GSI Helmholtzzentrum für Schwerionenforschung GmbH

Document Title

FAIR Control System Development Guideline “FESA Development Guideline”

Document Name
F-DG-C-01e

Date yyyy-mm-dd
2012-08-16

Abstract

This document is the FESA Development Guideline for the FAIR accelerator control system. This guideline serves as directive for designing FESA equipment software at the GSI and the FAIR facility.

FAIR Division/Group/Supplier
Controls Department

Prepared by:
FESA Team

Table of Contents

List of Tables

1. Purpose and Classification of the Document

The purpose of this document is to specify the development guideline as a directive for designing FESA equipment software for FAIR. Adherence to this guideline will improve product quality and maintainability of the FAIR accelerator control system.

The development guidelines complement the technical guidelines and detailed specifications for the FAIR control system in providing general rules and regulations for control system development.

Whenever regulations and requirements are specified in the General Specifications, Technical Guidelines, Common Specifications or Detailed Specifications of the Control System they are only referenced in this document. The related documents are listed in Appendix II.

No legal or contractual conditions are treated in this document. All related information is given in the General Specifications for FAIR II.

1.1. Responsibilities

The responsibilities with respect to changes and modifications of the present document are entirely in the hands of the Accelerator Controls and Electronics Department of the GSI Helmholtz Centre for Heavy Ion Research GmbH (GSI) Darmstadt. The technical responsibility is in the hands of the FESA Core Team within the Accelerator Controls and Electronics Department.

For initial information please contact the administration of the Accelerator Controls and Electronics Department.

Further information on the organigram, names of responsible persons and task leaders, as well as the agreed document release and approval procedure is summarized in the organizational note 'Controls Project for FAIR'.

2. Scope of this Development Guideline

This document is dedicated to FESA class developers and serves as guideline for all FESA class development done within the context of the FAIR control system (see also [3]). This guideline shall help to unify the way device interfaces are defined. Therefore, it contains conventions for the following aspects:

- A common syntax for names of properties and fields
- Requirements to operations and controls
- Recommendations for defining device APIs with (composite) properties and fields.

This document covers only new developments based on the FESA infrastructure.

The Software Architecture Guideline F-DG-C-03e [4] fully applies.

3. Introduction

This document is based on the CERN LEIR “Guidelines and conventions for defining interfaces of equipment developed using FESA” [1]. It presents guidelines for the definition of operational interfaces for equipment modules developed with FESA. Such operational interfaces are used by applications in the control room. Properties for equipment specialists are not covered by this document – equipment groups are free to define additional properties for their own use.

The reason for elaborating these guidelines is the following:

- Simplify operations by providing coherent data that can be easily correlated with information from other sources and consists of information about data-quality.
- Offer clear information about the status of an equipment with error messages when applicable.
- Simplify application development, by combining related data inside composite properties. This avoids that an application has to subscribe to many different properties and needs to re-combine them in a tedious and error prone process.
- Improve the overall reliability and performance of the control system.

This document covers the following items of standardization:

- Rules for names of properties and fields
- Definitions of standard properties with well-defined overall meaning, and standardized fields

The conventions in this document constitute a formal agreement between operations, application developers and equipment groups. The conventions will be officially supported by the FESA development environment at GSI.

To ease the usage of the conventions, a FESA - GSI class template exists which has to be used by class developers. As well most of the naming conventions are forced by the FESA XML-schema.

4. Requirements

The definition of the standardized properties is based on the following requirements. They have their origin in operational usage scenarios and from experience made during application development.

4.1. General Requirements

- FESA Equipment Software has to provide a consistent controls interface. There must be standard properties that have the same meaning and usage across all kinds of equipment.
- Property data should be as far as possible self–contained. The property should contain all information needed to use the data for different purposes, such as displaying, archiving, and correlation. To interpret a property value, it should not be necessary to retrieve data from other

sources (e.g. databases). It should also be possible to directly archive property values in a logging database, without having to process them (e.g. combine them with other information). Examples of information required:

  - Acquisition data must contain information on the timing context at which it was acquired.
  - Measurement data should contain information of how the instrument was configured at the moment of the acquisition (control values such as gain, calibration factors, etc.).
  - Data should contain information on problems associated with control values (e.g. acquisition problems, out–of–range values, deviations from the requested setting).

- Data from equipments should be easy to correlate. As it may be necessary to correlate different property values with each other, the following information is required:
  - multiplexing context
  - acquisition time
  - Information about the units in a format that is suitable for treatment in a program (e.g. for programmatic comparison of data).

