# ConfD Recent New Features

- CLI Turbo Parser
- TLS info in CDB
- Swagger for RESTCONF

*John Lawitzke*
*ConfD Product Manager*
*2019-10-22*

# CLI Turbo Parser

# CLI Turbo Parser

- Added in 7.2
- A faster method of parsing (and inserting into CDB) Cisco style configuration
- Used for loading a saved configuration from a file
- Accessible from CLI, MAAPI, and confd_load

# CLI Turbo Parser from the CLI

- To use in the CLI, “devtools” must be enabled
  - “devtools” are CLI commands useful for the developer during the development cycle
  - e.g.:  timecmd, xpath, turboparse
- In Oper Mode:
  - “devtools true”
- In Config Mode:
  - “turboparse file.txt”
  - “commit” or ”show configuration” or “revert no-confirm” etc.

# CLI Turbo Parser from MAAPI

- Invoked using maapi_load_config
- Use option “MAAPI_CONFIG_TURBO_C”
- maapi_load_config(sock, tid, MAAPI_CONFIG_TURBO_C, file)

# CLI Turbo Parser (cont.)

- There is a new format option 't' which stands for turbo.
- “confd_load -l -F t file.txt”

# TLS Data Storage in CDB

# TLS Data Storage in CDB

- Added in 7.2
- Previously, the private key, the private key's passphrase, the public key certificate, and CA certificates could only be stored in the file system
- These can now be stored in CDB
- Modeled in tailf-tls.yang

# TLS Data Storage in CDB (cont.)

- Must be enabled in confd.conf:

```
<webui>
  <transport>
    <ssl>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>8889</port>
      <readFromDb>true</readFromDb>
    </ssl>
  </transport>
</webui>
```

# TLS Data Storage in CDB (cont.)

- The database is populated with TLS data by configuring the /tailf-tls:tls/private-key, /tailf-tls:tls/certificate, and, optionally, /tailf-tls/ca-certificates
- It is possible to use password protected private keys
  - The passphrase leaf in the private-key container needs to be set to the password of the encrypted private key
- Unencrypted private key data can be supplied in both PKCS#8 and PKCS#1 format, while encrypted private key data needs to be supplied in PKCS#1 format.
- The SHA256 fingerprints of the public key certificate and the CA certificates can be accessed as operational data. e.g. “show tls”
  - The fingerprint is shown as a hex string. The first octet identifies what hashing algorithm is used, 04 is SHA256, and the following octets is the actual fingerprint.

# Generating Swagger for RESTCONF

# Generating Swagger for RESTCONF

- Added in 6.7
- Swagger is a documentation language used to describe RESTful APIs.
- Swagger specifications are used to both document APIs as well as generate clients in a variety of languages.
- In addition to ConfD User Guide documentation, an application note is available:
  - https://info.tail-f.com/appnote-restconf-yang-swagger

# Generating Swagger for RESTCONF (cont.)

- YANG and Swagger are two different languages serving slightly different purposes.
  - YANG is a data modeling language used to model configuration data, state data, RPCS, etc.
  - Swagger is an API definition language that documents API resource structure as well as HTTP body content validation for applicable HTTP request methods.
- Translation from YANG to Swagger is not perfect
  - There are certain constructs and features in YANG that are not possible to capture completely in Swagger.
- The design of the translation is designed such that the resulting Swagger definitions are more restrictive than what is expressed in the YANG definitions.
  - This means that there are certain cases where a client can do more in the RESTCONF API than what the Swagger definition expresses.
  - There is also a set of well-known resources defined in the RESTCONF RFC 8040 that are not part of the generated Swagger specification, notably resources related to event streams.

# Generating Swagger for RESTCONF (cont.)

- Swagger definition file generation is done using “yanger” with the “-f swagger” option

- “yanger --help” to see all the options:
  - Swagger output specific options:
  - --swagger-host Add host to the Swagger output
  - --swagger-basepath Add basePath to the Swagger output
  - --swagger-version Add version url to the Swagger output.
    - NOTE: this will override any revision in the yang file
  - --swagger-tag-mode Set tag mode to group resources. Valid values are: methods, resources, all [default: all]
  - --swagger-terms Add termsOfService to the Swagger output
  - --swagger-contact-name Add contact name to the Swagger output
  - Plus many more…

# Generating Swagger for RESTCONF (cont.)

```
yanger -p . -t expand -f swagger example-jukebox.yang \
              --swagger-host 127.0.0.1:8008 \
              --swagger-basepath /restconf \
              --swagger-version "My swagger version 1.0.0.1" \
              --swagger-tag-mode all \
              --swagger-terms "http://my-terms.example.com" \
              --swagger-contact-name "my contact name" \
              --swagger-contact-url "http://my-contact-url.example.com" \
              --swagger-contact-email "my-contact-email@example.com" \
              --swagger-license-name "my license name" \
              --swagger-license-url "http://my-license-url.example.com" \
              --swagger-top-resource all \
              --swagger-omit-query-params false \
              --swagger-omit-body-params false \
              --swagger-omit-form-params false \
              --swagger-omit-header-params false \
              --swagger-omit-path-params false \
              --swagger-omit-standard-statuses false \
              --swagger-methods "post, get, patch, put, delete, head, options"
```

# Generating Swagger for RESTCONF (cont.)

- The “RESTCONF, YANG, and Swagger” application note is available:
  - https://info.tail-f.com/appnote-restconf-yang-swagger

- In addition to showing how to generate Swagger definition files, the application note also shows how they can be used with the “swagger-editor” OSS Swagger tool


**Thank you for listening**

www.tail-f.com