Construction Project

ACADA – Telescope Camera Event and Trigger Data ICD

Prepared by Igor Oya Vallejo Digitally signed by Igor Oya Vallejo Date: 2021.11.18 15:09:19 +01'00' 2021-11-18

I. Oya, ACADA WP Coordinator Date

Etienne Lyard Digitally signed by Etienne Lyard Date: 2021.11.18 17:42:13 +01'00'

E. Lyard, ACADA-ADH Sub-task Coordinator

Approved by Nick Whyborn Digitally signed by Nick Whyborn Date: 2021.11.19 13:03:31 +01'00'

N. Whyborn, Lead Systems Engineer Date

Chiara Montanari Digitally signed by Chiara Montanari Date: 2021.11.19 11:21:20 +01'00'

C. Montanari, Interface Manager Date

Released by Wolfgang Wild Digitally signed by Wolfgang Wild Date: 2021.11.19 14:04:32 +01'00'

W. Wild, Project Manager Date

Change Log

Issue	**Revision**	**Date**	**Section/Page affected**	**Reason/ Remarks / Initiation Documents**
1	a	2021-11-18	All	Initial version

List of Contributors

Name	**Organization**	**Contribution**
Igor Oya	CTA PO	The document and current version of the interface
Etienne Lyard	University of Geneva	Event Data API and Data Format Specification
Adam Muraczewski	CAMK	Trigger Timestamp API
Chiara Montanari	CTA PO	Interface overall process management

List of ICD actors

Actor	Role	Agreed date and signature
I. Oya	ACADA Coordinator	
D. Mazin	LST	
D. Hoffmann	NectarCAM	
M. Barcelo	FlashCam	
R. White	SST Camera	
M. Gaug	Calibration Aspects	

Table of Contents

List of Acronyms

ACADA	Array Control and Data Acquisition
ADH	Array Data Handler
API	Application Programming Interface
CDTS	Clock Distribution and Timestamping (server)
CTA	Cherenkov Telescope Array
CTAO	CTA Observatory
DPPS	Data Processing and Preservation System
DVR	Data Volume Reduction
ICD	Interface Control Document
SAG	Science Alert Generation Pipeline
UCTS	Unified Clock and Timestamping System
SWAT	Software Array Trigger

1 Purpose and Scope

1.1 Purpose

This Interface control document (ICD) specifies the requirements of the interface between the Array Control and Data Acquisition (ACADA) System and CTA Telescope Cherenkov Cameras to handle raw camera data (R1, see [AD3] for the definition of CTA data levels). It also documents design decision affecting this interface.

This ICD describes the Camera Event and Trigger data exchange of the interface between the ACADA System (the target system) and a CTA Cherenkov Camera (the source system). The purpose of the ICD is to define the design of the interface(s) ensuring compatibility, modularity and independent testability among involved interface ends, by function and a common Application Programming Interface (API).

This ICD is managed by the CTAO Interface Manager (or their delegates) and represents an agreement between the relevant actors. The actors in this ICD are shown at the beginning of this document.

This ICD is used:

1. to document the interface definition,
2. to control the evolution of the interface,
3. to document the design solutions to be adhered to, for a particular interface,
4. as one of the means to ensure that the supplier design (and subsequent implementation) is consistent with the interface requirements,
5. as one of the means to ensure that the design (and subsequent implementation) of the participating interface ends are compatible.

This ICD tracks the necessary information required to effectively define the interface between a CTA Cherenkov Camera and ACADA for the R1/Event data flow, as well as any rules necessary to give the development team guidance on the architecture of the system to be developed.

The purpose of this ICD is also to clearly communicate the planned inputs and outputs from the affected systems for all potential actions whether they are internal to the system or transparent to system users.

Its intended audience is the ACADA and Telescope Cherenkov Camera development teams, and stakeholders interested in the interfacing of these systems, and the Systems Engineering personnel.

1.2 Scope

This ICD specifies the interfaces for

- Camera trigger timestamps (R1/Event/Trigger data message from Camera to SWAT, and event requests – R1/Event/Subarray data– from SWAT to Camera).
- Camera R1/Event/Telescope data (from the Camera Server) to ADH.

Control and monitoring interfacing aspects are not part of the scope of this document and are addressed elsewhere [RD1, RD2].

2 Applicable and Reference Documents

2.1 Applicable Documents

- AD1: MST, LST, SST, and Common Telescope Requirements in Jama.
- AD2: R1/Event Data Model Specification, Doc. No. CTA-SPE-COM-000000-0002, Issue 1, Rev. e, 12.2.2021.
- AD3: Top-level Data Model, Doc. No. CTA-SPE-OSO-000000-0001, Issue 1, Rev. b, 30.4.2020.
- AD4: LST R1 debugging events specification (under preparation).
- AD5: NectarCAM R1 debugging events specification (TBD).
- AD6: FlashCam R1 debugging events specification (TBD).
- AD7: SST Cam R1 debugging events specification (TBD).
- AD8: Requirement Specification for Array Control and Data Acquisition System. Doc. No. CTA-SPE-COM-303000-0001, Issue 2, Rev. h, 29.04.2020
- AD9: *adh-apis*. https://gitlab.cta-observatory.org/cta-computing/common/acada-array-elements/adh-apis/-/tree/master

