Preliminary draft 11:30 3 May 2018

3 May 2018
be-dep-co-cewg@cern.ch

Device Property behaviour and contextual data

Controls Evolution WorkingGroup (CEWG) [1]

Keywords: CEWG report; timing selector; timestamps; property behaviour; first update.

Summary

We propose to homogenise and clarify the requirements in terms of device property behaviour. This includes the valid timing selectors for the different operations on the different types of properties as well as the expected first updates (presence and number) following a subscription. We also propose homogenisation of the timestamp-related meta-data returned when interacting with a device server.

1 Introduction

The topic of the Timing selectors and the expected behaviour of the various types of Property was studied in the CEWG in the context of the upcoming LS2 and a request by TE-EPC to clarify a number of cases. We also looked at the contextual data that should be added to the Property data in the different cases. We propose to implement the following recommendation in order to restart after LS2 with a more homogeneous and clearer behaviour of the device servers. Note that this work slightly overlaps with the work conducted in the frame of the CO3 task-force on the Device Server Frameworks. This recommendation is compatible with it and will be integrated to the final document. However, the implementation target dates being very different, we believe a two-step approach is more suitable.

[1] For this topic, the CEWG was composed of: L. Burdzanowski, M. Buttner (Secretary), S. Deghaye (Chair), F. Hoguin, R. Gorbonosov, E. Gousiou, I. Kozsar, G. Kruk, S. Page, W. Sliwinski, T. Wlostowski, J. Wozniak.

2 Property types and operations

In the FESA3 framework, there are 3 types of properties: Setting, Acquisition, and Command. In the case of the FGC devices, this distinction does not exist. In the scope of this document, it is sufficient to say:

- Setting properties are read-write and only change when they are set;
- Acquisition properties are read-only and change periodically depending on the device's internal behaviour;
- Command properties are write-only and never change (as they can't be read).

Command properties are limited Setting properties and what is said about write operation of Setting properties applies. Therefore, we will not consider them further.

At the class design level, each property declares its capabilities. We propose that the Setting (and Command) properties have a *multiplexed* flag indicating that, by design, they are capable of handling PPM settings. For Acquisition properties, we propose that a new capability *cycle-bound* is introduced instead of the *multiplexed* flag. This new flag indicates that the property is capable of providing data bound to a specific accelerator cycle instance along with the required contextual data. At instantiation level (i.e. definition of a device), we also have two flags (*multiplexed* and *cycle-bound*). The final access point's (i.e. device-property pair) capability depends on the property capability definition AND the device configuration. For example, an access point is multiplexed only if both the property and the device definitions have the *multiplexed* flag set to true.

Three operations are available: Get, Set, and Subscribe/Update. The update operation is internal to the server and only make sense when there is a prior subscribe. Table 1 recaps the available operations for the different property types.

Table 1: Available operations per Property type

	Set	Get	Subscribe
Setting	Y	Y	Y
Acquisition	N	Y	Y
Command	Y	N	N

3 Timing selectors

A timing selector is composed of a triplet of values separated by a dot: timing domain, timing field, timing field value; for example, CPS.USER.SFTPRO. In the vast majority of the cases, the timing field is USER. *Subscribe* on acquisition property is the only operation that supports another timing field than USER.

The Timing selector requirement depends on the operation and on the device-property capability (i.e. multiplexed, cycle-bound). Today, the behaviour is inconsistent with values ignored in certain cases and overlap between the empty string and the ALL selectors. Therefore, we propose to enforce the following behaviour. Table 2 and Table 3 summarise the valid combinations of cycle selectors and the setting or acquisition properties respectively. In the tables, *DOM* represents the timing domain.

