Introduction to the MIP Driver Framework

Device drivers are an essential part of the XProtect VMS, interfacing between recording servers and the devices that you attach to the system.

The number of IP camera device models is ever-increasing, and other device types are becoming increasingly important.

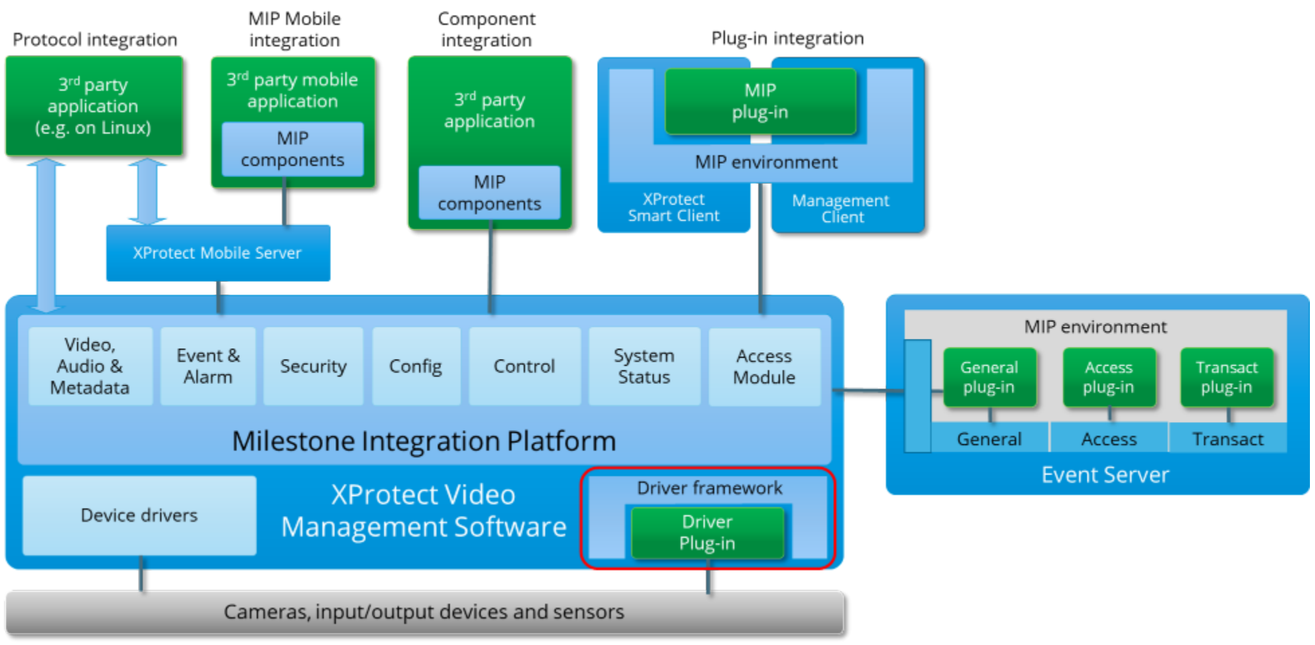
In addition to the dedicated drivers already available from Milestone Systems, XProtect offers these generic device drivers:

- Universal - video and audio streaming, but no metadata and no device control
- ONVIF - requires device support
- MIP driver - video streaming, metadata, and PTZ - requires device support or a proxy

The MIP Driver Framework is introduced to supplement the dedicated and generic device drivers available from Milestone Systems.

Using the MIP Driver Framework, you can create fully featured device drivers, similar to the way you create MIP plug-ins to extend the features of the Smart Client and the Management Client.

For VMS versions prior to 2021 R1, MIP Driver Framework-based device drivers can only support one video channel per hardware. From 2021 R1 there is no limitation.



