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

## Simplified Access Control Expression in Text Code

### Main Idea

To describe the access control of a parameter using the literals, introduced in the article *Access Controlling System Aspects*, you end up with an access control literal, that is quite a mouthful, for instance:

Object Set Public

Value Get Private

Value Set Private

The inaccessible system commands were left out already, even though their inaccessibility is also part of the access control.

In a diagram each of those access control elements can be represented by a quite easy to read access connector. But in textual form they look like quite a mouthful. In other programming languages they look easier.

That is why access control literals should be simplified. This article lists out the rules for simplifying access control literals.

### Admission

I have tried to make a good setup of simplification rules, but I have not been able finish it. However, I do find it important to be picked up in the future, to make a clear set of terms, that more easily expresses access control situations. In this article I will lay out the ideas I already have about it, even though it is not finished yet.

### Leave out the inaccessible

The first rule was already covered: leave out anything that is not accessible at all. You only end up with the ways you *can* access it.

Here is the list again of access control elements, that can be used:

Object Get Private

Object Get Public

Object Set Private

Object Set Public

Class Get Private

Class Get Public

Class Set Private

Class Set Public

Value Get Private

Value Get Public

Value Set Private

Value Set Public

Clone X Get Private

Clone X Get Public

Clone X Set Private

Clone X Set Public

Data Get Private

Data Get Public

Data Set Private

Data Set Public

Execute Private

Execute Public

New Private

New Public

Annul Private

Annul Public

( Command )

So only the accessible elements will end up in the access control literal.

### Aspects separated

The various system aspects will always produce their own separate piece of access control literal:

- Value

- Object

- Class

- Data

- Execute

- Creation ( New / Annul )

- Command / Object

These elements, if accessible at all, always each produce a *separate* piece of access control literal, and no rules will be introduced, that converge access control elements of different aspects to a single term.

### Order in a basic literal

In the texts above you have already seen the order of the keywords inside a single access control element, for instance:

Object Get Public

The order is:

- Aspect

- Assignment direction

- Access direction

So first one of the following:

- Value

- Data

- Clone X

- Object

- Class

- Execute

- New

- Annul

- Command

Then one of the following:

- Get

- Set

And then one of the following:

- Private

- Public

Those three keywords form a single access control element, for instance:

Object Get Public

### Ordered by system aspects

The access control elements themselves are also ordered. First I proposed to always order them by system aspect:

- Command

- Value

- Clone X

- Data

- Object

- Class

- Execute

- Creation

The order was dependent on a general likeliness for a system aspect to be accessed. But this ordering will not be used anymore. The chronological order described below will be used.

### Chronological order

Another option for ordering the access control elements, is to order them by the most likely chronological order, which is:

Set Public

Get Private

Set Private

Get Public

This order completely replaces the ordering by system aspects. The access control elements for different system aspects could be intermixed, producing for instance the following access control literal:

Object Set Public, Value Set Private

### In, Out and Thru

The terms I would like to see back in access control literals are the terms In, Out and Thru. It would greatly increase the readability if you could for instance say:

Object In, Value Out

Earlier on, this article introduced a way to use the terms In, Out and Thru, that I later had to withdraw. It was implied, that In is writing from the outside, and Out is writing from the inside, so it implied, that it was about the access direction of *writing*. However, now I figure that it is whether the inner object writes or reads, so In would mean reading from the inside and Out would mean writing on the inside. How public reading and writing will be called from that perspective, is not clear to me yet (2008-09-29). I will further work out the terms In, Out and Thru in the *Advanced Command Articles*.

Also: it was implied that in Object Set Public, Value Set Private, you do not privately *get* the object, that was *set* publically. But that was a mistake as well: you do first get the object privately in order to set it’s value. This all will be worked out in the *Advanced Command Articles*.

### Object Or Command

Whether a parameter is an object or a command, is not really access control over the parameter, but can be regarded part of the parameter *passing*, so it *is* involved here.

The parameter access control literal starts with the keyword Command when the parameter is a command. When the parameter is a normal object, the keyword Command is left out.

For instance:

Command Class Set Public

Or

Command Object Set Public, Execute Private

### Creation

By creation I mean the system commands New and Annul. An object might be created or annulled from the outside or created and annulled from the inside. You already have a low-level way to express parameter access control for this:

New Private

New Public

Annul Private

Annul Public

but you may want to express it with less words.