Table 2: Valid timing selectors for operations on setting properties

	Multiplexed		Non-multiplexed	
	Get/Set	Subscribe	Get/Set	Subscribe
DOM.USER.XYZ	**Y**	**Y**	N	N
DOM.USER.ALL	N	**Y**	N	N
DOM.OTHER.XYZ	N	N	N	N
"" (empty string)	N	N	**Y**	**Y**

Table 3: Valid timing selectors for operations on acquisition properties

	Cycle-bound		Non-cycle-bound	
	Get	Subscribe	Get	Subscribe
DOM.USER.XYZ	**Y**	**Y**	N	N
DOM.USER.ALL	N	**Y**	N	N
DOM.OTHER.XYZ	N	**Y**	N	N
"" (empty string)	N	N	**Y**	**Y**

4 Contextual data

Contextual data is the meta-data that come with property data on a get or an update. In the following definitions, we use names as explicit as possible without constraint on the existing situation and backward compatibility concerns. We suggest names for the different data item in 6.3 taking into account different aspects such as backward compatibility and data reliability. Figure 1 depicts a typical FEC process along with the measurement points of the different timestamps defined below.

- *Access timestamp*: The UTC time at which the CMW server received the request (any operation). This could easily be replaced by a reading of the system time at the client level;
- *Acquisition timestamp*: The UTC time at which the acquisition was triggered or to which the acquired value relates. By definition, this only makes sense for acquisition properties;

- *Cycle selector*: The timing selector that was used for a set or the user that was being played at acquisition time;
- *Cycle timestamp*: The UTC time of the start of the accelerator cycle that was being played at acquisition time;
- *Get timestamp*: The UTC time at which the read action was finished (e.g. in FESA, the get action's return time);
- *Set counter*: Counter to track the number of sets done on a setting property; eases checks on read-modify-write operations.
- *Set timestamp*: The UTC time at which a property was set (e.g. in FESA, the set action's return time). By definition, this only makes sense for setting properties.

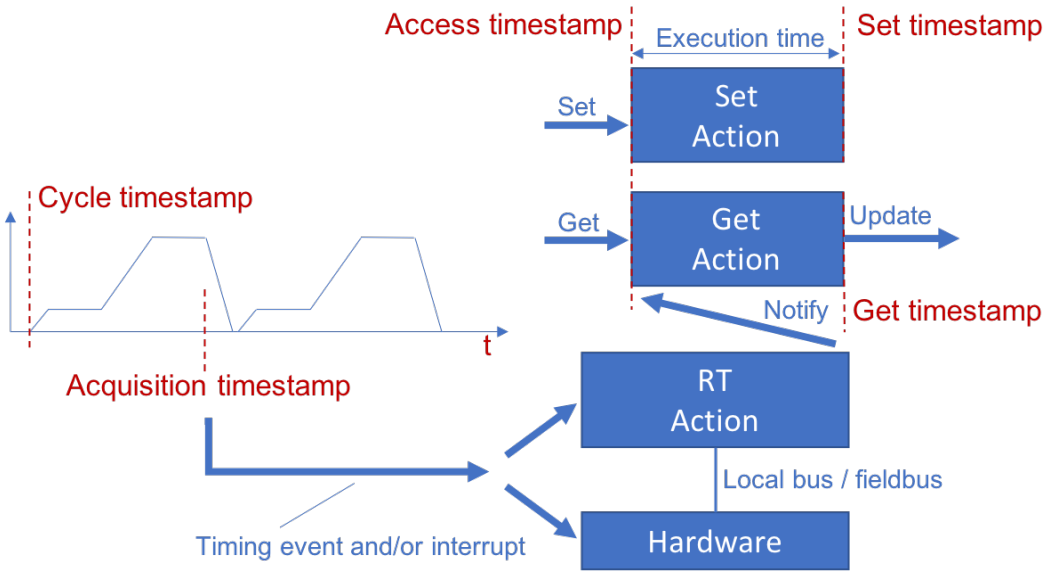

Figure 1: Typical front-end process and the associated timestamps

The timestamps and counters are updated only when the corresponding action is successful. For example, the *Set timestamp* and the *Set counter* are updated only if the *Set action* returns normally (i.e. no exception). On the other hand, setting several times the same property with the same values triggers an update of the *Set timestamp* and the *Set counter*.

While most of the timestamps and counters should be handled by framework code (e.g. FESA) the specific equipment software should have the possibility to override the default behaviour whenever this leads to a more precise value and does not change the semantic attached to the timestamps. For example, if the equipment software is able to compute the acquisition timestamp of the first point in a sampled waveform, this value should be used as it is more precise than the timestamp of timing event that triggered the acquisition.

