|  |
| --- |
| Circle Language Spec: Events |

## System Events

### Introduction

Without even defining any objects or class libraries, there are already a lot of standard events you can use. In most situations these will suffice.

They are events mostly related to changes to the data. Any type of change to the data can be picked up as an event.

More specifically: each system command has an event going off before it is executed and an event going off after it is executed: an Executing event (before) and an Executed event (after).

Derived from this comes the concept of being able to pick up a Value Set command going off. This event is called the Value Changed event. Picking up an event before the value is set also makes you able to cancel that change before it even happened. In case of the Value aspect this is the Value Changing event. So you have the opportunity to cancel a system call altogether.

Any other event of a call to a system command, such as Add, Remove and Execute can be picked up, before or after it happened, and could be cancelled.

Below is an overview of all events that derive from this concept. This results in a big list of standard events, varying in aspect, read/write direction and the before and after events. Something to denote is that Changed events are the ones most commonly used.

#### Changed Events

Value Changed

Object Changed

Class Changed

Name Changed

Data Changed

Interface Changed

Interface Merged Changed

Clone Written

#### Changing Events

Value Changing

Object Changing

Class Changing

Name Changing

Data Changing

Interface Changing

Interface Merged Changing

Writing Clone

#### Get Events

Getting Reference

Reference Gotten

Getting Object

Object Gotten

Getting Reference-Class

Reference-Class Gotten

Getting Object-Class

Object-Class Gotten

Getting Value

Value Gotten

Cloning

Cloned

Getting Name

Name Gotten

Getting Data

Data Gotten

Getting Reference-Interface

Reference-Interface Gotten

Getting Object-Interface

Object-Interface Gotten

Getting Interface Merged

Interface Merged Gotten

#### Use Events

Reference Used As Object

Using Reference As Object

Using As Class

Used As Class

Using Reference As Class

Reference Used As Class

Using As Interface

Used As Interface

Using Reference As Interface

Reference Used As Interface

#### List Events

Added (used more)

Adding (used less)

Removing (used more)

Removed (used less)

Storage Count Changed

Storage Count Changing

#### Execution & Existence Events

Executed

Executing

Created

Creating

Annulled

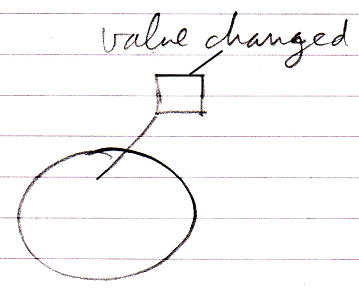
Annulling

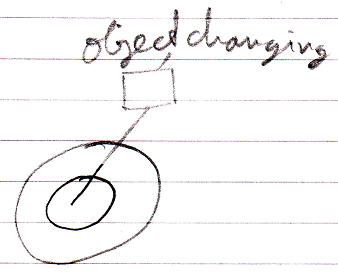
#### Pointer to Pointer Events

Perhaps the different overloads for pointer-to-pointer situations should have separate events as well, but this depends on design choices and this issue will be left open for now.

### System Events in Normal Notation

System can be displayed with the normal event notation, examples of which you will find below.





### Simplified System Event Notation

System events can also be displayed using a simpler notation that uses line types to denote the aspect, and leaves out the aspect out of the displayed event name.

In the matrix below, reference-bound aspects are displayed with a parent around the event source and object-bound aspects are displayed without a parent around the event source.





(It is not sure whether this will adequately disambiguate in practice. Hopefully using the diagram concept of automatic containment it will turn out not to be ambiguous, but that is not sure yet.)

Here follow the simplified notations for all the system events.

**Reference Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Getting Reference | Reference Gotten |  |  |
|  |  |  |  |

**Object Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Object Changed | Object Changing | Object Gotten | Getting Object |
|  |  |  |  |
|  |  |  |  |
| Reference Used As Object | Using Reference As Object |  |  |
|  |  |  |  |

**Class Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Class Changed | Class Changing |  |  |
|  |  |  |  |
|  |  |  |  |
| Used As Class | Using As Class | Reference Used As Class | Using Reference As Class |
|  |  |  |  |
|  |  |  |  |
| Reference-Class Gotten | Getting Reference-Class | Object-Class Gotten | Getting Object-Class |
|  |  |  |  |

**Interface Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Changed | Interface Changing |  |  |
|  |  |  |  |
|  |  |  |  |
| Used As Interface | Using As Interface | Reference Used As Interface | Using Reference As Interface |
|  |  |  |  |
|  |  |  |  |
| Reference-Interface Gotten | Getting Reference-Interface | Object-Interface Gotten | Getting Object-Interface |
|  |  |  |  |

**Interface Merged Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Merged Changed | Interface Merged Changing | Getting Interface Merged | Interface Merged Gotten |
|  |  |  |  |

**Value Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Value Changed | Value Changing | Getting Value | Value Gotten |
|  |  |  |  |

**Clone Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Clone Written | Writing Clone | Cloned | Cloning |
|  |  |  |  |

**Data Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Data Changed | Data Changing | Data Gotten | Getting Data |
|  |  |  |  |

**Name Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Name Changed | Name Changing | Getting Name | Name Gotten |
|  |  |  |  |

**List Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Added | Adding | Removing | Removed |
|  |  |  |  |
|  |  |  |  |
| Storage Count Changed | Storage Count Changing |  |  |
|  |  |  |  |

**Execution Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Executed | Executing |  |  |
|  |  |  |  |

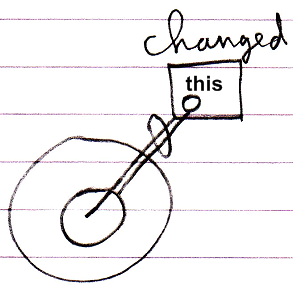
**Existence Aspect**

|  |  |  |  |
| --- | --- | --- | --- |
| Created | Creating | Annulled | Annulling |
|  |  |  |  |

Some aspects have no associated line type, so they are not shown with a simplified notation, but with the normal notation. The above are the connectors, but the actual event commands look almost the same, but then without the loose end at the top.