2.2 Reference Documents

- RD1: ICD for ACADA-Telescope Control Doc. No. CTA-ICD-SEI-000000-0002 Issue 2, Rev. f, 19.3.2020.
- RD2: ICD for ACADA - Array Element Monitoring. Doc. No. CTA-ICD-SEI-

000000-0004 Issue 1, Rev. a 16.11.2020

- RD3: CTAO. CTA On-site Data Flow Functional Description. v1.1, MAN-PO/160517, https://jama.cta-observatory.org/perspective.req#/items/28205?projectId=6.
- RD4: Requirement Specification On-Site ICT North Doc. No. CTA-SPE-COM-302000-0001_Issue 3, Rev a. *Link*

3 Interface Requirement specification

3.1 Overview

A CTA Telescope is a system composed of both a Camera and a Structure functional unit, each composed of hardware and software, and both coordinated by a Telescope Manager component.

The goal of the Camera functional unit is to collect Cherenkov light data from air showers and calibration light sources and send these data (R1 data level as defined in [AD3]) to ACADA for further processing. The Camera functional unit includes a *Camera Field Unit* installed onboard of the Cherenkov Telescope, and a *Camera Server* located in the On-site Data Centre. A very simplified version of Camera-ACADA interplay is the following (see also Figure 1).

1. Camera triggers on an event (shower, calibration, pedestal event).
2. Camera sends the trigger message with an associated high-precision timestamp (R1/Event/Trigger) to ACADA.
3. ACADA evaluates the camera trigger in the sub-array-level trigger (at the software array trigger, called SWAT hereafter) by comparing the trigger timestamps with the timestamps of other telescopes forming part of the sub-array. In addition, SWAT also processes trigger messages for local event triggers such as calibration events or local muons, which do not require a timestamp comparison.
4. ACADA sends a message (R1/Event/Subarray) back to Camera requesting those events of interest for further processing.
5. Camera applies the camera pre-calibration.
6. Camera sends the selected event data structures (R1/Event/Telescope) to the Array Data Handler (ADH) subsystem of ACADA.
7. ACADA applies the ACADA Data Volume Reduction (DVR) algorithms, forwards the reduced event to the Science Alert Generation Pipeline (SAG) pipeline and stores the data (DL0/Event) on disk.

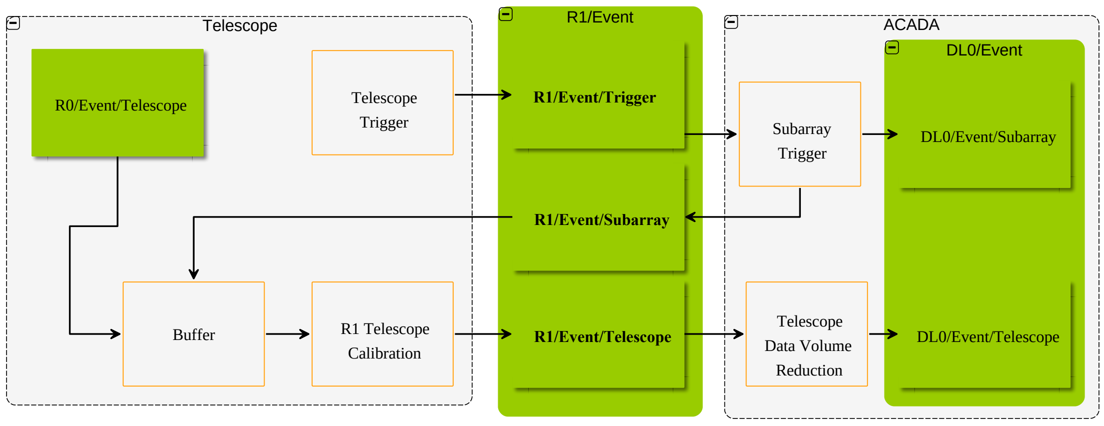


Figure 1: Simplified R1/Event data flow between Telescopes and ACADA, showing data streams (in green) and processing steps applied to them (orange).

This ICD specifies the interface requirements that the ACADA and Telescope System must meet.

This interface is a software interface, and thus it describes:

- the concept of operations for the interface,
- defines the message structure and protocols that govern the interchange of data,
- and identifies the communication paths along which the project team expects data to flow.

For each interface elements, the ICD provides the following information:

- A description of the data exchange format and protocol for exchange,
- A description of the interface,
- Assumptions where appropriate,
- Estimated size and frequency of data exchange.

3.2 Design Description

Data exchange is achieved by means of the *adh-apis* software [AD9]. R1/Event/Telescope data connections are achieved by the API using the ZeroMQ package[1] for R1/Event/Telescope data, which is a thin layer on top of TCP/IP connections. Google Protocol Buffers[2] are used for data serialization. A C++-based API is provided to telescope teams which permits them to abstract the usage of ZeroMQ and Protocol Buffers from the Camera software. The push/pull scheme is used for these streams, which allows to use load-balancing on the receiver's end. Camera servers and trigger sources are considered to be the

[1] https://zeromq.org/
[2] https://developers.google.com/protocol-buffers

servers, so that any failing component downstream can be replaced with no reconfiguration on the Cherenkov camera side.

