## Circle Language Spec Plan, System Objects Spec, Project Summary

*Author: JJ van Zon*

*Location: Oosterhout, The Netherlands*

*Date: August 3, 2008*

### Goal

Work out the System Objects article group,

required to work out the Assignment article group,

required to work out Commands as a Concept.

### Super-project

This project used to be part of the project ‘Command As A Concept’, which proved to be too large, so it was split up into multiple projects.

### Time

July 14, 2008 – July 24, 2008

11 days

32 hours of work

### Products

The following was produced:

*System Objects article group*

version *2008-07-24 00 2.0*

**38** articles:

*System Objects*

*System Objects in a Diagram*

*Objects Floating Around*

*Objects Floating Around in a Diagram*

*Related Item*

*Related Item in a Diagram*

*Controlling a Related Object*

*Controlling a Related Object in a Diagram*

*Pointer To Pointer*

*Pointer To Pointer in a Diagram*

*Symbol*

*Symbol in a Diagram*

*Object Get & Set*

*Object Get & Set in a Diagram*

*Pointer To Pointer Get & Set*

*Pointer To Pointer Get & Set in a Diagram*

*Value Get & Set*

*Value Get & Set in a Diagram*

*New & Annul*

*New & Annul in a Diagram*

*Related Lists*

*Related Lists in a Diagram*

*Related List Item*

*Related List Item in a Diagram*

*Add*

*Add in a Diagram*

*Remove*

*Remove in a Diagram*

*Names of Related Items & Related Lists*

*Names of Related Items & Related Lists in a Diagram*

*Calling a System Command*

*Calling a System Command in a Diagram*

*Related Items & Related Lists Collections*

*Related Items & Related Lists Collections in a Diagram*

*Extending the System Interface*

*Extending the System Interface in a Diagram*

*System Objects Summary*

*System Objects Summary in a Diagram*

### Project steps

- Create articles for each term

- Read over articles

- Reformulate articles

- Create diagram expression articles

- Cover details

Horribly much is involved.

The implementation of the system objects almost completely covers the code base.

It is essential for understanding the workings of the new computer language.

The functional description of the system objects is essential for understanding many of the basic concepts and principles, such as assignment, concepts and static members.

- Details:

- I think I want to add references to the articles of the members of a system interface to the articles about for instance Related Item, Related List and Related List Item.

> Anything in these brainstorm texts is probably worth mentioning somehwere in the documentation, because if I have trouble with it now, it is probably worth explaining to somebody else, that wiill be dealing with the material.

- Functional aspects:

- Related list item extension procedures

- I didn’t determine the notation for the list item extension procedures yet.

- After a lot of drawings, the final drawing was the final notation of it

- 2008-07-20 07

- 2008-07-21 01

> Draw it out differently: give related list items a reference to a list, and give the list a reference to a system procedure.

- 2008-07-21 02

- 2008-07-21 03

- 2008-07-21 04

- 2008-07-21 05

Notation of extending a system command of a related list item.

- 2008-07-21 06

Notation of all extensible parts shown in the system interface of a related list.

- Don’t draw out parameter passing

> In this part of the documentation it is not yet worked out how to pass parameters, so you can’t really draw out parameter passing yet. Do mention that in the documentation, that parameter passing is not drawn out, because it parameter passing will only be introduced later.

- This just covers the notation, not the way the system objects are tied together internally.

- This alters the following articles:

- Related Lists

Add the related list item command extension objects to the sum-up of system interface members.

- Related List Item

mainly add a small part about how the implementation is not covered there.

- Extending the System Interface

I have to say that the Item commands and Gets and Set in the system interface of the related list are not commands, but objects, with references to the extension procedures, because the Item Gets and Sets and commands in the related list are not executable commands, but *configurations* of the commands of the related list items, and a configuration is stored by an object.

- Extending the System Interface in a Diagram

Show how the extension of an item elements of the system interface of related list looks.

- System Interface Summary

Add the related list item command extension objects to the system interface of a related list.

- System Interface Summary in a Diagram

Show the diagram with all the extension objects of a related list.

- Extension procedure part of parent:

- But the extension procedures are not defined by the list object. The extension procedures are defined by the parent object.  
This counts for related items too. A related item’s extension procedures are defined by the parent object.  
Extension procedures need to be normal commands, not system commands.

> It is just more practical to make the extension procedures part of the parent object. Only a normal computer language object can define commands, that have all the possibilities of computer language. You can’t really change objects like related items or related lists, because they are system objects. Those system objects can only be extended, not changed.

