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

## Creation Behavior of Clauses

The article *Creation Behavior of Calls* talked about delaying the creation of a call’s private contents, until the command is about to be run, while the public contents of a command call are there straight away, as soon as the command call is created.

But if a command object does not have a definition, then it defines its own definition. For command objects that define their own definition, private contents *are* created all the time, because nothing else defines its private contents but the object itself.

This also counts for clauses.

Clauses are like command definitions inside another command.

Even when a clause is an execution, it is also a command definition. Command definitions are created permanently, so clauses are created permanently too, as well as clauses inside other clauses. Active clauses have added behavior compared to other sub-commands (command calls). They are like command definitions inside another command. An active clause’s private data is already created. Even when the clause structure inside a command is very deep, the *whole* depth of the clause structure is recursively created when the parent command is created. The clause structure can not have circularities and is always a limited tree structure, so that the process of creating the whole clause structure can never hang or anything like that.

Other active sub-commands (for instance command calls) behave differently. A command calls’ *private* data is not created until the command is actually run.

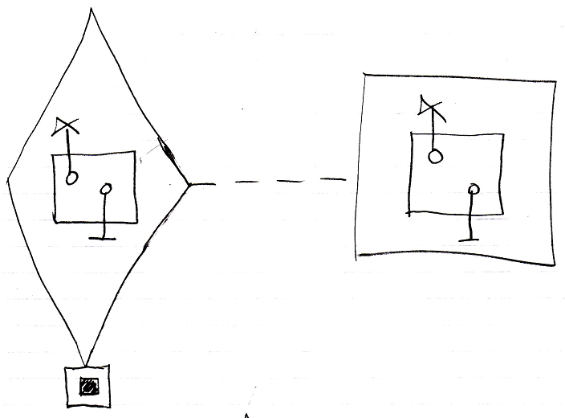
Clauses being permanently created as long as the parent command is created even counts for clauses inside a command call. Right before a call is executed, its private contents are created, including the whole depth of its clauses. In theory the definition of the clause could be pointing to the clauses inside the command call’s definition. The private contents of the clause could be created only just before the clause is run. But this is not done. As soon as a clause in a command call is copied from the definition, the clause has no connection anymore to the clause in the definition. Therefore, it needs to define its own private contents.

Note, that though everything of the clauses is created, parameters of an active clause are only *assigned* right before the clause is run.

### In a Diagram

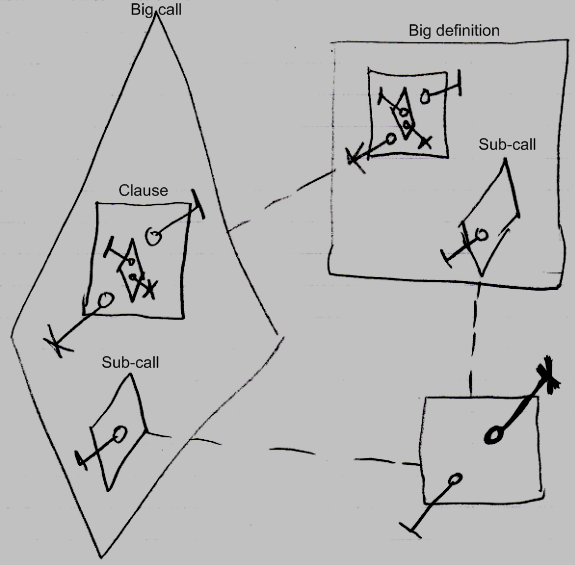
Just like other commands, that do not have a definition, a clause’s contents are created all the time, including its private contents.

Clauses being permanently created as long as the parent command is created even counts for clauses inside a command call. Right before a call is executed, its private contents are created, including the whole depth of its clauses.



In theory the definition of the clause could be pointing to the clause inside the command call’s definition. The private contents of the clause could be created only just before the clause is run. But this is not done. As soon as a clause in a command call is copied from the definition, the clause has no connection anymore to the clause in the definition. Therefore, it needs to define its own private contents.

Even when the clause structure inside a command is very deep, the *whole* depth of the clause structure is recursively created when the parent command is created.



The sub-call inside the big call and inside the big definition only have the public parameter created, not their private contents as neither of them is running. But the clause, with all its contents, are created inside the big call all anyway.