (or reuse an already created one) and a DataWriter with all the characteristics of the desired publication. Similarly, when an application wishes to receive data, it must create a Subscriber (or reuse an already created one) and a DataReader to define the subscription.

QoS (Quality of Service) is a general concept that is used to specify the behavior of a service. Programming service behavior by means of QoS settings offers the advantage that the application developer only indicates what is wanted rather than how this QoS should be achieved. Generally speaking, QoS is comprised of several QoS policies. Each QoS policy is then an independent description that associates a name with a value. Describing QoS by means of a list of independent QoS policies gives rise to more flexibility.

This specification is designed to allow a clear separation between the publish and the subscribe sides, so that an application process that only participates as a publisher can embed just what strictly relates to publication. Similarly, an application process that participates only as a subscriber can embed only what strictly relates to subscription.

Underlying any data-centric publish subscribe system is a data model. This model defines the global data space and specifies how Publishers and Subscribers refer to portions of this space. The data-model can be as simple as a set of unrelated datastructures, each identified by a topic and a type. The topic provides an identifier that uniquely identifies some data items within the global data space1. The type provides structural information needed to tell the middleware how to manipulate the data and also allows the middleware to provide a level of type safety. However, the target applications often require a higher-level data model that allows expression of aggregation and coherence relationships among data elements.

Another common need is a Data Local Reconstruction Layer (DLRL) that automatically reconstructs the data locally from the updates and allows the application to access the data as if it were local. In that case, the middleware not only propagates the information to all interested subscribers but also updates a local copy of the information.

2.2 Functional Allocation

----------ADD PER DEVICE COMMAND TYPES HERE----------------------

2.3 Data Transfer

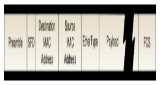
Hardware - Ethernet

Ethernet stations communicate by sending each other data packets, small blocks of data that are individually sent and delivered. As with other IEEE 802 LANs, each Ethernet station is given a single 48-bit MAC address, which is used both to specify the destination and the source of each data packet. Network interface cards (NICs) or chips normally do not accept packets addressed to other Ethernet stations. Adapters generally come programmed with a globally unique address, but this can be overridden, either to avoid an address change when an adapter is replaced, or to use locally administered addresses.

Main procedure

- **1. Frame ready for transmission**
- **2. Is medium idle? If not, wait until it becomes ready and wait the interframe gap period (9.6 Âµs in 10 Mbit/s Ethernet).**
- **3. Start transmitting**
- **4. Does a collision occur? If so, go to collision detected procedure.**
- **5. Reset retransmission counters and end frame transmission**
- **6. Collision detected procedure , Continue transmission until minimum packet time is reached (jam signal) to ensure that all receivers detect the collision**
- **7. Increment retransmission counter**
- **8. Is maximum number of transmission attempts reached? If so, abort transmission.**
- **9. Calculate and wait random backoff period based on number of collisions**
- **10. Re-enter main procedure at stage 1**

Frames are the format of data packets on the wire. Note that a frame viewed on the actual physical hardware would show start bits, sometimes called the preamble, and the trailing Frame Check Sequence. These are required by all physical hardware and is seen in all four following frame types. They are not displayed by packet sniffing software because these bits are removed by the Ethernet adapter before being passed on to the network protocol stack software.



Software - OMG DDS

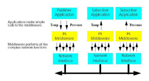
The OMG Data-Distribution Service (DDS) is a new specification for publish-subscribe data-distribution systems. The purpose of the specification is to provide a common application-level interface that clearly defines the data-distribution service. The specification describes the service using UML, providing a platform-independent model that can then be mapped into a variety of concrete platforms and programming languages.

The goal of the DDS specification is to facilitate the efficient distribution of data in a distributed system. Participants using DDS can read and write data efficiently and naturally with a typed interface. Underneath, the DDS middleware will distribute the data so that each reading participant can access the most-current values. In effect, the service creates a global data space that any participant can read and write. It also creates a name space to allow participants to find and share objects.

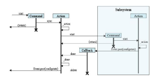
DDS targets real-time systems; the API and QoS are chosen to balance predictable behavior and implementation efficiency/performance.