- Object references inherently part of parent:

> Theoretically it does not matter whether extension procedures are defined by for instance inside a related item or inside the parent object of a related item. A related item is part of the parent object anyway, as opposed to the actual object the related item refers to. The reference to the object *is* part of the parent object. The object is not.

> Objects referred to by a parent object are not part of the parent object, but the *references* to the object *are* part of the parent object. Objects such as a related list or a related item are system objects that are inherently part of the parent object. Only the objects referred to are not part of the parent.

- System objects

- Rename the whole concept of System Interface to System Objects.

- Adapt the articles:

- System Objects

- Objects Floating Around

- Related Item

/ Controlling a Related Object

- Pointer to Pointer

/ Symbol

- Object Get & Set

- Pointer To Pointer Get & Set

- Value Get & Set

- New & Annul

- Related List

- Related List Item

/ Add

/ Remove

- Extending The System Interface

- Summary

- I need to define the term *system object* somewhere, as a term mainly used for objects such as related items, related lists and related list items.

- The system objects and commands are basic objects and commands implemented by the code base. The system objects are not normal objects. They are special deep-core system objects, that control relations between normal objects.  
In the first layer of the code base, the system objects and system commands are not comparable to normal objects and commands in the new computer language. But the code base gets reprogrammed using te new computer language itself and then the system objects and commands are implemented the same as any other object or command. But they are *still* the *system* objects, that have the special position of controlling the other objects.

- You have to make clear, that the notation of the system objects and the implementation of system objects are totally separate things.

- System objects such as Object . ID are also system objects: they are the *related objects*: an object reference embedded inside a related item or related list item. Perhaps I should put the term related object in the system interface summary, in the diagram that shows all the terms.

- Explain somewhere, why a system object is often called a system interface

The system object really represents an object. So usually you will see a system object as being the object itself, while the system object really only controls a pointer to the actual object. So when you see the members of target object of a system object, showing the actual members of the system object, is like showing the *system interface* of a related object. That’s why instead of calling it a system object, it is often callled a system interface of a related object.

- Writing style:

- It is stupid, that I can’t see the complete system interface of a related item in the article Related Item in a Diagram. I only get redirected to other articles for that. The reason I did that, was because I only explain the system commands in other articles. But I still want an overview of it, even though I haven’t explained anything about it yet.

> Related List Item

- Something sucks about the summary article, because not everything is mentioned in it.

- In the member articles (e.g. Object Get & Set) refer to the articles about the system objects to which the members apply.

- A parent object should be able to define multiple extension procedures for the same kind of extension, for the same related item or related list, because when you let an object support a concept, the concept can add another extension to a the system command. So an extension procedure is not a single-cast event.

- Access control

A lot of things about the system interface needs to be access controlled. Access control will only be worked out in the future. You do however have to mention it in the documentation everywhere, where access control has to be applied. Refer to the access control documentation, but also immediately mention the needed access control’s physical effect.

Do put in the preliminary access control documentation, that you need to consider everything about access control in the system interface.

- Notation of call to a system command.

- Consider related item’s ID In Parent…

> I’m thinking right now (2008-07-21 19:16), this is part of relation synchronization.

> No (2008-07-24 19:27). It is part of the concept of having a list of Related Items inside the parent object.

- System objects / code base / concepts,

2008-07-24

System Objects covers almost the complete code base

This means, that for a complete overview, you would still need the Related Items and Related Lists system objects. But the usage of system objects may not require knowledge of the objects Related Items and Related Lists. However, in order to author concepts, you have to have those collections available. You’ll also need to be able to extract an Attributes collection as well. And a methods collection.

It is worth considering completing the story like that.

But it already has been a lot of work, and I might want to just get on with it.

- A related item and related list have a name. The name should be a system member, because it needs to be used by concepts.

- Add members to summaries and articles related item, related list:

- ID In Parent

- Name

- Add something about the collections Related Items and Related Lists to the summary.

- I don’t have any references to the main article. I don’t redirect from the main article to the sub-articles.

### Reflection

It is annoying to first do the conceptual explanation and after that do all the diagram notation articles.

Then, if you work on the diagram notations, it is like you are not moving foreward, because were you already finished with the conceptual explaination.

But that was the only way to do it for the System Objects articles, because the conceptual explanation was created from a brainstorm, that was later split up into topics and pieces of text were dropped into the articles, and after that, the text was reformulated. And then, the diagram notation had yet to be done.