EDG-LCFG integration issues

CERN, 20/6/2002

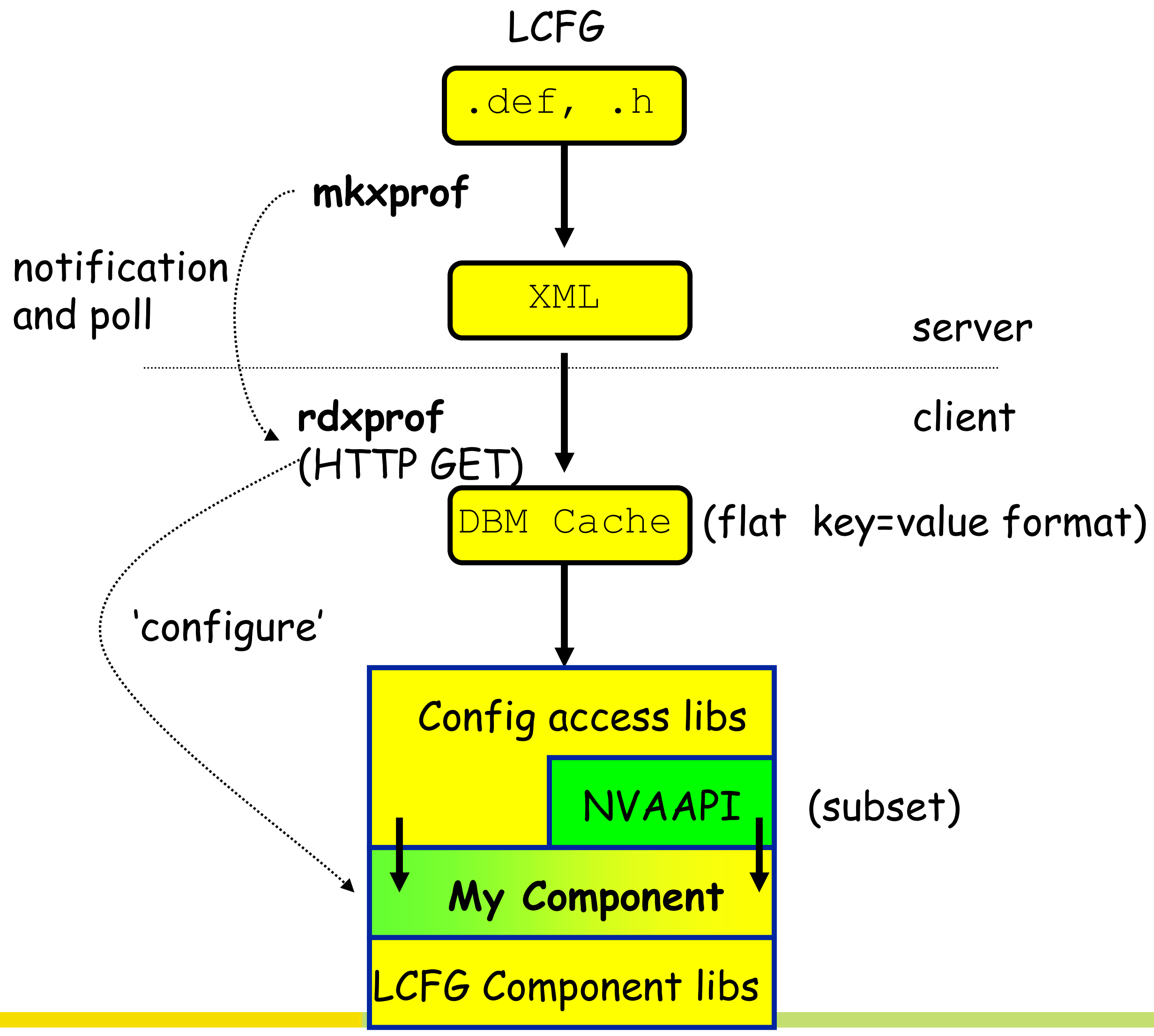
German.Cancio@cern.ch

Overview

