|  |
| --- |
| Circle Language Spec: Inheritance |

## Extending the System Objects

In the *System Objects* articles it has been repeatedly mentioned, that all the system commands can be extended with extra procedures. First, to demonstrate the concept, this article gives you the main examples of system command extension. After that, all the possible system command extensions will be listed out.

### Main examples of system command extension

By default the Object . Get and Object . Set commands just return and set the object. But calls to Object . Get and Object . Set can also be extended with procedures around the assignment and retrieval of the object.

The Object . Get command can be programmed. By default it just returns the target object of the related item or related list item, but you can program it to return any object, resulting from a procedure as complex as you want. One application of this is letting the Object . Get return a related item from another parent, so that another parent determines the related item. Functionally, this establishes a pointer to a pointer. From the outside, you don’t see this; the symbol just looks like the target object. (A pointer to pointer is indeed also possible in a more basic form. See the article *Pointer to Pointer* for that.)

The Object . Set command can also be programmed. Being able to program the Object . Set command is particularly handy. In the Object . Set you can add validation of the assigned object. The newly assigned object may also need to be applied to other objects. The changes to other objects can be immediately applied by a procedure extending the Object . Set command.

Actually, the *extension* of the system interface is the most important thing about even being *aware* of the system interface.

Even though a value is stored inside the object *itself*, the system commands of a *related item* or *related list item* allow you to build a procedure around assigning a value to a sub-object.

By default the Value . Get and Value . Set commands just return and set the value. But calls to Value . Get and Value . Set can be extended with procedures around the assignment and retrieval of the value.

### Command Extension Possibilities

Only the system *commands* of the system interface are extensible with procedures.

Related items and related list items have the following system commands:

Object . Get

Object . Set

Other Related Item . Get

Other Related Item . Set

Other Related List Item . Get

Other Related List Item . Set

Value . Get

Value . Set

New

Annul

Created . Get

Created . Set

Related list items also have the system command:

Remove

A related list has following the system command:

Add

Only the system *commands* are listed out above, not the other system members.

Extension can be done in three different ways:

- Pre-extend

- Post-extend

- Overriding

These types of extension can be applied to each system command. Pre-extension triggers a custom procedure *before* running the system command’s default procedure. Post-extension triggers a custom procedure *after* running the system command’s default procedure. The same system command can be both pre-extended and post-extended at the same time. A system command can also be overridden completely, and the default procedure completely omitted. Inside the overriding procedure you can always decide to call the default procedure anyway. So in theory, you only need overriding, because that gives you the ability to either pre-extend and post-extend. But it’s more intuitive to be able to immediately choose to *extend*, instead of first choosing override and *then* choose to extend.

Extension procedures may be implemented by the system interface as events raised by the system commands. But how to best implement procedure extension can be determined later.

A related item will have the following extension possibilities:

Object . Get . Pre-Extend

Object . Get . Post-Extend

Object . Get . Override

Object . Set . Pre-Extend

Object . Set . Post-Extend

Object . Set . Override

Other Related Item . Get . Pre-Extend

Other Related Item . Get . Post-Extend

Other Related Item . Get . Override

Other Related Item . Set . Pre-Extend

Other Related Item . Set . Post-Extend

Other Related Item . Set . Override

Other Related List Item . Get . Pre-Extend

Other Related List Item . Get . Post-Extend

Other Related List Item . Get . Override

Other Related List Item . Set . Pre-Extend

Other Related List Item . Set . Post-Extend

Other Related List Item . Set . Override

Value . Get . Pre-Extend

Value . Get . Post-Extend

Value . Get . Override

Value . Set . Pre-Extend

Value . Set . Post-Extend

Value . Set . Override

New . Pre-Extend

New . Post-Extend

New . Override

Annul . Pre-Extend

Annul . Post-Extend

Annul . Override

Created . Get . Pre-Extend

Created . Get . Post-Extend

Created . Get . Override

Created . Set . Pre-Extend

Created . Set . Post-Extend

Created . Set . Override

These are the new computer language’s equivalent of Getters, Setters and Property procedures.

This adds a lot more possiblities to surround a sub-object’s behavior, than just the one Getter and Setter, that other computer programming languages offer. You don’t have to define all of the extension procedures. When you don’t define a Getter or Setter, the related item will just have default Getter and Setter behavior.