2.4 Transactions

Publish/Subscribe



Command/Response



Notification/Alert

Logging service for collecting, recording, and analyzing system messages. The logging service supports multiple messages categories (e.g. Error, debug, informative) and message levels (e.g. Different levels of debug messages).

2.5 Security and Integrity

Hardware

Firewall

A firewall's basic task is to regulate the flow of traffic between computer networks of different trust levels. Typical examples are the Internet which is a zone with no trust and an internal network which is a zone of higher trust. A zone with an intermediate trust level, situated between the Internet and a trusted internal network, is often referred to as a perimeter network or Demilitarized zone (DMZ).

Packet filtering

Packet filters act by inspecting the packets which represent the basic unit of data transfer between computers on the Internet. If a packet matches the packet filter's set of rules, the packet filter will drop (silently discard) the packet, or reject it (discard it, and send error responses to the source).

This type of packet filtering pays no attention to whether a packet is part of an existing stream of traffic (it stores no information on connection state). Instead, it filters each packet based only on information contained in the packet itself (most commonly using a combination of the packet's source and destination address, its protocol, and, for TCP and UDP traffic, which comprises most internet communication, the port number).

Because TCP and UDP traffic by convention uses well known ports for particular types of traffic, a stateless packet filter can distinguish between, and thus control, those types of traffic (such as web browsing, remote printing, email transmission, file transfer), unless the machines on each side of the packet filter are both using the same non-standard ports. Second Generation firewalls do not simply examine the contents of each packet on an individual basis without regard to their placement within the packet series as their predecessors had done, rather they compare some key parts of the trusted database packets. This technology is generally referred to as a 'stateful firewall' as it maintains records of all connections passing through the firewall, and is able to determine whether a packet is the start of a new connection, or part of an existing connection. Though there is still a set of static rules in such a firewall, the state of a connection can in itself be one of the criteria which trigger specific rules.

This type of firewall can help prevent attacks which exploit existing connections, or certain Denial-of-service attacks, including the SYN flood which sends improper sequences of packets to consume resources on systems behind a firewall.

Private subnet

Firewalls often have network address translation (NAT) functionality, and the hosts protected behind a firewall commonly have addresses in the private address range, as defined in RFC 1918. Firewalls often have such functionality to hide the true address of protected hosts. Originally, the NAT function was developed to address the limited amount of IPv4 routable addresses that could be used or assigned to companies or individuals as well as reduce both the amount and therefore cost of obtaining enough public addresses for every computer in an organization. Hiding the addresses of protected devices has become an increasingly important defense against network reconnaissance.

Software

NDDS domains

The domain is the basic construct used to bind individual applications together for communication. A distributed application can elect to use a single domain for all its data-centric communications.

All Data Writers and Data Readers with like data types will communicate within this domain. DDS also has the capability to support multiple domains, thus providing developers a system that can scale with system needs or segregate based on different data types. When a specific data instance is published on one domain, it will not be received by subscribers residing on any other domains.

Multiple domains provide effective data isolation. One use case would be for a system to be designed whereby all Command/Control related data is exchanged via one domain while Status information is exchanged within another. Multiple domains are also a good way to control the introduction of new functionality into an existing system.

3.0 DETAILED INTERFACE REQUIREMENTS

3.1 Commanding Requirements

There are two basic classes of commands used : Lifecycle commands : commands used by OCS to control the lifecycle characteristics of applications. Users generally do not need to be concerned with the lifecycle commands because they are implemented by the underlying infrastructure.

Functional commands : commands that implement the specific functional characteristics of a subsystem components.

Functional operation is based on the Command/Action/Response model that isolates the transmission of the command from the resulting action that is performed. When an application receives a command, it validates any Configuration associated with that command and immediately accepts or rejects the command. If the command is accepted, the application then initiates an independent internal action to meet the conditions imposed by the command. Once those conditions have been met, an event is posted signifying the successful completion of the action (or the unsuccessful completion if the conditions can not be met).