Table 4 indicated which contextual data is relevant for the different property types.

Table 4: Contextual data for the different property types

	Access timestamp	Acquisition timestamp	Cycle selector	Cycle timestamp	Get timestamp	Set counter	Set timestamp
Multiplexed setting	**Y**	N	**Y**	N	**Y**	**Y**	**Y**
Non-multiplexed setting	**Y**	N	N	N	**Y**	**Y**	**Y**
Cycle-bound acquisition	**Y**	**Y**	**Y**	**Y**	**Y**	N	N
Non-cycle-bound acquisition	**Y**	**Y**	N	N	**Y**	N	N

5 First updates

The first update is an update sent as soon as possible after the creation of a subscription.

For setting properties, the first update is the only update that is sent by the device server until a set is done on the property. Due to this behaviour, the first update is absolutely required and the device server should always be able to produce one. When the property is multiplexed and the selector is ALL, a first update per user (the exact number depends on the timing domain) must be sent. Table 5 summarises the expected behaviour. Note that an update following a set of a setting property is called an immediate update.

Table 5: First update with setting properties

	Multiplexed	Non-multiplexed
DOM.USER.XYZ	1 first update for user XYZ	N.A.
DOM.USER.ALL	1 first update for each user of timing domain DOM	N.A.
"" (empty string)	N.A.	1 first update

For the acquisition properties, the situation is different as the update is triggered by the device server itself. The reasons to trigger are numerous and, as implementation details must be hidden from the client, are irrelevant. We have to take into account whether the property is cycle-bound as this defines the validity for the data. Table 6 summarises the

behaviour.

When the property is not cycle-bound, the latest acquired data is, by definition, still valid and therefore should be returned as a first update. Subsequent updates are sent whenever new data is available. Note that, due to the multi-threaded nature of the device servers, one cannot guarantee that the first update will always arrive before any other normal updates, or in the right order. Contextual data (timestamps), as discussed in 4, must be used to determine the update type and the acquisition time.

When the property is cycle-bound, the need for a first update is less obvious and has severe limitations. First, new data should be available next time the cycle is played (typical scenario is to subscribe to cycles being played) and, furthermore, the conditions of the accelerator when the acquisition was done is unknown making the data validity doubtful. In addition, due to data consistency requirements and limited memory on the FEC side, data is not always available for all of the cycles in the given timing domain. This leads to situations where, for subscriptions with ALL, most of the first updates are simply exceptions. For example, in the SPS with a super-cycle with 3 cycles, 3 first updates will be with data and 29 first updates will be with no-data exception. Nevertheless, there are several valid situations where the CS users, both in OP and equipment groups, require a first update as they are in conditions where it can be provided and is valid (e.g. subscription on a user that has just been played). They are also several valid cases where the first update is not required and actually needs to be excluded by the client code. This latter case requires the first update to be clearly identified as such and, ideally, our API should allow the user to specify whether or not a first update is needed.

Finally, for situations where the device server is not able to provide valid data, we propose to return a standardised *"no data"* exception indicating that the data is unavailable.

5.1 Future integration of on-line and historical data services

From client's point-of-view, the actual source of the first update is irrelevant. Furthermore, as described above, the device server is not always able to provide data and in some cases the cost of providing data can be high (e.g. when the data store is behind a low-bandwidth field bus). We propose to study the possibility to hide the access to the different services behind a Controls API. For the case at hand, we suggest to use in conjunction the device servers for live data and the logging system (i.e. NXCALS) for historical data. With such implementation, it would be possible to retrieve transparently not only data for the first updates but also data over an arbitrary time span in the past. Details and API should be studied in future CEWG meetings. The BE-CO Initiative to collect and follow up this topic is CS-170

Table 6: First update's behaviour for acquisition properties and proposed selector restrictions

	Cycle-bound	Non-cycle-bound
DOM.USER.XYZ	1 first update for user XYZ	N.A.
DOM.USER.ALL	1 first update for each user of timing domain DOM	N.A.
"" (empty string)	N.A.	1 first update

6 Implementation aspects

In the following sections, we detail what needs to be modified in order to put in place the proposed changes as well as how and when we suggest to perform the modifications. All the workpackages related to this proposal can be followed up through the BE-CO initiative CS-425.

