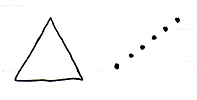
|  |
| --- |
| Circle Language Spec: Interfaces |

## Interface Aspect

While systems seem to be composed of a lot of different types of objects, all objects and types are in reality no more than a very limited set of types, that can be configured to look like other types. Those basic objects are called *system objects*. The behavior of those system objects is controlled by controlling their *system aspects*. System objects is covered in a separate chapter called *System Objects*, but there the *interface* aspect was completely left out of the story. In this part of the documentation everything about the interface aspect is laid out to make the concept of interfaces catch up with the *System Objects* story.

The Interface aspect allows you to control how objects look on the outside, while the insides of the objects can differ completely. Another concept which employs the word *interfaces* is an object (an interface) melting together with its container. From a systematic point of view they seem to be completely unrelated concepts, however in practice they are conjoinedly used. To distinguish them we call them the Interface aspect and the Interface Merging aspect. The main diagram symbolization of the Interface aspect is as follows:



The triangle is the symbolization of the Interface Merging aspect, and the dotted line is the symbolization of Interface aspect.

Interface Merging in the future will probably become an aspect that extends beyond the concept of interfaces, because it could for instance apply to the concept of default members too.

### System Commands for the Interface Aspect

The main system commands to control the Interface aspect are:

Use As Interface

Interface Set

Reference-Interface Get

Object-Interface Get

Use As Interface is a lot like the command Object Get, but then used for the purpose of making that object as the interface of another object. This is common usage of the interface aspect. It also makes you able to separately access control whether an object can be used as an interface.

The Interface Set command is executed on an object reference. The reference can then only point to object that supports the interface. This means either that the other object has the same interface, or that the object has it as sub-object that is an implicit interface. Interface Set applies only to references and not to objects, because the interface of an *object* can only be set upon creation.

The commands Reference-Interface Get and Object-Interface Get get the interface object that is associated with a reference or object. Those commands are less commonly used.

#### Interface is Both Object-Bound and Reference-Bound

The Interface aspect applies to both objects and references, but differently. An object can have a certain interface, that is set upon creation and throughout its lifetime. A reference can also have an interface, defining which classes of object you can assign to the reference. The interfaces of an object can never change. The interface of a reference can be changed, but only to an interface supported by the object it points to or arbitrarily while the reference is Nothing.

#### Pointer-to-Pointer Situations

In a standard situation the Use As Interface, Interface Set, Reference-Interface Get and Object-Interface Get commands are about making an object function as another object’s interface. However, you can also make something’s interface be yet again another reference. That means that another parent object determines the eventual interface.

(However, this might create difficulty for the system to maintain an object’s constant interface. You might want another parent to determine the initial interface, but the interface of an object should not change during its lifetime.)

#### Set Interface to Reference

To be able to set the Interface aspect to another related item, Interface Set has two overloads:

Interface Set 🡪 Set Interface to Other Related Item.

Interface Set 🡪 Set Interface to Other Related List Item

If you want a single name to express both situations, you could call it Set Interface to Reference.

#### Get Interface which is a Reference

Because the Interface aspect can be set to another related item, the Interface Set command gets extra overloads. Next to that, there are different overloads for the two types of Interface Get: Reference-Interface Get and Object-Interface Get. This creates the following overloads:

Reference-Interface Get 🡪 Get Reference-Interface which is Another Related Item

Reference-Interface Get 🡪 Get Reference-Interface which is Another Related List Item

Object-Interface Get 🡪 Get Object-Interface which is Another Related Item

Object-Interface Get 🡪 Get Object-Interface which is Another Related List Item

You could also call them Get Interface which is a Reference.

#### Use Reference As Interface

The Reference aspect can be access-controlled for different ways you can use it. Pointer-to-pointer situations require you to be able to use a reference as an *interface*. To be able to access control the different purposes for which you can use a reference, the Reference Get command gets the secondary implementation:

Use Reference As Interface

which delegates directly to the Reference Get command.