Commands return immediately but the actions that are initiated as a result of a command may take some time to complete. When the action completes, an action status event is posted that includes the completion status of that action. The subsystem generating the command monitors this status event prior to issuing the command on the remote system. While the monitoring is performed automatically by the command system, Subsystem developers may need to attach a callback to perform processing on action completion. This callback may be null if no processing is needed.

If a command is accepted by the subsystem it causes an independent action to begin. A response to the command is returned immediately. The action begins matching the current configuration to the new demand configuration. When the configurations match (i.e., the subsystem has performed the input operations) the action signals the successful end of the action. If the configurations cannot be matched (whether by hardware failure, external stop command, timeout, or some other fault) the action signals the unsuccessful end of the action.

The important features of the command/action/response model are:

- **Commands are never blocked. As soon as one command is started, another one can be issued. The behavior of the controller when two or more configurations are started can be configured on a per subsystem basis.**

- **The actions are performed using one or more separate threads. They can be tuned for priority, number of simultaneous actions, critical resources, or any other parameters.**

- **Action completions produce events that tell the state of the current configuration. Actions push the lifecycle of the configuration through to completion.**

- **Responses may be monitored by any other subsystems.**

Generic subsystem control state commands

Command	Description
start	Prepare the subsystem to accept functional commands
stop	Shutdown functional commanding capabilities
pause	Suspend functional activities , internal state is retained
resume	Resume functional actitivies (if possible)
online	Set subsystem ready for commands
offline	Take subsystem offline (must be brought back online before any other commanding is possible)

camera.CCS Commands

Subsystem	Device	Property	ActionParameter	
camera.CCS	control	exposure	abort	*n/a*
camera.CCS	control	exposure	set	seconds

camera.CCS	control	exposure	start	*n/a*
camera.CCS	control	exposure	stop	*n/a*
camera.CCS	filter	position	select	n
camera.CCS	guider	acquire	disable	*n/a*
camera.CCS	guider	acquire	enable	*n/a*
camera.CCS	shutter	motion	disable	*n/a*
camera.CCS	shutter	motion	enable	*n/a*
camera.CCS	wfsacq	acquire	disable	*n/a*
camera.CCS	wfsacq	acquire	enable	*n/a*

camera.FCS Commands

Subsystem	Device	Property	ActionParameter	
camera.FCS	control	lock	off	*n/a*
camera.FCS	control	lock	on	*n/a*
camera.FCS	control	position	select	n
camera.FCS	locking	brake1	on	*n/a*
camera.FCS	locking	pin1	in	*n/a*

camera.GAS Commands

Subsystem	Device	Property	ActionParameter	
camera.GAS	control	algorithm	select	name

camera.GAS	control	interval	set	nnn
camera.GAS	readout1	active	off	*n/a*
camera.GAS	readout1	active	on	*n/a*
camera.GAS	readout1	exposure	set	milliseconds
camera.GAS	readout1	region1	clear	*n/a*
camera.GAS	readout1	region1	set	pixels

camera.SCS Commands

Subsystem	Device	Property	ActionParameter	
camera.SCS	control	lock	off	*n/a*
camera.SCS	control	lock	on	*n/a*
camera.SCS	control	position	close	*n/a*
camera.SCS	control	position	open	*n/a*
camera.SCS	control	profile	load	URI
camera.SCS	control	profile	query	*n/a*
camera.SCS	locking	brake1	on	*n/a*

camera.SDS Commands

Subsystem	Device	Property	ActionParameter	
camera.SDS	ccd1	amplifier1	flush	*n/a*

camera.SDS	ccd1	amplifier1	off	*n/a*
camera.SDS	ccd1	amplifier1	on	*n/a*
camera.SDS	ccd1	amplifier1	readout	*n/a*
camera.SDS	ccd1	amplifier1	reset	*n/a*
camera.SDS	control	fiber1	disable	*n/a*
camera.SDS	control	fiber1	enable	*n/a*
camera.SDS	control	flash	disable	*n/a*
camera.SDS	control	flash	enable	*n/a*
camera.SDS	control	obsid	set	name
camera.SDS	control	readout	set	off
camera.SDS	control	readout	set	on
camera.SDS	fiber1	speed	set	nnn

