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
2020-01-07

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:
GSI's 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 organization chart, 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 developers of FESA equipment software and serves as guideline for all FESA development done within the context of the FAIR control system (see also F-DS-C-01e, “FEC software framework (FESA)”). 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 F-DG-C-03e “Accelerator Control System Architecture Guideline” fully applies.

3. Introduction

This document is based on the CERN LEIR “Guidelines and conventions for defining interfaces of equipment developed using FESA” (CERN LEIR “Guidelines and conventions for defining interfaces of equipment developed using FESA” [EDMS: 581892].). 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 easily be 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 is available. For FESA equipment software development at GSI it is recommended to use this template as a base. Wherever possible the naming conventions are supported by the underlying 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 equipment should be easy to correlate. As it may be necessary to correlate different property values with each other, the following information is required:
  - acquisition context, containing the time stamp and detailed beam information such as beam process and sequence IDs
  - 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).
- Interlock signaling: If a device is not operational because of a hardware failure (e.g. missing cooling water) the state must be signaled as an interlock in the Status property. The interlock is a device-specific state which depends on the device's features. Additional information about the device's state should be displayed using the detailed status bits.
- Ready for operation: A device is considered ready for operation if it is fully usable by applications and neither in a failure state nor operated locally. If the device is in a failure state or set to local operation the applications must be aware of this by information in the state of the opReady-bit. Whether a device is considered ready for operation must be confirmed per device. Some devices, e.g. bending magnets, may be powered off if they are ready for operation.
- 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.
- In order to ensure compatibility with operating software the data-type "float" must not be used for data fields. Instead the data-type "double" must be used.

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 within a property 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.
- Information about data units is kept within each value-item of a property and will be published with the data.

5. Property Types

In general FESA distinguishes two types of properties in a FESA class.

5.1. Acquisition Properties

The purpose of acquisition properties is to retrieve values from the device to the application.
Typical examples are the current and the voltage of a magnet's power supply, the actual position of a stepping motor or imaging data for beam diagnostics.
In exceptional cases acquisition properties should contain setting information as further explained in section 6.3.

5.2. Setting Properties

In contrast the purpose of setting properties is to transport setting information from the application to the device.
Examples for setting information are the next position of a stepping motor or the new values for current and voltage of a magnet's power supply.

6. Naming Conventions

A series of conventions are necessary to fulfill the requirement of a unified control interface. They concern names for FESA classes, device instances, property names, value-item and field names and more. For operational FESA3 software the names of classes, deploy-units, fields and properties and other design elements may not exceed a certain number of characters due to database restrictions. Table 1: List of allowed name lengths for operational FESA3 software lists the allowed lengths of names.

6.1. Syntax for naming Devices, Properties and Fields

- The **device class names** start with a capital letter and allow usage of camel case for significant parts as well as numbers and underscores.
  - Regular expression: [A-Z][A-Za-z0-9_]*
  - Examples: BdiStdProfile, SeptaMagnet, IonSource_2.
- The **device instance names** are predefined by the nomenclature–system. To find a proper device instance name, please refer to the person responsible for GSI nomenclatures. More information on the System for Nomenclatures of Accelerator Devices at FAIR & GSI is available at System for Nomenclatures of Accelerator Devices at FAIR & GSI https://www-acc.gsi.de/wiki/Accnomen.
- For **property names** the same rule as for device class names is applied . For the client the property name may be case insensitive. This implies that two properties can not be distinguished by case sensitivity.
  - Regular expression: [A-Z][A-Za-z0-9_]*
  - Examples: SummaryResult, ExpertSettings, ConfigureAxis_5
- The **field names** of a FESA class are written in camel 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 value-item name suffix is separated from its related field by an underscore, but underscores can be as well used for other purposes.
- The **value-item names** are strongly coupled to the referenced field names. They have to fulfill all naming restrictions of the field names.
- 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 2: List of predefined value item names shows all additional naming restrictions.

