ChangeMan ZMF skeleton coding techniques

- [Z-Practices] |
- [ZMF Administrator] |
- [Z-Wiki Subscribers] |
- [Z-Wiki Roadmap] |
- [1.1-Chimay]

Release info**Announcement date:** Mon, 06/08/2012
Applicable ChangeMan ZMF: Any ChangeMan ZMF release

To introduce this topic, consider these questions (and related answers):

1. **Question**: What's the difference between an install (CMN20) and backout (CMN50)?
   Answer: In both cases you typically copy from some input (distributed staging lib in CMN20, hot backup lib in CMN50) to some output (production lib in both CMN20 and CMN50).
2. **Question**: What's the difference between promote (CMNRPMCR) and install (CMN20)?
   Answer: it is more or less the same, it just happens in another environment (and in ChangeMan ZMF the skeleton logic and related variables are implemented via skeletons like CMN$$RPM versus CMN20).
3. **Question**:What's the difference between demote and backout?
   Answer: it is more or less the same, it just happens in another environment.

Here are a few samples to further illustrate the actual challenge:

- The DB2-option skeletons CMN21, CMN49, CMN56, etc (= ChangeMan ZMF components that quite often get customized) are a good sample of this also (compare CMN21 to CMN49 as delivered, and note how a lot of lines are exactly the same ...).
- Many (vendor versions of) ChangeMan ZMF staging skeletons all have the same set of about 20 to 40 lines related to PGM=CMNBAT90 and PGM=SERCOPY, used for all sorts of staging output processing. Additionally, many customized staging skeletons (to support additional programming languages via a custom staging procedure) are constructed via cloning (=cut-and-paste, not reusing!) of one of the vendor versions of staging skeletons (e.g. cloned from skeleton CMN$$LNK).

Think about the impact of all this when it comes to:

- building some ChangeMan ZMF customization.
- upgrading customizations from one ChangeMan ZMF release to another.
- how many updates are required to just add or change a DDName, a parm, etc.

We've been there, we've experienced the pains, and we've learned over the years how to minimizes these kind of challenges. That's what has grown over the years into what we've started to call **Object Oriented skeleton coding**. Here are some of the headlines of it:

- To support a special component type (like QMF, DDL, ACBGENs, JCL validations, etc), just construct a single (!!!) skeleton which is used for **functions** like checkout, stage, promote, install, backout, etc. To be able to reuse such skeleton over and over, it uses input and/or output DSNs based on the value of a variable to indicate the **function** being executed.

Copyright © 2000-2025 AbitMORE ® BV. All rights reserved

- Enforce the use of **local skeleton variables**, which are used to drive the output of what get's constructed from a skeleton (= the generated JCL). Such local skeleton variables typically receive a default value in the beginning of such skeleton (unless their value was set before imbedding the skeleton), and their value gets destroyed at the end of the skeleton (to avoid the local skeleton variable being used anywhere else).

Using these ChangeMan ZMF skeleton coding techniques, we've been able to support special component types like QMF, DDL, ACBGENs, JCL validations, etc. To do so, we typically construct a single skeleton (used for functions like checkout, stage, promote, install, backout, etc.) that will use input or output DSNs based on the value of a variable to indicate the function being executed. These techniques also facilitate the reuse of software components to develop more complex features. Probably the best illustration of this is Simplify CMNBAT90 - SERCOPY usage to activate components, which is like a function to handle all sorts of staging output processing via CMNBAT90 and SERCOPY job steps.

Here are some more Z-Issues for which their corresponding (amazing?) Z-Files that were developed compliant to these skeleton coding techniques:

- Add restart instructions in Xnode jobs.
- Analyse the content of a package.

For some more illustrations of this concept, read about it in:

- The benefits you can (will!) get from it, as illustrated in ChangeMan ZMF 7.1 - Measure customization upgrade progress (near the end).
- Some of the techniques used to Replace IEBCOPY by SERCOPY where possible (as mentioned near the end).

Source URL (retrieved on *2026-11-04 03:12*):
http://dr.chgman.com/z-wiki/z-practices/skeletons/guidelines

Copyright © 2000-2025 AbitMORE ® BV. All rights reserved