Property types

Tuning into a Digital TV physical channel and starting decoding it requires changing a set of parameters, in order to control the tuner, the demodulator, the Linear Low-noise Amplifier (LNA) and to set the antenna subsystem via Satellite Equipment Control - SEC (on satellite systems). The actual parameters are specific to each particular digital TV standards, and may change as the digital TV specs evolves.

In the past (up to DVB API version 3 - DVBv3), the strategy used was to have a union with the parameters needed to tune for DVB-S, DVB-C, DVB-T and ATSC delivery systems grouped there. The problem is that, as the second generation standards appeared, the size of such union was not big enough to group the structs that would be required for those new standards. Also, extending it would break userspace.

So, the legacy union/struct based approach was deprecated, in favor of a properties set approach. On such approach, :ref:`FE_GET_PROPERTY and FE_SET_PROPERTY <FE_GET_PROPERTY>` are used to setup the frontend and read its status.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\dvb\(linux-master)(Documentation)(userspace-api)(media)(dvb)dvbproperty.rst`, line 23);** *backlink*
>
> Unknown interpreted text role "ref".

The actual action is determined by a set of dtv_property cmd/data pairs. With one single ioctl, is possible to get/set up to 64 properties.

This section describes the new and recommended way to set the frontend, with supports all digital TV delivery systems.

> **Note**
>
> 1. On Linux DVB API version 3, setting a frontend was done via struct :c:type:`dvb_frontend_parameters`.
>
>   > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\dvb\(linux-master)(Documentation)(userspace-api)(media)(dvb)dvbproperty.rst`, line 36);** *backlink*
>   >
>   > Unknown interpreted text role "c:type".
>
> 2. Don't use DVB API version 3 calls on hardware with supports newer standards. Such API provides no support or a very limited support to new standards and/or new hardware.
>
> 3. Nowadays, most frontends support multiple delivery systems. Only with DVB API version 5 calls it is possible to switch between the multiple delivery systems supported by a frontend.
>
> 4. DVB API version 5 is also called *S2API*, as the first new standard added to it was DVB-S2.

Example: in order to set the hardware to tune into a DVB-C channel at 651 kHz, modulated with 256-QAM, FEC 3/4 and symbol rate of 5.217 Mbauds, those properties should be sent to :ref:`FE_SET_PROPERTY <FE_GET_PROPERTY>` ioctl:

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\dvb\(linux-master)(Documentation)(userspace-api)(media)(dvb)dvbproperty.rst`, line 50);** *backlink*
>
> Unknown interpreted text role "ref".

:ref:`DTV_DELIVERY_SYSTEM <DTV-DELIVERY-SYSTEM>` = SYS_DVBC_ANNEX_A

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\dvb\(linux-master)(Documentation)(userspace-api)(media)(dvb)dvbproperty.rst`, line 55);** *backlink*
>
> Unknown interpreted text role "ref".

:ref:`DTV_FREQUENCY <DTV-FREQUENCY>` = 651000000

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-`

:ref:`DTV_MODULATION <DTV-MODULATION>` = QAM_256

:ref:`DTV_INVERSION <DTV-INVERSION>` = INVERSION_AUTO

:ref:`DTV_SYMBOL_RATE <DTV-SYMBOL-RATE>` = 5217000

:ref:`DTV_INNER_FEC <DTV-INNER-FEC>` = FEC_3_4

:ref:`DTV_TUNE <DTV-TUNE>`

The code that would that would do the above is show in :ref:`dtv-prop-example`.

```
static struct dtv_property props[] = {
    { .cmd = DTV_DELIVERY_SYSTEM, .u.data = SYS_DVBC_ANNEX_A },
    { .cmd = DTV_FREQUENCY,       .u.data = 651000000 },
    { .cmd = DTV_MODULATION,      .u.data = QAM_256 },
    { .cmd = DTV_INVERSION,       .u.data = INVERSION_AUTO },
    { .cmd = DTV_SYMBOL_RATE,     .u.data = 5217000 },
    { .cmd = DTV_INNER_FEC,       .u.data = FEC_3_4 },
    { .cmd = DTV_TUNE }
};

static struct dtv_properties dtv_prop = {
    .num = 6, .props = props
};

int main(void)
{
    int fd = open("/dev/dvb/adapter0/frontend0", O_RDWR);

    if (!fd) {
        perror ("open");
        return -1;
    }
    if (ioctl(fd, FE_SET_PROPERTY, &dtv_prop) == -1) {
        perror("ioctl");
        return -1;
    }
    printf("Frontend set\\n");
    return 0;
}
```

Attention!

While it is possible to directly call the Kernel code like the above example, it is strongly recommended to use libdvbv5, as it provides abstraction to work with the supported digital TV standards and provides methods for usual operations like program scanning and to read/write channel descriptor files.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\dvb\(linux-master)(Documentation)(userspace-api)(media)(dvb)dvbproperty.rst`, line 118)

Unknown directive type "toctree".

```
.. toctree::
    :maxdepth: 1

    fe_property_parameters
    frontend-stat-properties
    frontend-property-terrestrial-systems
    frontend-property-cable-systems
    frontend-property-satellite-systems
    frontend-header
```