A control module development workflow

Clarity controls more than 1000 instruments using various control modules. If you want to add a new one, you can **develop the control module by yourself in cooperation with us**.

This document gives a detailed description of an external control module ("CTRL") development workflow and approval process. It concerns development through Ruby programming languages UNI Ruby, our C++ SDK or Agilent ICF interface.

Typically, it takes 3-6 months to develop the control module, based on its complexity. Each control module project includes three development states: **Under Development, Testing** and **Ready**. Requirements for the particular state are listed in *D081-External-SDK-control-module-development-checklist datasheet*. In order to achieve the desired state, the module has to meet all requirements for the given state without exception. While the Under Development state is for internal development, the Testing and Ready states are available in Clarity installer for public users.

The schema of CTRL development is shown in the figure below. It includes a summary of requested files from the developer and files provided by DataApex.

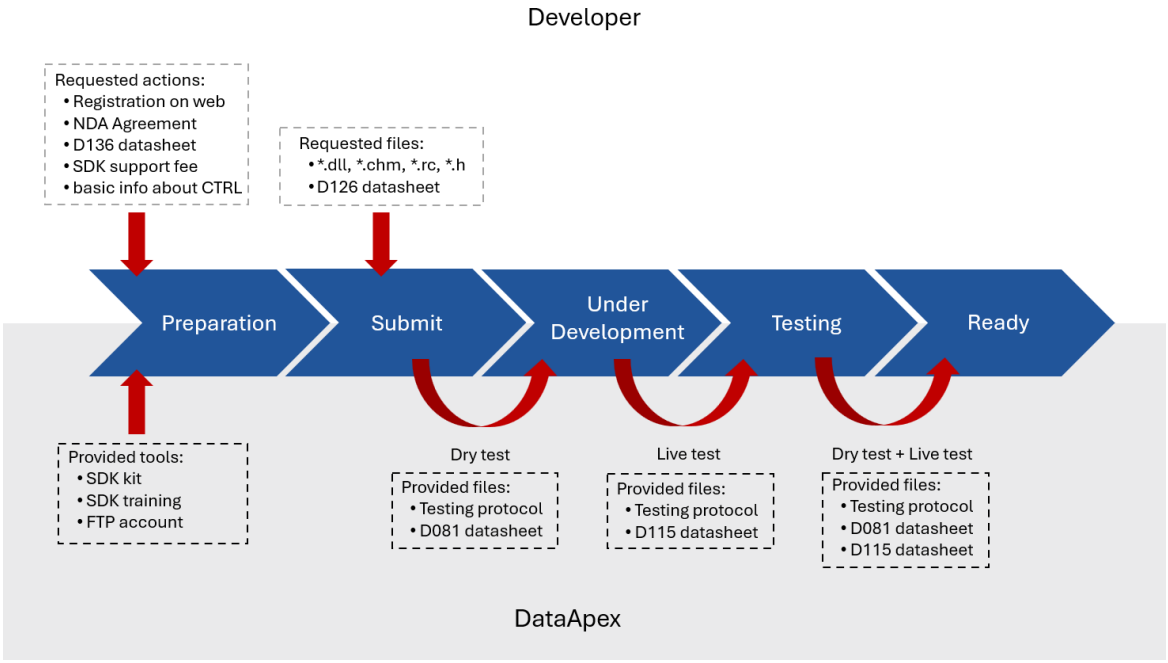

Fig.1 The workflow of CTRL development

Deadlines for inclusion of updated versions of control modules are available in Clarity Roadmap at the Developers section on our website.

By default, the control module is added to Clarity installer as a part of it. However, it is possible to install CTRL to external locations (other folders than the Clarity installation folder) and use these control modules in Clarity. Such modules are installed separately via control module package installers, which still need to pass the acceptance criteria for control modules. For more details refer to *D149-Package-Installation datasheet*.

SDK control modules must be tested and approved by DataApex before they are included in Clarity installer.

The workflow of testing is shown in the figures below. The first figure describes the first addition to installation and first release. Developer's part is Submit and re-submit the control module; the rest is done by us.

