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

## Target Command Definition

A target command definition is completely analogus to a *target class*.

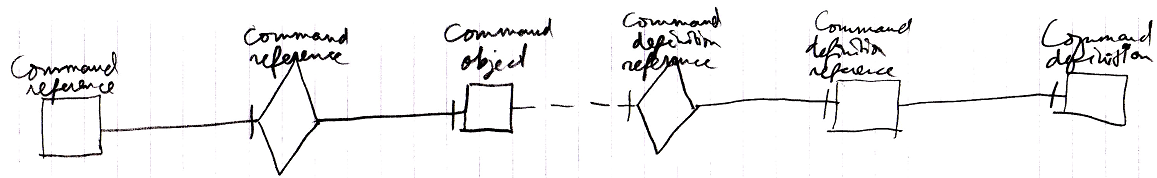
A target command definition is found by following the redirections, that lead to a symbol’s command definition.

When you want to find the definition of a command, and the command is actually a command reference, you first need to follow all object reference redirections, to find the target command object. When you found the target command object, you can find the target command definition, by following one class redirection. Do not follow more than one class redirection, because if a definition points out a definition again, then the second definition is *another* command object, that the first definition is just *based* on. If the definition is a command reference itself, you have to follow all object redirections to find the target *definition* object. Then you have found the target command definition. That’s where redirection following ends. If the definition object has a definition itself, you might be tempted to follow the definition object’s class redirections as well, to find the final target definition, but you should not do that. If a definition object has a definition itself, then the definition object is only based on another definition, but it *is* an object of its own. An object redirection is just a much tighter bond like that, than a class redirection.

### In a Diagram

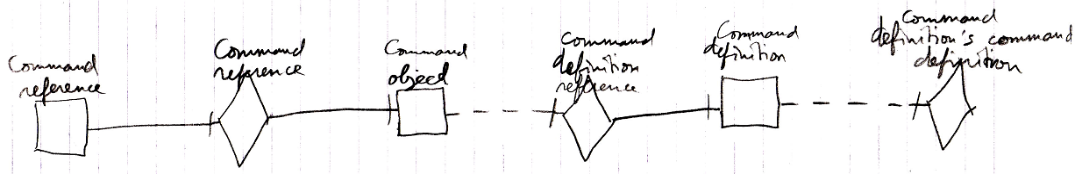
The concept of target command definitions is explained in the article *Target Command Definition*. This article only explains its expression in a diagram.

To find the target definition, you first follow *all* the object redirections, then *one* class redirection, then *all* the object redirections and there it ends.

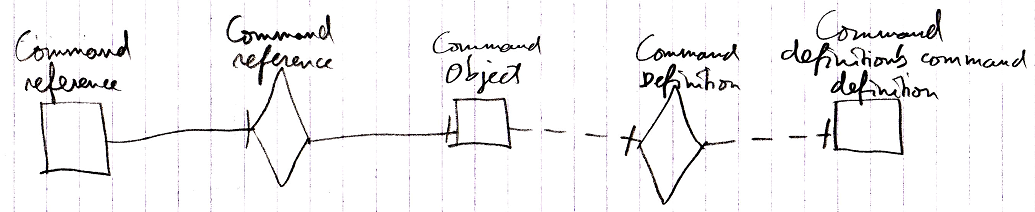


It does not matter if whether it is a diamond, that can be executed, or a square, that can not be executed.

If the definition has a definition as well, this does not redirect the original command object’s definition, because the second definition is *another* definition object, that the first definition is just *based* on. An object redirection is just a much tighter bond, than a class redirection.



The target definition of the first object reference is the symbol Command Definition, not the symbol Command definition’s command definition. The same counts for the diagram below.

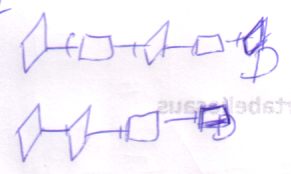


## Ideas

### Out of the original Symbol documentation

#### Definition Trace

The definition trace is quite easy: follow reference lines until you bump into a symbol without a reference line. That symbol’s the definition



#### Execution-Definition Trace

Just as with the object-type trace, the execution trace requires you to find the definition anyway, so when you need both, the execution trace will suffice. It’s called a *execution-definition trace* when you use an execution trace to find both execution and definition.