Runtime Configuration for MPC - Developper Manual

Sogeti High Tech

June 5, 2013

Contents

1 Introduction

Since MPC 2.5.0, a configuration system has been introduced through the module `MPC_Config`: it enables the user to setup some parameters at the runtime when running his binary with MPC.

This manual will explain to a developper how the configuration system is designed and how to add parameter into it.

2 The module MPC_Config

All the configuration system is implemented into the `MPC_Config` module which gives functionnalities to generate:

- The configuration structure;
- The UNIX man for options list and values;
- The parsing source code;
- The displaying function.

3 Sources of the modules MPC_Config

The module `MPC_Config` is subdivided into several directories:

- `bin`: contains the sources of the `mpc_print_config` executable;
- `doc`: contains the user and developper manuals describing the configuration system;
- `generated`: contains all the generated files needed at the runtime (see §3.1 for more details);
- `generators`: contains all XSLT files used to generate the files in the `generated` folder;
- `src`: contains the sources for the configuration parsor from XML files (see §3.2 for more details).

3.1 Details for the generated folder

Several files are generated using the XSLT in the `generators` folder:

- `sctk_runtime_config_struct.h`: define all the C data structures (`struct`, `enum`, etc.) of the MPC configuration structure;
- `sctk_runtime_config_struct_meta.c`: define meta-description (datatype, offset into the structure, etc.) to load the MPC configuration structure;
- `sctk_runtime_config_struct_defaults.h`: define functions prototypes initializing the MPC configuration structure with the default values;
- `sctk_runtime_config_struct_defaults.c`: initialize the MPC configuration structure with the default values;
- `global-config-meta.xml`: contains the contents of each `config-meta.xml` existing in MPC;
- `mpc_config.5`: UNIX man describing all the parameters (type, default value, doc) of the MPC configuration structure;
- `mpc-config.xsd`: schema to validate the final configuration file.

3.2 Details for the src folder

The `src` folder contains the following files:

- `sctk_runtime_config_mapper.{.h,.c}`: provide the functions to convert the XML configuration file to the C structure;
- `sctk_runtime_config_printer.{.h,.c}`: use by the `mpc_print_config` executable to display the parameters for the XML configuration file, using the file `sctk_runtime_config_struct_meta.c`;
- `sctk_runtime_config_selectors.{.h,.c}`: handle selectors to select dynamically profiles at execution time;
- `sctk_runtime_config_sources.{.h,.c}`: provide the functions to open XML configuration files and to select profiles to apply;
- `sctk_runtime_config_validation.{.h,.c}`: provide a function to overwrite parameters with environment variables, and a function to check the values of the parameters;
- `sctk_runtime_config_walk.{.h,.c}`: use to run over the C configruation structure in order to display its contents;
- `sctk_runtime_config.{.h,.c}`: provide the interface that will be used in the other MPC modules.

A module `sctk_libxml_helper.{.h, .c}` is also developped to use `libxml2` to read and write XML files.

4 Configuration management workflow

The workflow of configuration management can be described by steps:

1. Write a `config-meta.xml` for each MPC module which needs to be integrated into the configuration system;
2. Run `mpc_gen_runtime_config` which will :
   - Aggregate all the `config-meta.xml` to generate the `global-config-meta.xml`;
   - Apply XSLT transformations to generate source code for configuration management;
3. Compile MPC;

The Figure 1 summarized this process.

5 Steps for developper

A developper who wants to integrate his MPC module into the configuration system must:

1. Create a configuration file `config-meta.xml` in his module and define all the options he wants to parametrized;
2. Mark the dependency to the MPC_Config module by adding in the file `module_dep`: `need_module MPC_Config`;
3. Regenerate the MPC_Config auto-generated files by execution `./MPC_Tools/mpc_gen_runtime_config` from `mpc` directory;
4. Include the header `sctk_runtime_config.h` in his source files;
5. Use the function `sctk_runtime_config_get()` to access to the configuration structure.

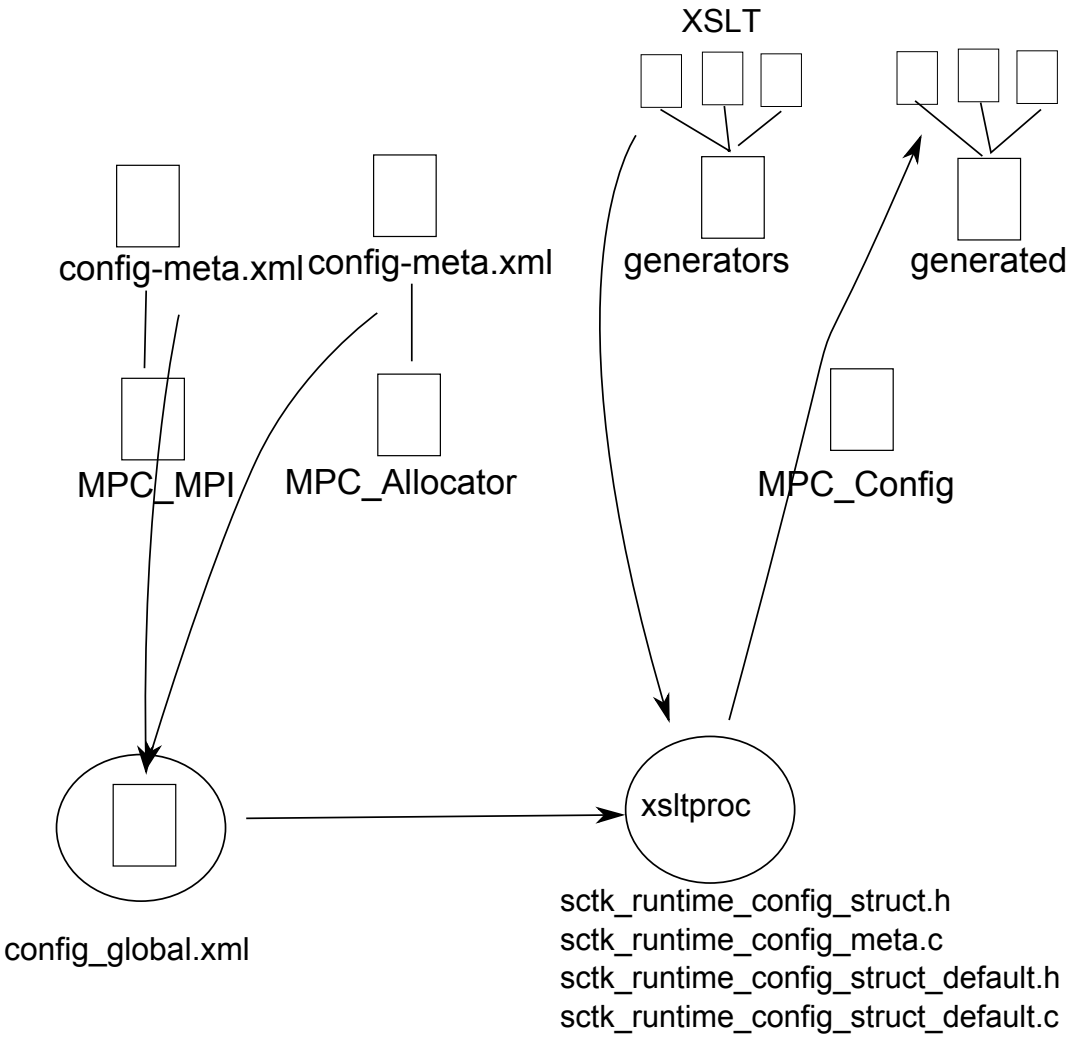

Figure 1: Representation of the workflow