The SWAT API is a C++-based API implemented with low-level Linux epoll and sockets[3] in order to stream trigger messages and event requests over a TCP session using a custom communication protocol. The API can be flexibly configured so that trigger reporting and event request reception is handled separately (i.e. using two separate processes, machines etc.). The API is provided as a set of C++ programming language functions and types declared in the swat_api.h file along with their implementations in the SWAT library.

3.3 Assumptions

ACADA and Cameras communicate via a set of common APIs, created by the ACADA team, and use the same data format specification.

3.3.1 Constraints

The aggregated rates and throughput supported by this interface are those described in Section 3.4.1.4.

3.3.2 Functional Allocation

This interface implements the following functions:

- Handle Cherenkov Camera event triggering at sub-array level
- Handle Cherenkov Camera event data.

3.3.3 Extension of the interface

Extensions to this interface are possible for debugging events. The “debugging” stream allows additional data elements to be included in the data stream, see Section 3.5.3.

3.3.4 Data Transfer

The following networks are relevant for this ICD (see details in [RD4][4]):

- Data Centre Network.
- Telescope Network.
- White Rabbit Network.

The camera event data (R1/Event/Telescope) are transferred thorough the Data Centre Network via Ethernet from the Camera Server located in the Onsite Data Centre to “worker nodes” allocated to ACADA, also located in the Onsite Data Centre (see [RD4]). The communication is done through the Onsite Data Centre switch stack.

[3] Refer to Linux man pages: epoll(7) (https://man7.org/linux/man-pages/man7/epoll.7.html) and socket(2) (https://man7.org/linux/man-pages/man2/socket.2.html)
[4] Note that formally the reference provided is only applicable to CTA-N, however, the underlying assumption is that the same requirements will apply also to CTA-S.

Trigger timestamps (R1/Event/Trigger) flow from elements in the Camera functional unit to SWAT, and then flow back to the Camera servers after SWAT evaluates them. The infrastructure used may be different depending on the type of Camera. ACADA always communicates with the "Camera Server" entity, regardless of where are the logical processes physically located.

Further details on the differences of the data flow between cameras is given in Appendix VI.

3.3.5 Security and Integrity

TBD: To be updated once security requirements are done.

3.4 Interface specification

This section specifies all elements of the ACADA to Camera Event Data interface.

3.4.1 Transactions

ACADA-CAMEVT-I-010 R1/Event/ Data Access Point. ACADA shall interface the Camera Server in order to exchange R1/Event data.

ACADA-CAMEVT-I-020 R1/Event/Telescope Data streaming. Data streaming shall be handled by the *adh-apis* software project [AD9].

3.4.1.1 Camera Event Data

ACADA-CAMEVT-I-030 Camera Event Data Transfer. The *adh-apis* [AD9] software project shall be used to enable the R1/Event/Telescope data transfer between the Cameras and ACADA.

3.4.1.2 Trigger timestamps

ACADA-CAMEVT-I-040 Trigger Timestamps and Event Requests Data Transfer. The SWAT Communication API from the *adh-apis* [AD9] software project shall be used at both ends to communicate the R1/Event/Trigger and R1/Event/Subarray data.

ACADA-CAMEVT-I-041 Event Trigger Timestamps and Event Requests Data Transfer. Cameras shall send to the SWAT R1/Event/Trigger (trigger timestamp) messages to indicate local and potential sub-array events (trigger_type == 0, 1 & 3 in [AD2], section A.4.1), and wait for event request messages to be received from the SWAT before submitting R1/Event/Telescope data to ACADA.

3.4.1.3 API Version Control

ACADA-CAMEVT-I-050 API Version Control. The following exact tags specify the versions of the APIs (adh-apis software project [AD9]) that shall be considered valid for this ICD issue: v0.1.0 (commit ID: 3dec678e).

3.4.1.4 Supported Data Rates and Throughput

ACADA-CAMEVT-I-060 Supported Data rates and Throughput. The Data rates and Throughput specified in the following table shall be supported by this interface

Note: Technically, it is planned to be achieved by aggregating several data streams in parallel.

Table 1: Supported rates and throughput.

Camera	**Trigger**	**Events**				
	Individual Camera trigger rate to ACADA (*) [kHz]	Individual Camera Event Rate handled by ACADA (*) (**) [kHz]	Camera Data Volumes [Gbit/s]	Local Event Collection Rate (**) [kHz]	Events for Calibration Purposes (#) (**) [kHz]	Events for Calibration Purposes (†) (**) [kHz]
LST	30 (a)	15 (d)(g)	24 (j)(p)	3 (m)(q)	15 (t)	3 (u)
MST	14 (b)	7 (e)(h)	12 (k)(p)	1.4 (n)(r)	7 (t)	1.4 (u)
SST	1.2 (c)	0.6 (f)(i)	2 (l)(p)	0.120 (o)(s)	0.6 (t)	0.120 (u)
SCT	TBD					