camera.T1S Commands

Subsystem	Device	Property	ActionParameter	
camera.T1S	valve1	flowrate	set	nnn
camera.T1S	valve1	position	close	*n/a*
camera.T1S	valve1	position	open	*n/a*

camera.TC Commands

Subsystem	Device	Property	ActionParameter	
camera.TC	control	algorithm	select	name
camera.TC	control	setpoint	set	temperature
camera.TC	zone1	interval	set	seconds
camera.TC	zone1	setpoint	set	temperature

camera.TCM Commands

Subsystem	Device	Property	ActionParameter	
camera.TCM	raft1	biases	load	URI
camera.TCM	raft1	clocks	load	URI

camera.VCS Commands

Subsystem	Device	Property	ActionParameter	
camera.VCS	control	algorithm	select	name
camera.VCS	line1	calib	load	URI
camera.VCS	line1	flowrate	set	nnn
camera.VCS	line1	pump	off	*n/a*
camera.VCS	line1	pump	on	*n/a*

3.1.1 Interface Processing Time Requirements

Command messages issued by OCS must be received by the computer system(s) of the commanded subsystem within TDB ms. A preliminary response (ACK) must be issued within TBD ms and received by OCS within TDB ms of the command origination time.

3.1.2 Message Requirements

Every stream includes items for consistency checking and performance monitoring support

Identifier	Description
private_revCode	crc of IDL source
private_sndStamp	system time of sender
private_rcvStamp	system time of receiver
private_seqNum	sequence number (process)
private_origin	IP subaddr and PID

3.2 Telemetry Requirements

Telemetry datastream topic : camera.Dewar_Cooler/Heater

This topic records application level data for the dewar heating and cooling systems. Target and actual statuses health , limits, etc.

Item	Type	Size	Freq.	Notes
Raw	Int	16	1	None
Calibrated	Float	32	1	None
Statistics	Float	8	1	None

Status	Byte	16	1	None

Telemetry datastream topic : camera.Electrical

Electrical monitoring for devices located in the camera subsystem.
Raw data, calibrated voltages, calibrated current, device power status.

Item	Type	Size	Freq.	Notes
Raw	Int	16	0.1	None
Calibrated	Float	32	0.1	None
Statistics	Float	96	0.1	None
Status	Byte	16	0.1	None

Telemetry datastream topic : camera.Filters

This topic records filter positioning and motion information
as well as lower level filter mechanism parameters.

Item	Type	Size	Freq.	Notes
Raw	Int	16	0.1	None
Positions	Int	32	0.1	None
Status	Byte	16	0.1	None

Telemetry datastream topic : camera.Guide_sensors

This topic records metadata concerning the state of the guide regions, and the results of the processing of the subimages. Items such as bax pixel counts, H/V profiles profile fit results etc.

Item	Type	Size	Freq.	Notes
Metadata	Int	8	5	None
Profiles	Float	64	5	None
Fits	Float	8	5	None
Centroids	Float	2	5	None
Status	Byte	4	5	None

Telemetry datastream topic : camera.Metrology

Position control for sensors located in the camera subsystem.
Raw sensor readings, calibrated positions, limit switches, status bits.

Item	Type	Size	Freq.	Notes
Raw	Int	16	100	None
Calibrated	Float	16	100	None
Limits	Byte	64	100	None
Status	Byte	64	100	None

Telemetry datastream topic : camera.RNA

This subsystem maintains data from each RNA unit. This will include performance characteristics, health checks, statuses etc. Camera Raft Modules consist of astronomical quality optical imaging sensors and the mechanical structure that holds them at their precise locations. The mechanical mounting system incorporates interfaces to the structural, thermal, vacuum, and electronics subsystems of the enclosing cryostat.

Arrays of 3Ã—3 sensors are mounted to raft structures that in turn mount to the integrating structure. Raft structure plates are double side lapped all at once to be flat, parallel and the same size to less than 1 Âµm. Thus, rafts assembled with sensors are flat, parallel and the same height to 6.5 Âµm peak to valley. The raft structure and the sensor substrate have a well matched coefficient of thermal expansion and good thermal conduction to maintain this flatness performance during operation, since the assembly is subject to large temperature changes (ambient to operating temperature).