### System Event Parameters

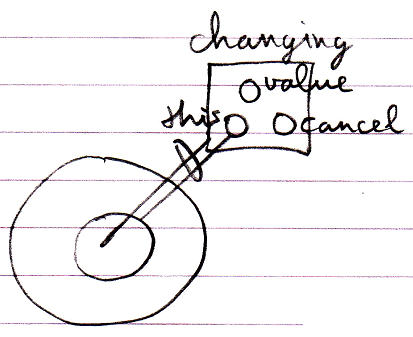
System events have parameters. First of all they have the This parameter, which identifies which object something’s happening to.



This can be an Object, but also for instance a Related Item.

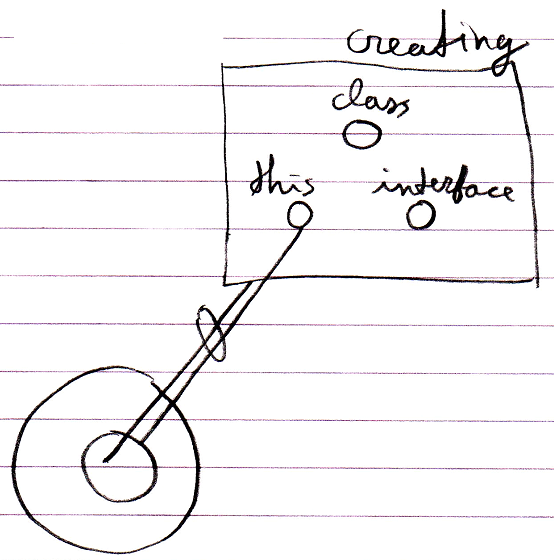
|  |  |
| --- | --- |
|  |  |
|  |  |
| *The* This *parameter is a* Related Item | *The* This *parameter is an* Object |

Second of all for systems events that announce something is about to happen, usually have a Value parameter which indicates which value is about to be assigned and also a Cancel Boolean parameter to be able to assign the value True to, to cancel the event.

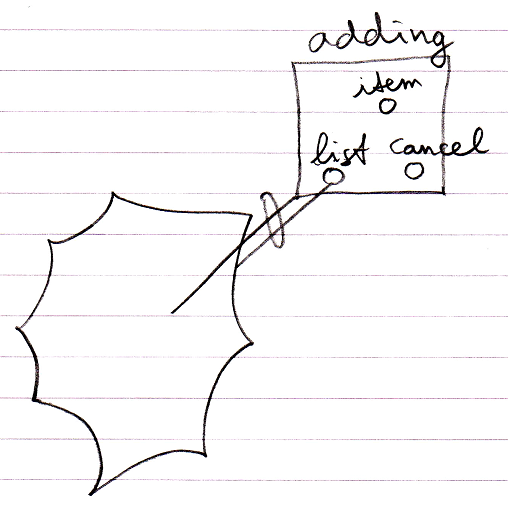


The List and Existence aspects have additional event parameters.

The Creating event have a Class and Interface parameter, which indicate which class or interface the new object will have.



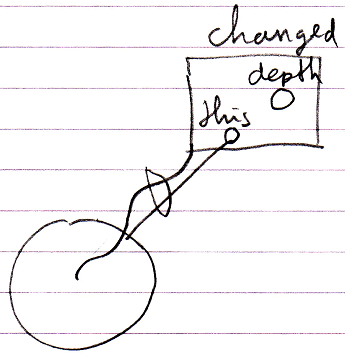
The Adding event has an Item parameter (an Object).

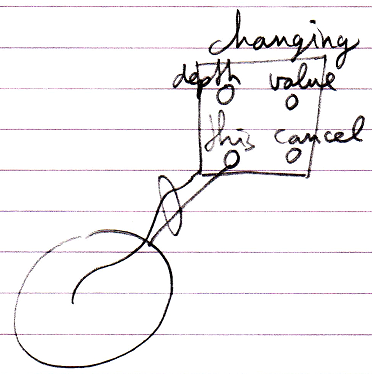


And the Added event has a List Item parameter (a Related List Item)

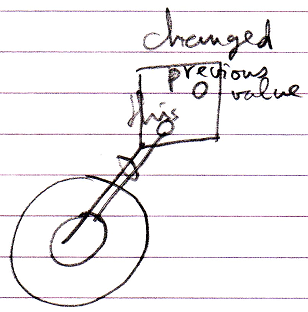


The Clone aspect’s events also have a Depth parameter.





Possibly a Changed event will also contain the Previous Value parameter.



In the earlier introduced connector notation, parameters are not shown. But the parameters *are* shown in the event *implementations*.

### System Event Design Choices

The names used for the events might not be the eventual names. For instance a more simplified design might simply let all system commands have a Executing and Executed event, instead of inventing different names for the different events, so that Value Changed is named Value Set . Executed. In another design choice you might not call it the Executing and Executed event, but the Before and After event, or the Happening and Happened events, or the Pre-Extension and Post-Extension events.

This does not change the concept. These are all just benign design choices. You could also call it Value Will Change instead of Value Changing, this is not really the point. It does not change the concept.

The concept is still derived from the fact that you can pick up the Executing and Executed event from any command, including all system commands.

Other design choices that could be made are that other aspect’s changed events may very well all derive from the Value Changed event. And the Class and Interface aspects’ events might all derive from the Object Changed event.