Notes	
(*) Averaged over 100 ms (**) Note: These are to be counted for the total volume in column 4) (#) when observations are not taking place, and valid during periods up to 5 seconds. *Note: the associated requirement is under revision.* (†) While observations are taking place	
Applicable Telescope requirements [AD1]: (a) B-LST-1285 (b) B-MST-1285 (c) B-SST-1285 (d) B-LST-1280 (e) B-MST-1280 (f) B-SST-1280 (j) B-LST-1450 (k) B-MST-1450 (l) B-SST-1450 (m) B-LST-1305 (n) B-MST-1305 (o) B-SST-1305	Applicable ACADA requirements [AD8]: (g) B-ACADA-0180 (h) B-ACADA-0182 (i) B-ACADA-0184 (p) B-ACADA-3015 (q) B-ACADA-0270 (r) B-ACADA-0272 (s) B-ACADA-0274 (t) B-ACADA-3135 (u) B-ACADA-3125

3.5 R1/Event Data Model and Format

3.5.1 Data Format

ACADA-CAMEVT-I-070 Compliance with R1 Data Model. The Data structures specified in this ICD shall be compliant with the data model presented in [AD2].

ACADA-CAMEVT-I-075 Abstracted R1/Event/Telescope Data Exchange. It shall be possible to use the C++ API in the CTA Gitlab repository [AD9] for the R1/Event/Telescope data exchange.

Notes:

- *The adh-apis will take care to internally transform the data into the format specified* *in* ACADA-CAMEVT-I-080

- *The valid version is specified in ACADA-CAMEVT-I-050.*

ACADA-CAMEVT-I-080 R1/Event/Telescope Data Format. It shall be possible to use the data format specified as protocol buffer data types in the CTA Gitlab repository under the file CTA_R1.proto [AD9] for the R1/Event/Telescope data exchange.

Note: The valid version is specified in ACADA-CAMEVT-I-050.

ACADA-CAMEVT-I-090 Trigger Timestamp Data Format. The data format specified as C++ structures used to form the required TCP payloads defined in the swat_packet.h header file [AD9] shall be used for the R1/Event/Trigger and R1/Event/Subarray data exchange.

Note: The valid version is specified in ACADA-CAMEVT-I-050.

3.5.2 Usage of data model optional fields

The R1/Event data model ([AD2]) is generic to suit all CTA cameras and specifies mandatory and optional data fields.

ACADA-CAMEVT-I-100 Mandatory Fields. All Cameras shall implement all mandatory fields identified in [AD2] as in the realization of this interface.

ACADA-CAMEVT-I-110 Optional Fields. This interface shall support the optional fields specified in Table 2 for the different CTA Cameras.

Table 2: Realization of optional R1/Event optional fields by the individual cameras.

	Event					CameraConfiguration		Trigger
Camera	num_chan	first_cell_id	pedestal_intensity	module_hires_local_clock_counter	GhostEvent	num_modules	module_id_map	hardware_stereo_trigger_mask
LST	2	Yes (array size 2120)	No	Yes (array size num_modules)	Yes	Yes	Yes (array size num_modules)	Yes
NectarCAM	2	No	No	Yes	Yes	Yes	Yes	No
FlashCam	1	No	Yes	No	No	No	No	No
SST	1	Yes (scalar)	No (?)	No	No	No	No	No
SCT	TBD							

Additionally, the following camera will submit to ACADA "long events" (event_type == 33 in Section A.4.1 of [AD2]):

- FlashCam.

3.5.3 Debugging Events

ACADA-CAMEVT-I-120 Debugging Events. ACADA shall support via this interface the delivery of Camera debugging events when Cameras are operating in a specific debugging mode. The data models of debugging events are extensions of the regular events specified in [AD2], different for each camera. The data model and format of these events shall be specified in dedicated documents [AD4, AD5, AD6. AD7].

3.6 Pipeline Configuration

This section describes the way to configure the APIs at the Camera functional unit and ACADA end in order to enable the event and trigger data transfer between the two elements.

3.6.1 Camera event messaging stream configuration

ACADA-CAMEVT-I-130 Camera event messaging stream configuration. Cameras shall use the configuration file provided as part of the adh-apis software project [AD9], and containing the information specified in Table 3, to configure the event message streaming.

Note: The configuration is currently read from a file called ZMQStreamerConfig.ini, except for the port parameter that is passed when initializing the API object. The API looks for this file at three different locations, in a specific order. First the file that came with the GitLab repository. The exact location on the system is assembled when compiling the project. If this file cannot be found, it looks in /opt/cta. If no configuration file can be found there either, it looks in the directory from where the executable was launched. In any case a message is printed in the output logs (either to the shell or to the ACS logging system) to indicate which file was used. If no configuration file can be found at either three locations, default values are used and a warning is added to the logs. The default values correspond to the values written in the file from the repository. Please note that this configuration scheme is temporary and will be superseded by the ACADA configuration system in future ACADA releases.

Table 3: Camera event messaging stream configuration parameters.