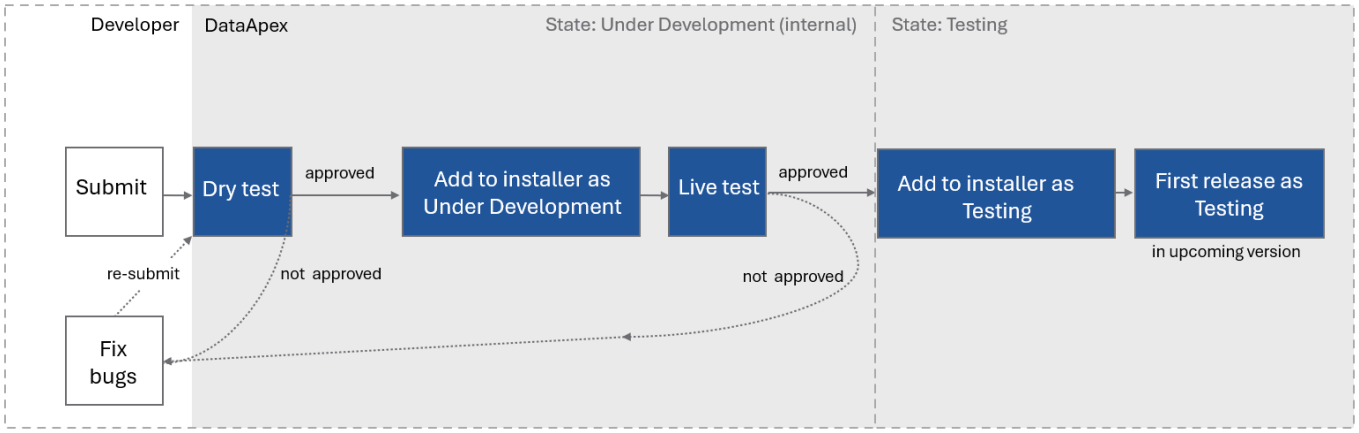

Fig. 2 CTRL testing before first public release

The second figure describes the testing of CTRL updates and the transition from Testing to Ready state. Developer's part is Submit and re-submit the control module; the rest is done by us.

