R1/Event Data Model Specification

	Name	Date	Signature
Prepared by	K. Kosack (CTAO, CEA Saclay), I. Oya (CTAO)		
Approved by	I. Oya (CTAO)		
Released by	S. Schlenstedt (CTAO)		

This Version:					
Issue	Rev.	Created	Comment	Ownership	Distribution
2	a	2023-08-10	Implement CTA-CRE-ACA-303000-0010 and CTA-CRE-ACA-303000-0011.	CTAO Computing	CTA

Change Log:				
Issue	Rev.	Created	Reason / Remarks / Initiation	Part Affected
1	g	2022-01-24	Fix typo in pixe_status data type in fig 3.2	
1	f	2021-12-03	Update hardware_stereo_trigger_mask information according to CTA-CRE-COM-000000-0002	
1	e	2021-02-12	A number of typos reported in Redmine were fixed. *coeff_stddev* type was corrected to float64 in fig 3.4. Optional field *num_modules* was added to CameraConfiguration	
1	d	2020-12-02	Updated after discussion with all stakeholders during September, October and November via teleconferences and RedMine forums.	
1	c	2020-04-29	Minor update to fix a few typos before the ACADA PDR. Same open points as in 1b apply.	
1	b	2020-03-18	Release version (for expert review). Comments from experts on version 1a addressed (except the open points)	
1	a	2020-01-23	circulated with Camera and ACADA experts	
draft	0.3	2019-10-08	addressed issues from Project Scientist and CTAO Computing	
draft	0.2	2019-08-02	circulated to Project Scientist and CTAO Computing	
draft	0.1	2019-07-03	split from old Raw Telescope Data Interface document	

List of Contributors:		
Contributor	Affiliation	Contribution Subject / Chapter
Karl Kosack	CTAO	
Igor Oya	CTAO	
Matthias Füßling	CTAO	

Table of Contents

1 Scope

This document specifies the model of the event-wise on-site raw data that is streamed from the CTA Telescopes (Cherenkov Cameras) to the ACADA system. This data category is classified as ***R1/Event*** in the CTA data model [4]. Additionally, the document covers some aspects of the ***R1/Monitoring*** data model, which are associated with the monitoring of the Camera pre-calibration process and event sanity. Other aspects of the Camera ***R1/Monitoring*** or ***R1/Service*** data are not covered in this document.

2 Context

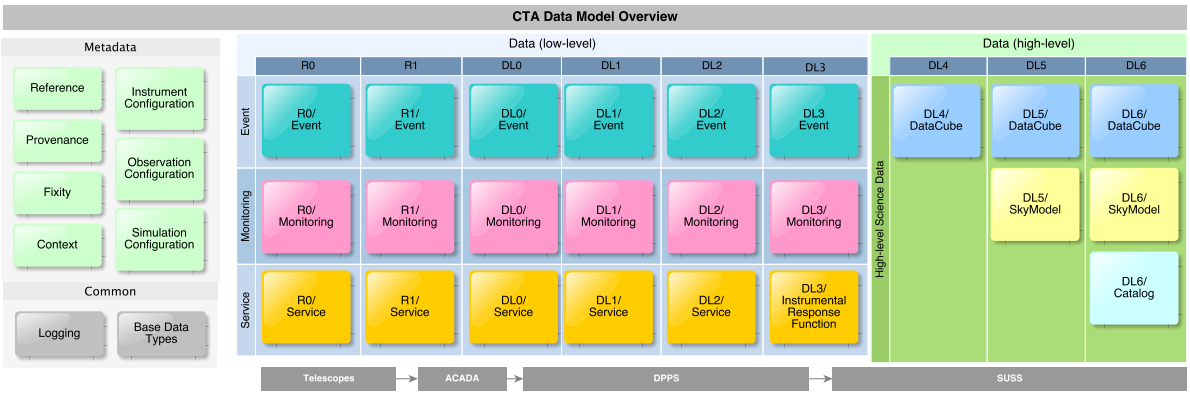


Figure 2.1 – Overview of the CTA Data Model (taken from [4]). This document pertains only to the box labeled ***R1/Event***; other models are described in separate documents.

This document describes the ***R1/Event*** data model, which is a subset of the overall CTAO data model shown in Figure 2.1.

The ***R1/Event*** data model is part of the main data interface between Cherenkov Telescopes and Array Control and Data Acquisition (ACADA) as shown in Figure 2.2. Cameras send to ACADA trigger timestamps, then ACADA sends back to Cameras trigger confirmations, and then finally Cameras send event data of the confirmed triggers to ACADA. Additionally, *debugging* events—which are events containing regular event data and additional fields useful for troubleshooting—can be transmitted to ACADA when the situation requires (see Sec. 3 for further details). The Camera configurations and the monitoring information of the calibration coefficient set applied to the R1 event data are also transferred from the Cameras to ACADA. The mechanisms used to transfer data from the Cameras to ACADA will be specified in a Camera–ACADA Interface Control Document (ICD).

This data model contains the following sub-models:

R1/Event/Telescope the data model associated with Telescope Triggering (Camera) and Camera Events

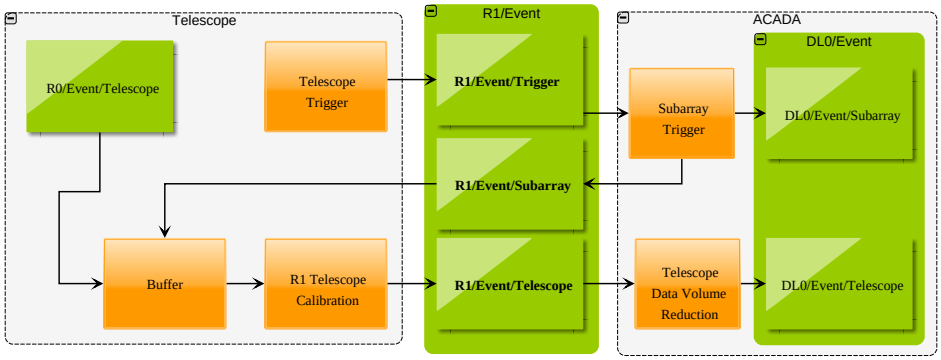


Figure 2.2 – Simplified data flow between Telescopes and ACADA, showing data streams (in green) and processing steps applied to them (orange). A more detailed view is given in [2]. For more information on the timing of this process, see Section A.2

from a single Cherenkov Telescope.

R1/Event/Subarray the data model for Array-Level Events associated with a Sub-array of Telescopes; in this case, this pertains only to the Sub-array Trigger response to the Telescopes.

R1/Monitoring/Calibration the data model of the monitoring information associated with the R1 data pre-calibration happening at the Camera server.

R1/Monitoring/Event/Debugging the data model of the monitoring information associated with debugging information for events with issues.

