CERN
CH-1211 Geneva 23
Switzerland

Document Name
GUIDELINES_FESA_PROP_V_1.0.1

CERN Dep/Group

EDMS Document No.

Date : 2005-06-15

Deleted: 4

Deleted: 2

GUIDELINES AND CONVENTIONS FOR DEFINING INTERFACES OF EQUIPMENT DEVELOPED USING FESA (FIRST APPLICATION IN LEIR)

Abstract

The new controls infrastructure (with FESA, CMW and JAPC) supports composite properties. This document shall help unifying the way device interfaces are defined using this new facility. This document contains conventions for the following items:

- A common syntax for names of properties and fields
- Definitions of standard properties with well-defined meaning, standard format, and standard values
- Recommendations for defining device APIs with (composite) properties and fields.

This document covers only new developments based on the FESA infrastructure. The first application of this specification is the LEIR machine, but nothing is specific to LEIR, so these conventions should be eventually used also for other machines.

Prepared by :	***Checked by :***	
Vito Baggiolini Ana Guerrero Fabio Follin Eric Roux José-Luis Sanchez		

Approval List:

History of Changes

Rev. No.	Date	Pages	Description of Changes
0.1	14-02-05		First version in this format
0.2	22-02-05		Integration of comments, added sections on Requirements and Error Reporting. Added description of each standard property in section Standard Properties.
0.6	10-03-05		Integration of comments from equipment groups and further discussions with representatives from LSA, JAPC and FESA.
0.7	21-03-05		Changes in Status property; Described relation to Standard Alarm Interface; Clarification that suffixes are read-only. Improvement of documentation.
0.8	31-03-05		Modified DEVICE_STATUS; DEVICE_MODE according to last proposal. Corrected small inco herencies from last version.
0.9	02-04-05		Finalized specification for LEIR after conclusive meeting. Changes in Status property and in AQN_STATUS. Changed to fixed units instead of dynamic units, therefore removed the UNITS enum and the _units suffix.
1.0	15-04-05		Prepared for EDMS. Proof reading with correction of mistakes. integration of comments from readers.
1.0.1	15-06-05		Removed all the cycle-dependent fields from Status (because Status is not cycle-dependent); Increased the constant MAX_ERR to 32 (used by FESA only); modifed enum numbers of DEVICE_CONTROL {REMOTE=5, LOCAL=6}

Table of Contents

1. INTRODUCTION

This document presents guidelines for the definition of properties of operational interfaces for equipment modules developed with FESA. Such operational interfaces are used in the control room, both during daily operations and MD.

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 (1) coherent data that can be easily corellated with information from other sources, and (2) 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 naming conventions and a subset of the defined properties will be officially supported by the FESA development environment.

2. 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.

2.1 GENERAL REQUIREMENTS

- Equipment should provide a uniform controls interface. There should 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), except for static configuration data. It should also be possible to directly archive property values in a logging database, without having to process them (e.g. combine themit with other information).
  Examples of information required:
  - Acquisition data should 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
  - standard cyle ids, acquisition time, the beam id, etc. These values must have identical meaning and format across all kinds of devices.
  - Information about the units in a format that is suitable for treatment in a program (e.g. for programmatic comparison of data).
- Measurement values should 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 should be transmitted to the user, with of course an indication of the lower quality of the data.

2.2 ADDITIONAL REQUIREMENTS FOR OPERATIONS

- Supervision of device status should be supported. Operators need to have an overview which devices are in an abnormal state. If there is a problem with a device, error messages need to be available for further diagnosis by piquet people or equipment experts.
  It should be easy for operators to determine where the problem is: in the controls infrastructure, in the device itself or in the environment (e.g. an access interlock). This information is necessary for operators to decide on how to react to a problem, i.e. whether they shall try to fix the problem themselves (e.g. by doing a reset), whether they shall check the environment (in case of an external condition) or whether they need to call the equipment expert or the controls piquet.
- Supervision of control values should be supported. Operators need to quickly get an overview whether a control value is at its reference or not. For instance, it is important for operations to know when 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.

