|  |
| --- |
| Circle Language Spec: Black Boxes |

## Public & Private Connections

This section covers any kind of connection between objects, that are a *result* of accessing system aspects.

#### Object Connections

|  |  |
| --- | --- |
| Public Object Get Connection | Friend Object Get Connection |
|  |  |
|  |  |
| Public Object Set Connection | Friend Object Set Connection |
|  |  |

#### Class Connections

|  |  |
| --- | --- |
| Public Use As Class Connection | Friend Use As Class Connection |
|  |  |
|  |  |
| Public Class Set Connection | Friend Class Set  Connection |
|  |  |

#### Value Connections

|  |  |
| --- | --- |
| Public Value Get Connection | Friend Value Get Connection |
|  |  |
|  |  |
| Public Value Set Connection | Friend Value Set Connection |
|  |  |

#### Clone Connections

|  |  |
| --- | --- |
| Public Clone (2) Get Connection | Friend Clone (2) Get Connection |
|  |  |
|  |  |
| Public Clone (2) Set Connection | Friend Clone (2) Set Connection |
|  |  |

#### New & Annul Calls

New and Annul connections are *calls* to the New and Annul system commands.

There are the following possible New and Annul calls:

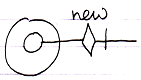
|  |  |
| --- | --- |
| Public New Call | Friend New Call |
|  |  |
|  |  |
| Public Annul Call | Friend Annul Call |
|  |  |

But the *result* of accessing New and Annul are the reference’s being Nothing or Something, which look as follows:

|  |  |
| --- | --- |
| Nothing | Something |
|  |  |

(As with the Set connections of the Object and Class aspects, actually *accessing* the symbol is history and it does not matter anymore if it was Public of Friend access that caused the result, so you do not need to see it in the diagram of Nothing and Something whether it was established through Public access or Friend access.)

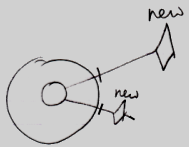
The access *connector* notation of New and Annul were preliminary:



And here it becomes apparent why. The following diagrams show New and Annul calls and also the access connectors displayed right in the picture:

|  |  |
| --- | --- |
|  |  |

An access connector stands for a potential connection and should look like the potential connection. In the pictures above, the connector does not look like the actual connection, so the notation may need to change in the future. A potential connection is usually expressed as a loose end, so the potential system command call should also be expressed as a loose end somehow, so perhaps the following notation for access controlling a system command would be better:



For now the notation of system command connectors is open to discussion.

#### Execute Connections

< TO DO: You have to address the details in *Using Command Symbols* and the preliminariness of the system command notation must eventually be removed. And then you have to display just the actual connections (executable references to the command) that are already visible inside the more complex pictures now shown below. >

Access controlling the Execute system command for command objects may seem like access controlling the ability to call a command. But that is not true. You can access control a command’s Use As Class connector to make a command callable or not. You do not need to access control the Execute system command for that.

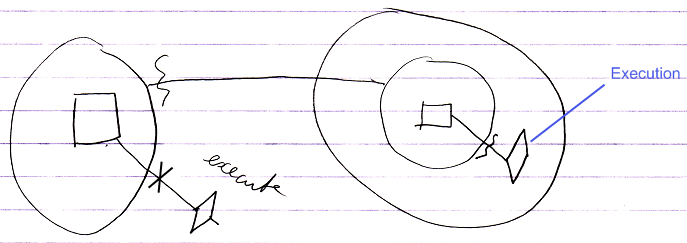
The essence of it is, that each system aspect can be separately access controlled. They all get their own access connector. The uses and implications of access controlling the Execute command are less important for now. Access controlling commands is covered later separately.