🡪n related *list items* have the same system commands as 🡪1 related items, but the extension of a related list item’s system commands is the same for every item of the list, so the extension procedures for related list items are defined by the related list. The related list items have a reference to the list they belong to, and also store the position in the list, as well as a Remove command, that removes the item from the list. In the new computer language’s code base, a related list item automatically calls the extension procedures defined in the list, passing the position of the item in the list to the extension procedure as a parameter. Here is a list of all the extension procedures a list can define for a related list’s items:

Item Object . Get . Pre-Extend

Item Object . Get . Post-Extend

Item Object . Get . Override

Item Object . Set . Pre-Extend

Item Object . Set . Post-Extend

Item Object . Set . Override

Item Other Related Item . Get . Pre-Extend

Item Other Related Item . Get . Post-Extend

Item Other Related Item . Get . Override

Item Other Related Item . Set . Pre-Extend

Item Other Related Item . Set . Post-Extend

Item Other Related Item . Set . Override

Item Other Related List Item . Get . Pre-Extend

Item Other Related List Item . Get . Post-Extend

Item Other Related List Item . Get . Override

Item Other Related List Item . Set . Pre-Extend

Item Other Related List Item . Set . Post-Extend

Item Other Related List Item . Set . Override

Item Value . Get . Pre-Extend

Item Value . Get . Post-Extend

Item Value . Get . Override

Item Value . Set . Pre-Extend

Item Value . Set . Post-Extend

Item Value . Set . Override

Item New . Pre-Extend

Item New . Post-Extend

Item New . Override

Item Annul . Pre-Extend

Item Annul . Post-Extend

Item Annul . Override

Item Created . Get . Pre-Extend

Item Created . Get . Post-Extend

Item Created . Get . Override

Item Created . Set . Pre-Extend

Item Created . Set . Post-Extend

Item Created . Set . Override

Item Remove . Pre-Extend

Item Remove . Pre-Extend

Item Remove . Override

References to these extension procedures are stored inside the related list inside the following system objects:

Item New

Item Annul

Item Remove

Item Created . Get

Item Created . Set

Item Object . Get

Item Object . Set

Item Other Related Item . Get

Item Other Related Item . Set

Item Other Related List Item . Get

Item Other Related List Item . Set

Item Value . Get

Item Value . Set

So those are all objects, not commands. The objects contain references to the extension procedures for a related list item.

A related list itself only has the following system command:

Add

The list object only has one system command. Its extension procedures are:

Add . Pre-Extend

Add . Post-Extend

Add . Override

The other extension procedures, a related list defines, are extensions for commands of the list *items*, not extensions for commands of the list object itself.

You are most likely to only use the Add . Post-Extend to extend the addition of an item and use the Item Remove . Pre-Extend to finalize an item ready to be removed. But the other extension possibilities of Add and Remove are there, because this allows you full control over the behavior of related objects, regardless of what would be most logical to do.

Extension of the system interface’s members allows you to control the inner workings of a sub-object.

### Extend system interface with more members

Other features of the new computer language can add more items to the system interface, to define the behavior of a related item. The system interface is *extensible*. For instance the concept of *the class* will add more items to the configuration of a related item. So apart from extending a system interface’s existing commands, new commands and objects can be added to the system interface.

The system interface members added by other features will automatically get all the extension possibilities as well.

### Extension regardless of value change

The various Get commands only really execute their procedure when a value is actually *changed*. That is the default behavior of the new computer language. It prevents a lot of unnecessary system activity. By default, pre- and post-extension procedures are only executed for getters, when a *different* value is actually assigned. However, a feature, that may be added to the new computer language could be:

Pre-extension regardless of value change

Post-extension regardless of value change

That feature adds two more extension possibilities to every Get command, to execute a procedure extension even when the value does not change.

### Extension procedures are events

The implementation of an extension procedure looks the same as the implementation of an event procedure. The concept of *events* has not been covered yet, but it looks the same as the notation used for system command extensions.

The reason why an extension procedure *has to be* an event, is because it is a signal from a sub-object to a parent object. A signal from a sub-object to a parent is always an event.

A system command extension event is a multi-cast event, even though you may only define one extension procedure yourself. Different concepts can automatically add more extensions to system commands, so the same extension has multiple implementations, that are all run when a system command is fired.

The parent object is the only one that can implement the system command extension event. This also counts for concepts too. Concepts just extend the parent object, so it is still only the parent object extending a system command. This is has to be access controlled. Access control is worked out later in the *Access Control* articles.