2.3 ADDITIONAL REQUIREMENTS FOR CONTROLS

- Data that belongs together should be kept together. If a coherent set of information is available on the front–end computer (e.g. in shared memory), it should be kept together and transmitted “en bloc” 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, not only for displaying in a GUI. While Strings are normally the best choice for displaying information in a GUI, numerically coded values (e.g. enumerations that map to integers) are more appropriate for machine treatment. Similarly, units must be specified an a format that makes it easy to do calculations. For instance, it must be possible to relate values from different equipment that are expressed in different powers of 10 (e.g. comparison of 100 kHz with 0.5 MHz).
- 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.

3. NAMING CONVENTIONS

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

3.1 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, not underscores. Examples: BdiStdProfile, SeptaMagnetic.

The **device instance names** implementing this interface follow the conventions described in [1] and [2]. They are not specified here any further.

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, not underscores. Examples: SummaryResult, ExpertSettings.

The **field names** within each property are written in mixed case, starting with a lower-case letter. To separate meaningful parts of the name a capital letter is used, not underscores. Examples: timeStamp, highVoltage.

Suffixes to field names are used to add information to the field. This is further explained in section 3.3. The suffixes are in mixed case, starting with a lower-case letter. They are separated from the field name by an underscore. Examples: highVoltage_min, highVoltage_tolAbs.

Constants are written in capital letters. To separate meaningful parts of the name an underscore is used in constants. Examples: INIT_DEV_STATE.

3.2 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 amount of information they need. Experts need more information than normal users. For properties with set access (R/W properties), it also allows to restrict to experts only the privilege of modifying certain fields.

Adding different levels of detail is done with a naming convention for the property names, which adds a prefix to the property name:

<Property> (e.g. Setting)	the property with all information needed for operations
Expert<Property> (e.g. ExpertSetting) Summary<Property> (e.g. SummarySetting)	adds information for equipment experts reduced contents settings (e.g. to reduce resources needed to store the property in a database).

3.3 SUFFIXES TO ADD INFORMATION TO A FIELD

A property typically contains several values from acquisition, one in each field. For instance, in an imaginary device, there might be a property with two fields:

Acquisition

- position
- highVoltage

It may be necessary to add information to each of these fields, e.g. a "status" information to determine whether the value in the field is valid. This should be done by adding a suffix to the field name, with the syntax **<field>_<suffix>**. For the above device, this would look as follows:

Acquisition

- position
- position_status
- highVoltage
- highVoltage_status

A number of suffixes have been defined for standardization (others may follow):

SUFFIXES		
Data Field Name	**Data Field Type**	**Data Field Description**
_status	AQN_STATUS	Additional information about problems or anomalies in the corresponding main value that must be taken into account when interpreting the main 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.
_min, _max	<same as property>	Mininmal and maximal values for c ontinuous properties.
_tolAbs	<same as property>	Absolute tolerance, expressed in the same units as the field it refers to. (taking into account a possibly dynamic unitExponent).
_tolRel	<% of property>	Relative tolerance, expressed in percent (%). The tolerance specifies how much an acquisition value can deviate from the control setting. If the aqn value is outside the value +/– _tolRel range, the _status field should flag an "DIFFERENT_FROM_SETTING" error.
_tolCheckMode	TOL_CHECK_MODE	Describes how the tolerance is controlled
_unitExponent	long	Exponent to express milli, kilo, mega, etc.For instance, a field "delay" expressed in milliseconds will have a field "delay_unitFactor" = –3 This suffix is only used if the exponent is really dynamic. Otherwise, the unitExponent are defined statically in the FESA device description.

An important aspect about suffixes is that they are read-only. In other words, a field with a suffix cannot be modified, and attempts to modify it will simply be ignored by the equipment server. The reason is that suffixes contain meta–data that characterizes the main control value, but that should not be modifiable by normal users. If it is necessary to change the values of the suffixes remotely, a normal field must be added to the equipment interface. The naming convention for such a field is to remove the underscore and to attach the suffix in mixed–case. For instance, to change the meta –data position_min, a field "positionMin" should be defined.

