THE DIRECTORY SERVICE FOR THE CERN ACCELERATOR CONTROL APPLICATION PROGRAMS

J. Cuperus, M. Peryt, E. Roux, J. Schinzel

CERN, Geneva, Switzerland

ABSTRACT

The controls infrastructure of CERN's accelerator complex is converging towards a unified architecture. High level control room application programs are written in Java and they interact with the accelerator environment through several interface libraries. One of these libraries is the Directory Service. This library provides information about the accelerator equipment through a number of Java classes that get their data from a relational Configuration Database via JDBC.

On a higher level, application programs are called from a generic console manager that can be configured for an operational environment through the Directory Service, which provides information about menu trees of software programs to be called plus the attributes and parameters of these programs.

INTRODUCTION

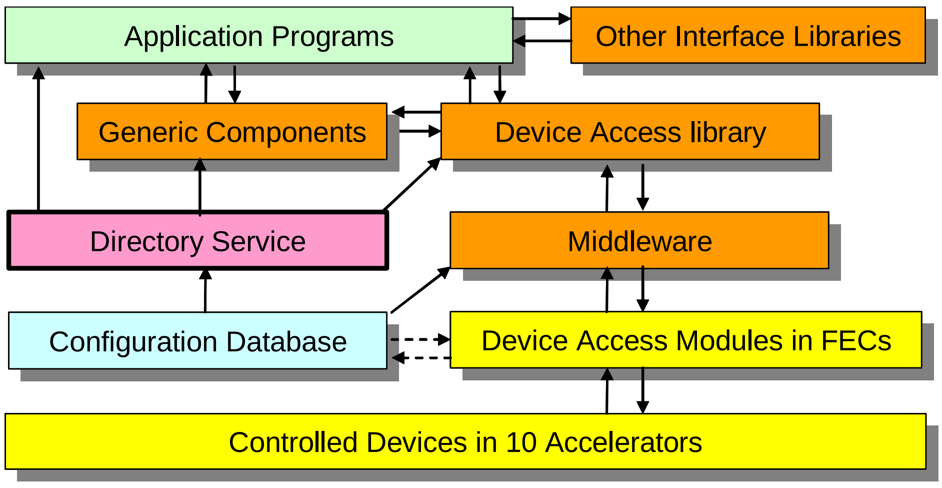


Fig.1: Block diagram of the CERN Accelerator Control System

Fig.1 shows the block diagram of the control system. The Directory Service library, written in Java, gets its data from the Configuration Database [1] via JDBC (Java Database Connectivity) and delivers them, in a suitable form, to the Java application programs. This provides information about devices, and the way to control them, in 10 accelerators: a proton linac, an heavy-ion linac, the PSB booster, the 30 Gev PS accelerator, the 450GeV SPS accelerator and the 2x7TeV Large Hadron Collider (LHC) storage rings, plus the interconnecting beam lines, an Antiproton Decelerator (AD), an ion accumulator ring (LEIR), the isotope facility (ISOLDE), and the new electron linac test facility (CTF). Note that many subsystems, like the cryogenics for the LHC, are controlled by industrial electronics, with only summary control from the accelerator control system. The application programs may access any number of other access libraries for timing, logging, alarms, … Solid arrows indicate operational data flow. Dotted arrows indicate design-time data flow.

DEVICE ACCESS

The controlled devices are power supplies, actuators, pumps, valves, beam monitors, timings, function generators, … They are controlled via device access modules. The accelerators up to the

PS are mainly controlled by General Modules (GM) [2]. New device access modules follow the Front End System Access (FESA) model [3]. The SPS is mainly controlled by yet another technology: SL-EQUIP[4]. Both GM and FESA are object-oriented with similar devices grouped in device classes and their design is stored in the database, so that full information is available. SL-EQUIP is not object oriented, except for some beam-monitoring modules, and most of its parameters are not centrally available. SL-EQUIP and GM modules are destined to be re-written in FESA but the priority is now for new design for the LHC and replacement may take many years. The Directory Service provides the available information in a uniform way, independent of the underlying access technology.

Summary information about devices, and how to access them, is available in records of the database table DEVICES with among the important fields: *devicename, computer, server, accelerator, timing, classname,* … Note also that most of the database tables, including DEVICES have a *description* field, in general 80 characters long, which is important for documenting the system and to give information to the accelerator operators. However, these fields are not essential for operation and, unfortunately, they are not always filled in.