  These values must have identical meaning and format across all kinds of devices.
- Measurement values must have information about their quality. It may happen that a measurement is taken under bad circumstances, e.g. with noise or an insufficient number of particles. Even if a measurement is not perfect, it has to be transmitted to the user, of course with an indication of the lower quality of the data.

4.2. Additional Requirements from Operations

- Supervision of device status must be supported. Operators need to have an overview which devices are in an abnormal state. If there is a problem with a device, error information needs to be available for further diagnosis. It should be easy for operators to determine where the problem is: In the device itself or in the environment (e.g. an access interlock).
- Supervision of actual values should be supported. Operators need to get an overview quickly whether a control value is at its set value or not. For instance, it is important for operations to know that the current of a power converter has deviated from the requested control value. If possible, the supervision and interpretation of the data should be done at the front-end level.

4.3. Additional Requirements from Controls

- Data that belongs together should be kept together. If a coherent set of information is available on the front–end computer, it should be kept together and transmitted as “block” to the application layer. Splitting up data into several properties has a negative effect especially when subscription is used, because the application layer has to subscribe to several small properties and recombine the complete data set. This is a tedious and error prone process that should be avoided.

- Property data should be suitable for machine processing. Numerically coded values (e.g. enumerations that map to integers) are more appropriate for machine treatment.
- On-change publishing of data should be supported. To optimize the network traffic, it is recommended that data is published only when it has changed. A mechanism to specify a threshold is foreseen for later FESA versions.
- Information about data-units is kept within each field and will be published with the data.

5. Naming Conventions

A series of conventions are necessary to fulfill the requirement of a unified controls interface. They concern names for device classes and instances, for property names, for field names and field characteristics.

5.1. Syntax for naming devices, properties and fields

- The **device class names** are mixed-case starting with a capital letter. To separate meaningful parts of the name a capital letter is used.
  - Regular expression: ([A-Z])([A-Za-z0-9])*
  - Examples: BdiStdProfile, SeptaMagnet.
- The **device instance names** are already fixed by the nomenclature–system. To find a proper device instance name, please refer to the GSI nomenclature-responsible. More information on the System for Nomenclatures of Accelerator Devices at FAIR & GSI is available at [2].
- The **property names** are written in mixed-case starting with a capital letter. To separate meaningful parts of the name a capital letter is used. For the client the property name can be case-insensitive. This as well means that two properties can not be distinguished by case sensitivity.
  - Regular expression: ([A-Z])([A-Za-z0-9])*
  - Examples: SummaryResult, ExpertSettings, ConfigureAxis5
- The **field names** of a FESA-class are written in mixed case starting with a lower-case letter. To separate meaningful parts of the name a capital letter is used. Field names should represent the physical value they describe.
  - Regular expression: ([a-z])([A-Za-z0-9_])*
  - Examples: timeStamp, highVoltage_status, current_unit, flowChannel_03
  - A field name suffix is separated from its related field by an underscore, but underscores can be as well used for other purpose.
- The **value-item-names** are strongly coupled to the field-names. They have to fulfill all naming-restrictions of the fields.
- Besides that, there are fixed value-item-names that have predefined meanings and cannot be used in a different sense because they are reserved by the JAPC client interface which will be used at GSI. Table 1 shows all additional naming restrictions.

value-item-name	Usage
~~value~~	This value-item-name is reserved and must not be used.
acqStamp ~~timestamp~~ ~~timeStamp~~ ~~TimeStamp~~ ~~timeNano~~	Different value-item-names for timestamps. Some must not be used because of historical reasons. The standard one, that should be used is acqStamp (in nanoseconds).
~~cycleId~~ ~~scNumber~~	These two value-item-names are reserved by JAPC and must not be used.
cycleStamp	The timestamp which indicates the start of a cycle
cycleName	This is the value-item-name for the full name of a cycle. (ex. "SIS.USER.VACC_11“)
~~dim_~~****	The prefix dim_ must not be used because of historical reasons.

Table 1: List of predefined value-item names

- **Constants** are written in capital letters. To separate meaningful parts of the name an underscore is used in constants.
    - Regular expression: ([A-Z])([A-Z0-9_])*
    - Examples: INIT_DEV_STATE.

5.2. Convention to define different levels of detail for a property

There may be variants of the same property, presenting information with a different level of detail. This provides each user category with the needed amount of information. Experts need more information than normal users. For properties there will be a concept to restrict the access level, so that only experts are allowed to change them.