2.1 Requirements

The R1 data model is based on the following requirements (see linked Jama pages for up-to-date versions) [1]:

- A-OBS-1120 Reproduceability of Data Processing: Prior versions of data processing software, calibration coefficients and reference/ look-up tables must be preserved, ensuring that data processing procedures are reproduceable.
- B-TEL-1250 Pixel Readout: The Camera must be able to deliver signal amplitude information from all camera pixels in each event together with all available time domain information, unless it can be shown that for a given partial readout scheme, using end-to-end simulations, that the achievable sensitivity is not reduced by more than 3%. Incomplete or mixed Events will be dropped and count towards dead-time and/or availability.
- B-TEL-1370: Pedestal Subtraction: During Observations the Camera must measure the Pedestal in each pixel and the event-to-event RMS of this quantity with an uncertainty no greater than 20% of the event-to-event RMS or 1.2 photons (if greater). The Pedestal must be subtracted from the data prior to delivery to ACADA (R1 level) and the Pedestal value being subtracted made available to ACADA at least once every 2 seconds as part of the monitoring stream.
- B-TEL-1610: R1 Event Data: Camera Event Data delivered to ACADA (R1 level) must be time-ordered and use a compact common format, occupying at most 16-bits per sample time-slice for data-cube information.
- B-TEL-1630: R1 Intensity Resolution: It must be possible to extract a signal estimate from the R1 data delivered to ACADA, using previously determined per-pixel coefficients, to reach the Intensity resolution shown in the figure below and table attached, degraded with respect to B-TEL-1010.

[1] We replaced OES with ACADA everywhere in the requirement texts.

- B-TEL-1640: R1 Time Uncertainty: It must be possible to extract pixel signal timing information from the R1 data delivered to ACADA, using previously determined per-pixel coefficients, such that the time resolution requirement (B-TEL-1030) relaxed by a factor of 1.5 is satisfied.

- B-TEL-1650: R1 Trigger Time Information: In addition to an absolute time derived from the system wide precision time distribution system, the (R1) Event data sent to the ACADA must contain an event sequence number and/or NTP-based event time identifier, to be used for monitoring and diagnostic purposes.

- B-ACADA-0125 Readout Request: ACADA shall send to each Camera in a sub-array lists of the Trigger Timestamps from that Camera, indicating for which Events data should be sent to ACADA for further processing, with a maximum latency of 0.8 seconds from the arrival of a given Camera Trigger Timestamp at ACADA

- B-ACADA-0130: Trigger Information: ACADA shall collect and store all Trigger Timestamp information for sub-array coincidences, including the ability or intention of all Cameras to deliver camera data to ACADA for a given Event

2.2 Preconditions

At the R1 level, all Telescopes shall have already applied some basic and *reversible*[2] calibration, as outlined in the requirements. This calibration need not meet the full CTA offline systematics or precision requirements, and is intended to facilitate the ACADA Science Alert Generation Pipeline (SAG), which contains *data volume reduction* and on-line shower reconstruction. The final image should be calibrated such that strong signal pixels can be identified with no extra calibration steps.

This will include:

- The reduction of effects of ADC inhomogeneities in time (ring-sampler corrections), when applicable to the Camera type. These corrections do not need to be reversible.

- The subtraction of a rough electronic + NSB noise pedestal value for each pixel, such that the values are in units roughly equivalent (within calibration systematics and numerical precision limitations) to Cherenkov light signal photo-electrons (PE), with 0 standing for 0 PE.

- The selection of the appropriate channel if multiple gain channels exist such that the readout value is within a linear range (marking the pixel as saturated if no channel is in the linear range)[3].

- The application of a gain linearization function if a non-linear gain is used.*Note: This does not imply that the gain has to be made linear over the full range. The linearization is intended for small pulses, so that it is possible distinguish signal from noise. For very bright pulses, even if non-linear, we will easily detect them and not remove them during DVR.*

- The application of a pixel-wise multiplicative gain correction factor to correct for differences in camera uniformity between pixels (flat-fielding).

- The application of a global (not per-pixel) *scale* and *shift* to the signals so that they efficiently fit in a 16-bit unsigned integer value with the required dynamic range (see Equation 2.1). These values are readout-hardware-dependent, i.e. they depend on the ADC bit-depth and amplification range of a particular Camera, and do not change in time. Note that using unsigned values plus a scale and shift (rather than a signed integer) allows for choosing an asymmetric dynamic range where positive values use more bits than negative, which is what we want since the negative fluctuations are small compared with the signal in general.

- The recording of all applied corrections as service ***Service*** data, or (***Monitoring***) streams (see also [4]). Such recording will be in floating point format, and contain at least per-pixel multiplicative and subtractive factors of the below equation, in order to ensure reversibility.

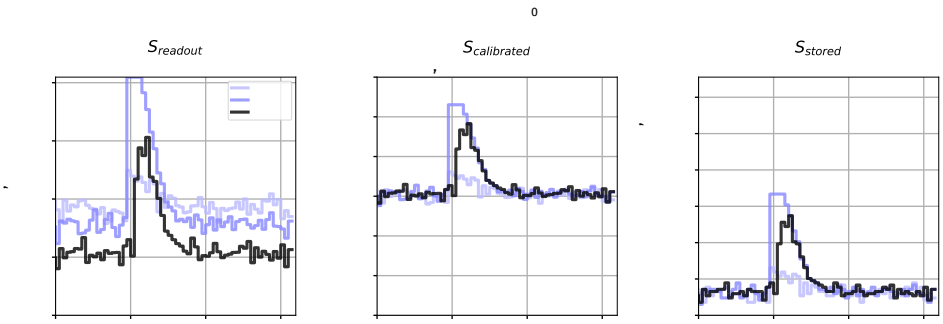


Figure 2.3 – Example of R1 calibration steps. The left plot shows the signal as read-out by the camera, with the pedestal level changing in time due to e.g. NSB or temperature variations. The middle figure shows the signal after R1 calibration has been applied, and the right figure shows how this can be packed into a uint16. The main purpose of these steps is to ensure the signal offset (Δ) is constant, and that signals from all amplifier gains are in the same scale.

This can be summarized in Figure 2.3 and in the following equations, where P is a rough (e.g. integer) noise pedestal measurement, G is a pixel gain-correction value hat allows both gain channels to be comparable, and Δ, κ are the shift and scale applied to the signal to fit it efficiently into a 16 bit integer with more dynamic range used for positive signals:

$$
\begin{aligned}
S_i^{\text{calibrated}} &= \begin{cases} (S_{0,i}^{\text{readout}} - P_{0,i})G_{0,i} & channel = 0 \\ (S_{1,i}^{\text{readout}} - P_{1,i})G_{1,i} & channel = 1 \end{cases} \\
S_i^{\text{stored}} &= \begin{cases} \left(S_{0,i}^{\text{calibrated}} + \Delta_0\right)\kappa_0 & channel = 0 \\ \left(S_{1,i}^{\text{calibrated}} + \Delta_1\right)\kappa_1 & channel = 1 \end{cases}
\end{aligned}
\tag{2.1}
$$

In the previous equation S_i^{stored} is assumed to be positive, with values increasing with increasing readout values. This implies that certain Cherenkov Camera types may need to flip the readout values, or to include G as a negative number.