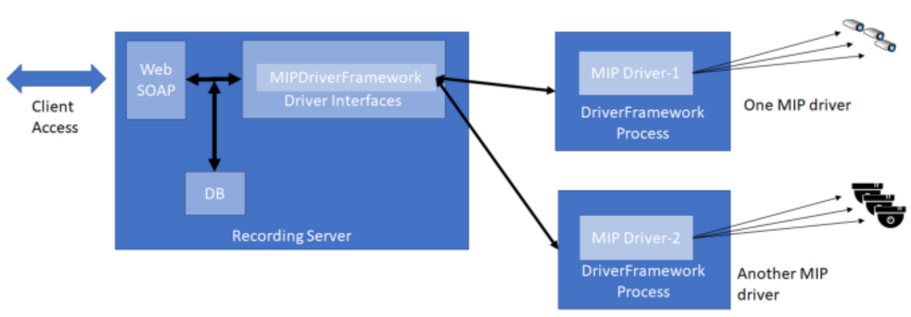
MIP Driver Framework features

The MIP Driver Framework supports all the features of dedicated device drivers:

- Device types
  - Camera. Until and including the 2020 R3 release, only one video channel per hardware was supported.
  - Microphone
  - Speaker
  - Input
  - Output
  - Metadata
- Codec
  - All codecs supported
- Streams
  - Multiple streams and stream configuration
- Settings
  - Available for all device types and hardware
- Event support can be defined for:
  - All devices
  - Hardware
- I/O support
  - Set high/low or trigger
- PTZ
- Edge support
  - All device types
  - Remote playback
- Device discovery
  - Range scan
  - Express scan (from XProtect 2023 R1)
- Firmware upgrade (from XProtect 2023 R3)

MIP Driver Framework architecture

MIP Driver Framework drivers run in processes separate from the Recording Server process. This improves scalability and also prevents a buggy driver from bringing down the Recording Server.



The MIP Driver Framework classes

The MIP Driver Framework lives in the namespace `VideoOS.Platform.DriverFramework`.

`VideoOS.Platform.DriverFramework` declares a number of abstract classes that you override to implement the device features that you need.

The main entry point for the driver is the `DriverDefinition`, the first class loaded by the Recording server. `DriverDefinition` has two members, `DriverInfo` and `Container`.

`DriverInfo` identifies the driver and provides householding information about the driver.

`Container` contains a number of managers that each takes care of a specific class of device features. For example, the `StreamManager` takes care of video, audio and metadata streams.

A few of these managers are built-in, implemented by `VideoOS.Platform.DriverFramework`, and take care of system related features, for example settings and event subscriptions.

You must implement the `ConnectionManager` (the connection method supported by the device) and the `ConfigurationManager` (device specific configuration).

Except for these two managers, you need only implement as many of the optional managers as you need.

Using the MIP Driver Framework

The MIP Driver Visual Studio template

The SDK will automatically install the MIPDriver template for Visual Studio. Use this template when creating your project to automatically get the classes most commonly needed included in your project.

Next, you can then remove the classes you will not need and fill in the missing pieces of code for communicating with your hardware. At the very least go through the various TODOs in the generated project. Especially be aware of the `SpeakerStream1RefId` in `Template.Constants`, which has to be manually updated to a new unique GUID if not removed. All other GUIDs in the generated project are automatically replaced with new unique ones, but due to a limitation in Visual Studio only 10 GUIDs can be generated.

The Demo Driver sample

The Demo Driver sample implements all features of the MIP Driver Framework.

The `.def` file

Similar to the way MIP plug-ins are loaded, for each MIP Driver Framework driver, a small XML file specifies the `.dll` to load.

```
<plugin>
   <file name="DemoDriver.dll" debug="false" />
</plugin>
```

If a directory named `MIPDrivers` is present in parallel to the Recording Server folder, the recording Server will recursively scan the directory for files with the extension `.def`. If the `.def` file references a `.dll`, the Recording Server will attempt to load and run the .dll in a separate DriverFramework process. Make sure the user which is running "Recording Server" has access to the `MIPDrivers` folder.

Relevant samples

Plug-in integration

- Demo Driver