The following example illustrates how this could be implemented in practice:

```
Setting {
  position (read/write)
  position_min (read-only)
  position_max (read–only)
}
ExpertSetting {
  positionMin (read/write)
  positionMax (read/write)
}
```

The above specification fulfills the following needs. It contains meta–data (_min and _max) that is handy in a user interface to limit possible input values. It enforces that users with write–access privileges to the Setting property cannot change the _min, _max values. These values can only be changed by privileged users that are granted write–access to the ExpertSettings property.

4. STANDARD PROPERTIES

This section specifies standard properties for the device interfaces. As a general rule, there is no obligation to implement a property or field that is not applicable (e.g. if it does not make sense for the equipment). However, there is an obligation to use standardized names for properties or fields where they are available. In other words: if a standardized name/property exists for a concept or information, this name should be used, and no new name should be invented.

This document contains a non-exhaustive list of standardized properties and fields. More standard properties/fields can be added later.

4.1 STANDARDIZED PROPERTIES

The following properties should be present on all devices. They are part of a Standard FESA Interface.

- **Status** – the overall status of the device (not cycle dependent).
- **Acquisition** – information retrieved from the hardware (typically at every cycle).
- **Setting** – property used to control the device. It contains requested control values.
- **Reset** – used to (re–)initialize the device.

Variants of these properties can be used, e.g. ExpertSettings, SummaryAcquisition.

The two properties Setting and Acquisition are related, and it is recommended to use the same fields in both properties where this is applicable. For instance, for a Motor, both the Setting and the Acquisition property should contain a field “position”. The position field in Setting is used to control the position, whereas the position field in Acquisition is used to return the measured value.

In the following descriptions, fields with an asterisk (*) need to be there only if applicable. For instance, a cycle time stamp (cycleStamp) is applicable only for timing-dependent data.

4.1.1 STATUS

The status property contains information about the status of the device. It shall give the operator an overview whether the device works correctly, and if not, what is wrong.

If the Status property shows a warning or an error, the operator must also have enough information (1) to try to correct the problem and (2) to decide whether to call the equipment expert, the technical services or the controls piquet service. This information is contained in the mandatory errorMsg[] field, that contains corresponding error messages.

The information in the Status property is closely related to the Alarms an equipment can send out. In particular, the mandatory status field {OK, WARNING, ERROR } is automatically calculated by FESA according to the active fault states in the Standard

Alarm interface. Similarly, the errorMsg[] field reflects the descriptions associated with active Alarm fault states [3].

The mandatory fields “status” and “mode” are non-extensible and contain mutually exclusive values. Equipment specialists are encouraged to define additional fields “detailedStatus” and “detailedMode” which are equipment specific and contain more information about the state of the device.

The Status property shall be used to only report errors in the hardware, as described in section 6. Errors in the control system shall be reported via Exceptions.

The difference between the Status property and the _status suffix (c.f. type AQN_STATUS) is the following:

- Status represents the status of the whole device, whereas _status refers to one control value only. If several control values are contained in the Acquisition property, there may be a _status suffix for each of them.
- Status is not cycle dependent, whereas _status can be cycle dependent.

Property Name : Status			
Multiplexing	**Subscribable**	**Usage**	**Get / Set**
NONE	Yes	Standard Operations	Get
Data Field	**Data Field Type**	**Data Field Description**	
acqStamp	long long	Acquisition Time Stamp (UTC in nanoseconds).	
status	DEVICE_STATUS	Summary of the device status (OK, WARNING, ERROR). Should be coherent with active alarm fault states.	
mode	DEVICE_MODE	Summary of the device mode (ON, OFF, STANDBY, etc).	
control	DEVICE_CONTROL	Device control mode; describes whether the device can be controlled remotely.	
errorMsg	char[MAX_ERR][MSG_SIZE]	Active error messages (if the status field flags an error or a warning). Should be coherent with active alarm fault states.	
externalCondition	Boolean	True if an external condition (e.g. a safety interlock) is active. An external condition is a situation in the environment of the device that prevents the device from being operated normally. It tells the operator that trying to use the device would create problems (e.g. damage to material or persons). In case of an external condition, the operator should check the environment, not call the equipment specialist or piquet service. An external conditionshould not be used to signal a failure in the device. Even failures in the services needed for a device (e.g. lack of cooling water for a magnet) should not be flagged as an external condition, but asdevice errors using the appropriate fields (status, detailedStatus, errorMsg).	
detailedStatus	<some Bit Enum Type >	Equipment–specific, detailed information about the status. (Not mutually exclusive). This value is used to complement the information contained in the status field.	
detailedMode	<some Bit Enum Type >	Equipment–specific, detailed information about the mode. (Not mutually exclusive). This value is used to complement the information in the mode field. Normally, the mode field will have the value “OTHER”.	
Error Message		**Problem Description**	