### Extension procedures part of parent

Although the system objects define the extension events, the *parent object* defines the implementaiton of the extension procedures. The extension procedures are defined by the parent object. This counts for related item, related lists and related list items. 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 cannot really change objects like related items or related lists, because they are system objects. Those system objects can only be extended, not changed.

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.

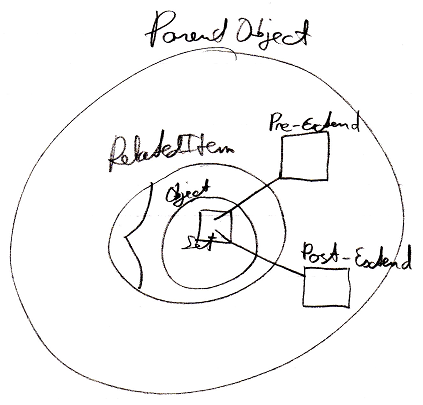
### Obsolete members: Initialize Item & Terminate Item

In an earlier version of the new computer language, version 2.0, a related list also had the extension procedures Initialize Item and Terminate Item. In the new version of the new computer language, version 3.0, Initialize Item is replaced by Add . Post-Extend and Terminate Item is replaced by Remove . Pre-Extend.

### Extension of System Objects in a Diagram

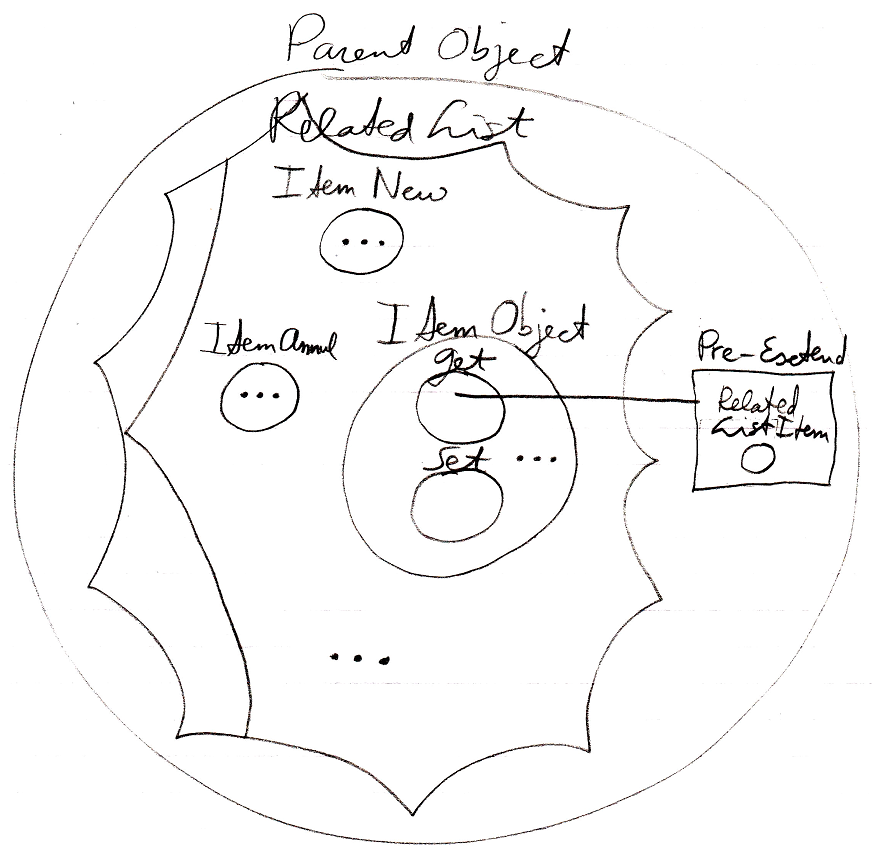
The concept of extending the system objects is already explained in the article *Extending the System Objects*. The current article demonstrates its expression in a diagram.

Not all possible extensions of the system commands will be displayed in this article. Any extension procedure of the system interface commands is displayed the same way. Only a couple of extension procedures will be displayed, to demonstrate the idea.



In the picture above, a Related Item is shown inside a Parent Object. The system interface of the related item is opened up. You can see the Object part of the system interface. Inside the Object part of the system interface is displayed the Object . Set command. Lines are going out of the Object . Set command to the Pre-Extend and Post-Extend procedures, that are implemented by the Parent Object.