Below are described three options for simplification of New and Annul access control literals:

- Implied by Object aspect

- Use of keywords In, Out and Thru

- Use of the keyword Existance

#### Implied by Object aspect

New and Annul are also sort of like Object aspect writing. So Object Set Public may also mean you can publically execute New on the parameter, as well as Annul. So maybe Object Set implies also that you can execute New and Annul. New and Annul under the surface will probably use Object Set anyway, and *above* the surface you may be able to use Object Set as a New and Annul after all (but I am saying nothing final about that yet).

But perhaps you should not go with such implication and insist you always express it when you can use New or Annul on something.

#### In, Out and Thru

Since New and Annul areboth writing, you may use the terms In and Out for it.

New In would mean you can execute New from the outside and New Out means you can execute New on the inside. This produces the following, easier to interpret literals:

New In

New Out

Annul In

Annul Out

And when New or Annul are both accessible In and Out:

New Thru

Annul Thru

#### The Existance keyword

Perhaps the terms New and Annul could converge into a single term, when they are both accessible the same way, for instance to the keyword Existance:

Existance In

= New In, Annul In

= New Public, Annul Public

Existance Out

= New Out, Annul Out

= New Private, Annul Private

Existance Thru

= Existance In, Existance Out

= New In, Annul In, New Out, Annul Out

= New Public, Annul Public, New Private, Annul Private

### Leave out the accessible

When *everything* is accessible, then the access control literal may become more complicated. Perhaps lateron you should also invent rules to simplify the access control literal, when most of the things are accessible. A sort of substractive or exclusion kind of access control literal. Now it is an additive or inclusion kind of access control literal.

### All in all

All in all, the access control literals for parameters can get the following names in the following (chronological) order:

( Command )

Object Set Public

New In

Annul In

Value Set Public

Data Set Public

Class Set Public

Object Get Private

Value Get Private

Data Get Private

Class Get Private

Execute Private

Object Set Private

New Out

Annul Out

Value Set Private

Data Set Private

Class Set Private

Object Get Public

Class Get Public

Value Get Public

Data Get Public

Execute Public

For instance:

Object Set Public, Object Get Private, Value Set Private, Value Get Public

It is lengthy, but clear.

If New and Annul have the same access control, it might be replaced by a single access control literal on Existance.

Existance In

Existance Out

Existance Thru

The only thing I need to make me completely satisfied with the simplified access control literals, is that good usage of the terms In, Out and Thru will replace the terms Get Public , Set Public, Get Private and Set Private.

### Replacement that didn’t work out

I thought about replacement of separate access control elements by single ones.

I thought about replacing Public and Private by Accessible when the same system command was both publically and privately accessible.

I also thought about replacing Get and Set by Accessible when the Get and Set of a system aspect were both accessible the same way.

But that naming resulted in overlap in possibilities. The same access control situation could be expressed in several different ways. And I could *not* come up with an exact rule to pick the clearest naming from a set of overlapping naming possibilities.

When I also tried to apply the terms In, Out and Thru, then I got really confused. At that point I decided, that I would no longer think about it anymore, and only summarize my existing, unfinished ideas in this article. It did not even turn out that bad and it is a good start anyway.

### Old naming

Reference Out was a parameter access control literal at one point. It meant writing the object on the outside and reading the value from the inside. This is now expressed with the following literal:

Object Set Public, Value Set Private

You can immediately see what happens from that literal.

What you expect *now* from access controlled parameters, the words Reference Out could imply you write the object on the inside, so the literal for that now is:

Object Set Private

And Object Out *used* to be the name for reading the object from the outside, and then *possibly* writing it on the inside, which is now the literal:

Object Get Public

The point is: the naming has changed.

Below you find a table of new literals for the old parameter passings. This also gives you an idea of the old parameter passings I had in mind:

|  |  |
| --- | --- |
| *Old Name* | *New Name* |
| Value In | Value Set Public |
| Value Out | Value Get Public |
| Value Thru | Value Set Public, Value Get Public |
| Reference In | Object Set Public  Object, Value, Clone, Data & Class Get Private |
| Reference Out | Object Set Public  Object, Value, Clone, Data & Class Set Private |
| Reference Thru | Object Set Public  Object, Value, Clone, Data & Class Get & Set Private |
| Object Out | Object Get Public  Object, Value, Clone, Data & Class Get & Set Private |