Deleted: cycleName * ... [1]

(*) fields that need to be there only if applicable (e.g. if the device is cycle-dependent).

This information is aimed at standard users. If more information is necessary, it can be put into a property named ExpertStatus.

4.1.2 ACQUISITION

This is the property used by operations to retrieve data from the hardware. It also contains information on the circumstances (e.g. timing context and control values) under which the acquisition was made.

This property is used both for measurement data and for supervision of control values.

Property Name: Acquisition (to be decided)			
Multiplexing	**Subscribable**	**Usage**	**Get/Set**
User	Yes	Standard Operations	Get
Data Field	**Data Field Type**	**Data Field Description**	
cycleName *	char[NAME_SIZE]	Name of the cycle (e.g. SFTPRO, EASTA).	
beamID *	long	Unique identifier of a beam type (not instance). It represents a beam production process through the succession of machines. It is repeated every time the beam is produced.	
cycleStamp *	long long	(Unique) time stamp that identifies a unique cycle occurrence. UTC (in nanoseconds). Under certain circumstances, this is automatically provided by FESA (c.f. Annex A).	
acqStamp	long long	Acquisition Time Stamp (UTC in nanoseconds).	
...			
Error Message		**Problem Description**	

(*) fields that need to be present only if applicable (e.g. if the device is cycle-dependent).

This information is aimed at standard users. If more information is necessary, it can be put into a property named ExpertAcquisition.

4.1.3 SETTING

This property is used to control the device. It contains all requested settings of the device which the operators are allowed to modify in daily operations.

The value of a field read back from the FESA server is *exactly* the same as the value previously set. This is even true if a DAC is used in the underlying hardware, which due to its limited resolution cannot handle a precise floating point number. In other words, the control value in a Settings field may differ slightly from the value read back from the corresponding DAC. If an expert needs to know the value in the DAC, a separate field must be used.

Note that a requested setting may be completely different from the actual control value in the hardware (as returned in the Acquisition property). The reason is the following. When the user does a Set action, the requested control value is first stored in memory in the device server. It is only downloaded to hardware when the correct timing event occurs, which may be only after a while.
In other words, if a user does a Get operation on the Setting property and on the Acquisition property, corresponding fields do not necessarily return corresponding values. However, a Get operation on the Setting property always returns the last value send to the property with a Set operation.

Property Name: Setting			
Multiplexing	**Subscribable**	**Usage**	**Get/Set**
User	Yes (onChange upon Set)	Standard Operations	Get and Set
Data Field	**Data Field Type**	**Data Field Description**	
acqStamp	long long	Acquisition Time Stamp (UTC in nanoseconds)	
mode *	DEVICE_MODE_SETTING	Used to control the device mode (c.f. Status.mode). This field is only present if operations can change the device mode.	

...	...
Error Message	**Problem Description**

(*) fields that need to be present only if applicable (e.g. if the device is cycle-dependent).

This information is aimed at standard users. Privileged users may need to change other information that should not be accessible to standard. For this purpose, it is recommended to use a property named ExpertSetting.

4.1.4 RESET

This property is used to reinitialize the device. It is a write-only operation. The progress of re-initialization can be read back from the Status property (in the mode field). Note however, that it may not be possible to read values from the device while it is reinitializing.

Property Name: Reset			
Multiplexing	**Subscribable**	**Scope**	**Get/Set**
None	No	Device	Set
Data Field	**Data Field Type**	**Data Field Description**	
reset	boolean	Dummy value as requested by FESA implementation	
Error Message		**Problem Description**	

5. CUSTOM TYPES

This chapter describes the custom types used in this interface.