Extension of system commands of a related list item works a bit differently, because the extension procedures of a related list item are the same for all items in the list, thus the extension procedures are defined by the related list. The related list defines a number of objects, that contain references to the extension procedures. When you open up the system interface of a related list, you can specify the extension procedures.



The examples shows the system interface of a related list, but not all its members. The following system objects are shown: Item New, Item Annul and Item Object. The rest of the system objects are left out. The system object Item Object contains the Get and Set objects. Objects like Item Object . Get, that have the same name as a system command of a related list item, define the extension procedures. In the example above, the system object Item Object . Get has a pre- extension defined. The extension is displayed as a line going out of Item Object . Get to a procedure named Pre-Extend, defined inside the parent object. The extension procedure is passed a parameter, namely: the Related List Item for which the extension procedure was callled.

## Ideas

*The texts below are loose ideas yet to be turned into good documentation.*

### Extension (cross-out leftovers from System Objects)

Assignment and retrieval of the related item can be decorated with extra procedures.

Decorating assignment and retrieval of a related item with extra procedures will be covered by the article *Extending the System Interface*.

Each item in the list has different data in its system object, but extensions to the procedures around assignment or retrieval of an item in the list are the same for each item. That’s why the extension procedures of related list items are defined inside the related list.

A related list’s system interface has objects, that define the extension procedures of related list items’ system commands:

Item New

Item Annul

Item Created . Get

Item Created . Set

Item Object . Get

Item Object . Set

Item Other Related Item . Get

Item Other Related Item . Set

Item Other Related List Item . Get

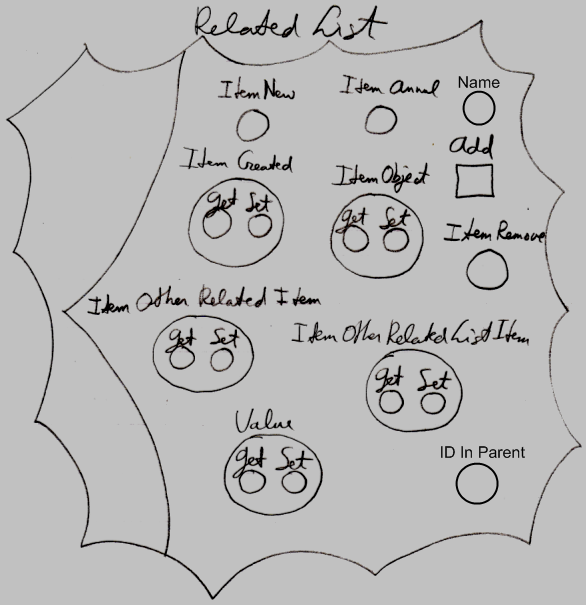
Item Other Related List Item . Set

Item Value . Get

Item Value . Set

Item Remove

The complete system interface of the related list system object looks like this:



So most of those are objects, not commands. The objects only contain references to the extension procedures for related list items.

For a related *list item* extension procedures of assignment and retrieval are delegated to the list, because the assignment and retrieval procedure extensions are the same for every item in the list.

At first there was the idea, that a related list item could get the same behavior as a related item. The reason why a related *list item* is not the same as a related *item*, is because a related *list* *item* does not define its own decoration with procedures. The decoration of the assignment and retrieval of the a related list item is delegated to the related list and is the same for all the items in the list.

Actually, system objects would not be an equivalent of Getters and Setters if it weren’t for the *extension* of system commands. A summary of the system objects would not be complete without a list of all possible system commands and all its possible procedure extensions. Below you will find lists of all the extension possibilities of the system objects. For explanations about the extension possibilities, refer to the article *Extending The System Objects*.

Related item extension possibilities:

Object . Get . Pre-Extend

Object . Get . Post-Extend

Object . Get . Override

Object . Set . Pre-Extend

Object . Set . Post-Extend

Object . Set . Override

Other Related Item . Get . Pre-Extend

Other Related Item . Get . Post-Extend

Other Related Item . Get . Override

Other Related Item . Set . Pre-Extend

Other Related Item . Set . Post-Extend

Other Related Item . Set . Override

Other Related List Item . Get . Pre-Extend

Other Related List Item . Get . Post-Extend

Other Related List Item . Get . Override

Other Related List Item . Set . Pre-Extend

Other Related List Item . Set . Post-Extend

Other Related List Item . Set . Override

Value . Get . Pre-Extend

Value . Get . Post-Extend

Value . Get . Override

Value . Set . Pre-Extend

Value . Set . Post-Extend

Value . Set . Override

New . Pre-Extend

New . Post-Extend

New . Override

Annul . Pre-Extend

Annul . Post-Extend

Annul . Override

Created . Get . Pre-Extend

Created . Get . Post-Extend

Created . Get . Override

Created . Set . Pre-Extend

Created . Set . Post-Extend

Created . Set . Override

Related list extension possibilities:

