



CLI to the Max

Overview

The NPCLI Engine is a powerful, feature rich CLI engine that provides a host of features to facilitate command development. NPCLI supports standard CLI features like tab completion, context sensitive help, case insensitive commands, Cisco-like command modes and standard hotkeys for command line editing. NPCLI operates as an integral part of the NP BloX framework, but functions equally well as a standalone utility. An easy-to-use IDE allows the user to define commands in a simple textual format (Command Definition Language), enter backend instrumentation for the command and its parameters or enter help information.

Feature

Advantage

Init and exit callbacks: Supports definition of init /exit callback for any command or command node, which can function as an initializer or as entry/exit checks.

Useful for implementing commands which need to perform context checks before allowing the user to enter a particular command node.

Parameter Validation: Extensive support for standard networking data types like ipaddress and macaddress in the CLI.

Engine takes care of input validation and range checks without any custom coding. Parameter validation also extends to support any user-defined data types defined in the PDP.

Dynamic Enumerations: Allows the developer to associate a dynamically generated list of allowed keywords to a parameter. End user can invoke tab completion and context sensitive help on this list.

Useful for constructing commands where set of valid inputs are dynamic and can only be determined at runtime.

Password: Supports password type parameters.

Password type parameters can be entered in the command line without being echoed.

Prompt Configuration: Comprehensive options to customize command prompts.

Prompt configuration can be specified through the IDE and can be defined to contain dynamic as well as static strings.

Output Formatting: Generates the formatted output data for typical presentation patterns, page display, table display, chained display, etc. The output patterns are specifiable in simple notation in the XML definitions or through the IDE.

Manual coding for output formatting for every command is eliminated. Saves on command development time, as well as code size, as a generic handler can take care of all output formatting.

Alias Definition: Supports definition of aliases and partial aliases

Built-in commands to define, save and load aliases makes it easy to define and save command aliases.

100% code generation: The CLI compiler generates 100% functional code to support new command tree, as well as interfacing with the backend through the NPSEAL layer. Create, delete, set, and get are translated to the generic DAL request formats, sent to the appropriate backend modules, and responses are parsed and displayed to complete the command.

Fully functional command tree with command handlers ready with no custom coding. Reduced development and testing cycles, as custom coding is eliminated at the CLI engine level.

Highlights

- Small Footprint
- Support Cisco like command definition
- Parameter level authentication
- Command History save and load features
- Exec/replay of commands
- Complete commands from any node
- Internationalized Help strings support
- Seamless integration with NPSEAL layer
- 100% Code Generation