Adding different levels of detail is done with a naming convention for the property names by the use of a prefix:

- <PropertyName> (e.g. Setting, SetFlow) The property with all information needed for operations.
- Expert<PropertyName> (e.g. ExpertSetting, ExpertSetFlow) Adds information for equipment experts (the person which is responsible for the device)
- Summary<PropertyName> (e.g. SummarySetting, SummarySetFlow) Reduced contents settings (e.g. to reduce resources needed to store the property in a database).

5.3. Value-item names across properties

If a value can be set in a Setting-Property and an Acquisition-Property is used to read back the corresponding measurement value, than the used value-items should have the same name for both properties.

5.4. Suffixes to add information to a field

A property typically contains several value-items. For instance, in an imaginary device, there might be a property with two value-items:

- MyProperty
  - position
  - highVoltage

It may be necessary to add information to each of these items, e.g. a “status” information to determine whether the value of “position” is valid. This is done by adding an additional value item, using the naming syntax <ref-item>_<suffix>. For the above device, this would look as follows:

- MyProperty
  - position
  - position_status
  - highVoltage
  - highVoltage_status

A number of suffixes has been defined already for standardization:

Data Field Name	Data Field Type	Description
_status	**AQN_STATUS**	Additional information about problems in the corresponding field-value that must be taken into account when interpreting the field-value. If _status = 0 everything is OK and the value is fully normal and valid. Problems signaled with this suffix include deviations from the requested setting, reduced measurement quality, ongoing movements, problems occurred in the acquisition, etc. If several control values are contained in the Acquisition property, there may be a _status suffix for each of them.
_min, _max	**<same type as field>**	Minimal and maximal values for continuous properties
_tolAbs	**<same type as field>**	Absolute tolerance, expressed in the same units as the field it refers to. The tolerance specifies how much an acquisition value can deviate from the control setting. If the aqn value is outside range, the _status field must flag an “DIFFERENT_FROM_SETTING” error.
_tolRel	**double**	Relative tolerance in percent. See description of _tolAbs
_tolCheckMode	**TOL_CHECK_MODE**	Describes how the tolerance is controlled.
_acqStamp	**long long**	The concrete time, when the related field was measured in UTC (in nanoseconds)
_unit	**char[10]**	The unit, in which the field-value is saved.

Table 2: Fieldname Suffixes

All values have to be saved in the default unit. E.g. it is forbidden to save a field in the unit “mA”, instead the data needs to be saved always in the base-unit, which is “A”.

An important aspect about suffixes is that they are read-only for the client. The reason is that suffixes contain meta-data that characterizes the main control value, but that is not modifiable by normal users.

6. Standard Properties

This section specifies standard properties for the device interface. Each standard property needs to be implemented by each device, even if the property is not used and only returns an error.

The following property-types must be present on all devices. They are part of a standard FESA Interface at GSI:

- GSI-Status-Property
    - Acquisition Property, used to show the overall status of the device (not cycle dependent)
    - Only one instance of this property-type is allowed
- GSI-Power-Property
    - Setting property, used to enable or disable a device
    - Only one instance of this property-type is allowed
- GSI-Acquisition-Property
    - Returns acquisition data, retrieved from the hardware.
    - Any number of instances are allowed
- GSI-Setting-Property
    - Used to set parameters to the hardware
    - Contains any control values
    - Any number of instances are allowed
- GSI-Reset-Property
    - Control-Property, used to reset the device (all persistent data is kept)
    - Only one instance of this property-type is allowed
- GSI-Init-Property
    - Control-Property, used to initialize the device with default values from the device-instance xml file
    - Only one instance of this property-type is allowed
- GSI-Version-Property
    - Acquisition-Property, which returns the current software and hardware-versions of an equipment
    - Only one instance of this property-type is allowed

In order to avoid documentation duplication, all GSI-Property documentation is stored directly within the GSI specific html-documentation in the class-design (FESA-browser).

The GSI standard class template already contains the necessary standard properties and custom types, in order to allow an easy usage.

7. Design Decisions

Some elements have a GSI-specific code-generation, C++ implementation or Plugin behavior. This chapter explains, why these design decisions were taken and how they are realized.

7.1. The GSI-multiplexing-field

The field “GSI-multiplexing-context” was introduced to the GSI-Acquisition-Property for the following reasons:

- Per default FESA only returns a cycle-stamp and a cycle-name when subscription is used as connection method. The GSI-multiplexing-context field ensures that as well via a client-”Get” all multiplexing data is obtained.
- A requirement of the application group of the Accelerator Controls and Electronics Department was, to always have the multiplexing context and/or measurement-timestamp within the acquisition-data. These stamps e.g. are needed in order to correlate measurement values.