Configuration parameter	Type	Purpose
Port	Unsigned integer	Select which TCP/IP port will be used on the server side (i.e. Camera Server) to establish a connection with ACADA
Num_extra_streaming_threads	Unsigned integer	Define how many extra threads can be used for the network IOs. At least one thread is used. Under the ZMQ implementation, this

		is the number of thread assigned for ZMQ internal use.
Catch_signals	Boolean	Define if POSIX signals SIGINT and SIGTERM should terminate the streaming operations right away, or if it should rather gracefully terminate by flushing pipelines and closing any active connection.
Statistics_period	Unsigned int	Defines how often (in µs) throughput statistics should be sent via port 12800. 0 means that statistics are disabled. Statistics include number of events per second and averaged throughput over the period. This service uses the ZMQ PUBLISH distribution pattern, and having a receiving peer or not has no impact on overall performances. As the data sent is very small, it has no measurable impact on overall performances as long as the period remains long enough (> 1 second)
Data_period	Unsigned int	Defines how often (in µs) data items are duplicated via port 12800. 0 means no duplication. Default is localhost. This service uses the ZMQ PUBLISH distribution pattern, and having a receiving per or not has no impact on overall performance. Activating the feature consumes extra throughput, i.e. event_size*period_i
Monitor_hostname	String	Defines the name of the host to which statistics and

		duplicated data items should be sent
Poll_timeout	Unsigned int	How long (in msec) should a call to receive block before returning. 0 means forever.
Send_timeout	Unsigned int	How long (in msec) should a call to send block before returning if no peer is connected or if outbound buffers are full.
Max_num_waiting_send	Unsigned int	Number of messages that can wait in the inbound queue. Default is 100.
Max_num_waiting_receive	Unsigned int	Number of messages that can wait in the outbound queue. Default is 100.
Linger_duration	Unsigned int	Max. duration in msec during which the API will block when being destroyed in case messages are yet to be sent out. Default is 1 second.

3.6.2 SWAT API configuration

ACADA-CAMEVT-I-140 SWAT API Configuration. Cameras shall use the SWAT configuration file provided as part of the of the adh-apis software project [AD9], and containing the information specified in Table 4, to configure the SWAT message streaming.

Note: The configuration of the API is performed using an .ini configuration file, except for the IP address and SWAT channel parameters which are passed programmatically when initializing the SWAT API. Its location and filename shall be provided as a string using the swat_api_configure_from_file(…) function in order to facilitate a flexible directory tree setup (the path can be read, for instance, as a command line parameter). The software can also be started without a configuration file, using default values.

Please note that this configuration scheme is temporary and will be superseded by the ACADA configuration system in future ACADA releases.

Table 4: SWAT API Configuration.

Configuration parameter	Type	Purpose
IP_address	String	The IP address of the machine running the SWAT server instance the API should connect to.
Channel_number	Integer	The number of the channel reserved in the SWAT server

		configuration for the telescope that uses the API instance in question.
TAI_offset	Integer	Current TAI-UTC offset to be used by SWAT's timing-related functions if kernel CLOCK_TAI is misconfigured. Use 0 to treat CLOCK_TAI absence as critical failure.
Port	Integer	The TCP port number the SWAT server listens on.
Send_flag	Integer	Defines if this client instance will send triggers: 0 if it will not; 1 if it will.
Receive_flag	Integer	Defines if this client instance shall receive event requests: 0 if it shall not; 1 if it shall.
Sort_flag	Integer	Defines if the stream of trigger timestamps is sorted: 0 if sorting is not guaranteed, 1 if timestamps are sorted.
Net_timeout_ms	Integer	Defines the timeout for socket operations in milliseconds.

4 Appendix I: Level B requirements applicable to this interface

This interface addresses the following requirements, repeated here from Jama [AD1] and the ACADA Level-B Requirement Specification document [AD8] for convenience. Please note that OES (Observation Execution System) is the earlier name given to the ACADA system now, and should be understood as a synonym.

- **B-LST-1280** Event Rate The LST Camera must be able to collect and deliver for further processing Events arriving at a sustained event rate of at least 15 kHz with a random distribution in time. At 7.5 kHz event rate the deadtime requirement of B-TEL-1260 must be met.
- **B-LST-1285** Trigger Rate Cap: The LST Camera must deliver Trigger information to the OES at a rate (averaged over 100 ms) of less than 30 kHz for events arriving at random.
- **B-LST-1305** Local Event Rate The Camera must deliver events without associated array level coincidences (Local Events) to the OES at a rate of less than 3 kHz. All such events must be flagged as Local Events.
- **B-LST-1450** Camera Output Data Rate The Camera must deliver Event data (including both Local Events and Array-Level Events) to the OES at a rate never (in any second) exceeding 24 Gb/s.
- **B-MST-1280** Event Rate The MST Camera must be able to collect and deliver for further processing Events arriving at a sustained event rate of at least 7000 Hz with a random distribution in time.
- **B-MST-1285** Trigger Rate Cap The MST Camera must deliver Trigger information to the OES at a rate (averaged over 100 ms) of less than 14000 Hz for events arriving at random.
- **B-MST-1305** Local Event Rate The Camera must deliver events without associated array level coincidences (Local Events) to the OES at a rate of less than 1400 Hz. Such events must be flagged as Local Events.
- **B-MST-1450** Camera Output Data Rate The Camera must deliver Event data (including both Local Events and Array-Level Events) to the OES at a rate never (in any second) exceeding 12 Gb/s.
- **B-SST-1280** Event Rate The SST Camera must be able to collect and deliver for further processing Events arriving at a sustained event rate of at least 600 Hz with a random distribution in time.
- **B-SST-1285** Trigger Rate Cap The SST Camera must deliver Trigger information to the OES at a rate (averaged over 100 ms) of less than 1200 Hz for events arriving at random.
- **B-SST-1305** Local Event Rate The Camera must deliver events without associated array level coincidences (Local Events) to the OES at a rate of less than 120 Hz. Such events must be flagged as Local Events.
- **B-SST-1450** Camera Output Data Rate The Camera must deliver Event data (including both Local Events and Array-Level Events) to the OES at a rate never (in any second) exceeding 2 Gb/s.