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

The length of names of certain FESA software design elements is restricted. The following table summarizes the limitations:

Name of FESA Design Element	**Allowed Length**
FESA Class	20
FESA Deploy-Unit	32
FESA Device-Instance	30
Custom Event	100
Custom Event Source	30
Data Element	60
Value-Item / Field	60
Global Instance	30
Hardware Address Hostname	30
Logical Event / Logical Event Group	30
Prio Management / Type	100
Property	30
Scheduling Unit	60
Scheduling Layer	100
Servername (<class name>. <hostname>)	50
Simulated Event	30

Table 1: List of allowed name lengths for operational FESA3 software

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.
****_min	Minimum value of a related field. See 6.4 for more details.
****_max	Maximum value of a related field. See 6.4 for more details.

Table 2: List of predefined value item names

6.2. 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. Equipment experts may need more information than normal users. With respect to this difference, the following naming convention is introduced:

- <PropertyName> (e.g. Setting, SetFlow) Properties with all information needed for operations.
- Expert<PropertyName> (e.g. ExpertSetting, ExpertSetFlow) Properties with additional information for equipment experts (the person who is responsible for the device)

There will also be an access restriction concept to ensure that only experts can change ExpertProperties.

Beside this naming-convention, the attribute “visibility” has to be used in order to classify each property correctly. Visibility can have the following values:

- **operational** - This property may be used by the operating.
- **deprecated** - This property is outdated and should not be used anymore. If backward compatibility is not required anymore this property should be removed.
- **expert** - This property only should be used by device experts.
- **development** - This property is currently under development.

6.3. Convention when adding Setting Information to Acquisition Properties

If applicable acquisition properties should contain information about the used setting values for a measurement, the names of the used fields and value-items have to be suffixed by the string “Set”. An example is the transfer of the voltage which was used during a measurement in an acquisition property. The name of

the corresponding value-item and the name of the referenced acquisition field should be “voltageSet”.

6.3.1. How to derive the Setting Information

The information about the used setting values for a measurement (`xxxxSet`) should be derived from a component as close as possible to the equipment (hardware, electronics, etc.).

- If a setting value is stored in the equipment and can be read back, the `xxxxSet` value should be derived from this hardware value.
- If a setting value can not be read back from the equipment but is stored in its device driver (a software component), the `xxxxSet` value should be derived from this value.
- If a setting value (the property value-item `xxxx`) was scaled to meet the hardware requirements before it was stored, it must be re-scaled for the `xxxxSet` value. In other words, the `xxxx` value and the corresponding `xxxxSet` value must be directly comparable. Example

  1. Setting.current = 47.11 A
  2. scaled and stored DAC-value = 0x17F0
  3. re-scaled Acquisition.currentSet = 47.13 A

  Note: Due to scaling and re-scaling, `xxxx` and `xxxxSet` may slightly deviate.
- If a setting value can not be stored in the equipment or its device driver, the `xxxx` value-item must be copied into the `xxxxSet` value-item. It is recommended to implement it as follows:

  1. RT-action that does the setting: `buffer = xxxx`
  2. RT-action that reads the acquisition: `xxxxSet = buffer`

  Note that the `buffer` is not multiplexed. Note also that the scaling / re-scaling requirement applies.

It is highly recommended to note in the FESA class documentation (the equipment model) where the setting information in the Acquisition property is exactly derived from.

6.4. Suffixes to add Information to a Value-Item

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 to foster standardization:

Data Field / Value-Item Name	Data Field / Value-Item 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. Make use of the xml elements “min-value-item” and “max-value-item”.
_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 a “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)
_units	char[10]	The units, in which the field’s value is saved. Makes use of the xml element “units-value-item”.

Table 3: Fieldname Suffixes

All values have to be presented in the base 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 characterize the main control value, but that is not modifiable by normal users.

For operational properties the appropriate suffix items such as minimal and maximal values as well as the unit of each value item need to be available whenever it is possible to provide them. The availability of such suffixes may influence the appearance of generic GUI applications.