[2] A-OBS-1120: Reproduce-ability of Data Processing
[3] it will however be possible to read out and store both gain channels for specialized calibration runs

3 Data Model

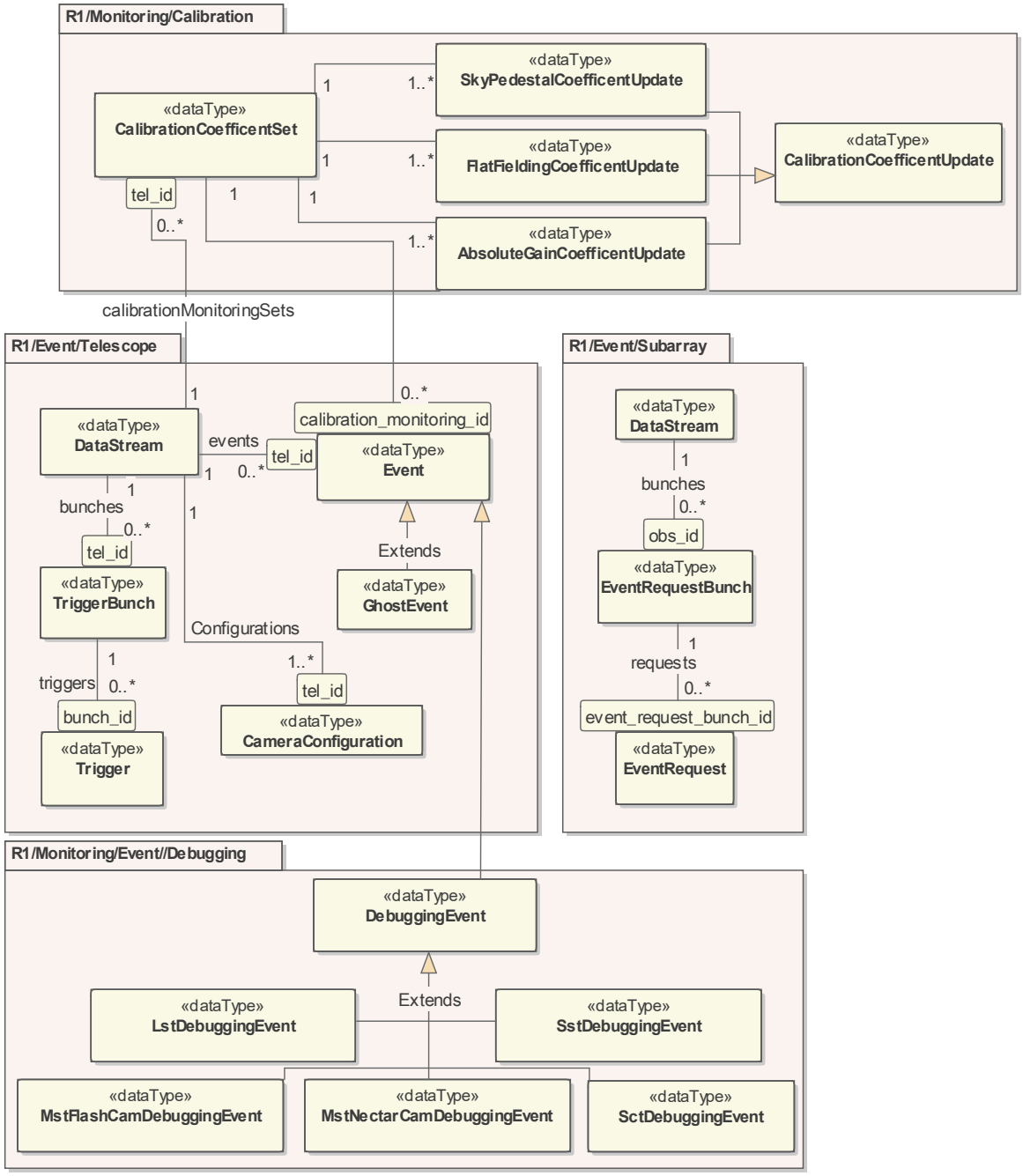


Figure 3.1 – Overview of the data model presented in this document, in UML form. Keys linking data are depicted as *association qualifiers* (the small rectangles attached to the larger rectangles indicating data types).

Figure 3.1 provides an overview of the information that all Telescopes and Monte-Carlo Simulations will generate during normal operations, for each event, the information exchanged with the Sub-array Trig-

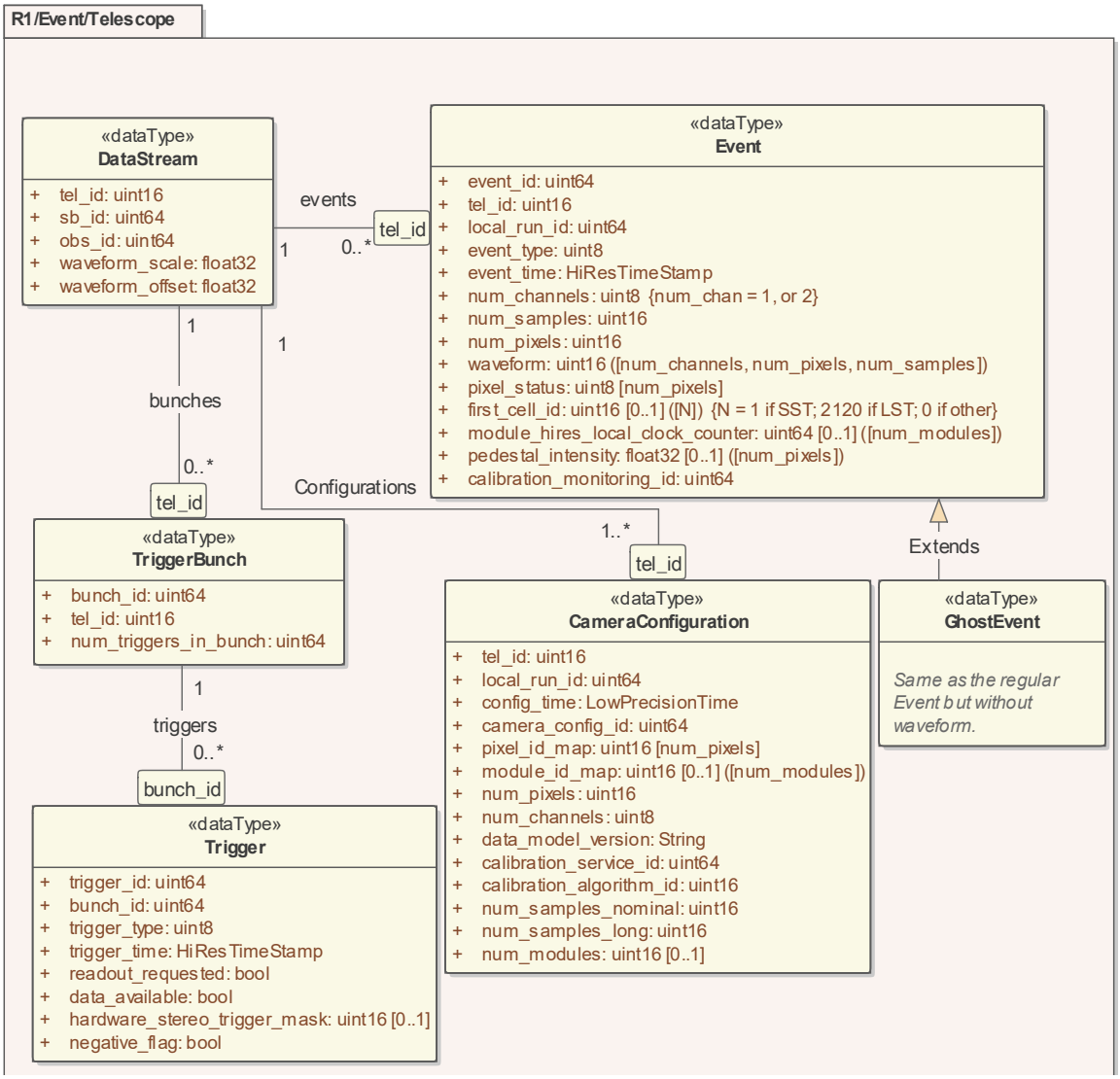


Figure 3.2 – ***R1/Event/Telescope*** data model, in UML form, not including any Camera-specific attributes. *optional* fields are indicated as fields with [0..1] multiplicity, arrays are specified within parenthesis, and constraints within curly brackets. For example, "`+ first_cell_id: uint32 [0..1] ([N]) { N = 1 if SST; 2120 if LST; 0 if other }` " meas that the field *first_cell_id* is optional, is an array of 32 unsigned integers of size N, N being either 1 for SSTs or 2120 for LSTs.

