NIF DEVICE HEALTH MONITORING

R. Fleming, J. Fisher, E. Stout and C. Estes

Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, California, USA

Abstract

The Integrated Computer Control System (ICCS) at the National Ignition Facility (NIF) uses Front-End Processors (FEPs) controlling over 60,000 devices. Often device faults are not discovered until a device is needed during a shot, creating run-time errors that delay the laser shot. This paper discusses a new ICCS framework feature for FEPs to monitor devices and report their overall health, allowing for earlier identification of problem devices. Each FEP has different devices with their own definitions of healthy. The ICCS software uses an object oriented approach using polymorphism so FEPs can determine their health status and report it in a consistent way. This generic approach provides consistent GUI indication and the display of detailed information of device problems. It allows for operators to be informed quickly of faults and provides them with the information necessary to pinpoint and resolve issues. When fully implemented by device software, operators will know before starting a shot if the control system is ready, thereby reducing time and material lost due to a failure and improving overall control system reliability and availability.

INTRODUCTION

A control system as large as ICCS for NIF controls a wide variety of devices that use many different means of communication. As many of these devices are not communicated with until they are needed, problems that may have happened sometime in the past are not recognized until the device is needed. This late detection of problems can bring the entire facility to a stop where earlier detection would have allowed time to determine a solution at reduced impact and cost.

Currently, there is a variety of GUI elements used for various device types to display health. Often a user would have to visit several different GUI screens to get a complete view of system health. We recognized the need for a more consistent and higher-level mechanism to report device failures to users.

The Device Health Monitoring framework is a set of features that improve the detection and reporting of device failures.

The first feature added monitoring of remote embedded networked systems used by FEPs. A script was developed that periodically pings the controllers to determine if they are turned on and available on the network. A new GUI page to show the status of these controllers was added that extends the existing centralized system status display.

The next feature, which is the focus of this paper, provides the framework for ICCS FEP processes to report and display failures with devices they control. It extends the existing centralized process status display with a new 'unhealthy' state. Each device class is responsible for implementing its own mechanism for detecting health.

With this new framework mechanism in place, FEP development can now begin to enhance the detection and reporting of problems with devices in the FEPs.

MONITORING FRAMEWORK

The Device Health Monitoring framework has two main components. The first is a singleton FEP health agent that is instantiated in each FEP process and acts as a device state aggregator and reporter (Figure 1). The second component is the extension of the System Manager GUI to receive and display the new FEP process 'unhealthy' state (Figure 2).

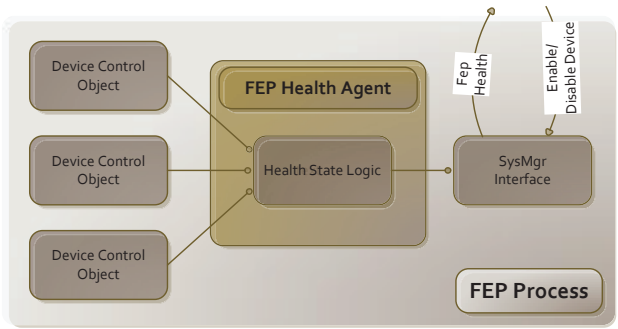

Figure 1: FEP Health Monitor Agent.

At FEP start up, the health agent subscribes to each device's health state change event after all the devices have been configured. Device control objects derive from a common base class that has the ability to publish state changes.

When a device health state changes the FEP health agent invokes logic to determine the overall FEP process health. If there is a change in overall FEP health, the new state is published to the System Manager for display on the GUI.

A new process state, 'Unhealthy,' has been added to the list of possible process states. It indicates that a process is running and communicating with the rest of the ICCS system, but has experienced a problem with one or more devices that it controls. When all the device problems are resolved, the process state will go back to the normal running state.

ISBN 978-3-95450-139-7

The default FEP health state algorithm simply declares the process as 'unhealthy' if any device is not in a healthy state. This algorithm is easily overridden by FEP developers if there are situations where device health differs from the default logic.

The device state change event that the FEP health agent receives contains the device state and text information that is intended to aid operators in identifying and resolving the issue. The text is generated at run-time, allowing for a high degree of detail that should reduce the time to diagnose device problems.

In order to detect device issues as they occur, device classes will require added logic to monitor their communications link and device hardware status. This will be an on-going process to add active device monitoring to existing device classes.

GUI PRESENTATION

The second component of the Device Health Monitoring framework is the presentation of FEP health on the GUI. The primary goal of this feature is to provide operators with timely notification of device problems and information to help them quickly diagnose and fix issues.

The existing ICCS System Manager GUI has a centralized way to show process states using color coded icons. The FEP Health Monitoring framework added the new 'unhealthy' state which is displayed using a new off-red color (Figure 2).