7.2. The GSI-error_collection-field

The GSI-error_collection-field keeps the most recent errors on class-level. It was introduced as GSI-specific field for the following reasons:

- It allows to indicate the error-state in the “Status” property, not only in the FEC-specific logfiles.
- It allows to introduce a ring-buffer which is not cycle-dependant (the FESA-rolling buffer only works for multiplexed fields).
- Easy usage, provided by a GSI-specific implementation. The developer just uses the GSI-specific method addError(...), which automatically will add a timestamp, a device-name and a cycle-name. As well a logging entry automatically will be written, using the GSI-specific logging system.
- Possibility to provide a default-server action for the struct-data-type.

8. Custom Types

In order to avoid documentation duplication, most custom-type documentation is stored directly within the GSI specific html-documentation of the according fields in the class-design (FESA-browser).

This chapter provides documentation for all custom types for which it is not yet possible to include their documentation into the FESA-browser. (This will be done, as soon as lab-specific custom-types are supported by FESA).

8.1. TOL_CHECK_MODE

This constant defines possible modes to check whether a control value is inside the tolerance values (see 5.4, "_tolCheckMode" suffix).

Used to give information on how the tolerance field is used to calculate the xxx_status information.

Enum identifier	Enum value	Usage
ABS	**0**	Use the absolute tolerance _tolAbs.
REL	**1**	Use the relative tolerance _tolRel.

Table 3: Custom type field: TOL_CHECK_MODE, datatype: enum

8.2. AQN_STATUS

Possible values to describe the acquisition status of a field (see 6.4, "_status" suffix). If this suffix is missing, it means that no additional status information is provided for the corresponding field. If all bits are 0, this means that the corresponding field is OK. Only the lower 16 bits are standardized, the upper 16 bits can be defined by the equipment specialist.

Enum Identifier	Enum Value	Usage
NOT_OK	**2^0**	Some problem occurred that is not represented by the other bits. This property is called NOT_OK so that it is not mixed up with ERROR or WARNING in the Status property
BAD_QUALITY	**2^1**	The value was acquired with a degraded quality. This is typically used for measurements.
DIFFERENT_ FROM_SETTING	**2^2**	Different from the requested control value (for discrete values) or out of tolerance (for continuous values).
OUT_OF_ RANGE	**2^3**	The value is out of the normal range (e.g. a temperature is too high or too low).
BUSY	**2^4**	The property value is changing in response to receiving a new control value (e.g. moving to a new position, charging a capacitor, …). If the value change does not reach the requested new value within the maximum timeout, the BUSY bit should remain=1 and the TIMEOUT bit must be turned on.
TIMEOUT	**2^5**	A timeout occurred, because the property did not reach the reqested new control value within the maximum allowable time. A timeout normally indicates a problem to be addressed by the equipment specialist. This is typically used for slow changing control values that are BUSY while they change.
<reserved>	**2^6 … 2^15**	Reserved for future standardization.
<class-specific>	**>= 2^16**	Equipment-specific problem indicators. A bit which is set to 1 should indicate a problem. If the whole xxx_status field = 0, this indicates that the status is OK.

Table 4: Custom type field: AQN_STATUS, datatype: bit-enum-32bit

9. Filters

A filter (context) can be used to get/set only parts of a property.

E.g. if the client only needs to receive one field of a property instead of all fields, this can be specified using a filter.

The filter (context) which the client can send to the FESA class is of the type "rdaData" and can consist of any number of filter-elements. If a FESA class supports the usage of filters, it is recommended to use the standardized names for these filter items.

Tag	Type	Value	Description
[FieldName]Filter [1]	**String**	**add**	Only added fields will be sent to the client
[FieldName]Filter [1]	**String**	**remove**	The removed field will not be sent to the client
[FieldName]Filter	**String**	**addPartial[3]**	Only the 3.rd Element of the array will be set/get
[FieldName]Filter	**String**	**addPartial[3][6]**	Only element [3][6] of the matrix will be set/get
[FieldName]Filter	**String**	**addPartial[][6]**	Only the 6th row of the matrix will be set/get
[FieldName]Filter	**String**	**addPartial[structItem]**	Only the defined structure item will be set/get

Table 5: Filter items

If a partial set is triggered for a Setting property, it is not needed to send a list of filter items for the fields which are to set. Simply all provided fields inside the data container will be set if the Setting property is configured to handle the request properly.

10. Class Relationships

For some types of hardware equipment it makes sense to define a relation between different software classes. FESA already predefines some inter-class relations. If the concrete equipment fit's in one of the following predefined relation concepts, the adequate FESA concept should be used.

Additional information can be found in the FESA-class documentation.

