

DIY Zoning: Components

by Vadim Tkachenko

1. Macro Components

Macro components have one-to-one mapping to operating system processes. Basically, there's a shell script for each of them. The components may communicate to each other over the network (currently mostly TCP and some UDP, see [Protocols](#) section for more details).

1.1. Configurator

Currently, this is the most derelict component - it was created in old times when DZ was simple, and there were no active users. Project progress calls for improvement, and the 0.1p7 release is the target release for the new configurator. The configurator as it exists today will be retired completely.

1.2. DAC

DAC (Data ACquisition) module. Actually, it does more than that, partly due to design, partly due to technical limitations.

Following is a list of services DAC currently provides:

- Data acquisition. Currently, this is implemented using 1-Wire® devices, but there's nothing that prevents it from using different hardware.
- Actuator control. This service doesn't really belong here, but it has to be in the DAC due to the way Java 1-Wire® API is implemented.
- Data logging. Two alternatives exist, one is based on [RRDTool](#), the other is based on [JRobin](#). It is a design decision for this service to be here, though it is possible to have it elsewhere.
- Data broadcast and control. In other words, communicating with other DZ modules.

1.3. CORE

This module is the heart of DZ. It performs data gathering from one or more DACs, business logic of zone control, and issuing control signals to the dampers and the HVAC unit[s].

1.4. Scheduler

Currently, the scheduler is an integral part of CORE. However, it is clear at this point that it doesn't really belong there and can be made an external part, which will greatly improve flexibility and interoperability.

1.5. VIEW

Obviously, the View is a representation of a system and the focal point for real-time system control.

Currently, it is implemented as a Java Swing application, but it should be noted that it is also possible to have the view to be the adapter between CORE and actual hardware, such as a touch-screen wall controller.

2. Micro Components

Initially, DZ design was much more monolithic than it is today. As the project develops, a need for modules to move here or there becomes more and more apparent, therefore, there is a plan now to split the system into micro components and provide the data flow based framework where the components would be defined as requiring certain inputs, and providing certain outputs. The framework would then configure the runtime environment to place the modules where they requested to be, and provide proper data channels between them.

DAC is already implemented this way, now all that's left is to provide a way to migrate micro components across macro component boundaries.

Following is an incomplete list of micro components, in no particular order, with no regard to inheritance (there will be a lot of it):

- A sensor (temperature, humidity, pressure)
- A switch
- Data source (producer)
- Data sink (logger). At this time, the following loggers are available as a part of [v.2](#) transition: trace file, RRD (Jrobin and RRDTTool based), xAP, xPL.
- PnP advertiser
- PnP listener
- Secure connector
- Connector factory
- Process controller (hysteresis, PID)
- Hardware Abstraction Layer

DIY Zoning: Components

- Arrival/departure watcher
- Pattern analyzer

Additional information can be found in [v.2 Notes](#) chapter.

FIXME (VT):

Provide dependency graph, "requires" and "provides".

© 2004 Vadim Tkachenko