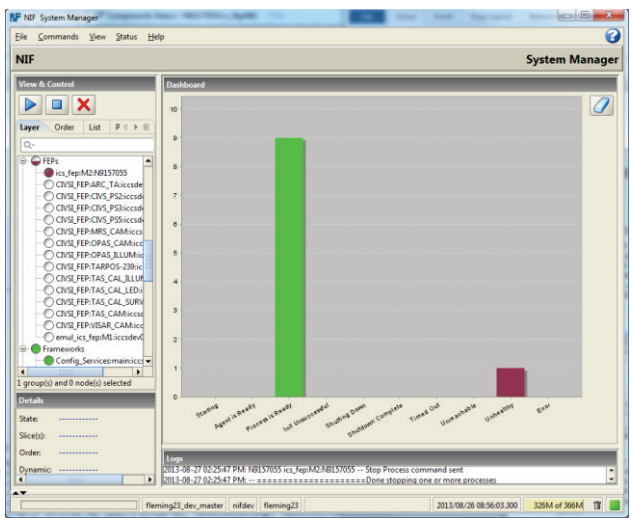

Figure 2: Device Information on the GUI.

ICCS operators are already accustomed to looking at the System Manager GUI for process state information. Right-clicking on a process in the tree list on the left displays a dialog containing detailed information about devices in that process (Figure 3).

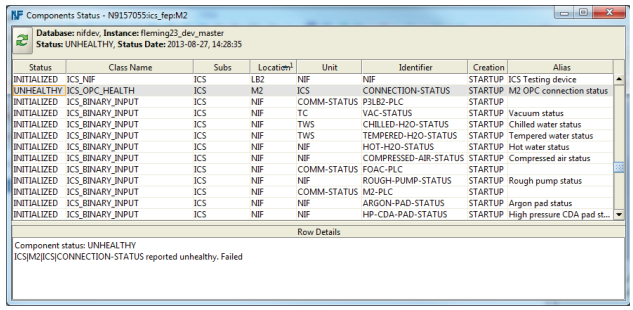

Figure 3: Detailed Device Fault Information.

Each device control object in the FEP is displayed in the dialog. By selecting an object in the display, the text details that were published in the state change event are displayed at the bottom. This way operators can quickly identify processes that are unhealthy, determine which devices are experiencing problems and get details on what is wrong.

The System Manager GUI may not always be visible on screen. To ensure that operators are informed quickly when a FEP goes unhealthy, a dialog containing information about unhealthy processes is displayed and forced to be on top of other screens until dismissed by the user. This alerts operators that a problem has occurred and needs to resolved, see Figure 4.

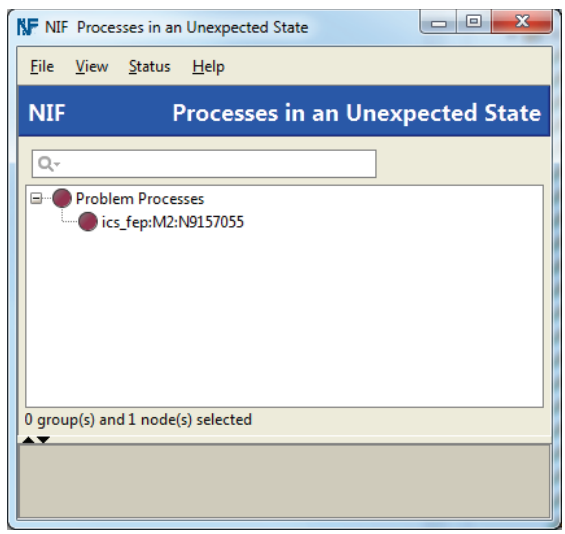

Figure 4: Problem Process Pop-up Display.

ADOPTION BY FEP DEVELOPERS

Providing the framework for device health monitoring is the first step. With this feature now in place, FEP developers are beginning to add problem detection to their device control objects and reporting changes of state to the Health Monitoring agent.

The first FEP to use this feature uses OPC to get hardware status from two PLCs. When there are network problems or if a PLC goes offline, the OPC link is lost and the data the FEP provides to the higher layers becomes stale. This FEP now provides the state of the

ISBN 978-3-95450-139-7

OPC connection to the Health Monitor Agent. If the OPC connection state goes to a faulted state, information provided by the OPC client software is provided through the agent and displayed on the GUI. This allows operators to diagnose the problem and determine if it appears to be network related or a problem with the PLC.

The next group of devices to be monitored will be devices that control remote hardware. These devices communicate via Ethernet, GPIB, RS-232 or other mechanisms. Monitoring logic will be added to the device classes to periodically communicate with the remote devices to ascertain their ability to operate properly.

FUTURE CAPABILITIES

Continuing the effort to detect and report system health problems may result in projects that add monitoring of other system resources, such as database health and processor and network resources, such as CPU, memory and network bandwidth usage.

ACKNOWLEDGEMENT

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
LLNL-CONF-644475.

ISBN 978-3-95450-139-7