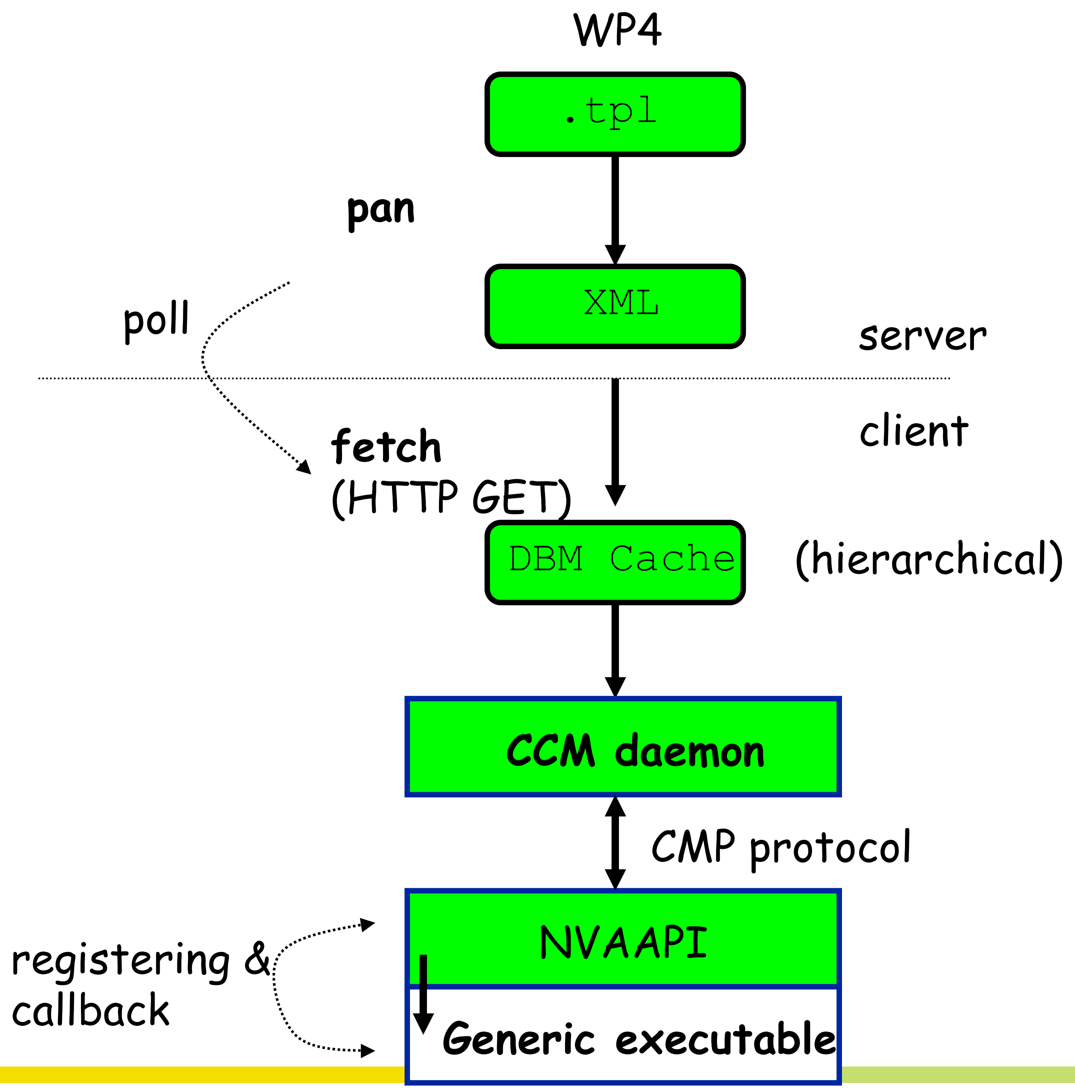
- LCFGng vs. WP4-config
- Choices for interfacing LCFG with WP4-config
- Discussion