Item	Type	Size	Freq.	Notes
Raw	Int	16	10	None
Calibrated	Float	32	10	None
Statistics	Float	96	10	None
Status	Byte	16	10	None

Telemetry datastream topic : camera.Science_sensor_metadata

This topic records the science ccd metadata. Items such as chip voltages, biasaes, health, per-chip temps etc.

Item	Type	Size	Freq.	Notes
Raw	Int	16	0.1	None
Calibrated	Float	32	0.1	None
Statistics	Float	96	0.1	None
Status	Byte	16	0.1	None

Telemetry datastream topic : camera.Shutter

This topic records shutter positioning and motion information as well as lower level shutter mechanism parameters.

Item	Type	Size	Freq.	Notes
Raw	Int	1024	1	None
Position	Float	32	1	None
Status	Byte	16	1	None

Telemetry datastream topic : camera.Temps

Temperature sensing information for sensors located in the camera subsystem. Raw sensor readings, calibrated temperatures, time-series statistics, sensor health.

Item	Type	Size	Freq.	Notes
Raw	Int	16	0.1	None
Calibrated	Float	16	0.1	None
Statistics	Float	48	0.1	None
Health	Byte	16	0.1	None

Telemetry datastream topic : camera.Vacuum

This topic records application level data for the dewar vacuum systems. Target and actual statuses health , limits, etc.

Item	Type	Size	Freq.	Notes

Raw	Int	16	0.1	None
Calibrated	Float	32	0.1	None
Statistics	Float	8	0.1	None
Status	Byte	16	0.1	None

Telemetry datastream topic : camera.Wavefront_sensors

This topic records metadata concerning the state of the wavefront sensors, and the results of the processing of images. Items such as chip voltages, health, per-chip temps bad pixel/line/column counts, image pair counts, zernike results and so on.

Item	Type	Size	Freq.	Notes
Metadata	Float	32	0.1	None
Results	Float	96	0.1	None
Status	Byte	16	0.1	None

3.3 Notifications Requirements

Any application may post notifications and/or subscribe to notifications posted elsewhere. The notification service is robust and high performance. A notification consists of a topic and a severity. A sequence of notifications with the same topic is referred to as an event.

The topic is used to identify publishers to subscribers. The severity may be used as a filter by notification subscribers.

The notification service has the following general properties: An notification topic represents a many to many mapping: notifications may be posted to the topic from more than one source and received by zero or more targets. (Typically, however, most topics will have a single source.)

Notifications posted by a single source into an notification topic are received by all targets in the same order as they were posted.

Delivery of notifications to one subscriber cannot be blocked by the actions of another subscriber. An notification stream is an abstract concept: a subscriber may subscribe to an notification stream using a wildcarded name in which case the notifications it receives are the merging of all published notifications whose names match that wildcarded name.

Notification are not queued by the service. A late subscriber will not see earlier notifications.

The service does not drop notifications. A published notification will be delivered to all subscribers.

The notification service supports arbitrary notification topics.

Notifications are automatically tagged with the source and a timestamp.

3.4 Communication Methods

Initiation : NDDS discovery

The process by which domain participants find out about each others entities Each participant maintains database on other participants in the domain and their entities happens automatically behind the scenes (anonymous publish-subscribe)

