SDK and UNI Ruby development environments comparison

SDK (Software Development Kit) and UNI Ruby represent a set of tools for development of control modules which can directly control a specific instrument. They provide a stable, well-documented **interface between Clarity software** and **the instrument**.

This document is intended to help you decide which environment is more suitable for developing your control module.

The **most important** parameters:

- UNI Ruby - for simple modules (autosamplers, pumps, detectors), Ruby language, limited UI and communication, no library support
- SDK - for complex modules (GC, HPLC systems), C++ language, no limitations

SDK and UNI Ruby are **available for free download**. Documentation and examples are also provided for free. Support, control module testing and addition into Clarity installation is charged.

Both environments enable control module development, system configuration settings, method editing, sending method to device, waiting for getting ready, starting acquisition, data collection, printing and localization.

Main differences are the following:

	UNI Ruby	SDK Control
Programming language	Ruby	C++
Suitable for control modules	simple modules	complex modules
Development Cooperation Agreement	not required	required
Control module testing and approval	not required when used for own purposes	required
External Libraries support	no	Microsoft MFC and ATL
User interface (UI)	simple (table with parameters and values/checkboxes/ buttons or time table)	rich
Wizard	no	yes
Communication channel(s)	1x only (TCP/IP, UDP, USB, COM or GSIOC)	no limitations
Combination of different subdevices	limited	no limitations
Multiple Instruments assignment	limited	no limitations
Advantages	fast simple easy programming	variable support external libraries covers the full scope of UNI Ruby unlimited UI
Disadvantages	limited UI limited communication options limited debugging options external libraries not supported	steep learning curve more complex