

# PatternFactory Developer Standards

## Fundamental Idea

The PatternFactory patterns are class objects of high using comfort for developers and users with highest possible flexibility and independence of existing classes, but without limitations. It should be possible to primarily use the classes by external settings (inheriting, replacing, scoped messages and field values). Mandatory amendments to the action diagram should be kept to an absolute minimum. The generated code should be optimized and compact without unnecessary instructions (no overhead).

**“To build your application by gathering (is a) and setting options (replacing)!”**

## Keep Things Compact and Tidy (Structure)

Making use of classes often can be quite tricky, even if you know them well. Often you have to gather the necessary objects from various places, having to browse objects from the same or even different object types. Therefore, the goal is to facilitate this process:

- Hold all necessary class objects together by scoping them, if possible, by a container entity. Exceptions are objects which cannot be scoped, like variables or names.
- If your pattern library consists of several sub classes, build a container entity for each of them and scope them all by one superordinate container entity.
- In addition, arrange your class objects by dividing your primary container entity into further group objects like entities or functions:
  - an entity called Aux for any kind of auxiliary objects, if existing (e.g. functions, fields, messages, scripts, source codes)
  - an entity called ReplObjects for abstractly defined objects used by the class itself (e.g. views, functions, fields) to be replaced in the end when going life
  - the actual class object called Class, mostly being a function or an entity



Figure 1

## Action Diagramming Rules

- Put your detail code into subroutines.
- Append the function name in brackets at the end of your own subroutine names. Makes it very useful when merging functions by multiple inheritance to see the origin of every subroutine.
- Build subroutines in the same way as the standard Pattern libraries by CA do.
- Append the function name in brackets at the end of your own edit point names. This way in the local modification view and in the edit point dialog of the AD Editor you always know immediately where the edit point is located.
- Place your subroutines in Pre Point Subroutines. This way the Post Point Subroutines stays free for any user amendment and thus stays easier to navigate.

```

Sub Execute (5250.API.Class.Server.Dolt)
  +++Define Field: FIELDS/+Subroutine
  Edit Point Start Execute (5250.API.Class.Server.Dolt)
  +If Field: FIELDS/+Subroutine
    Seq Prepare input parameters of 5250 program
      Set Local/CalledProgram = Input/CalledProgram
      Go Sub Receive 5250 program parameters (5250.[Aux].Fnc.GetParmFromDb)

    Seq Call 5250 program
      Edit Point Additional API call preparations (5250.API.Class.Server.Dolt)
      Name Function: 5250.API.Class.5250Pgm, Environment<"CalledFunctionName">, .Scoped
      Call 5250.API.Class.5250Pgm
      Go Sub Check error
      Edit Point Post API call preparations (5250.API.Class.Server.Dolt)

    Go Sub Prepare returned output parameters of 5250 program (5250.API.Class.Server.Dolt)
  Edit Point End Execute (5250.API.Class.Server.Dolt)
  +++Undefine Field: FIELDS/+Subroutine

```

Figure 2

- Avoid unnecessary generated code by using meta conditioning where ever you can.

## Documentation

- Again, to keep things compact and tidy we chose to use the Plex model narratives facility to document the patterns. Enter your documentation into the narratives of the class container entity.
- The documentation should consist of:
  - Date and Author
  - Title
  - Description of the pattern
  - Implementation rules, detailing the steps to take to achieve the requested application object.



Figure 3

## Packaging

- Pack the contents of your group model directory into a zip file. The subdirectories \Backup and \Locks are not necessary to be delivered and for this reason may be omitted.
- Enclose a ReadMe text file in the zip file with the following contents:

```

21.01.2003/P.Fabel
Copyright © 2002 PatternFactory. All Rights Reserved.

PatternFactory ReadMe
-----

- Unzip the delivered group model objects into a directory of your choice.

- Attach the group model of the selected PatternFactory pattern as library to your application model.

- The pattern documentation may be found in the narratives of the main class entities.
    
```