LCFGng: configuration flow

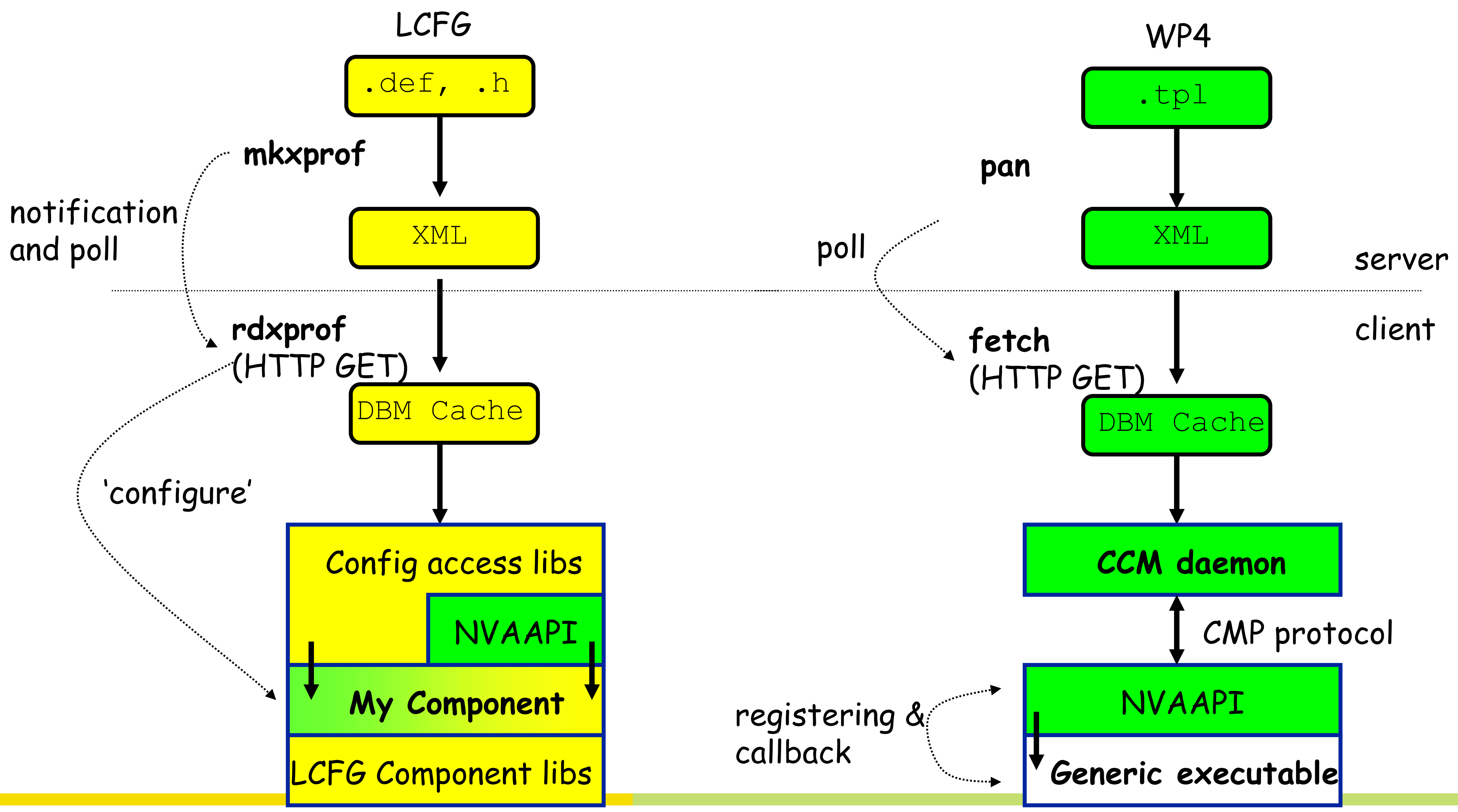
🟨 LCFG developments
🟩 EDG developments



LCFGng vs. WP4-config flows



Comparison



LCFG
.def, .h
mkxprof
notification and poll
XML
rdxprof (HTTP GET)
DBM Cache
'configure'
Config access libs
NVAAPI
My Component
LCFG Component libs
WP4
.tpl
pan
poll
XML
server
client
fetch (HTTP GET)
DBM Cache
CCM daemon
CMP protocol
NVAAPI
registering & callback
Generic executable

Interfacing LCFG with WP4-config

Approach #1: Replace `mkxprof` with `pan`, use common XML

- Currently, `pan` can generate LCFG-compatible XML.
- However, current LCFG components require resource definition information (.`def` files) for 'flattening' resources into key-value pairs. This legacy information is not necessary in HLD. Two possibilities:
  - A) incorporate component resource definition information into special templates, and send over XML
  - B) copy resource definition information to the client side and evaluate it by rdxprof.
- Advantages:
  - Backwards compatible
  - Clear cut in responsibilities LCFG <-> WP4-Config
- Issues:
  - Needs a much more precise definition of the XML profile structures!
  - Compiler needs to support ordered named lists (not the case currently)
  - The NVA API implementation has to 'unflatten' key-value resources -> complex!
  - Resource definition information uses a different syntax than the HLD.
  - Both the HLD and the resource definition info have to be kept in sync.
  - Does not support the global schema.

Interfacing LCFG with WP4-config

Approach #2: same as #1, but with implicit resource definitions

- For new components, rdxprof could use implicit conventions for avoiding the `.def` files.
- Advantages:
  - Same as #1
  - New components don't require .def files
- Disadvantages:
  - No support for global schema.
  - NVA API has to unflatten key-value resources -> complex.

Interfacing LCFG with WP4-config

Approach #3: change LCFG flat DBM format

- Use the CCM DBM format, which allows for config caching *without* information loss (no key-value resources, no need for 'resource definitions').

- Advantages:
  - Direct access for NVA API
  - No complex resource flatting/unflatting
  - Would allow global schema (needs some additional work for callback notifications)
- Disadvantages:
  - Breaks backwards compatibility (no key-value config access)

Interfacing LCFG with WP4-config

Approach #4: same as #3, keeping .def files for old components

- A backwards compatibility module allows to flat out resources for old-style components using the .def file.
- Advantages:
  - Same as #3
  - Backwards compatibility
- Disadvantages:
  - Complexity of implementation?
  - Needs further detailed thought.

Interfacing LCFG with WP4-config

Approach #5: use CCM

- Replace the LCFG DBM access libraries and the NVA API subset by the CCM and full NVA API.
- Consider using fetch instead of rdxprof for reading the profile information.
- Advantages:
  - Same as #3
- Disadvantages:
  - Same as #3
  - CCM probably needs a reimplementation
  - Additional LCFG rdxprof and config access libs functionality needs to be understood and implemented if needed.