ger, and associated monitoring information. Figures 3.2, 3.3 and 3.4 provide the details of the data model for Camera events (***R1/Event/Telescope***), sub-array level trigger ***R1/Event/Subarray***, and Camera pre-calibration monitoring ***R1/Monitoring/Calibration***, respectively. Section A.3 provides a description of each field of this data model. The data model of the debugging events (***R1/Monitoring/Event/Debugging***) is not further detailed in this document since it is Camera type specific.

Extra telescope-specific information (such as trigger sector hit patterns) may be added to this upon discussion, however *it is not guaranteed to be stored later in the **DL0/Event/Telescope** output* due to data volume limitations; nevertheless samples or averages of these telescope-specific fields may be transmitted to the ACADA monitoring system as ***DL0/Monitoring/Telescope*** data, or event-wise in short specialized calibration runs (in which case a specialized subclass of Event will be added to the data model to reflect the additional information). Any telescope-specific information that has to be transmitted to ACADA must be specified in an ICD.

The `waveform` is the main data readout per pixel of a Camera, and is an array of time samples relative to the `event_time` (see Figure A.2 for details). Note that the telescope *trigger_time* is the main information sent to the Sub-array Trigger to form time coincidences (see Figure A.3). Once this subarray event is formed, the Sub-array Trigger then assigns an `event_id` and sends a set of ***EventRequestBunches*** to

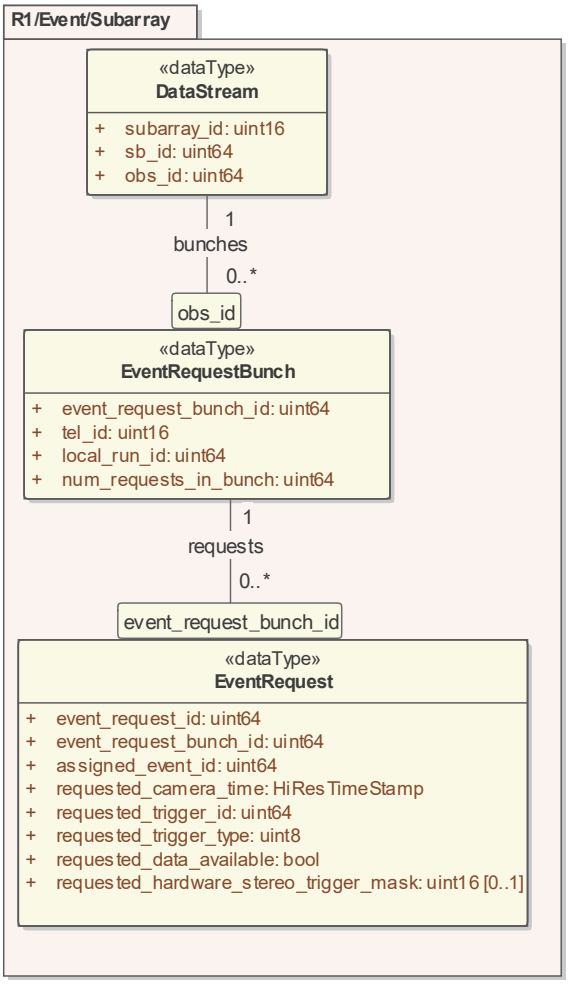


Figure 3.3 – ***R1/Event/Subarray*** data model, in UML form.

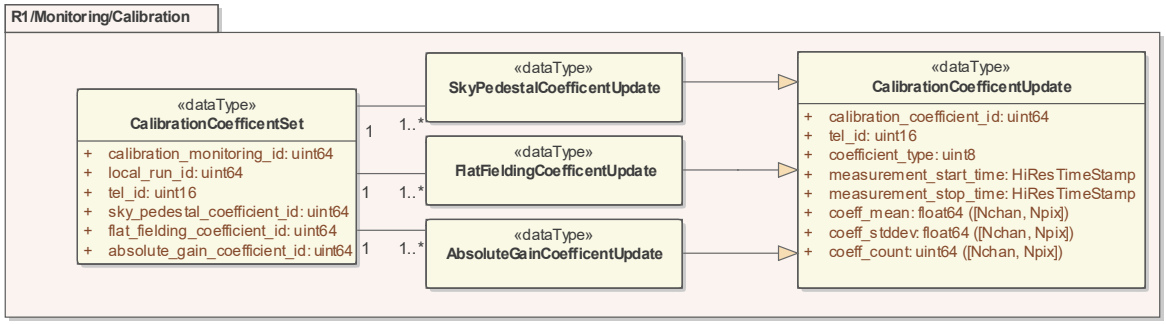


Figure 3.4 – ***R1/Monitoring/Calibration*** data model, in UML form, not including any Camera-specific attributes.

each Camera.

In the data model specified in Figure 3.2, the following aspects are considered:

- N_{pix} is a fixed-length (at the R1 level), therefore information for all pixels is always read out, with either random or zero data in pixels with no or faulty signal. In most cases N_{chan} should be 1, assuming the Camera has either a single gain channel or has applied gain channel selection and marked which channel has been applied in `pixel_status`, however for certain event types such as

calibration events (see below), both channels may be read out on Cameras that have two.