#### The Overloads Recapitulated

Do not wreck your brain over all this delegation and overloading. It is just for pointer-to-pointer situations to have the same command names as standard situations, and also to be able to separately access-control the specific *uses* of references or objects. You will not usually see the pointer-related commands, because they will be implicitly delegated to by the main commands. This leaves us with the following commands:

Use As Interface

Interface Set

Reference-Interface Get

Object-Interface Get

Use Reference As Interface

Detail: For that last command you might want to overload Object Get. But that does not work. You can not overload it, because they will both take a pointer to an object as an argument. To disambiguate, they have to have a different name and you have to point to a *specific* command.

### System Commands for the Interface Merging Aspect

The commands for the Interface Merging aspect are:

Interface Merged Get

Interface Merged Set

Interface Merging is a reference-bound aspect: it only applies to references, not objects.

Interface Merged is a Boolean, that can be either True or False.

When an interface is merged it means that its members merge with its container and are accessible as if part of the container. Also, the container can be referenced as if it has the type of the interface.

### Interface Aspect in System Interface

System aspects can be shown in the system interface notation as worked out in the *System Interfaces* articles in the *System Objects* chapter. The Interface aspect was not covered there, so this is a supplement to that documentation.

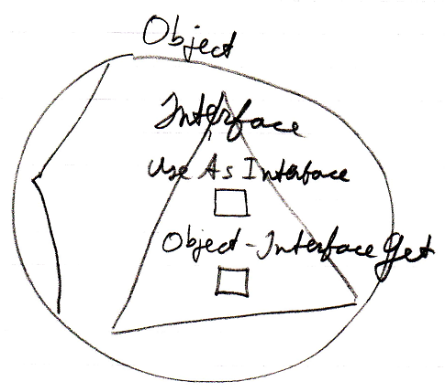
#### Object-Interface in System Interface

The Interface aspect has *five* system commands, but only *two* of them apply to Objects. The other ones apply to references. The Interface aspect of an Object is controlled through the following commands:

Use As Interface

Object-Interface Get

The commands are placed inside the system interface inside a triangle that wraps together the members of the Interface aspect:



#### Reference-Interface in System Interface

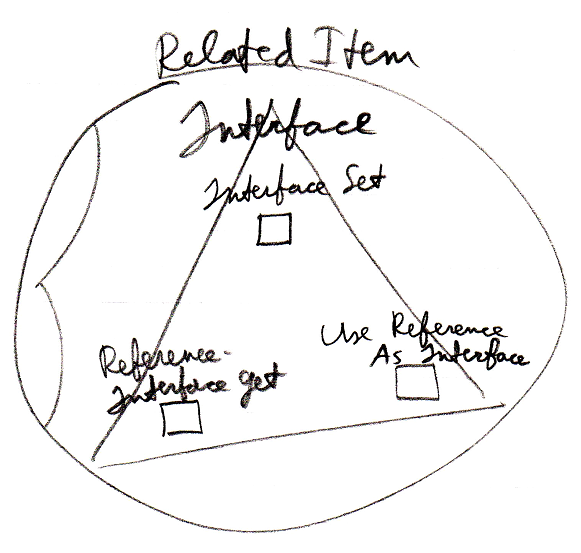
The Interface aspect has *five* system commands, but only *three* of them apply to Related Items:

Interface Set

Reference-Interface Get

Use Reference As Interface

The commands are placed inside a triangle, that wraps together the members of the Interface aspect:

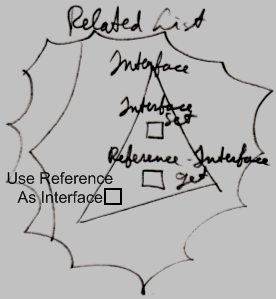


It works the same for a Related List Item as it does for a Related Item.

#### Reference-Interface in System Interface of Related List

A list can centrally control the aspects of its items. Therefore, a list can control the Interface aspect of its items, centrally setting the Interface that all the items must support.