10.1. Composition

If hardware equipment is clustered into different sub devices and all these devices shall be controlled by the same FEC, the class developer should segment the equipment-software to several devices and fit them together in a so called "Composition".

Advantages:

- it is possible to re-use already existing classes
- a replacement of sub-classes of a composition can be done without much effort

[1] Only for acquisition properties

Example:

A slit-control, which consists of 2 step-motors and 4 end-switches

10.2. Inheritance

For widely used device-types often the developer does not need to specify a new interface, but is able to inherit from an already defined base FESA class. Before starting development the developer should check the list of available base classes. Together with the FE group and the FESA core team a matching base-class can be chosen, if available.

Advantages:

- interface standardization
- less work for the class developer

Example:

A power supply class, which inherits from a “BasePowerSupply” class.

10.3. Association

If one FESA class serves as client for another FESA class, the relationship “Association” should be used. In this relationship it is not necessary to run both classes on the same FEC.

Advantages:

- less implementation work for the class developer

Example:

Consider an imaging setup with a GigE camera, lens and image intensifier. A dedicated FESA class on a high end computer acquires the images from the camera and performs the pre-analysis. Control of the lens aperture, operating voltages of the image intensifier etc. may be done by a separate slow control system, i.e. via a PLC which in turn is controlled by another FESA class running on a normal computer. Using the "Association" relationship, the FESA class acquiring the images may also act as a client to the FESA class for the slow control (e.g. make settings, do acquisition, ...). It can thus present a complete interface of the device to the GUI client.

In general an "Association" relationship is similar to the "Composition" relationship, but with the different FESA classes running independently on different FECs. It should be used in cases where a single device is controlled by more than one FESA class running on different FECs. It is intended to present the GUI client with a single device view and access instead of forcing the GUI to deal with different FESA classes controlling different aspects of a single device.

11. Source Code Repository

The usage of the source code repository provided by the control system supplier is essential for FESA class-development at GSI. The repository is separated into the following main-structures:

- “class”, used to store the FESA-classes itself
- “deploy-unit”, used to store deploy-units. Separated from the class-folder, since a deploy-unit can rely on more than one class.
- “driver”, all hardware-drivers, used in the different FESA-classes

Each group has its own sub-folder, in order to cluster device-software per group.

Furthermore each class, deploy-unit and driver in SVN has to provide the sub-folders “trunk”, “branches” and “tags”. According to commons SVN standards, releases of the class should be saved in “tags”, whereas the current development should be saved in “trunk” or “branches”.

A FESA-class only can be delivered and deployed as productive class, if it is stored in this source code repository, taking into account the above policy.

12. Logging System

Messages which are dedicated to the FEC specialist, the class developer or the hardware specialist can be sent using the Diagnostic Logging System [5].

13. Alarm System

Whenever a device is in a critical state, an alarm should be raised. This should be done by using the mechanisms provided by the Alarm System [6].

14. State Machines

A pre-configured state-machine will be provided by the fesa-core-gsi library. This implementation should be used, whenever a state-machine is needed.

I. Attached Documents

List of abbreviations for controls (Abbreviations_Controls.pdf).

II. Related Documentation

[1] CERN LEIR “Guidelines and conventions for defining interfaces of equipment developed using FESA” [EDMS: 581892].

[2] System for Nomenclatures of Accelerator Devices at FAIR & GSI https://www-acc.gsi.de/wiki/Accnomen

[3] F-DS-C-01e, “FEC software framework (FESA)”

[4] F-DG-C-03e “Accelerator Control System Architecture Guideline”

[5] F-DS-C-10e, FAIR Detailed Specification “Diagnostic Logging System”

[6] F-DS-C-09e, FAIR Detailed Specification “Alarm System”

III. Document Information

III.1. Document History

Version	Date	Description	Author	Review / Approval
0.1	19. Jan. 2011	Draft version	A. Schwinn	
0.2	24. Jan. 2011	Overall revision	S. Matthies	
0.3	28. Jan. 2011	revision	A. Schwinn	
0.4	04. Mar. 2011	revision (After Appl.-Team meeting1)	A. Schwinn	
0.6	18. Mar. 2011	revision (After Appl.-Team meeting2)	A. Schwinn	
0.8	31. May. 2011	Added Appl-Team field name conventions	A. Schwinn	
0.10	05. Dec. 2011	Moved parts into the Lab-Specific FESA-html-Doku	A. Schwinn	
0.11	16. Mar. 2012	Added TODO's and some minor notes	A. Schwinn	
1.0	07. Aug. 2012	Draft version for FAIR contracts, created .doc	CCT	
1.1	16. Aug. 2012	small corrections	A. Schwinn	