7. Standard Properties

This section specifies standard properties for the device interface. The following property types should be present on all devices. They are part of a standard FESA interface at GSI:

Name	Realization	Type	Purpose
Status	GSI-Status-Property	Acquisition	Used to display the (cycle independent) overall status of the device. Detailed status information may be additionally added to this property. Detailed status should consist of an array of boolean values considered as detailed status information as well as a corresponding string array containing keys to illustrate the meaning of the detailed status information. The keys should be defined in coordination with CSCOAP.
ModuleStatus	GSI-ModuleStatus-Property	Acquisition	Gives detailed information on the state of 3rd party hardware and software components which are required to operate the device.
Power	GSI-Power-Property	Setting	Used to enable or disable a device
Reset	GSI-Reset-Property	Setting	Control property, used to reset to a starting state. Performs a hardware-reset and acknowledges failure states of the hardware-device, if any. Resets internal states of the front-end software to starting conditions. The current setting data is kept.
Init	GSI-Init-Property	Setting	Control property, used to initialize the hardware-device with default values from the device instantiation file. Additionally performs a “Reset” of the device (see property “Reset”)
Version	GSI-Version-Property	Acquisition	Returns the current software and hardware versions of a piece of equipment
Setting	GSI-Setting-Property	Setting	Used for setting hardware parameters for controlling the device
Acquisition	GSI-Acquisition-Property	Acquisition	Used for returning acquisition data which is retrieved from the hardware
DeviceDescription	GSI-DeviceDescription-Property	Global Acquisition	Provides general information about the binary. It as well keeps a list of all available properties and devices.

Table 4: Standard Properties of an operational Device Interface

In order to avoid documentation duplication, the GSI property documentation is stored directly within the GSI specific documentation. This documentation may be accessed using an HTML browser.

The location of the internal FESA documentation is:

FESA3 Installation	External Access
/opt/fesa/fesa-model-gsi/<FESA-version>/xml/design/docgen/html/design-doc.html	https://www-acc.gsi.de/data/documentation/fesa-gsi/metamodel/<FESA-version>/design-doc.html
/opt/fesa/fesa-model-gsi/<FESA-version>/xml/deployment/docgen/html/deployment-doc.html	https://www-acc.gsi.de/data/documentation/fesa-gsi/metamodel/<FESA-version>/deployment-doc.html
/opt/fesa/fesa-model-gsi/<FESA-version>/xml/instantiation/docgen/html/instantiation-doc.html	https://www-acc.gsi.de/data/documentation/fesa-gsi/metamodel/<FESA-version>/instantiation-doc.html

Table 5: Documentation Locations

To simplify developments the standard GSI class template contains the necessary standard properties and custom types per default.

7.1. Third-Party Modules

If a device requires specific third party hardware or software modules, the state of these modules has to be shown in the property “ModuleStatus”.

E.g. if White Rabbit based timing is used the status of the timing-receiver and the status of the Saftlib daemon has to be shown in “ModlueStatus”.

8. Design Decisions

Some elements have a GSI specific code generation or C++ implementation. The intention of this chapter is to explain why these design decisions were taken and how they are realized.

8.1. GSI specific Fields

8.1.1. The GSI-acquisition-context-field

The field “GSI-acquisition-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 field “GSI-acquisition-context” ensures that as well during a client’s get access all multiplexing data is obtained.
- A requirement of the application group of the Accelerator Controls and Electronics Department is to have access to the multiplexing context and/or measurement timestamp within the acquisition data. These stamps are required for correlation of measurement values.

8.1.2. The field “GSI-error_collection”

The field “GSI-error_collection” 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 log files.

- It allows to introduce a ring buffer which is not cycle dependent (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.

9. Custom Types

In order to avoid documentation duplication, the custom type documentation is stored directly within the GSI specific documentation of the according fields in the class design. The location of the documentation is supplied in chapter 6. Standard Properties.

This chapter provides additional documentation for all custom types.