- At the R0 level, there may be multiple time-stamps, e.g. from a central location and the local Camera server. At the R1 level and beyond, we assume the correct time stamp has been chosen and any corrections have been applied. CTAO will define the origin of the telescope event time-stamp recorded here.
- For Cameras that additionally compute pedestals per event, this data model currently assumes those are used only for monitoring of the R1 pedestal subtraction and are not stored, however a `pedestal_intensity` optional field exists in the event data structure, and moreover pedestal-type events may be used to optionally store these.
- the `trigger_type` field allows us to differentiate various ways the Telescopes are read out. This includes not just Cherenkov triggers (on real events and muons, for example), but also interleaved calibration events (e.g. a trigger from a pedestal pulser, or a flat-fielding flash). In practice, these may even get written to different streams.
- A unique (within an Observation Block) `event_id` is assigned by (and only by) the Sub-array Trigger, even in the case of an Internal Trigger from a calibration source or other non-stereo trigger type, where the telescope sets the `readout_requested` bit to *true*. This assigned id is sent back to the Cameras in the *EventRequest*.
- Cameras may self-trigger on a muon-like event or other calibration source, and should set the `readout_requested` flag in the ***Trigger*** message, which signals to the Sub-array Trigger that the event should be assigned an `event_id` and included in an ***EventRequest***, regardless if it participated in a stereo trigger. *Note: This means that the Camera just needs to apply some rough pixel multiplicity cut to flag events that might be muons. The muonness, muon ring parameters and subsequent optical efficiency parameters are estimated off-line in the Cat-B and Cat-C pipelines of the DPPS.*
- Cameras assign a unique identifier `local_run_id` every time a data acquisition operation is started. This identifier is local for each Camera and is not to be confused with Observation Blocks, which are identifiers assigned by ACADA at sub-array level.
- The `calibration_monitoring_id` at `Event` points to the calibration coefficient set used for the pre-calibration of this particular event. The calibration coefficient set data model is specified in Figure 3.3.
- LST and NectarCam are able to send to ACADA the two gains of calibration and pedestal events (event types 0, 1, 2, see Sec. A.4.3) in the same data structure. Whether to send one or both gains is decided at runtime by the Camera. This is encoded in this data model by using the field `num_channels`. When `num_channels` is 1, a single-gain event with a `waveform[1, num_pixels, num_samples]` is sent. When `num_channels` is 2, a waveform with a shape of `waveform[2, num_pixels, num_samples] is sent, corresponding to a two-gain event`. For the latter case, index 0 corresponds to high gain, and 1 to low gain.
- FlashCam and SST are able to send to ACADA shower candidate events of two different lengths, *nominal* and *long* (event types 32 and 33 in Sec. A.4.3, respectively). Which kind of event to send to ACADA is decided at runtime by the Camera. This is encoded in this data model by using the field `num_samples`. Regular events of nominal length are sent with `num_samples = num_samples_nominal` (see `CameraConfiguration`). Long events are sent with `num_samples = num_samples_long` (see `CameraConfiguration`). Despite this data model being compatible with long-two-gain events, we note that in the current CTA configuration, those cameras requiring long events do not use two gains and vice-versa.
- The LST cameras can send *ghost* events to ACADA. These are events with the same fields as the `Event` data structure in 3.2, but that do not include `waveform` data or related fields like `n_pixels`. `GhostEvents` permit the application of certain offline corrections[1] to the LST data in DPPS, and for the reversibility of the pre-calibration.

[1] Whereas most of the issues in the data created by inhomogeneities in the LST Cameras DRS4 can be fixed online, a few of them cannot, and may need to be (re)applied in DPPS to optimize the sensitivity.

- The `CameraConfiguration` data structure specified in Fig 3.2 is the sub-set of the Camera configuration settings needed by ACADA and DPPS to decode the event data. The complete Camera configuration is Camera-type specific, and to be specified in an ACADA–Camera ICD.
- The debugging event types indicated in Fig. 3.1 are events used for problem identification and troubleshooting. They can be transmitted to ACADA in two ways: (i) as an alarm, indicating an faulty condition during regular operations, and as a continuous stream of *debugging* data activated via a Camera debugging mode.

A Details

A.1 R1/Event Data Flow

Figure A.1 shows an overview of the R1/Event data flow between a Cherenkov Camera and ACADA.

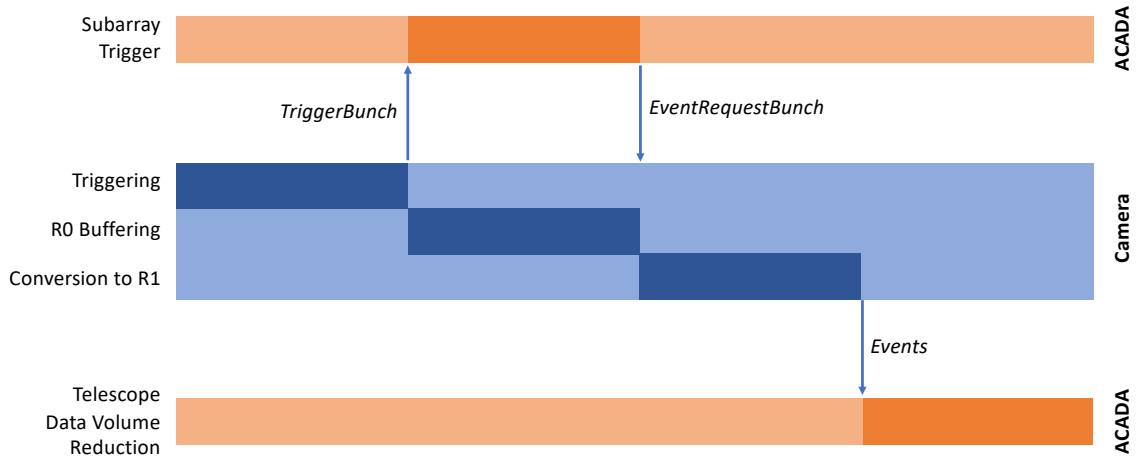


Figure A.1 – Overview of Camera and ACADA R1/Event data exchanges. Camera submits trigger timestamps to ACADA in ***TriggerBunches*** to limit the amount of small data packets sent over the network. The sub-array trigger at ACADA gathers and compares camera triggers from all the Cameras of the sub-array, and then sends event request in ***EventRequestBunches*** to individual Cameras. Cameras convert R0 data into R1, which includes the pre-calibration step. Finally, the camera event data corresponding to the ***EventRequestBunch*** is sent to the Array Data Handler of the ACADA system.

A.2 Triggers and Timing

Figure A.2 shows the relationships between the timestamps and constants used in the data model. Figure A.3 shows in more detail the concepts of a stereo trigger (multiple telescopes that trigger within a time window) and an event bunch (multiple events from a single telescope, grouped together for data transfer purposes).

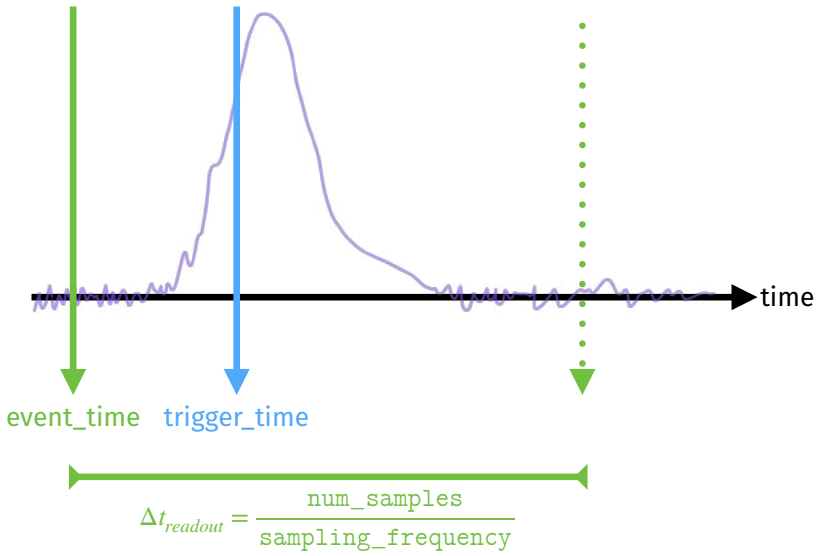


Figure A.2 – Trigger timing (simplified). This shows the relationship between the trigger time, i.e. when a Telescope decides that something interesting has happened that merits readout, and the event time, which is what is later the start of the readout window after an ***EventRequest*** is received.