CONTROLDEVICE CLASS AND DEVICE SETS

All available information about a device can be obtained, directly or indirectly, through the Java class ControlDevice, which has some 30 public methods for getting data or instances of related Java classes. An instance of the ControlDevice class can be obtained with:

- ControlDevice device = ControlDevice.getControlDevice(String deviceName);

An array of these devices can be obtained with:

- ControlDevice[] devices = ControlDevice.getControldDevices(String query, String order);

Where *query* is any SQL query statement against the columns of table DEVICES, like: query = ” accelerator=’PSB’ and classname=’POW’ “; and *order* is a comma-separated list of table columns according to which you want the result ordered.

A more complicated set of devices is the working set, which is an array of array of devices (see Fig.2).

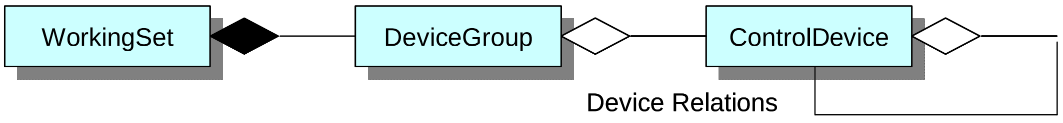


Fig.2: Composition of working sets and relations between devices.

The working set itself can be obtained with:

- WorkingSet wset = WorkingSet.getWorkingSetByName(String wsetName);

It is also possible to get a working set by query:

- WorkingSet wset = WorkingSet.getWorkingsetByQuery(String query);

For queried working sets, the devices are grouped by device class.

For each device, it is possible to get the associated devices and the relation they have to the device (e.g. timxxx is start-trigger for powyyy …).

DEVICE CLASSES AND PROPERTIES

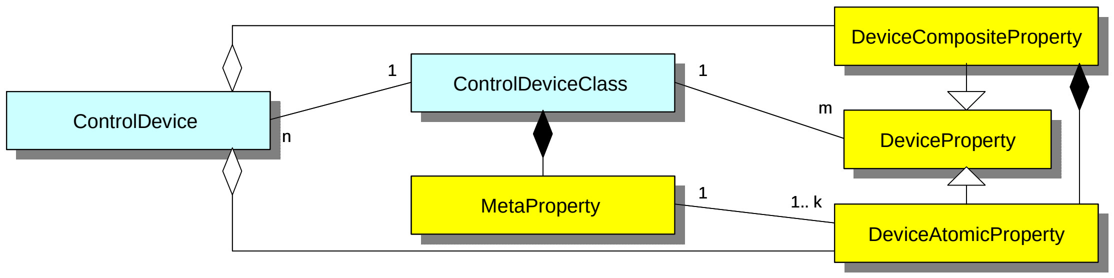


Fig.3: Properties for a device.

Our devices belong to a GM or FESA device class or else can be grouped into a number of virtual device classes. Device classes can have properties for getting information about the devices from the device access packages in the FECs. The association between device classes and properties can be ambiguous for 4 reasons:

1. Our system evolves and classes can have versions with different properties installed in different parts of the system.
2. Our equipment diversity is very large and similar devices, but with slightly different properties and implementations, may be grouped in the same device class.
3. Properties can have attributes (like minval and maxval) that may be associated with the device instance rather than with the device class.
4. Virtual device classes have no direct information about the ‘properties’ of their devices (but they can have MetaProperty information - see further).

For these reasons, device properties are associated with ControlDevice rather than with ControlDeviceClass, at least from the point of view of the user of the Directory Service.