- **B-TEL-1300** Muon Trigger The Camera must be able to trigger on, and flag prior to transmission of R1 data to ACADA, fully contained (within the sensitive area of the camera) muon rings, from muons impacting the mirror with an energy > 20 GeV with an overall efficiency greater than 90% (excluding dead-time). The fraction of falsely flagged muons in the data stream to ACADA may be high as long as this efficiency requirement is met and the overall Local Event limit (B-*ST-1305) is not exceeded.
- **B-ACADA-3125** Calibration Data Acquisition ACADA shall acquire all Event data flagged as of interest for calibration purposes that is sent from Cameras during Observations up to the following average rates: LST: 3kHz, MST: 1.4kHz, SST: 120 Hz. Note: These data are included in the total data volume received as described in B-ACADA-3015.
- **B-ACADA-3135** High Rate Calibration Data ACADA shall be able to acquire all Event data flagged as of interest for calibration purposes that is sent from Cameras up to the following limits over timescales of up to 5 seconds when Observations are not taking place. LST: 15 kHz, MST: 7 kHz, SST: 0.6 kHz. *Note: This requirement is under revision.*
- **B-ACADA-0180** Individual LST Trigger Rate ACADA shall identify all sub-array-level coincidences with sustained individual LST trigger rates (averaged over 100 ms) of at least 15 kHz.
- **B-ACADA-0182** Individual MST Trigger Rate ACADA shall identify all sub-array-level coincidences with sustained individual MST trigger rates (averaged over 100 ms) of at least 14 kHz.
- **B-ACADA-0184** Individual SST Trigger Rate ACADA shall identify all sub-array-level coincidences with sustained individual SST trigger rates (averaged over 100 ms) of at least 1.2 kHz.
- **B-ACADA-0190** Trigger rate ACADA shall identify coincidence Events with a total rate over all sub-arrays of at least 40 kHz. Note that this rate shall be met simultaneously with the deadtime requirement B-ACADA-0200.
- **B-ACADA-0270** Local LST Event RateACADA shall collect flagged Local Events arriving irrespectively of array coincidence from individual LSTs at rates of at least 3000 Hz. Note that Local Events will be indicated as part of the Event Class, see B-TEL-1310
- **B-ACADA- 0272** Local MST Event Rate ACADA shall collect flagged Local Events arriving irrespectively of array coincidence from individual MSTs at rates of at least 1400 Hz. Note that Local Events will be indicated as part of the Event Class, see B-TEL-1310
- **B-ACADA-0274** Local SST Event RateACADA shall collect flagged Local Events arriving irrespectively of array coincidence from individual SSTs at rates of at least 120 Hz. Note that Local Events will be indicated as part of the Event Class, see B-TEL-1310
- **B-ACADA-3125** Calibration Data Acquisition ACADA shall acquire all Event data flagged as of interest for calibration purposes that is sent from Cameras during Observations up to the following average rates: LST: 3kHz, MST: 1.4kHz, SST: 120

Hz. Note: These data are included in the total data volume received as described in #3015.

- **B-ACADA-3015** Receiving Camera Data Volume The ACADA system shall handle up to the following Camera Data volumes: 24 Gbps for each LST (up to four telescopes), 12 Gbps for each MST (up to 25 telescopes), and 2 Gbps for each SST (up to 70 telescopes).

5 Appendix II: Data Format Specification.

The valid data format for Event is specified in tag v0.1.1 (commit ID: 3dec678e).