The system commands to do this are placed inside a triangle that wraps together the members to control the Interface aspect:



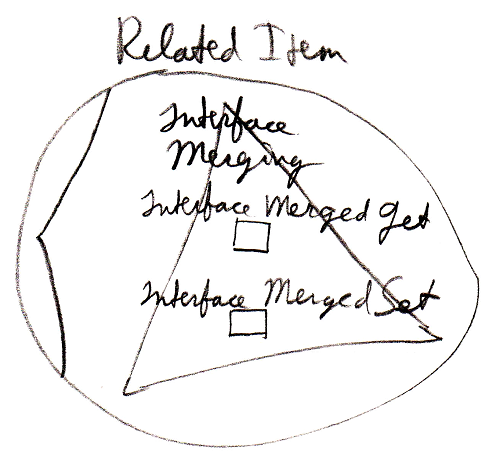
#### Interface Merging in System Interface

The Interface Merging aspect only applies to Related Items. It is controlled through two commands:

Interface Merged Get

Interface Merged Set

The commands are placed inside the system interface inside a triangle, that wraps together the members of the Interface Merging aspect:



Interface Merging does not apply to Related List Items, because it does not seem to be practical there.

### Interface Assignment Types

This section is a mere supplement to the *Assignment* articles in the *System Objects* chapter. For a general explanation of assignment, look there. This section only makes the Interface aspect catch up with the rest of the explanations.

|  |
| --- |
| Interface Assignment |
|  |
| Use As Interface 🡨 (~= Object Get)  Interface Set 🡪 |
| *Makes the source become*  *the interface for the target.* |
|  |
| Interface Pointer Assignment |
|  |
| Use Reference As Interface 🡨 (~= Reference Get)  Interface Set 🡪 (~= Set Interface to Other Related Item) |
|  |
|  |
| Use Reference As Interface 🡨 (~= Reference Get)  Interface Set 🡪 (~= Set Interface to Other Related List Item) |
|  |
|  |
| Use Reference As Interface 🡨 (~= Reference Get)  Interface Set 🡪 (~= Set Interface to Other Related List Item) |
|  |
| Interface Assignment With Pointer Source |
|  |
|  |
| Use As Interface 🡨 (~= Use Reference As Interface)  Interface Set 🡪 (~= Set Interface to Other Related Item) |
|  |
|  |
| Use As Interface 🡨 (~= Use Reference As Interface)  Interface Set 🡪 (~= Set Interface to Other Related List Item) |
|  |
| Cross-Aspect Interface Assignment |
|  |
| Object-Interface to Object Assignment |
|  |
| Object-Interface Get 🡨  Object Set 🡪 |
|  |
| Reference-Interface to Object Assignment |
|  |
| Reference-Interface Get 🡨  Object Set 🡪 |
|  |
| Object-Interface to Reference-Class Assignment |
|  |
| Object-Interface Get 🡨  Reference-Class Set 🡪 |
|  |
| Object-Class to Reference-Interface Assignment |
|  |
| Object-Class Get 🡨  Reference-Interface Set 🡪 |
|  |
| Reference-Class to Reference-Interface Assignment |
|  |
| Reference-Class Get 🡨  Reference-Interface Set 🡪 |

There are more implementations of the cross-aspect assignments for cases when the source of the assignment is a pointer-to-pointer, but they are not shown here. To understand the concept look in the *Cross-Aspect Assignments* article in the *System Objects* chapter.

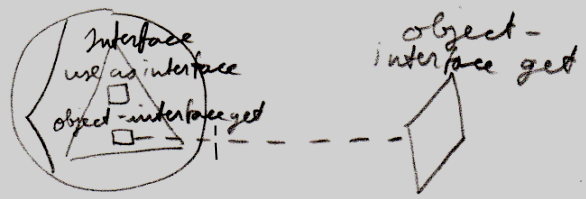
### Interface System Command Calls

The concepts of system command call notations is not explained here. Refer to the *System Command Call Notations* article in the *System Objects* chapter for that. this is a supplement to that documentation, covering it for the Interface aspect. The Interface Merged aspect is also covered.

Cross-aspect situations can apply to the Interface aspect, but they are not covered. Cross-aspect situations between the Object and Class aspects are described in the *System Objects* chapter and adequately shed light on the topic.

#### Interface - System Interface - Command Call Notation

Only one example of a call to a system command of the Interface aspect using the system interface notation will be given here. Below you will find this notation for a call to the Object-Interface Get command:



#### Simplified System Command Call Notation

Below you will find the various types of calls to a system command on the Interface aspect in the *simplified system command call notation*.

|  |  |
| --- | --- |
| Use As Interface | Use Reference As Interface |
|  |  |
|  |  |
| Interface Set |  |
|  |  |
|  |  |
| Object-Interface Get | Reference-Interface Get |
|  |  |
|  |  |
|  |  |
| Interface Merged Get | Interface Merged Set |
|  |  |

#### System Command Call Notation with Argument

Below you will find the various types of calls to a system command on the Interface aspect in the *system command call with argument notation*.