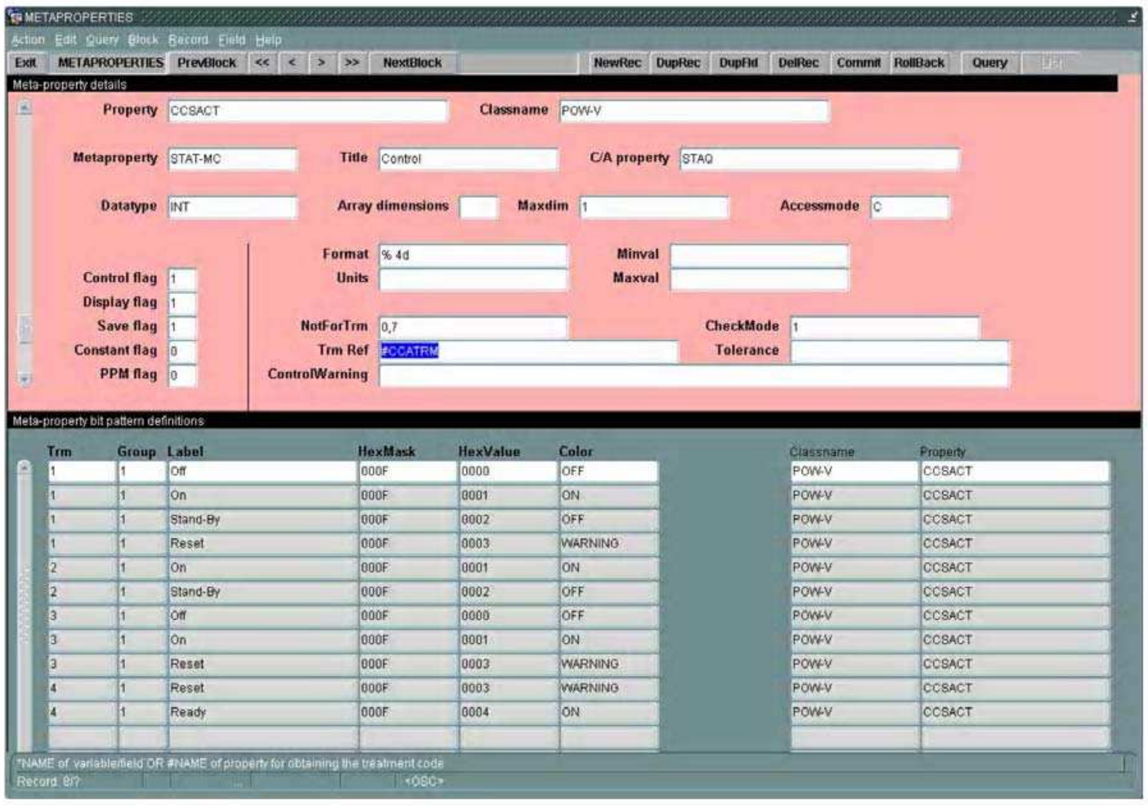
Generic components and programs must be able to work with devices and classes they know nothing about. So we need information in very general terms. This information can be obtained from the Java class MetaProperty.

META PROPERTIES

Each ControlDeviceClass can be associated with a number of MetaProperty instances. A metaproperty indicates an important function of the class in general terms, understandable by generic programs that have no a priory knowledge about the ControlDeviceClass and its properties. What the nature of the metaproperty data is can best be illustrated by showing a database input form used for creating and updating the metadata (fig. 4).

The example is for MetaProperty ‘STAT-MC’ which is recognized by generic programs and components as meaning: main status control. The device can be controlled through DeviceAtomicProperty CCSACT and the effective status of the equipment can be obtained with STAQ.

For the status word, the meaning of the bits can be described. For a value metaproperty (with a name like ‘VAL-MC’ or ‘VAL-A’, …) the Units, Minval, Maxval are more likely to be filled in. In many cases, the data filled in can be either a literal value, a *reference to a GM or FESA variable which the directory service will resolve automatically, or a #reference to a DeviceAtomicProperty that the application program has to call to obtain the value.


METAPROPERTIES
Action Edit Query Block Record Field Help
Exit METAPROPERTIES PrevBlock << < > >> NextBlock NewRec DupRec DupFld DelRec Commit RollBack Query

Meta-property details

Property	CCSACT	Classname	POW-V		
Metaproperty	STAT-MC	Title	Control	C/A property	STAQ
Datatype	INT	Array dimensions		Maxdim	1
Accessmode	C				
Format	% 4d	Minval			
Units		Maxval			
Control flag	1	Display flag	1	Save flag	1
Constant flag	0	PPM flag	0		
NotForTrm	0,7	CheckMode	1		
Trm Ref	#CCATRM	Tolerance			
ControlWarning					

Meta-property bit pattern definitions

Trm	Group	Label	HexMask	HexValue	Color	Classname	Property
1	1	Off	000F	0000	OFF	POW-V	CCSACT
1	1	On	000F	0001	ON	POW-V	CCSACT
1	1	Stand-By	000F	0002	OFF	POW-V	CCSACT
1	1	Reset	000F	0003	WARNING	POW-V	CCSACT
2	1	On	000F	0001	ON	POW-V	CCSACT
2	1	Stand-By	000F	0002	OFF	POW-V	CCSACT
3	1	Off	000F	0000	OFF	POW-V	CCSACT
3	1	On	000F	0001	ON	POW-V	CCSACT
3	1	Reset	000F	0003	WARNING	POW-V	CCSACT
4	1	Reset	000F	0003	WARNING	POW-V	CCSACT
4	1	Ready	000F	0004	ON	POW-V	CCSACT

*NAME of variable/field OR #NAME of property for obtaining the treatment code

Record: 8/? <OSC>

Fig.4: Input Form for metadata.