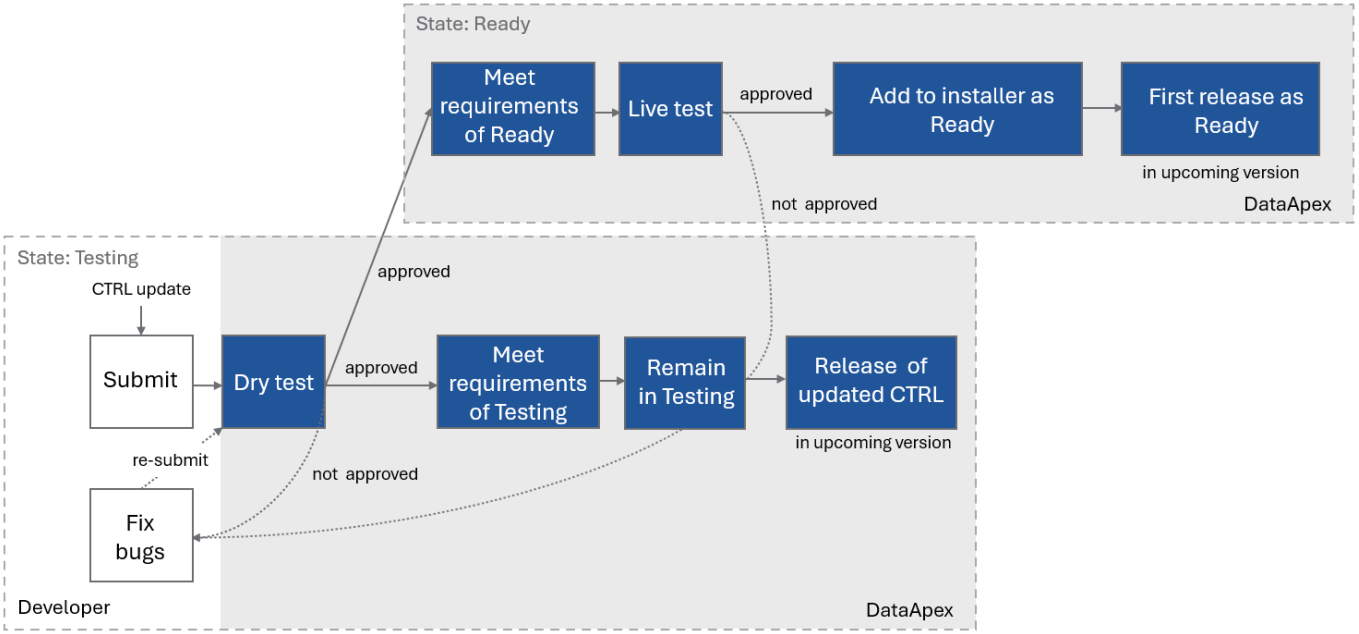

Fig. 3 Testing of CTRL updates

Let's follow the process step by step from the point of view of requirements for an external developer.

1. Preparation

- The developer/manufacturer contacts us and informs us that he wants to develop his own control module.
- Developer fills in the registration on our website and *D136-Clarity-External-Developer-Questionnaire datasheet* sent via email.
- Business conditions are discussed (expected sales, exclusivity, ...) and NDA agreement is signed.
- DataApex creates FTP account for sharing the files and upload SDK interface on it.
- The developer provides basic information about the device (protocols, manuals, figures).

- The developer pays the SDK support fee, which includes SDK support, training, CTRL testing and implementation into installation.

2. **Developing state**
   - When the control module is ready, the developer submits it for testing at DataApex. All necessary files should be uploaded on FTP account and the developer should inform us about it by email send to support@dataapex.com.
   - Control module is tested on our side according to our internal procedure to determine whether it meets requirements for corresponding state to be implemented in Clarity. Requirements are listed in *D081-External-SDK-control-module-development-checklist*.
   - This so called **dry test** is performed in demo mode (without the real instrument). Results are presented in the Testing Protocol that is available within two to three weeks. It contains DataApex tester's comments, questions, suggestions and reported bugs.
   - Based on results and requirements that the CTRL meets, it is approved or not approved to be added into Clarity installer and the Testing Protocol is sent to the developer via email.
   - Usually it takes several "loops" (submit-testing-fix-resubmit-testing...) before the module is finally approved in dry test.
   - After approval, we add the module into Clarity in **"Under development"** state, which is an internal state. First testing build is then created by us.

3. **Testing state**
   - When the build is ready, the next step is a **live test**.
   - It is performed remotely over TeamViewer at developer's side with the real instrument. It is done by a tester from DataApex according to procedure described in *D115-Control-module-approval* datasheet.
   - After the session, the developer will obtain the updated Testing Protocol, including results of the live test.
   - Once the module is good on live test too, it may be included in Clarity in **"Testing"** state.
   - The control module is released with the upcoming Clarity version.

4. **Ready state**
   - After the first release, the developer works on updates and fixes reported bugs in order to meet requirements for Ready state.
   - Updates are submitted for the test. The workflow is the same (dry test, testing build).
   - Live test is required in case CTRL meets the requirements of Ready stage.
   - Live test is usually not required in case when requirements of Ready state are not met, and the module remains in Testing state. In such case, the CTRL is updated directly.
   - When changing the state of completeness, a live test is performed and once the module is good on live test too, it is included in Clarity in **"Ready"** state.
   - The control module is released with the upcoming Clarity version.

Related documents:

D004-Clarity-Controls-List-of-Controlled-instruments
D081-External-SDK-control-module-development-checklist
D115-Control-module-approval
D126-CTRL-Module-Specification
D136-Clarity-External-Developer-Questionnaire
D149-Package-Installation datasheet
D187-SDK-and-UNI-Ruby-comparison-table datasheet