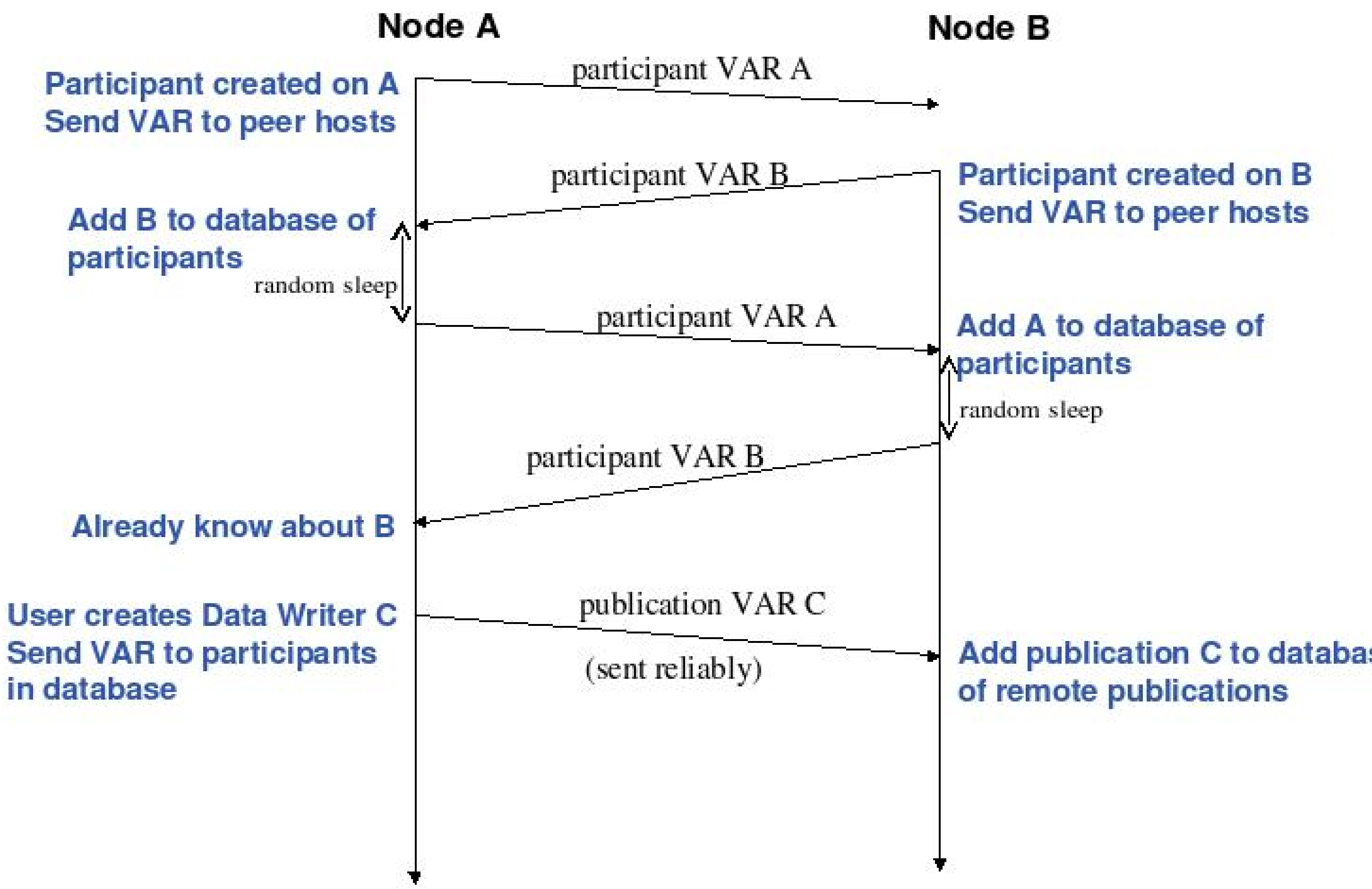
- **Does not cross domain boundaries**

- **Dynamic discovery**
- **Participants must refresh their presence in the domain or will be aged out of database**
- **QoS changes are propagated to remote participants**
- **Two consecutive phases**
- **Participant discovery phase**
- **Participants discover each other**
- **Best-effort communication**
- **Endpoint discovery phase**
- **Participants exchange information about their datawriter and datareader entities**
- **Reliable communication**
- **Steady state traffic to maintain liveliness of participants**
- **Participants periodically announce their presence using RTPS VAR message**
- **Contains participant GUID, transport locators, QoS**
- **Initially sent to all participants in initial peers list, then sent periodically to all discovered participants**
- **Sent using best-effort**