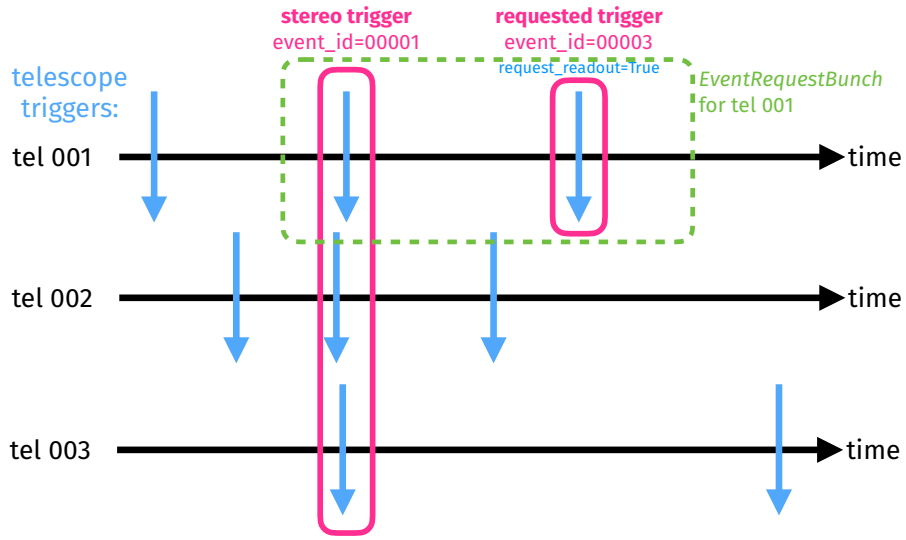


Figure A.3 – Sub-array Trigger timing (simplified). This example shows triggers from several Telescopes being formed into a stereo-trigger by the Sub-array Trigger, which collects multiple ***EventRequests*** into a single ***EventRequestBunch*** for each telescope. It then sends the ***EventRequestBunch*** to the telescope for readout.

A.3 Data Model Definitions

A definition of the data model elements presented in this document is provided in Tables A.1 and A.3.

Table A.1 – R1/Event/Telescope data definitions.

Name	Type	Description
Event		
event_id	uint64	The event ID assigned by the subarray trigger (*primary key*).
tel_id	uint16	The ID of the Telescope whose Camera sent the event.
local_run_id	uint64	A local unique identifier assigned by the Camera. Not to be confused with the Scheduling and Observation Block IDs assigned by ACADA (not included in this data model).
event_type	uint8	The type of event, see definition in Sec. A.4.3.
event_time	HiResTimeStamp	High-precision timestamp assigned by the Cherenkov Camera.
num_channels	uint8	The number of channels in this event.
num_pixels	uint16	Number pixels.
num_samples	uint16	Number of samples for each pixel.
waveform	uint16 [num_channels, num_pixels, num_samples]	R1-calibrated time-sampled image series in photoelectrons. Note: It is also possible to store images in DCs when the associated pre-calibration coefficients are set to 1.
pixel_status	uint8 [num_pixels]	See definition in Sec. A.5.
first_cell_id	uint16 [N]	The cell id of the first cell in the ring-sampler that is read out (*optional*, N is 1 for SST, and 2120 for LST (265 modules * 8 chips/module))
module_hires_local_clock_counter	uint64[265]	The 133 MHZ counter of from the camera modules (Optional).
pedestal_intensity	float32 [num_pixels]	Pedestal (baseline) intensity in each pixel in DC, and potentially additional information from the baseline. (*optional*).
calibration_monitoring_id	uint64	ID of the CalibrationMonitoringSet containing the applied pre-calibration parameters (*foreign key*).
GhostEvent		
Same as the regular Event with without waveform and num_channels.		
DataStream		
tel_id	uint16	The ID of the Telescope providing this data stream (*primary key*).
sb_id	uint64	The ID of the Scheduling Block to which this stream is associated to.
obs_id	uint64	The ID of ID of the current Observation Block (*primary key*).
waveform_scale	float32	Global scale to fit in the dynamic range.
waveform_offset	float32	Global shift to fit in the dynamic range in PE.
TriggerBunch		
bunch_id	uint64	The bunch ID assigned by the Cherenkov Camera (*primary key*).
tel_id	uint16	The ID of the telescope whose Camera sent the trigger bunch (*foreign key*).
num_triggers_in_bunch	uint64	The number of triggers in the bunch.
Trigger		
trigger_id	uint64	The trigger ID assigned by the Camera trigger (*primary key*).
bunch_id	uint64	The bunch ID assigned by the Cherenkov Camera (*foreign key*).
trigger_type	uint8	The type of trigger, see definition in Sec. A.4.1
trigger_time	HiResTimeStamp	High-precision timestamp assigned by the Cherenkov Camera.
readout_requested	bool	True if Camera is requesting to readout this event irrespective of array coincidences (e.g. a muon).
data_available	bool	False if event data are not available for that particular trigger (busy trigger).
hardware_stereo_trigger_mask	uint16	The hardware trigger mask (*optional*). See Sec. A.4.2 for the definitions of the allowed values.
negative_flag	bool	When set to True the event request is negative i.e. the assigned_event_id is set to 0 and the Camera is informed that all earlier unrequested events (including the one indicated) will not be read out and can be safely discarded.

Table A.2 – R1/Event/Telescope Configuration data definitions.

Name	Type	Description
CameraConfiguration		
tel_id	uint16	The ID of the Telescope whose Camera sent the event.
local_run_id	uint64	A local unique identifier assigned by the Camera.
config_time	LowPrecisionTime	Low-precision timestamp of the time when this configuration is applied. See [3] for details.
camera_config_id	uint64	The ID of the used Camera configuration parameter set.
pixel_id_map	uint16[num_pixel]	The mapping of pixel IDs to array positions in the waveform field in the Event data structure.
module_id_map	uint16[num_modules	The mapping of module IDs to array positions (*optional*).
num_pixels	uint16	The number of pixels of this camera.
num_channels	uint8	The number of channels (for CTA Cameras it is either 1 or 2).
num_samples_nominal	uint16	number of samples for event_type $\leq$ 33.
num_samples_long	uint16	number of samples for event_type = 33.
data_model_version	String	The version of the data model being used.
calibration_service_id	uint64	The ID of the calibration service data being used in the Camera pre-calibration.
calibration_algorithm_id	uint16	The ID of the algorithm used in the Camera pre-calibration.
num_modules	uint16	The number of modules in this camera (*optional*).

Table A.3 – R1/Event/Subarray data definitions.

Name	Type	Description
DataStream		
subarray_id	uint16	The ID of the sub-array associated to this data stream (*primary key*).
sb_id	uint64	The ID of the Scheduling Block to which this stream is associated to.
obs_id	uint64	The ID of ID of the current Observation Block (*primary key*).
EventRequestBunch		
event_request_bunch_id	uint64	The ID assigned to this event request bunch (*primary key*).
tel_id	uint16	The ID of the telescope whose Camera receives this request (*foreign key*).
local_run_id	uint64	A local unique identifier assigned by the Camera.
num_requests_in_bunch	uint64	The number of events requested in this bunch.
EventRequest		
event_request_id	uint64	ID of the event request (*primary key*).
event_request_bunch_id	uint64	ID of the bunch this event belongs to.
assigned_event_id	uint64	The event ID assigned by the subarray trigger.
requested_camera_time	HiResTimeStamp	High-precision timestamp for the requested event, normally the same as the corresponding trigger_time except for ACADA-requested triggers (trigger_type == 3 in A.4.1).
requested_trigger_id	uint64	The ID of the requested trigger.
requested_trigger_type	uint8	The trigger type of the requested event, see definition in Sec. A.4.1.
requested_data_available	bool	False if event data are not available for that particular trigger (busy trigger).
requested_hardware_stereo_trigger_mask	uint16 [0..1]	The hardware trigger mask (optional). See Sec. A.4.2 for the definitions of the allowed values.