The data format for Events data is specified at https://gitlab.cta-observatory.org/cta-computing/common/acada-array-elements/adh-apis/-/blob/master/data_model/raw/CTA_R1.proto

The valid data format for Triggers and Event requests is specified in the swat_packet.h header at https://gitlab.cta-observatory.org/cta-computing/common/acada-array-elements/adh-apis/-/blob/swat_c_api/swat/swat_packet.h

6 Appendix III: Example of Trigger Interface API Implementation

An example implementation of an application that uses the SWAT API to send triggers and receive event requests can be found in the adh-apis gitlab repository: https://gitlab.cta-observatory.org/cta-computing/common/acada-array-elements/adh-apis/-/blob/aadadf8608aa896c716c599798aec8d998872af5/ExecExampleSWATClient.c

7 Appendix IV: Example Event Data API Implementation

An example implementation can be found at https://gitlab.cta-observatory.org/cta-computing/common/acada-array-elements/adh-apis/-/blob/master/ExecExampleEventsServerFlashcamStyle.cpp

A companion program so that the API can be exercised is also provided at https://gitlab.cta-observatory.org/cta-computing/common/acada-array-elements/adh-apis/-/blob/master/ExecExampleEventsReaderFlashcamStyle.cpp

8 Appendix V: API design choices justification.

The Camera Event data transfer mechanisms used in the adh-apis make usage of the 3rd party libraries ZeroMQ and Protocol Buffers. The usage of such libraries was decided by the ACADA team –following an ACTL review recommendation– to minimize the cost of producing the necessary software to transfer the data while supporting the throughput and efficiency needed by CTA. The original Protocol-Buffers-based API was tested extensively and successfully with real data from the LST1 installation at La Palma.

Based on the feedback of the FlashCam and SST teams, and with the aim to reduce the risk of future maintenance costs of the Camera software, a library encapsulating the individual software modules (including 3rd party) in the form of a C++-based API is provided by ACADA. This library provides a common ACADA-defined API serving as a lightweight abstraction of the serialisation and transport technologies.

However, the LST and NectarCAM teams have already developed a mature and optimized software using the original Protocol Buffers based API and express that porting this software to the C++ abstraction will require a significant manpower effort from the side of these teams. Enforcing such a porting would create a significant delay in the integration of ACADA with the telescopes deployed in CTA-N. As a management decision, and since ACADA will use internally the Protocol Buffer serialization anyway, it is also offered as a way of realizing this interface. This means that Cameras can choose to use the abstracted C++ API (see ACADA-CAMEVT-I-075) or its actual implementation (see ACADA-CAMEVT-I-080). By choosing to use the implementation rather than the abstracted API, camera teams take the risk to have to update their interface to ACADA in their camera server, would the implementation be replaced by another technology in the future. We want to note that we estimate that this arrangement provides negligible additional effort to the involved teams and is the least risky choice at the moment of defining this ICD.

It is encouraged to all teams to join the effort of defining the common C++ abstracted API (see ACADA-CAMEVT-I-075), and to eventually transition to only use it.

We note that for the SWAT API, all cameras will use the same common API (See ACADA-CAMEVT-I-090).

9 Appendix VI: Detailed context of the interface

This section provides a more detailled description of the context of the interface specified in this document.

9.1 SWAT Behavior

9.1.1 Trigger and event types

In what follows, a few terms defined in the R1 data model specification document [R2], section "A.4.1 Trigger_type Definition" will be used extensively. We reproduce them here for convenience:

event_id: *The event ID assigned by the subarray trigger*

trigger_time: *High-precision timestamp assigned by the Cherenkov Camera.*

trigger_type: *The type of trigger, according to these definitions:*
• 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.

TO provide context to the reader, the Event data include the following data types, which are related to trigger_types:

Event_types*: The type of event, according to these definitions:*
• trigger_type == 0 subtypes:
– event_type == 0: Flat Field.
– event_type == 1: Single PE.
– event_type == 2: Pedestal.

• trigger_type == 1 subtypes:
– event_type == 16: Muon.
– event_type == 17: LST hardware stereo trigger.

• trigger_type == 2 subtypes:
– event_type == 24: ACADA requested this event be captured (e.g. working with a central Illuminator).

• 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.

