

Note : The Pams Client SDK final implementation can be delayed until the very last milestone as long as the interfaces are well-defined and a dummy implementation granting the rights systematically is provided.

Information Screen

Messages Reader Module

Note : in the communication session and the protocol commands, Message Polling might also be introduced at this stage, in addition to Message Retrieval.

Read Variables

Oscilloscope Module

Note : Variable Group Polling Feature Services might also be introduced at this stage

About box Module

License Info Module

Preferences Module

\* Using the ProtocolEncapsulator means that after having used the current underlying Protocol library to translate the operation to a byte array to be sent on the wire, we would post-process the resulting byte array. In order to do so, the ProtocolEncapsulator would have an ordered list of Protocol libraries – a protocol chain – implementing the IEncapsulator interface. This interface would expose an *Encapsulate* method that would transform the byte array in order to wrap the specified command to be understood by the said protocol as an order to relay the command to the next ECU. After *Encapsulate* has been called on all the protocol libraries, it is ready to be sent on the wire to a device understanding the last layer of encapsulation.

Typically, the ProtocolEncapsulator would be called before the command hits the wire, but would be injected as a service that the Communication Service would rely on.

The protocols chain is populated by a FeatureService returning any second-level ECUs connected to the first ECU. If more than 2 level of ECUs are present, the operation will be conducted recursively until it bubbles to the ECU to which Sculi is directly connected.

If no tunneling is required – i.e. Sculi is directly connected to the target ECU – the ProtocolEncapsulator would act as a pass-through.

