NERC
NORTH AMERICAN ELECTRIC RELIABILITY CORPORATION

Lesson Learned

Loss of Automatic Generation Control During Routine Update

Primary Interest Groups
Transmission Operators (TOPs)
Generator Operators (GOPs)
Balancing Authorities (BAs)
Reliability Coordinators (RCs)

Problem Statement
During a weekly automatic generation control (AGC) software update, a critical AGC task aborted at one of two fully redundant control centers, and the same critical task aborted at the other control center four minutes later. As a result, generation schedules could not be set, and area control error (ACE) could not be automatically calculated until the issue was resolved. The cause was a modified line of code.

Details
On December 10, 2019, a typical weekly AGC system build was being deployed. AGC was controlled from Control Center #1 with a completely redundant hot-standby system at Control Center #2. The standard process of work is to complete the deployment on the Control Center #2 AGC system and have the dispatcher transfer control to Control Center #2 AGC so the deployment can be completed on Control Center #1 AGC once that is complete. After the completion of that work, the dispatcher switches back to control using Control Center #1 AGC and remains in that state.

When the Control Center #2 AGC deployment was complete, the generation dispatcher was informed by software support staff that they could transfer control to the Control Center #2 AGC system so the software update could be completed on the Control Center #1 AGC system. Approximately 1 hour and 15 minutes after switching control to Control Center #2, a critical AGC task aborted and critical control functionality was lost. The software support staff immediately became aware of the problem and, after being unable to restart the task, advised the dispatcher to switch control back to Control Center #1 AGC since its software deployment had been completed. The dispatcher switched control to Control Center #1 AGC, and the same critical task aborted on the Control Center #1 AGC. At this point, the dispatcher contacted the generation plants under AGC control to confirm plant schedules and to instruct them to remain at their current generation levels. The RC was notified.

It was discovered that there was a change made to the primary inadvertent interchange (PII) alarm text during this AGC update that caused the failure. Prior to the update, when the PII value exceeded +/- 999 MW, the value in the alarm text defaulted to "****," preventing dispatchers from having an accurate indication of the amount of PII change. This alarm comes in when the PII value has a change greater than +/-200MW. During this deployment, the two MW value fields in the PII alarm text were modified from i4 to i5 (4 digit integers to 5 digit integers) to allow for an additional digit. While the dispatcher was controlling from the Control Center #2 AGC, the PII value went from +71 to +338 (267 MW change), causing the alarm. The original alarm text array had 79 characters which almost hit the max character limit of 80, and the

change from i4 to i5 resulted in the alarm text increasing by 2 characters to 81 characters in length. Writing an 81 character text string into an 80 character fixed length array resulted in a run-time abort of the task. When the dispatcher transferred control to the Control Center #1 AGC system, the Control Center #1 system began processing alarms and issued the same PII alarm, resulting in the same task to abort and loss of critical functionality on the Control Center #1 AGC. To immediately remedy the problem, code was written to tell the critical task to ignore the PII alarm. The dispatcher was able to go back on control on Control Center #1 AGC. Later that day, a more permanent remedy was implemented, and the alarm text statement was shortened to 72 characters while still allowing 5 digits in the MW value fields.

PRIOR TO BUILD: LARGE CHANGE IN PII FROM XXXX TO XXXX MW. SELECT ACCEPT TO ALLOW PII TO UPDATE. (79 characters)

AFTER BUILD: LARGE CHANGE IN PII FROM XXXXX TO XXXXX MW. SELECT ACCEPT TO ALLOW PII TO UPDATE. (81 characters)

CURRENT: LARGE CHANGE IN PII FROM XXXXX TO XXXXX MW. SELECT ACCEPT TO UPDATE PII. (72 characters)

Corrective Actions

A script was written to parse alarm format strings to see the theoretical limit for string max length to determine if any other alarms have text lengths that exceed the local array character limit.

The process was modified by the group that programs and tests AGC modifications. Prior to an update occurring, there will be fields in the system used to track control center work to prove that the code was tested. The coder name, tester name, and test evidence that proves the successful test are required.

Lesson Learned

- No matter how small the change is, validate changes in a test environment first. This requires using a thorough test script for the changes.
- A completed software testing process is critical to guarantee that products meet their requirements. In general, the process has four components:
  - **Test Scope:** to define test environment requirements and setup, features/functions that need to be tested, documentations to refer and produce as output, approval workflows, etc.
  - **Test Design:** to design test cases that are necessary to validate the system/functions/features being built, against its requirements. Typically, regression testing and incremental testing are necessary
  - **Test Execution:** to execute tests in many different ways
  - **Test Closure:** to consider the exit criteria for signaling completion of the test cycle and readiness for a release

- After implementing a code change at a control center, allow it to have enough time to determine that no adverse condition has been introduced prior putting that center in control (operate parallel) and prior to attempting to implement the change on the other control center.
- Buffer overflows must be avoided by maintaining a high degree of correctness in code that performs buffer management. It has also long been recommended to avoid standard library functions that are not bounds checked. Well-written and tested abstract data type libraries that centralize and automatically perform buffer management, including bounds checking, can reduce the occurrence and impact of buffer overflows.

NERC's goal with publishing lessons learned is to provide industry with technical and understandable information that assists them with maintaining the reliability of the bulk power system. NERC is asking entities who have taken action on this lesson learned to respond to the short survey provided in the link below.

Click here for: Lesson Learned Comment Form

For more Information please contact:

NERC – Lessons Learned (via email)	WECC Event Analysis
Source of Lesson Learned:	WECC
Lesson Learned #:	20200403
Date Published:	April 14, 2020
Category:	Communications

This document is designed to convey lessons learned from NERC's various activities. It is not intended to establish new requirements under NERC's Reliability Standards or to modify the requirements in any existing Reliability Standards. Compliance will continue to be determined based on language in the NERC Reliability Standards as they may be amended from time to time. Implementation of this lesson learned is not a substitute for compliance with requirements in NERC's Reliability Standards.