The first category of components is the device server providers. At the FEC level, there are FESA and FGC. At application level, japc-ext-remote is impacted as well as the latest generation of concentrators based on japc-flux. As the InCA server can be accessed via JAPC, server-side modifications are required to ensure a conform behaviour. For FESA, the modifications must be part of the LS2 Baseline in order to ensure a wide adoption during LS2 and a restart for run3 with the new scheme for the vast majority of the device servers.

The second category is the client applications and the core communications libraries such as JAPC and perhaps CMW.

As usual, the CCS will be impacted whenever there is a change in the configuration aspects (e.g. device definition).

6.1 Multiplexed and Cycle-bound flags

There are two modifications proposed here. The first one is to rename the *multiplexed* (or *PPM*) flag at the acquisition property design level to *cycle-bound*. The second change is the introduce an additional flag cycle-bound at the device definition level. For FESA devices, the initialisation of the new flag could be done based on the timing domain information following the logic (timing domain = none ⇒ cycle-bound = false).

Impacted: FESA, FESA Navigator, FGC, CCS, InCA, japc-ext-dirservice, CCDA.

6.2 Timing selector

The stricter handling of the timing selectors needs to be supported by all the device server providers. Note that servers running with the old scheme are forward-compatible as they are less strict. However, clients should not rely on that fact in order to allow a smooth transition to the new scheme.

For FESA, this could be part of the next major version i.e. the LS2 Baseline version. For FGC, this should be adopted during LS2 (part of the proposal was triggered by them) For the JAPC extensions, this should be part of the baseline.

Impacted: FESA, FESA Navigator, FGC, InCA, japc-ext-remote, japc-flux.

6.3 Contextual data

Currently, with CMW-RDA3, there are 3 contextual elements that are provided (cycle timestamp, cycle name, acquisition stamp). They are stored in the *AcquiredContext* object included in the *AcquiredData*, which contains the property data. Unfortunately, acquisition stamp provided by the *getAcqStamp* method does not always match the definitions given in 4. This can be a source of confusion even though this short cut is convenient for the application developers who are only interested by a vague notion of time irrespective of the property type.

We do not believe all the timestamps identified in 4 should be added. There are several reasons:

- some backward compatibility must be kept;
- the application developer's effort must not be increased without bringing a clear added value;
- several timestamps are mainly for experts and do not need to be standardised.

The *Access timestamp*, and the *Set count* do not bring enough added value and we propose not to use them. The cycle timestamp and the cycle selector are already directly available in the *AcquiredContext* and should be kept that way. We proposed to merge the table 4 and to set the *AcquiredContext's AcqStamp* as shown in table 7.

Table 7: Source of the *AcquiredContext's AcqStamp* for the different property types

	Acquisition timestamp	Get timestamp
Multiplexed setting		**X**
Non-multiplexed setting		**X**
Cycle-bound acquisition	**X**	
Non-cycle-bound acquisition	**X**	

We also propose that the *Set timestamp* is added, when available as a *setStamp* entry in the data field available in the *AcquiredContext*. In situation where the *Set timestamp* is not

available, the entry can either be omitted or set to 0. To avoid duplication of the entry name, we propose that the CMW-RDA-CERN layer standardise the entry.

Impacted: FESA, FESA Navigator, FGC, JAPC, CMW.

6.4 First update

We propose to keep the status quo in the first update behaviour. Nevertheless, we recommend to introduce a specific exception *no-data* at the CMW-RDA-CERN level that should be used by all the frameworks instead of the current generic exception with string-parsing processing at application level (e.g. FESA_13021 ≡ no data).

The cost of introducing a flag in the subscription request and the added value is not clear especially since the handling of such a flag requires important changes in the subscription concentration in JAPC. Therefore, we recommend postponing this modification until more profound changes such as described in 5.1 are implemented.

Impacted: CMW, FESA, FGC, JAPC, InCA, any client processing exceptions such as *FESA_13021*.