Add . Pre-Extend

Add . Post-Extend

Add . Override

Related list item extension possibilities:

Item Object . Get . Pre-Extend

Item Object . Get . Post-Extend

Item Object . Get . Override

Item Object . Set . Pre-Extend

Item Object . Set . Post-Extend

Item Object . Set . Override

Item Other Related Item . Get . Pre-Extend

Item Other Related Item . Get . Post-Extend

Item Other Related Item . Get . Override

Item Other Related Item . Set . Pre-Extend

Item Other Related Item . Set . Post-Extend

Item Other Related Item . Set . Override

Item Other Related List Item . Get . Pre-Extend

Item Other Related List Item . Get . Post-Extend

Item Other Related List Item . Get . Override

Item Other Related List Item . Set . Pre-Extend

Item Other Related List Item . Set . Post-Extend

Item Other Related List Item . Set . Override

Item Value . Get . Pre-Extend

Item Value . Get . Post-Extend

Item Value . Get . Override

Item Value . Set . Pre-Extend

Item Value . Set . Post-Extend

Item Value . Set . Override

Item New . Pre-Extend

Item New . Post-Extend

Item New . Override

Item Annul . Pre-Extend

Item Annul . Post-Extend

Item Annul . Override

Item Created . Get . Pre-Extend

Item Created . Get . Post-Extend

Item Created . Get . Override

Item Created . Set . Pre-Extend

Item Created . Set . Post-Extend

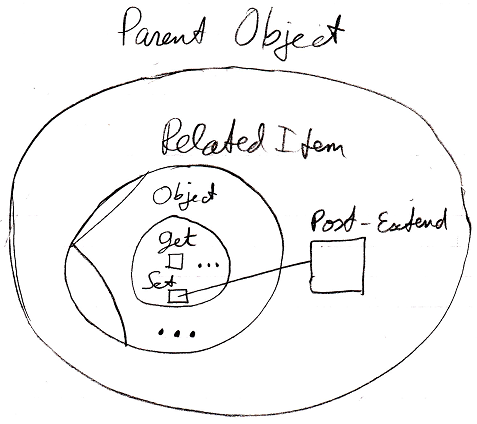
Item Created . Set . Override

Item Remove . Pre-Extend

Item Remove . Pre-Extend

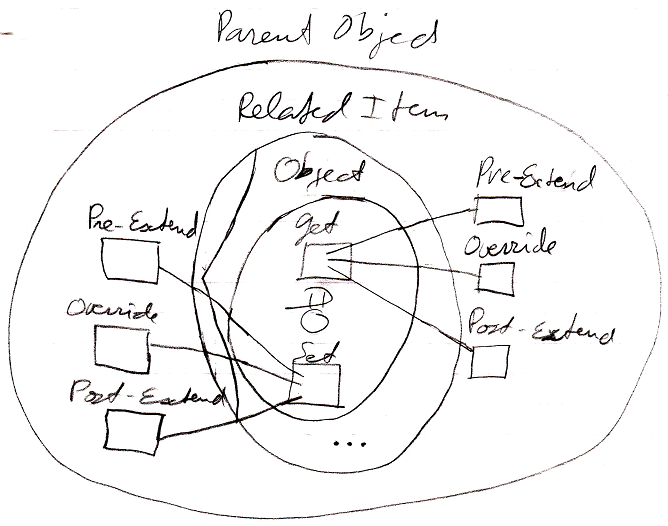
Item Remove . Override

Usually not all members of a system object are shown, but only the parts that are extended or used. In that case the diagram shows an ellipsis in the spots where the additional members are not displayed:

