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

## 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.