Use Case Exploration

**Case 1: As a CM online advisor**

As a broadband connecting device, the CM needs to register onto CMTS by a series of steps to communicate messages with each other. The settings from CM/CMTS or environment (diplexer/splitter/attenuator) might introduce problems and the CM fails to go online. Usually the CM console logs have enough information showing the issues. The OSS module helps diagnose and gives some suggestions after consuming the console logs. The OSS does this by referring to a domain knowledgebase after parsing the templates and its parameters for each log. Other module engineers may try this tool before contacting the DOCSIS people.

**Case 2: As an auxiliary checker for automatic test**

It depends on what nightly build test on the CM. It specially suits tests of DOCSIS related functionalities, like various ATP tests. Although there are test reports always, the CM console logs anomaly detection results still can provide help for developers if some failures happen.

To have DeepLog exhibit its power, we need to get some complete logs from the STE team for training and updating the model.

**Case 3: As a debug helper**

For issue producing, we might use hours/days for continuously logging. As a human engineer, I always check the CM logs with any debugging off to get an overview. If needed, turn on some related debugging for further logging.

For DeepLog, CM logs with debugging off is the first selection because our current training model probably can detect the anomalies. For debugging, we need to select what kind of commands to turn on.

Debugging commands:

“CM/DocsisCtl> log\_messages” and “CM/CmHal> log” should be off always because they turn on message content instead of more detailed messages back and forth. The machine cannot understand them.

“CM> zone” and “CM/power> debug” include many sub-options and some of them provide more detailed messages in context. We can turn them on to collect logs for training.

**Case 4: As a CSP case finder**

After parsing the log file, we get one or more CSP numbers that might be related with the anomalies we detected by DeepLog.

First, we use DeepLog to detect the anomalies in the attached logs in one CSP, and then we get a mapping data sample (anomaly set -> CSP number). We do the same thing on all available CSPs, and we get a mapping dataset between anomalies and CSP numbers.

Second, we use the mapping dataset to train another model (called CSP model). This new model learns the connections between anomalies and CSP numbers.

In prediction, a log file containing anomalies will get into DeepLog model firstly and then the detected anomalies enter into CSP model to find the possible CSP numbers.

*Prerequisites* that determine if the idea above is feasible in practice:

1) The DeepLog can handle the customization logs and older reference logs like BFC6.1.2. We can overcome it by using more logs from old versions for training and by ignoring log sequences that contain unknown log templates.

2) The anomalies in the log files attached in each CSP have one to one relationship to CSP numbers.

3) Enough number of CSPs and the quality of log files in each CSP.

**Case 5: As a security detector**

We suppose doing this on the RG side, which has firewall/ids logs, etc. How to get enough training data is an issue.

**Case 6: As a messed log corrector**

The template library has complete log templates. We can run the Drain module on logs that include messed ones, and then find them (including its count) in the huge log set via their ids. We can even automatically correct them for most cases in some way.

Open-minded for utilizing this feature in other areas.

**Case 7: As a real-time monitor**

Monitoring the device status in real-time by prediction. CPEs ssh CM/RG and the stdout logs redirect (PIPE) to Loganalyzer for processing. It is a challenge for CM logs, which have multi-line format and occasional messed stuff by multi-thread.

**About RG log adaption:**

The console logs, which come from different modules, like system booting, wifi, moca, etc.

The log files the system produces, like firewall logs.

We need to consider how to differentiate logs from different functional modules on RG.