Introduction to INDIGO development

Introduction

INDIGO is a platform for the communication between software entities over a kind of software *bus*. These entities are typically either in a *device* or a *client* role, but there are special entities often referred as *agents* which are in both the *device* and the *client* role. A piece of code registering one or more *devices* on the *bus* is referred as the *driver*.

To be able to communicate over the *bus*, software entity have to register providing a structure containing pointers to callback functions called by the *bus*.

Messages sent over the *bus* are abstraction of INDI messages. The messages sent from a device to a client are *definition of a property*, *update of property item values* and *deletion of a property*. The messages sent from a client to a device are *request for definition of available properties*, *request for change of property item values*.

For the list of the *well known* properties (but *device* can define specific properties as well) see PROPERTIES.md.

Different busses can be connected to a hierarchical structure, but from a *driver* or a *client* point of view it is fully transparent.

For the description of XML and JSON INDIGO protocols used for communication between different INDIGO busses see PROTOCOLS.md.

A common API

A basic common API (shared by both *driver* and *client* roles) is defined in indigo_bus.h. The most important structures are *indigo_item* (a definition of property item container), *indigo_property* (a definition of property container), *indigo_device* (a definition of a logical device made available by *driver* containing both driver private data and pointers to callback functions) and *indigo_client* (a definition of a client containing both client private data and pointers to callback funtions).

The *bus* instance should be initialized or started before by *indigo_start()* call and stopped by *indigo_stop()* call. A device should be attached or detached from the *bus* by *indigo_attach_device()* or *indigo_detach_device()* call while a client should be attached by *indigo_attach_client()* or *indigo_detach_client()* call.

Messages from a *device* to a *client* are sent by *indigo_define_property()*, *indigo_update_property()* and *indigo_delete_property()* calls, while messages from a *client* to a *device* are sent by *indigo_enumerate_properties()* and *indigo_change_property()* calls.

Properties are within a *driver* defined by *indigo_init_XXX_property()* calls and items by *indigo_init_XXX_items()* calls.

For a complete list of available functions and more detailed description see indigo_bus.h.

Client API

Structures and helper functions for a client code are defined in *indigo_client.h*. There are three groups of structures and functions – for management of standard local *drivers* (loaded as dynamic libraries and

connected to the local *bus*), executables local *drivers* (loaded as executables, e.g. legacy INDI drivers, and connected to the local bus over pipes) and remote *servers* (connected to the local bus over a network).

An open source examples of client API usage are the following pieces of code:

indigo_test/client.c - API example

indigo_tools/indigo_prop_tool.c - command line tool

Linux control panel project

PixInsight INDIGO client project - under development

Driver API

Structures, helper functions and macros for a driver code are defined in *indigo_driver.h* and *indigo_XXX_driver.h*. For examples of different device drivers see indigo_drivers.