Note the use of treatment codes (Trm) for distinguishing between different types of similar equipment. This is preferred over subclassing which would get out of hand due to our large variability of equipment. These treatment codes are for internal use only and are not seen explicitly by the users of the Directory Service.

For GM and FESA equipment, these MetaProperty attributes complement the information available from the DeviceAtomicProperty attributes. For the SL-EQUIP implementation, it is the only potential property information available.

CONSOLE MENUS

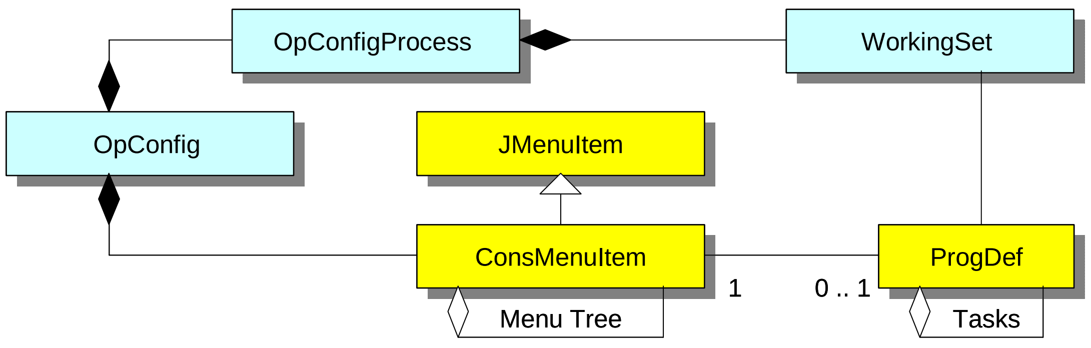


Fig.5: Console Menu and Working Set Classes.

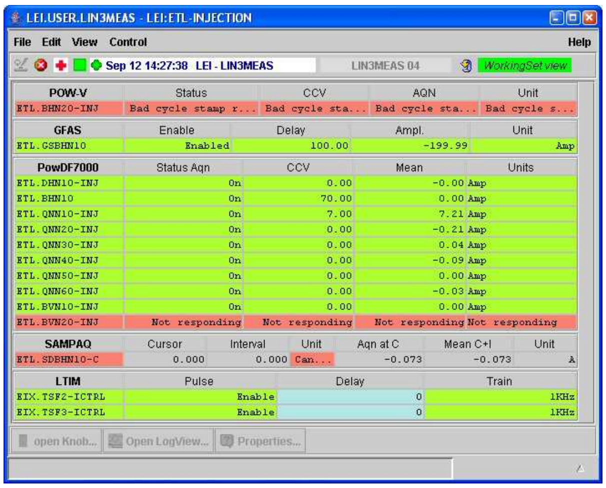
An operational configuration, or OpConfig is mainly a label for an environment in which to work, like CPSOP which denotes operation of the CPS accelerator. An OpConfigProcess is a part of this operation, like particle ejection.

For each OpConfig, a number menu trees are available from the directory service (Fig.5). These menu trees can be added directly to the menu bar of the Java console. The leaves of these menu trees call application programs with defined arguments. Some of these application programs can be Tasks, which are ordered lists of application programs, to be executed sequentially.

The first menu tree is dedicated to calling hand-made working sets with a generic working set display program and control. This program provides an overview of the status of the devices in the working set and allows manual control of the devices.

GENERIC COMPONENTS AND PROGRAMS

A number of generic programs and components (Java Beans) allow display and control of devices with information about the devices provided by the directory service (see Fig.6). The set of devices to be worked on is provided by a named or queried working set. Then, for each device, the ControlDeviceClass can be obtained with its list of MetaProperty, which provides the DeviceAtomicProperty instances for monitoring and controlling the devices.



POW-V	Status	CCV	AQN	Unit
ETL.BHN20-INJ	Bad cycle stamp r...	Bad cycle sta...	Bad cycle sta...	Bad cycle s...

