|  |
| --- |
| Circle Language Spec: Commands |

## Parent Controls Its Sub-Executions

A parent command needs full control over the execution of its sub-commands. The parent command makes the decision about the exact moment, that the sub-command runs.

To enforce this general rule, three rules are imposed to make sure a parent command never looses control over the execution of one of its sub-commands:

- Sub-commands are never referenced

See the article *Sub-Commands Are Never Referenced*

- Beware of active command references in commands

See the article *Beware of Active Command References in Commands*

- Sub-commands are never manually started

See the article *Sub-Commands Not Manually Started*

With these rules, there’s no way to point to sub-commands, and the point is made, that you have to beware when you are using active command references inside commands. And there is no way for a user to start the sub-command at any arbitrary moment. There’s no way in, and you have to be aware of the way out, so the sub-command is always an isolated command object, the execution of which is initiated by the parent command, or if it is an active command reference, you have to be aware of giving up control. An external force can not execute the sub-command it before the parent command chooses to execute it, unless you use an active command reference.

Any command object, potentially executed within another command definition, needs to be protected against this risk. The rules to protect the system against this hazard could be changed in the future. But the *reason* for this protection does not change.

For instance: the rule ‘sub-commands are never referenced’, may be changed to being able to reference sub-commands after all, but never to be able to *execute* a sub-command through a reference, even if the reference is active. But this change of rules is just an example. It might prove not to be practicle after all.