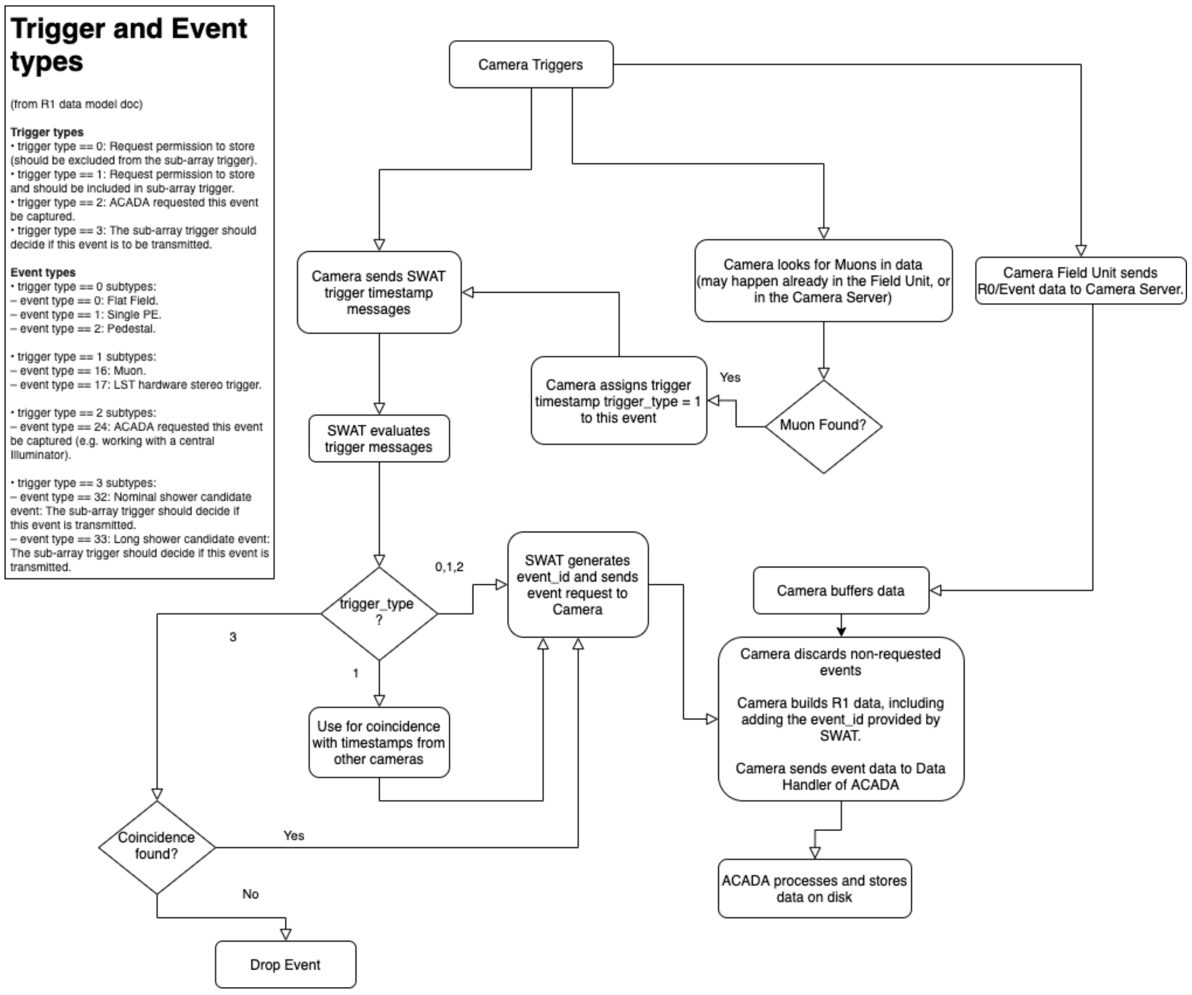
9.1.2 Handling trigger messages and event requests

When handling trigger messages of a sub-array, the following process happens in ACADA ad the telescopes (see also *Figure 2*):

1. Camera triggers
2. Camera looks for muons candidate events in data
3. Camera Sends ACADA (SWAT) trigger timestamps messages which include a trigger_type and trigger_time
4. Camera buffers the corresponding event data waiting for SWAT's decision
5. SWAT evaluates the trigger messages by:
    a. If the trigger_type is 3, the SWAT compares the trigger_time with the one from trigger messages of trigger_types 3 and 1 from other telescopes participating in the subarray. If a coincidence is found, an event request and event_id are created
    b. If trigger_type is 0, 1 or 2, an event_id and event request are created
    c. In addition, for trigger_types 1, the timestamp information is also used by SWAT to look for time coincidences for events arriving from other cameras (e.g. other cameras triggering also for an LST HW stereo trigger)
6. SWAT sends an *event request bunch* to each camera. This contains a set of individual event requests and event_id-s.
7. Cameras send to ACADA the event image data for those events that were requested. Each event includes the event_id assigned by SWAT, as well as trigger_time.

Caveats:

- Event_id-s are unique within an OB, and can be generated only by SWAT.
- ACADA is the deciding authority on which event is stored by CTA. This allows managing cameras misbehaving, e.g. if triggering too much from ACADA though no procedure for this has been considered yet.
- Muon identification (trigger_type = 1) must be assigned by cameras to the trigger timestamp message before sending it to SWAT. It is the camera's responsibility to channel timestamps internally and to decide to request those muon candidate events.



Trigger and Event types

(from R1 data model doc)

Trigger types
• 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.

Event types
• trigger type == 0 subtypes:
– event type == 0: Flat Field.
– event type == 1: Single PE.
– event type == 2: Pedestal.

• trigger type == 1 subtypes:
– event type == 16: Muon.
– event type == 17: LST hardware stereo trigger.

• trigger type == 2 subtypes:
– event type == 24: ACADA requested this event be captured (e.g. working with a central Illuminator).

• 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.

Camera Triggers
Camera sends SWAT trigger timestamp messages
Camera looks for Muons in data (may happen already in the Field Unit, or in the Camera Server)
Camera Field Unit sends R0/Event data to Camera Server.
Camera assigns trigger timestamp trigger_type = 1 to this event
Yes
Muon Found?
SWAT evaluates trigger messages
0,1,2
SWAT generates event_id and sends event request to Camera
Camera buffers data
trigger_type ?
3
1
Camera discards non-requested events
Camera builds R1 data, including adding the event_id provided by SWAT.
Camera sends event data to Data Handler of ACADA
Use for coincidence with timestamps from other cameras
Coincidence found?
Yes
No
ACADA processes and stores data on disk
Drop Event

Figure 2: Schematic view of the process of handling sub-array triggers by SWAT. Internal details of Cameras are omitted.