GFAS	Enable	Delay	Ampl.	Unit
ETL.GSBHN10	Enabled	100.00	-199.99	Amp

PowDF7000	Status Aqn	CCV	Mean	Units
ETL.DHN10-INJ	On	0.00	-0.00	Amp
ETL.BHN10	On	70.00	0.00	Amp
ETL.QNN10-INJ	On	7.00	7.21	Amp
ETL.QNN20-INJ	On	0.00	-0.21	Amp
ETL.QNN30-INJ	On	0.00	0.04	Amp
ETL.QNN40-INJ	On	0.00	-0.09	Amp
ETL.QNN50-INJ	On	0.00	0.00	Amp
ETL.QNN60-INJ	On	0.00	-0.03	Amp
ETL.BVN10-INJ	On	0.00	0.00	Amp
ETL.BVN20-INJ	Not responding	Not responding	Not responding	Not responding

SAMPAQ	Cursor	Interval	Unit	Aqn at C	Mean C+I	Unit
ETL.SDBHN10-C	0.000	0.000	Can...	-0.073	-0.073	A

LTIM	Pulse	Delay	Train
EIX.TSF2-ICTRL	Enable	0	1KHz
EIX.TSF3-ICTRL	Enable	0	1KHz

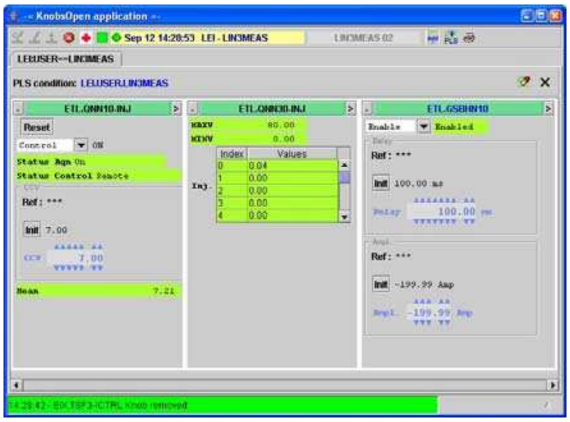


Fig.6: At left, an example of a generic working set display program that displays the status and values of the devices in a small working set for injection into LEIR. Knobs for controlling individual devices can be created. At right, a container with such generic knob components.

Support is also given for interactive browsers where the device structure can be explored and detailed information can be obtained from the devices. This is useful for trouble shooting and for familiarizing the operators with the accelerators.

Up to now, we really only talked about controlling electronic extensions of devices, sitting in a rack somewhere. It would be useful to relate these extensions with the accelerator hardware, like magnets, RF cavities, and beam monitors. This would make it possible to make a generic Synoptic program that could display data about any set of devices related to their position in the accelerators. This information about the relations between electronic and accelerator devices exists partially in automatic beam steering programs (ABS) and it would be very useful to extend this information and make it generally available through the Directory Service.

INITIALISATION PARAMETERS

Any status or value control parameter for a device can be stored as 'REFERENCE'. This includes arrays for function generators. A particularity of our control system is that it allows to define up to 64 virtual machines (VM) which configure our system of interlocking accelerators for a particular operation like injecting protons in the LHC. Most of our accelerators can be reconfigured, on a pulse-to pulse basis from one VM to another. There is a REFERENCE set for all devices for each VM. Generic operations like set-in-reference or get-from-reference can be done for individual devices or for any set of devices.

Particular sets of values can be stored in named ARCHIVES which contain a date, a comment, and the settings for a named working set and a named VM. These archives can be used for setting up the accelerators for a particular operation.

The directory service provides the interfaces for storing and retrieving references and archives. For the moment it is only used for GM devices but it could easily be used for all devices. The mechanism should however not be used for storing the functions for controlling the beam in storage rings, for which more secure mechanisms are available.

DOCUMENTATION

The public Java Classes and Methods of the Directory Service can be consulted on the Web through documentation generated with the JavaDoc facility. This is enhanced by providing a detailed description of each class, method, and method argument. When writing applications with a Java design tool, this documentation can be consulted on-line as part of the design process.

CONCLUSIONS

Some of the accelerator equipment is over 40 years old and some is brand new, with a great diversity of equipment types and control technologies. The Directory Service presents this diversity in a uniform way to the application programs so that generic programs and software components can be written that can handle many control and display tasks, based on the configuration data.

REFERENCES

[1] The Configuration Database for the CERN Accelerator Control System. J.Cuperus, R.Billen, M.Lelaizant, Icalepcs2003, Geongeoeng, South Korea.

[2] Process Equipment Data Organization in CERN PS Controls. L.Casalegno, J.Cuperus, CH.Sicard, Nuclear Instruments and Methods in Physics Research A293 (1990), pp412-415.

[3] Equipment Software Modeling for Accelerator Controls. M.Arruat, S.Jackson, J-L.Nougaret, M. Peryt, Icalepcs2005, Geneve, Switzerland.

[4] Accessing equipment in the SPS-LEP controls infrastructure: the SL-EQUIP package. P.Charrue, CERN-SL-Tech-Note-93-086-CO.