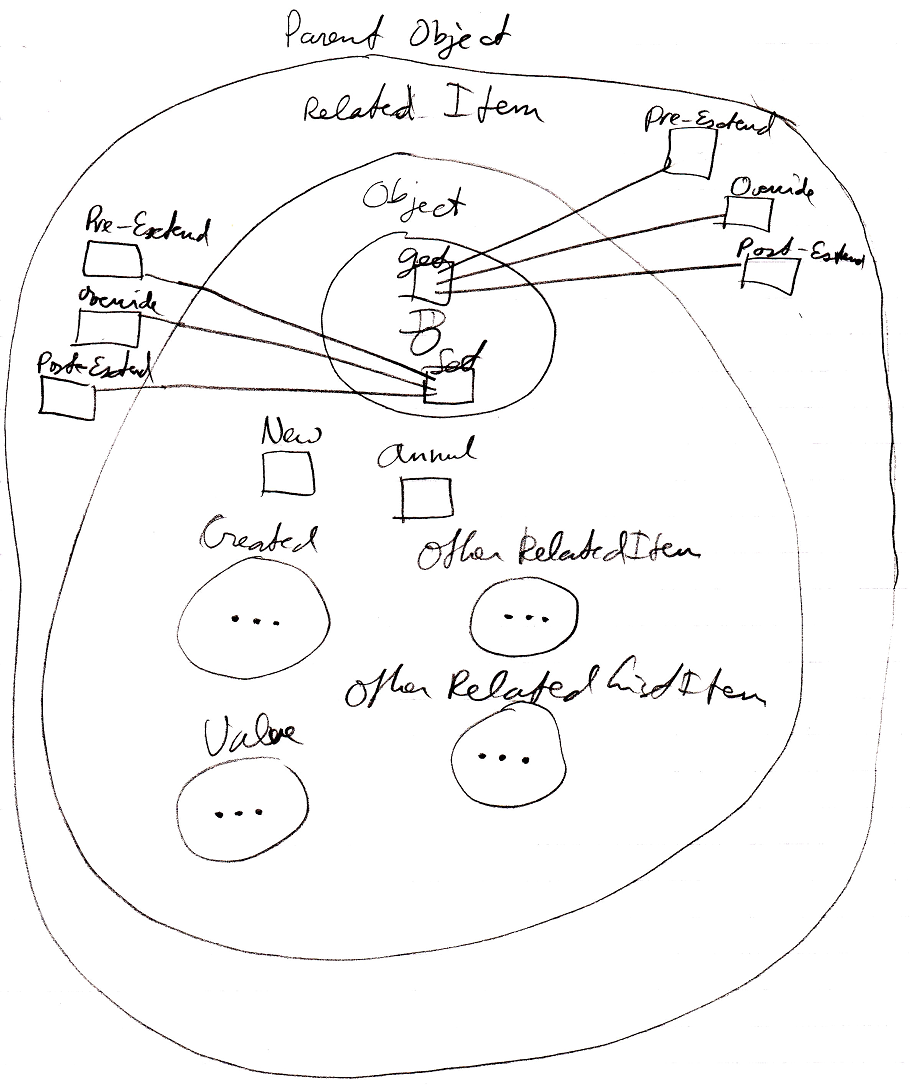


Each ellipsis can be double cliked or clicked, or something, to show all the other system object members and extension possiblities.

When you double click the ellipsis inside the Object part of the system interface, all the Object members are shown and their extension possibilities:



If you double click the ellipsis at the bottom of the related item symbol, all system interface members are shown, but not *their* members.



To also display the members and extension possibilities of another part of the system interface, you have to double-click another ellipsis.

A system interface is the new computer language’s equivalent of Getters, Setters and Property procedures. But the new computer language offers a lot more to control the behavior of assigning or retrieving a related object compared to other programming languages.

System Objects implementation,

2008-07-23

- System object implementation and notation are separate things

- The extension procedures for related list items must become procedure references, part of the system interface of a related list.

> Actually, internally all system command extensions everywhere are procedure extensions, but in the diagram it is just differently expressed. The internal workings of system objects is different from the way it is expressed in a diagram.

> The system interface of a related list must be extended with all system members a related list item has, but then the members are not callable but only extendable.

- Implementing the tie between related list item and related list:

> If a related items, related lists and related list items get a reference to the parent object, then this makes the system command extensions of the parent object accessible. Each of the system objects’ commands has to be tied to a specific command extension, though.  
For related items and related lists, each system command’s extension is individually assignable. But related list items have the special feature, that they are not individually assignable. Therefore, each related list item as a reference to the related list, that then has a binding to a system procedure extension.

- I stated, that the related item needed a reference to the list to call the extension procedure of the list object.

JJ

Parameters for System Commands,

2009-08-31

Parameters for System Commands should also belong to extension, but it is currently inside the System Objects article group.

JJ