We will group these constants in the 3 categories available in the FESA model:

- The **Constants,** used here essentially to define array dimensions
- The **Enum Types**, used to select/reflect a single option/state within a list.
- The **Bit Enum Types**, used to select/reflect one or several options/states within a list.

5.1 CONSTANTS

Constant Name	Type	Value	Constant Usage
NAME_SIZE	unsigned-int	64	Used to define string length for names
MSG_SIZE	unsigned-int	256	Used to define string length for messages
MAX_ERR	unsigned-int	32	Used to define the buffer of active error messages

Deleted: 4

5.2 ENUM TYPES

5.2.1 DEVICE_STATUS

Possible (mutually exclusive) values to describe the device status.

Enum Name	Enum Type	Enum Value Usage
DEVICE_STATUS	Enum	Used to define the status of a device.
Enum Identifier	**Enum Value**	**Enum Identifier Usage**

UNKNOWN	0	The device status is unknown.
OK	1	The device is in a fully operational state
WARNING	2	The device is not fully operational; A device in WARNING state can still be used operationally, but the operator should be informed of a problem that might become worse. Details are explained in the errorMsg field.
ERROR	3	The device is in a fault state. Details are explained in the errorMsg field.

5.2.2 DEVICE_MODE

Possible (mutually exclusive) values to describe the operational mode of the device.

Enum Name	**Enum Type**	**Enum Value Usage**
DEVICE_MODE	enum	Used to define the mode of a device.
Enum Identifier	**Enum Value**	**Enum Identifier Usage**
UNKNOWN	0	It is not possible to determine the mode.
ON	1	The device is in a fully operational state.
OFF	2	The device is turned off.
STANDBY	3	The device is in a stand-by mode. This mode is a sort of "parking mode" in which the device can stay for hours or even days. It is defined by the following characteristics: It is safe, it does not wear out, it consumes little energy. Furthermore, it takes a short time to go from STANDBY to ON mode.
INITIALIZING	4	The device is initializing. This field is activated when the Reset property has been Set, and remains active until the mode has reached one of the other values (e.g. STANDBY).
OTHER	127	The device is in an other, equipment-specific mode, to be looked up in the field detailedMode.

5.2.3 DEVICE_MODE_SETTING

A Enum Type used to control the operational mode of the device. Its values are a subset of those in the DEVICE_MODE type.

Enum Name	**Enum Type**	**Enum Value Usage**
DEVICE_MODE_SETTING	enum	Used tocontrol the mode of a device.
Enum Identifier	**Enum Value**	**Enum Identifier Usage**
ON	1	Used to set the device to mode=ON
OFF	2	Used to set the device to mode=OFF
STANDBY	3	Used to set the device to mode=STANDBY

5.2.4 DEVICE_CONTROL

Possible values to describe the control mode of a device. Currently only two control modes (LOCAL, REMOTE) are defined, but it is envisageable to introduce others.

Enum Name	**Enum Type**	**Enum Value Usage**
DEVICE_CONTROL	enum	Used to describe the control mode of a device.
Enum Identifier	**Enum Value**	**Enum Identifier Usage**
REMOTE	5	The device can be controlled normally through the control system
LOCAL	6	The device can be controlled only locally (but it can be accessed in read-only mode via the control system).

Deleted: 0

Deleted: 1

5.3 BIT ENUM TYPES

5.3.1 AQN_STATUS

Possible values to describe the acquisition status of a field (in the _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, it 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.

The difference between the Status property and the _status suffix is described in the section on the Status property.

Bit Enum Name	**Bit Enum Type**	**Enum Value Usage**
AQN_STATUS	bit-enum-32bits	Additional information about problems or anomalies in the corresponding main value that must be taken into account when interpreting the main value.
Bit Enum Identifier	**Bit Enum Value**	**Enum Identifier 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 (forcontinuous 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 changingin 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 should be turned on.
TIMEOUT	2^5	A timeout occurred, because the property did not reach the reqested newcontrol 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.
<equipment–specific>	>= 2^{16}	Equipment-specific problem indicators. A bit set to 1 should indicate a problem. If the whole xxx_status field = 0, this indicates that the status is OK.