Table A.4 – R1/Monitoring/Calibration data definitions.

Name	Type	Description
CalibrationCoefficientSet		
calibration_monitoring_id	uint64	ID of the calibration MON data entry containing the applied pre-calibration parameters.
tel_id	uint16	The ID of the Telescope associated with this event
local_run_id	uint64	ID of the current Local Run.
sky_pedestal_coefficient_id	uint64	ID of the sky pedestal coefficient data structure used in this Calibration Coefficient Set.
flat_fielding_coefficient_id	uint64	ID of the flat fielding coefficient data structure used in this Calibration Coefficient Set.
absolute_gain_coefficient_id	uint64	ID of the absolute gain coefficient data structure used in this Calibration Coefficient Set.

A.4 Trigger and Event Types

This document refers to three different trigger levels. Below we describe a classification of these triggers in terms of their level:

- **Level 0**: Below Camera level, not globally defined (e.g. pixel trigger as described in the pixel_status field).
- **Level 1**: Camera-level trigger, sent to the subarray trigger for coincidence search. A level 1 trigger has an trigger_id associated field.
- **Level 2**: Subarray-level trigger, which are those events to be kept for storage. A level 2 trigger has an associated event_id field.

Additionally, the LSTs use a hardware (HW) stereo trigger. The HW stereo trigger mask is transmitted in the ***Trigger*** data structure, see Sec. A.4.2 for further details.

A.4.1 Trigger_type Definition.

The type *trigger_type encodes* the following types of triggers:

- **trigger_type == 0**: Request permission to store (should be excluded from the sub-array trigger).
- **trigger_type == 1**: Request permission to store and should be included in sub-array trigger.
- **trigger_type == 2**: ACADA requested this event be captured.
- **trigger_type == 3**: The sub-array trigger should decide if this event is to be transmitted.
- **trigger_type == 4–255**: Free, not used at the moment.

A.4.2 LST hardware stereo trigger mask Definition.

The hardware_stereo_trigger_mask is used as a bit pattern. The description of each bit is given in the following table:

A.4.3 Event_type Definition.

The type event_type encodes the following event types, which are sub-types of the trigger_types:

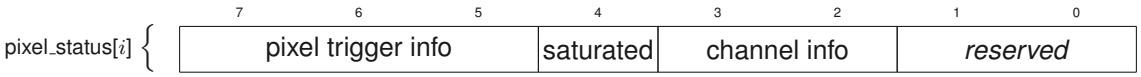
Bit [0 = LSB	Description
0	Mono trigger
1	Stereo trigger
2	Calibration trigger from Calibration light source
3	Single photo-electron trigger (Calibration box input trigger)
4	Software trigger
5	Pedestal trigger
6	Slow control
7	Free
8	Neighbour 1
9	Neighbour 2
10	Neighbour 3
11	Neighbour 4
12	Neighbour 5
13	Neighbour 6
14	Neighbour 7
15	Free

- **trigger_type == 0 subtypes**:
  - **event_type == 0**: Flat Field.
  - **event_type == 1**: Single PE.
  - **event_type == 2**: Pedestal.
  - **event_type == 3–15**: Free (can be extended with additional trigger_type == 0 subtypes).
- **trigger_type == 1 subtypes**:
  - **event_type == 16**: Muon.
  - **event_type == 17**: LST hardware stereo trigger.
  - **event_type == 18–23**: Free (can be extended with additional trigger_type == 1 subtypes).
- **trigger_type == 2 subtypes**:
  - **event_type == 24**: ACADA requested this event be captured (e.g. working with a central Illuminator).
  - **event_type == 25–31**: Free (can be extended with additional trigger_type == 2 subtypes).
- **trigger_type == 3 subtypes**:
  - **event_type == 32**: Nominal shower candidate event: The sub-array trigger should decide if this event is transmitted.
  - **event_type == 33**: Long shower candidate event: The sub-array trigger should decide if this event is transmitted.
- **event_type == 33–255**: Free.

Note that there are different types of pedestals: *sky*, *dark*, and *electronic*. The distinction between these pedestal types is handled at DPPS, based on monitoring data information transmitted to ACADA via the Monitoring system interface.

A.5 Pixel-wise Status Info

The layout of the `pixel_status` information should be as follows for each pixel, where bit 0 is the least-significant bit:



- *bit 0-1* is reserved for a DL0 DVR flag (not used at the R1 level, since all waveforms are read out)
- *bits 2-3* channel info: 0x0=pixel off/broken, 0x1=high-gain channel stored, 0x2=low-gain channel stored, 0x3=both channels stored
- *bit 4* saturation flag (0=ok, 1=saturated beyond acceptable gain range)
- *bits 5-7* the last 3 bits are for recording pixel-wise (or sector-wise) trigger info (contents to be determined)

A.6 Multi-dimensional Array Ordering

Some of the attributes described in this document contain multi-dimensional arrays, like those used to store a Cherenkov image with time evolution (waveforms). In order that all instruments store these in a common way, multi-dimensional arrays must be stored *contiguously* in row-major (C-like) ordering, with little-Endian byte-order. This means that the offset δ of an element with indices $\vec{n}$ and Data Shape $\vec{S}$ in an array with d dimensions is:

$$\delta = \sum_{i=0}^{d-1} \left(\prod_{j=i+1}^{d-1} S_j \right) n_i \tag{A.1}$$

For example, if the Data Shape of a 3D array called `waveform` is (L, M, N), the code to access element (i, j, k) would be the following in *C++*:

```
size_t index_3d(i, j, k){
  return i * (M*N) + j * N + k;
}

std::array<unsigned short, L*N*M> waveform;
...
// if i,j,k = 3,5,2:
value = waveform[index_3d(3,5,2)];
```

In *Python*, this would be equivalent to a *NumPY NDArray* with it's *shape* attribute set to `(L,M,N)` and it's `C_CONTIGUOUS` flag set to `True`:

```
waveform  = np.frombuffer(data, dtype=np.uint16).reshape((L, M, N), order="C")
value = waveform[3, 5, 2]
```

A.7 High Resolution Time Stamps

Since a standard way to represent times is essential in CTA for trigger synchronization and science analysis[1], the storage of times is standardized based on the following requirements and recommendations (from e.g. [1], [5]):

- The internal time-stamp value must be monotonically increasing, with no corrections for leap-seconds needed.

[1] note that this definition will eventually move to a document on "Common Base Data Types", part of the CTA Data Model

- The storage must meet CTA inter-telescope time precision requirements (A-PERF-0605, A-PERF-0610), which require a precision of $< 10\mathrm{ns} \pm 2\mathrm{ns}$, for the life of the instrument (to at least 30 years from the start of data taking).
- Time stamps should be stored compactly (not wasting too much space) and in a format that is not too difficult to parse.
- Time-stamps should not be susceptible to machine rounding error when subtracted, e.g. not stored in floating-point fields.

