How To Develop Driver, Part 1

In the first part of this lesson, we will look at the basic principles in developing a Plugin of type Driver, designed for integration of external systems.

Specific extensions of the Driver, such as integration of access control system or client integration of camera, will be covered in separate lessons.

A Driver is a component connecting the C4 System with an external system. Its task is to monitor and control individual parts of the external device in the C4 System environment. A Driver communicates with the external system non-stop, from initiation of the communication until its termination.

Each instance of the Driver always communicates with a set of devices that have one specific network connection point. If there are multiple devices with various connection points, the Driver is launched in several instances, for each connection point separately. This means that the C4 System automatically creates and manages as many instances of the Driver as there are connection points defined in the device tree. It is important for the developer to take these basic principles into account when designing the integration of an external system. It is strongly recommended not to integrate all managed devices with a single instance of the Driver. Such an approach puts an extreme load on the internal resources of the C4 System, which are dedicated to managing the integration of external systems.

In the C4 System, each instance of the Driver has separately defined configuration of the device that it manages. This configuration contains all subsystems of the particular device. Therefore, if we have two devices that differ in configuration (as we can see in our case, one device has one Output, and the other has two Outputs), their trees in the C4 System will also differ. The Driver must be programmed to be able to work with all possible configurations supported by the device manufacturer.

If developers use Gamanet's generated functions, they will have prepared codes for various types of subsystems at their disposal. Their task is to write a code for a specific object, for example an Output, in a way that allows it to work in multiple instances. When initializing the Device Context, these instances are automatically created according to the configuration in the device tree. This significantly saves the developer's work and time. Considering the Driver's architecture, it is highly recommended to avoid using static variables or objects when programming.

Device Logic is a module programmed by a developer, where the behavior Logic of the device is implemented. It represents physical components of the system, and at the same time serves as a bridge between them and their representation in the C4 System. In the Driver, we aim to mirror and simulate Logic of Device's behavior as accurately as possible, as well as implement all the conditions according to which the Device behaves. As a result, this Logic will allow us to control the device, or send information

about statuses of the device to the C4 System.

Mirroring of the device in the Driver is divided into parts that represent individual subsystems. For example, if the device has two Outputs, there will be two instances of the object of type Output. Each of them contains the same Logic, but different data.

From a functional perspective, the internal architecture of the Driver consists of several logical components.

Framework 4 is an SDK provided by Gamanet for developers of Drivers. It includes a submodule Link, which provides the communication channel with the device; Packet Processor, which enables physical communication of packets - sending data and processing responses; and a submodule called Access Calculator, which helps the developers to manage access rights settings in devices managing credentials. Each of these parts will be described in more detail in a separate lesson.

Device Logic is a block of code developed by developers of the Drivers. While Framework 4 is supplied by Gamanet, Device Logic must be programmed by the developer.

Since each device uses a different protocol, every Driver also has different types of packets. Device Logic is responsible for creating packets which are sent to the device, as well as for processing responses from the device. It also handles processing of commands sent from the C4 System to the Driver. Of course, one of the main tasks of the Device Logic is sending data to the C4 System, whether it's events or information about statuses of devices.

Structure of the Driver described in this lesson is universal and applicable when integrating majority of external systems. In the upcoming lessons, we will describe how to use the modules of Framework 4 with different types of systems.