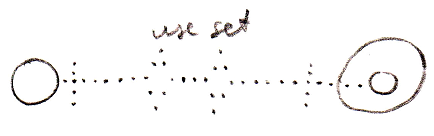
|  |  |  |
| --- | --- | --- |
| Use As Interface | Use Reference As Interface | Interface Set |
|  |  |  |
| Use As Interface 🡨 (Interface Set 🡪) | Use Reference As Interface 🡨  (Interface Set 🡪) | (Interface Get 🡪)  Interface Set 🡨 |
|  |  |  |
| Object-Interface Get | Reference-Interface Get |  |
|  |  |  |
| Object-Interface Get 🡨  (Interface Set 🡪) | Reference-Interface Get 🡨  (Interface Set 🡪) |  |

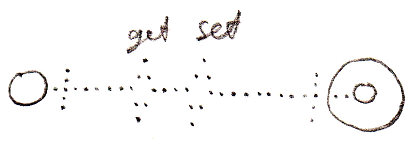
|  |  |  |
| --- | --- | --- |
| Interface Merged Get | Interface Merged Set |  |
|  |  |  |
| Interface Merged Get 🡨  (Value Set 🡪) | (Value Get 🡪)  Interface Merged Set 🡨 |  |

*(For the* Interface Merged *commands the argument is a* Boolean*.)*

#### Explicit Get & Set Notation

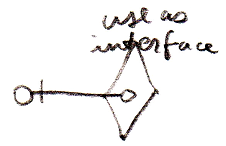
Below you will find some examples of the *explicit Get and Set* notation working on the Interface aspect.

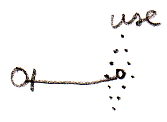


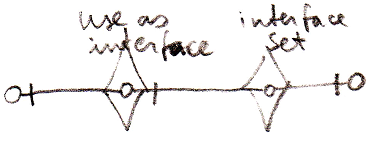


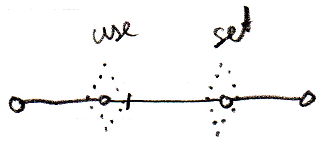
#### Explicit Get & Set Arguments Notation

Below you find various notations where Get and Set arguments are explicitly drawn out, in which the Interface aspect is addressed.









### Interface Connectors

Below you will find an overview of the possible connectors for the Interface aspect and the Interface Merged aspect.

|  |  |
| --- | --- |
| Use As Interface | Use Reference As Interface |
|  |  |
|  |  |
| Interface Set |  |
|  |  |
|  |  |
| Object-Interface Get | Reference-Interface Get |
|  |  |
|  |  |
| Interface Merged Get | Interface Merged Set |
|  |  |

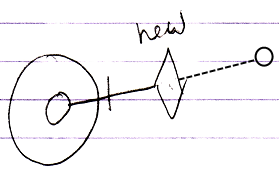
### Interface Connections

Below you will find the possible connections for the Interface aspect and the Interface Merged aspect. A connection is the result of a call to a system command. However, the result of call to the Interface Merged Set command is that the reference becomes a triangle or a circle shape. This result is also presented in this overview.

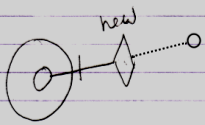
|  |  |
| --- | --- |
| Use As Interface | Interface Set |
|  |  |
|  |  |
| Interface Merged = True | Interface Merged = False |
|  |  |

### New Command with Interface Parameter

The New command was introduced in the *System Objects* chapter, having an optional Class parameter:



But now also the Interface can be specified upon creation of an object. This gives the New command an extra implementation:



Now the New command’s Interface parameter is filled in. The new object will have a fixed interface, but the object does not have a class. This means the Publics are fixed by the interface, but an implementation is free to fill in.

You do not need to set both Class and Interface parameter of the New method, because if the Class parameter is set, it determines the Interface too.

It is also not required to set either parameter, because that will create an empty object in which you can put arbitrary content, that is not bound to a class or interface at all.