The Execute access connector was introduced as follows:



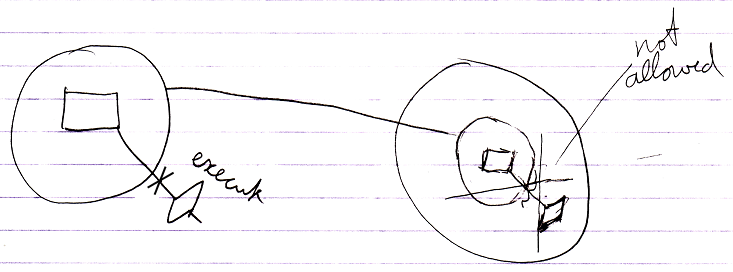
The result of access controlling the Execute command is that the command can only be executed by Friends. The command can not be referenced by a diamond shape (an *execution*) unless it is from a Friend object. Access controlling the Execute aspect *disallows* execution of the command.

So this is allowed (see **Execution** in the diagram):

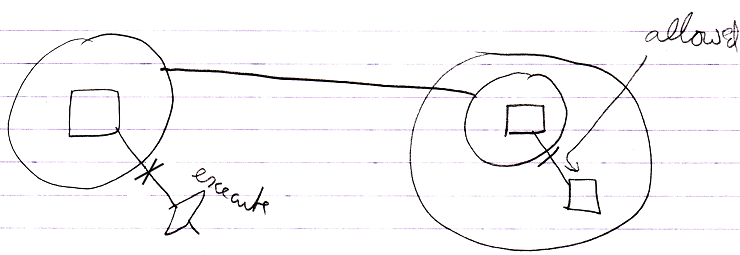


It is allowed, because of the friend declaration on the left.

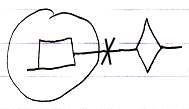
When you take away the friend declaration, execution is no longer allowed:



But *non-executing* references to the command *are* allowed:



The notation for access controlling system commands is not final yet. But because an access connector should be a depiction of a potential connection, the access connector for the Execute aspect might eventually have to look as follows:



With not even the word ‘execute’ in it anymore.

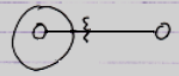
#### < Add And Remove Connections … >

…

#### Remarks

##### Friend and Private Connections

Because there is not distinction between a Friend connection and a Private *connection*, Friend connections:



are called Private connections as well.

##### Object Set Connections

The Public and Friend Object Set connections look the same.

|  |  |
| --- | --- |
| Public Object Set Connection | Friend Object Set Connection |
|  |  |

Public and Friend does not matter for outward connections. Public or Friend access only mattered when *establishing* the outward connection. Outward connections are passive connections: direct connections to the target object. *Inward* connections, though, cause a Get command on each consult of the source symbol, so in that case an actual *access* takes place, and the distinction between Friend or Public access does matter.

##### Class Connections

Use As Class and Class Set are covered here. Class Get is not covered here, because that is only used in an assignment, and assignments are addressed in other section: *Assignment of System Aspects*.

##### Value Connections

Note that the indication of *direction* for Get and Set is switched for the Value aspect, compared to the Object aspect: Object Get is inward, while Value Get is *outward*.

Value connections are shown here in the section *Connections to System Aspects*, and not in the section *Assignment of System Aspects*, but Value connections are assignments, but they do not have an ‘assignment notation’.

What you can also notice is that outward Value connections do have access marks, while outward connections for the Object and Class aspects do *not* show access marks. This is because *outward* Object and Class connections are passive connections and outward Value connections are *active* connections, because an Value connection always requires an active assignment.

##### Data Connections

A Data connector such as the following:

|  |
| --- |
| Public Data Get |
|  |

is an abstract connector: it represents any possible Get connection to a sub-sub-object.

Even though there is a Data *connector*, there are no Data *connections*. Any Get connection to a sub-object is be considered a Data Get connection.

##### Clone Connections

The diagrams above show access connections for cloning at a depth of 2. That is just an example. Any cloning depth other than 2 could be used.

Cloning has the exact same notation as the Value aspect, but then with a number near the access mark. Cloning is that closely related to the Value concept.

##### Existance & Execute Connections

The remaining aspects are Existance and Execute.

The essence of it is, that all system commands are separately access controlled. They all get their own access connector. The uses and implications are less important.

This part of the documentation displays connections that are a result of accessing system aspects. But for the Value aspect that means the display of an assignment.

And for Existance and Execute that also means something different.

Here you see again, that all system aspects behave totally differently.