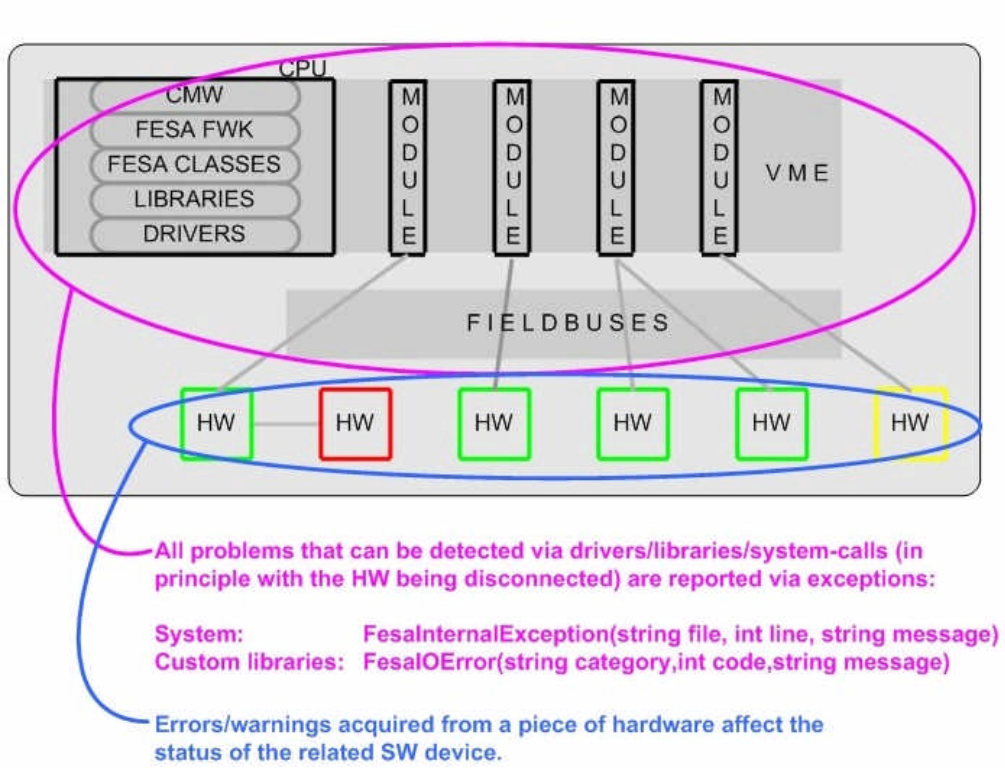
5.3.2 TOL_CHECK_MODE

This constant defines possible modes to check whether a control value is inside the tolerance values.

Bit Enum Name	**Bit Enum Type**	**Enum Value Usage**
TOL_CHECK_MODE	bit-enum-16bits	Used to give information on how the tolerance fields should be used to calculate the xxx_status information.
Bit Enum Identifier	**Bit Enum Value**	**Enum Identifier Usage**
ABS	2^0	Use the absolute tolerance _tolAbs
REL	2^1	Use the relative tolerance _tolRel

6. ERROR REPORTING

There are two kinds of errors that can occur: (1) problems in the hardware and (2) problems in the control system. Problems in the hardware are reported in the standardized Status property, while problems in the control system are reported by throwing an exception in the Get or Set call, or through the appropriate subscription handler call-back.



7. REFERENCES

[1] Basic Syntactic Rules for naming of LHC entities and their parameters for the CCC [EDMS: 473086]

[2] Naming of LHC entities and their parameters for the CCC [EDMS: 473091]

[3] FESA Interface Specification: “Interface with the Alarm System”

Page 9: [1] Deleted **vbaggiol** **6/14/2005 7:54:00 PM**

cycleName *	char[NAME_SIZE]	Name of the cycle (e.g. SFTPRO, EASTA).
beamID *	Long	Unique identifier of a beam type (not instance). It represents a beam production process through the succession of machines. The same number is repeated every time the same kind of beam is produced.
cycleStamp *	long long	(Unique) time stamp that identifies a unique cycle occurrence. UTC (in nanoseconds). Under certain circumstances, this is automatically provided by FESA (c.f. Annex A).