9.1. TOL_CHECK_MODE

This constant defines possible modes to check whether a control value is inside the tolerance values (see Suffixes to add Information to a , “_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 6: Custom type field: TOL_CHECK_MODE, datatype: enum AQN_STATUS

9.2. AQN_STATUS

Possible values to describe the acquisition status of a value-item (see 6.4, “_status” suffix). If this suffix is missing, it means that no additional status information is provided for the corresponding value-item. If all bits are 0, this means that the corresponding value-item 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 property “Status”.
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 requested 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 7: Custom type field: AQN_STATUS, data type: bit-enum-32bit

10. 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 binary is of the type “rdaData” and can consist of any number of filter elements. If a property supports the usage of filters, it is recommended to use the standardized names for the 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 8: 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.

1 Only for acquisition properties

11. Conditions

At GSI system-wide unique (error) conditions should be used to help illustrate the state of a device. The term condition is used to not only illustrate errors and problems but also positive device states. Examples for conditions are out of range errors, illegal device states, device specific states, etc. .
Text messages can be retrieved by applications in several languages (e.g. english or german) for each condition.

A set of tools was developed to support the usage of conditions. These condition tools allow generating / updating / storing conditions in the database and to generate headers for inclusion in the device software. The database is the base for generating / updating conditions. An XML file is generated from the database to simplify creating / editing / updating of the conditions, their texts and descriptions.

It is recommended to display a device's (error) state by throwing appropriate conditions. The exception handling classes of the FESA framework are used to signal conditions.
To define a new set of conditions for FESA software a facility ID is required. To obtain one please refer to Udo Krause or Ludwig Hechler (CSCOFE).

12. Internal Software Design Recommendation

When implementing FESA device software a few issues should be considered to help ease testing the software from the beginning on and from different point of views.

Application developers desire permanent access to FESA devices to be able to adjust and test their applications. FESA device installations should be offered permanently in an environment well separated from the accelerator to the applications developers. Since providing dedicated equipment for testing the applications would be too costly, the FESA classes should run without equipment hardware on central servers. This implies that timing information will not be available on these servers. The FESA devices for testing purposes should be permanently installed in a test environment that is clearly separated from the production accelerator installation. This implies that accelerator timing information will not be available.

The idea is to offer FESA device software as mock devices. These mock devices should offer the same set of properties and a partially similar behavior as the productive version which is installed in the accelerator. This does not imply to develop the FESA software twice. Ideally the same FESA class implementation runs together with hardware equipment and within the test environment.

Using the same FESA class implementation in two environments can be achieved by separating the FESA class implementation from the interaction with the equipment: creation of an adapter layer which implements the interaction with the equipment.

It is generally recommended to

- Never directly access the hardware components in the FESA actions but use an intermediate software layer (“device adapter”) that provides access to the device’s properties (server actions, RT-actions). E.g. provide a procedure/method ‘setGain(float gain)’ which encapsulates writing to the gain-register of the equipment. Vice versa implement functions/methods which provide data read from the equipment.
- Design FESA device software that uses the timing system in the way that server actions run without the need to use real-time actions. This implies that in the test environment without timing different return values have to be expected.

In addition to the adapter for hardware access, a separate mock adapter should be foreseen which provides the same interface to the FESA class, but without implementing the interaction with the equipment hardware. Only the interface to the FESA device software needs to be served, that is the mock adapter only accepts and provides data. A simple use case for the FESA device mock adapter is setting reference values and retrieving actual values. Actual values may be either the reference values itself or a slight variation.

The intermediate adapter layer provides the properties of the hardware equipment for the FESA device software. When running the FESA device software in production the equipment adapter links the FESA class to the hardware equipment. When running the FESA device software within a test environment the mock adapter simulates the hardware equipment.

To distinguish the productive from the test environment different devices should be instantiated differently during start-up of the FESA device software in the specific environments. The recommendation is to mark test instances by a preceding x in the name. The reason is that the letter x is not used by GSI's nomenclature system by convention.

Examples: (taken from FESA device instantiation document)

- Production system: <device-instance name="PLLEVM1">
- Tests: <device-instance name="xPLLEVM1">

A full device simulation will not be easily realized and is therefore not expected. However a partial simulation of the main properties will help to test FESA device software at an early stage and to develop application software in parallel.