For these reasons, we require the following format to store time-stamps for low-level data where time precision is necessary:

- *Scale*: TAI[2]
- *Format:* POSIX (UNIX)
- *Reference:* 1970-01-01 00:00:00.0 [3]
- *Storage:* 2 unsigned 32-bit integers: (number of seconds since reference time, number of quarter-nanoseconds since the recorded second), as shown in Figure A.4. It is recommended for ease of use that if time stamps are stored in tabular output, they be broken into two colums such as `time_s` and `time_qns`, rather than stored as a single 64-bit value.

This gives enough precision $(250\mathrm{ps})$ and supports time stamps through the date *2106-02-07*.

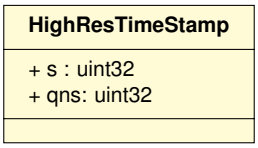


Figure A.4 – High Resolution Time Stamp data model

A.8 Known Data Model Issues

- The LST and NectarCam team cannot guarantee yet that all pre-calibraton steps, specifically the conversion to photo-electrons and the flat-fielding, can be implemented into the Camera Server.
- Requirements for the pre-calibration to be done within the Camera need to be specified. The work is already started by the Project Scientist.
- Usage of the pixel_trigger bit files must be specified by Camera teams in ACADA–Telescope ICDs.
- Support for SCT Cameras needs some consideration for a future data model update, see Sec. A.9.

A.9 Plans for Model Extension to Support SCT Cameras

The SCT Cameras do not fit in the Camera–ACADA data flow paradigm and the data model presented in this document due to the following reasons:

- The SCT Camera event rate is peaking at 10 kHz, which can be brought down to 2.5 kHz via the SCT DIAT trigger. Such a volume will not fit in the allocated number of fibres that connect the SCT Cameras to ACADA, and thus would require an additional data reduction step in the Camera Server.

[2]TAI is used by White Rabbit, and is supported by NTP servers

[3]Note that UTC at this reference time is 1969-12-31 23:59:52.000, so UTC and TAI are not identical at the reference, and of course differ more at later times as leap-seconds are added.

- The SCT Cameras have a large number of pixels (more than 11,000). This would require the possibility of sending to ACADA a sparse matrix of pixels with waveforms, or even in some cases (if possible) a time and integrated charge to reduce the data volume further.
- The SCT needs Camera trigger masks (to mask out some Camera modules during data taking) and trigger patterns (a bitmap of triggered pixels used for the trigger decision).

Since these features are not required for the Cameras of CTA *Phase 1*, implementing them now would complicate the data model presented in this document. Instead, they will be addressed in a future upgrade when is the time of integrating SCTs into CTA. Such an upgrade will be handled by either extending the data model for all Cameras or handling the SCT events via special methods. The future extension will consider the following aspects:

- An extension of the DVR concept will be considered, by considering data volume steps happening in the SCT Camera server in addition to ACADA, in contrast with the standard DVR happening only in ACADA. Arriving R1 events would have been already reduced. Such processing settings and provenance data should be incorporated into the event data. This new DVR step includes discarding unnecessary pixel information – some pixel data will be converted from waveforms to integrated charge and pulse arrival times, other pixels will be thrown away completely, and for most relevant pixels the whole waveform will be sent. The processing applied for each pixel and each event could be identified by using the *pixels_status* field. The waveform matrix has to be flexible to only send information for the pixels they need, and we need to avoid sending ”0”-s. This means we may need variable-length arrays or another clever way to handle this.
- Camera trigger masks are static configurations within the SBs, and can be added into the *CameraConfig* data structure.
- The *Pixel Status* bits 5-7 could be used for trigger patterns.
- The content of the Data structure of the debugging stream can be specified when known by the SCT team.

References

[1] A. Balzer. CDTS Trigger Message Content. v. 1.0, https://forge.in2p3.fr/attachments/download/13913/CDTSTriggerMessageContent.pdf, 5 2016.

[2] CTAO. CTA On-site Data Flow Functional Description. v1.1, *MAN-PO/160517*, https://jama.cta-observatory.org/perspective.req#/items/28205?projectId=6.

[3] CTAO Computing. DL0 Data Model Specification. CTA-SPE-COM-000000-0004. Rev 1c.

[4] CTAO Computing. Top-Level Data Model. CTA-SPE-OSO-000000-0001. Rev 1a.

[5] International Astronomical Union. SOFA Time Scale and Calendar Tools, 6 2019. http://www.iausofa.org/2019_0722_C/sofa/sofa_ts_c.pdf.

Glossary

ACADA (Array Control and Data Acquisition). A software system responsible for the control and monitoring of telescopes and auxiliary (non-telescope) instruments at a CTA site, for the efficient scheduling and execution of pre-scheduled observations and those triggered dynamically, for the monitoring of the system performance, for the data acquisition and volume reduction as well as the automatic generation of science alerts. (CTA-GLOS-245) 5

ADC Analog-to-Digital Converter 7

Array-Level Event A single Air-Shower Event Triggering multiple Cherenkov Cameras and judged suitable for storage. (CTA-GLOS-193) 6

Camera Event Set of data from the pixels of a Cherenkov Camera associated with a Trigger and including time and amplitude information. (CTA-GLOS-190) 5

Cherenkov Telescope A system composed of a Cherenkov Camera and Telescope Structure which is used to collect and image Cherenkov light from Air Showers. (CTA-GLOS-216) 5, 6

CTAO The Cherenkov Telescope Array Observatory, an international user facility distributed over four primary sites: a Headquarters, Science Data Management Centre, and two array-sites in the norhtern and southern hemispheres: CTA-N and CTA-S. (CTA-GLOS-206) 1, 2, 5, 12

Data Shape a tuple of integers representing the definition of a multi-dimensional array, where element i is the number of elements in the ith dimension, e.g. a 3D array with 10 elements in the first dimension, 2 in the second and 6 in the third would have a Data Shape of (10,2,3). 21

DVR Data Volume Reduction system, e.g. lossy-compression/zero-suppression/data-dropping 21, 23

Internal Trigger Triggering of a Camera due to a flash of light. (CTA-GLOS-187) 12

NSB Night Sky Background (Light) (CTA-GLOS-229) 7

Observation Block The smallest scheduling unit, a continuous observation with a sub-array during which science data is collected on the Merged Target of the parent Scheduling Block . During Observation Blocks, the Camera configuration and Telescope Target remain fixed. (CTA-GLOS-284) 12

SAG Science Alert Generation Pipeline. This is the ”Real-Time-Analysis” pipeline of CTA that produces Category-A Data products (formerly “Level A Analysis”) 7

Sub-array A sub-set of the Cherenkov Telescopes at one of the Array Sites. May be qualified by a Cherenkov Telescope type (LST, MST, SST) to denote the sub-set of all telescopes of that type on a given site. (CTA-GLOS-214) 6

Sub-array Trigger the stereo coincidence trigger system that considers triggers received from telescopes to identify Cherenkov showers, for example the “SWAT” (SoftWare Array Trigger) 9, 10, 12, 16

TAI International Atomic Time scale 22

Triggering (Triggering) Identification by a Cherenkov Camera of a block of time for which data may be of interest for analysis and to which a Trigger Timestamp can be associated. (CTA-GLOS-186) 5