DataWriter/DataReader discovery

- **Send out pub/sub VAR to every new participant**

- **NACK for pub/sub info if not received from a known participant**
- **Send out changes/additions/deletions to each participant**
- **Uses reliable communication between participants**
- **Data Distribution Service matches up local and remote entities to establish communication paths**



Discovery is implemented using DDS entities known as Built-in Data Writers and Built-in Data Readers

- **Uses same infrastructure as user defined Data Writers/Data Readers**
- **Participant data is sent best effort**
- **Publication/subscription data is sent reliably**

Three Built-in topics (keyed):

- **DCPSParticipant**
- **DCPSPublication**
- **DCPSSubscription**

Each participant on the same host and in the same domain requires a unique participant index

For given domain, participant index determines port numbers used by the participant

```
      Unicast meta-traffic:
7400 + ((100 * participant_index) + domain) * 10
Multicast meta-traffic:
7400 + (domain * 10) + 2
Unicast user-traffic:
7400 + (((100 * participant_index) + domain) * 10) + 3
Multicast user-traffic:
7400 + (domain * 10) + 1
```

Flow Control : NDDS topics

Topics provide the basic connection point between publishers and subscribers. The Topic of a given publisher on one node must match the Topic of an associated subscriber on any other node. If the Topics do not match, communication will not take place.

A Topic is comprised of a Topic Name and a Topic Type. The Topic Name is a string that uniquely identifies the Topic within a domain. The Topic Type is the definition of the data contained within the Topic. Topics must be uniquely defined within any one particular domain. Two Topics with different Topic Names but the same Topic Type definition would be considered two different Topics within the DDS infrastructure.

3.5 Security Requirements

Message timestamps

Message integrity is enhanced by the inclusion of egree-time and arrival time (local system clocks) field in every topic (command , notification, and telemetry). The SAL software automatically performs validation to ensure early detection of clock slew or other time related problems.

Software versioning checksums

Communications consistency and security is supported by the inclusion of CRC checksum fields in every topic definition (command , notification, and telemetry). The SAL software automatically checks that the publisher and subscribers are running code generated using identical (at the source code level) topic definitions. This prevents problems associated with maintaining consistent inter-subsystem interfaces across a widely distributed software development infrastructure.

4.0 QUALIFICATION METHODS

System dictionary

A systemwide dictionary of all subsystems, devices, actions and states is maintained. All the interactions between subsystems are automatically checked to verify that only objects defined in the dictionary can be used or referenced.

Command definition database

A database of permissible commands is maintained on a per subsystem basis. The database references the system dictonary and contains 1 record per command. Each command is constrained in terms of target subsystems, parameter ranges, maximum frequency, timeout, pause/hold potential,and failure severity.

The database is used to automatically generate application level code to perform all command level interactions. This code is thus guaranteed to be consistent system wide.

Telemetry datastream Definition database

All telemetry datastream definitions are stored in a database. Each definition is automatically verified for compliance with the system dictionary. Telemetry items are detailed in terms of type, size, frequency, units, and value ranges. Any item with a physical correlate has an SI unit associated with it.

The database is used to automatically generate application level code to perform all datastream topic references. This code is thus guaranteed to be consistent system wide.

Code generation

The primary implementation of the software interace described in this document will be automatically generated. A software abstraction layer (SAL) provides a standardized wrapper for the low-level OMG DDS functionality which provides the transport later.

The permissible commands, datastream contents, and issuable alerts are all defined by the controls system database and their nomenclature is controlled by the system dictionary. All intersubsystem messages formats are autogenerated. Low level data transfers include versioning checksums based on the source level record definition.

Testing

Test servers and clients are generated which implement the full set of commands, datastreams, and notifications are defined by the controls system database. Tests may be configured for a variable number of servers/clients and automatically monitored to ensure compliance with bandwidth and latency requirements. All test results are archived to the facility database for future examination.

5.0 NOTES

6.0 APPENDICES

7.0 APPROVALS

8.0 RECORD OF CHANGES

Version	Date	Author	Notes
1.0	December 12 2008	Mills., D.	Initial release