The concept of mock adapters should be used to provide FESA devices already in a very early stage of development, before the FESA class implementation is finished. To provide a test environment for applications development should start with the mock adapter.

The final recommendation for development of FESA device software is:

1. Define the main properties which will be used by the operations applications

2.
    a. Implement a basic FESA class that supports the main properties
    b. In parallel implement a mock adapter to simulate the equipment
3. Implement the adapter for hardware access
4. Refine the FESA class, enhance the adapters, implement other properties

13. Class Relationships

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

Additional information can be found in the FESA documentation.

13.1. Composition

If hardware equipment is clustered into different sub devices and all these devices shall be controlled by the same FEC, the developer should segment the equipment software to several devices and combine them 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

Example:

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

13.2. Association

If a 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 developer

Example:

Consider an imaging setup with a GigE camera, lens and image intensifier. A dedicated FESA binary 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 binary 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.

14. Documentation

FESA provides the possibility to document the interface of a FESA class. Description elements may be added to certain elements such as properties and data elements in the design of a FESA class. HTML documentation may be created on the push of a button.
This allows to provide additional information for the users of FESA class in a human readable form.
It is generally recommended to document the meaning of properties, data items etc. as it will improve understanding of the FESA software interface.

15. Source Code Repository

The usage of the source code repository provided by the control system supplier is essential for FESA 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 FESA 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 subfolder, 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 common SVN standards, releases of the class should be saved in “tags”, whereas the current development should be saved in “trunk” or “branches”.

A productive FESA binary should only be delivered and deployed if its sources are stored in this source code repository, taking into account the above policy.

16. Logging System

Messages which are dedicated to the FEC specialist, the developer or the hardware specialist can be sent using the Diagnostic Logging System F-DS-C-10e, FAIR Detailed Specification “Diagnostic Logging System”.

17. Alarm System

Whenever a device or a piece of equipment is in a critical state, an alarm should be raised. This should be done by using the mechanisms provided by the Alarm System F-DS-C-09e, FAIR Detailed Specification “Alarm System”.

Attached Documents

List of abbreviations for controls (Abbreviations_Controls.pdf).

I. 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”

I. 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 bugfix	A.Schwinn	
1.2	20.Feb.2013	re-definition of chapter 5.3, to solve conflict	A.Schwinn	
1.3	01.Mar.2013	Slight changes in chapter 5.2, rechecked usage of concept “FESA-class”	A.Schwinn	
1.4	12.03.2013	Complete revision, Clarification of terms and explanations, expansion of examples, extraction of references to not available FESA features, preparations for extensions, insertion of references to internal FESA documentation, update to GSI’s current organigram	S.Matthies	
1.5	21.03.2013 /	Section on documentation	S. Matthies	

	22.05.2013	introduced, Section on types of properties added Information on detailed status added		
1.6	26.09.2013	- “Setting” and “Acquisition” are optional	A.Schwinn	
1.7	24.06.2014	Section on mockup device software added	S. Matthies / U. Krause	
1.8	11.09.2014	Naming restrictions for class- and instance names of operational FESA3 software extended, Naming recommendation for test device instances added	S. Matthies	
1.9	04.05.2015	Finetuning, preparations for upcoming FESA release Document ported to LibreOffice Paragraph on Inheritance (postponed FESA feature) removed, clarity of presentation of standard properties, details	S. Matthies A.Schwinn	
1.10	02.09.2015	Added GSI new Properties	A.Schwinn	
1.11	23.02.2016	New: GSI's Conditions	S. Matthies	
1.12	13.07.2016	Adapted naming conventions (new: underscore allowable) More precise definition of Standard Properties such as Reset and Init	A.Schwinn	
1.13	27.06.2017	Information on Interlock and Ready for Operation definition added	S. Matthies	
1.14	11.05.2018	Chapter “6.3.1 How to derive the Setting Information” added	L.Hechler	
1.15	07.01.2020	8.1.1: GSI-multiplexing-field updated to current name GSI-acquisition-context-field	S. Matthies	