International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 7, Issue 9 (July2013), PP. 16-21

Implementation of Sensor Data Record Generation Automation Tool for Baseboard Management Controller

Robert Pebam[1], Smitha G.R.[2]
[1] M.Tech, Software Engineering,
[2] Assistant Professor
[1, 2] Department of Information Science, R.V.College of Engineering, Bangalore

Abstract:- Intelligent Platform Management Interface (IPMI) defines common interfaces to server's hardware that used to monitor a server's physical health such as fans, voltage, temperature, power supplies, power status, and chassis. The server's physical health gets report to Baseboard Management Controller (BMC) by different sensors that built into the system. All the sensors information is provided by Sensor Data Records (SDRs). SDRs provide the data records that contain information about the type and number of sensors in the platform, sensor threshold support, chassis intrusion, power supply monitoring, fan speeds, fan presence, board voltages, board temperatures, bus errors, memory errors. SDR are created manually by the users using the information provided by Configuration sheet. In this paper the need to automate the generation of SDR is discussed in two ways: the SDR mapping to the spreadsheets and provided the details without spreadsheets. This paper proposed a solution by implementing of the SDR creation automation tool. We proposed a prototype including an implementation of IPMI's Sensor Data Records format module that has been applied to different configuration sheet provided by different engineers. Any changes to the configuration sheet values get updated by using the SDR generation tool.

Keywords:- Baseboard Management controller, Field Replacement U nit, IPMI, sensors

I. INTRODUCTION

With rapid increase in business, network application, the load on server system continues to increase. Management and monitoring the server system in a datacentre becomes more and more difficult as the load and requirement increases. Server systems are often composed of many different brands, different architecture, different product composition, different products interface and different working platforms, to bring a unified centralized management more difficult.[1] In order to meet the needs of remote monitoring management, in 1997,Intel,HP,Dell,NEC four companies started to develop management standards, in 1998 made a preliminary Intelligent Platform Management Interface specification, 2004 IPMI2.0 version available, additional encryption, authentication, VLAN, Serial Over. LAN (SOL) and other functions, and backward compatible with 1.0 and 1.5 version of the specification, thus providing greater security, independence and versatility, and offer-of-band management capabilities, present, this specification has been more than 180 manufacturers support.

IPMI defines a standardized, message-based intelligent management platform, and to describe the platform standardization of records management equipment, has formed a server and other systems, including hardware, management standards, and has good cross-platform portability and features. [2] In addition, IPMI can be in different states of the system to operate, even in the server is down or the case of power failure, IPMI monitoring can still be carried out normally. Therefore, through the IPMI specification can be achieved with a cross-platform configuration, monitoring, management, and alarm functions such as remote management platform.

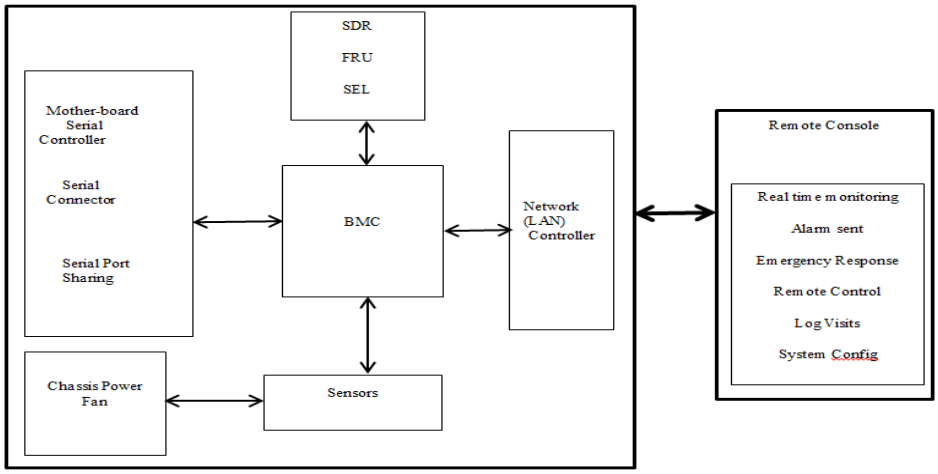

Fig 1 BMC Architecture

As shown in fig 1, IPMI system architecture, Baseboard Management Controller (BMC) is the heart of IPMI, which is independent of the server, has self-power microprocessors. [3].The servers have various sensors that built upon it in order control different hardware like fan, power, and voltages. The various sensors collect the information of fan, power, voltage and reports to BMC through IPMB bus.

In order to manage and monitor the health of the system IPMI defines a set of common interfaces. The FRUSDR contains two components-the FRU (Field Replacement Unit) and the SDR (Sensor Data Records). [4] FRUSDR is a non-volatile storage that is accessible through the BMC using standard IPMI commands. FRU is a components, part or assembly that can be quickly and easily removed from a system and replaced by the user or technician without having to send the entire system to repair facility. The IPMI specification includes support for storing and accessing multiple sets of non-volatile FRU information for different modules in the system. The FRU data includes information such as serial number, part number, model, assets tag. Sensor Data Records (SDR) describes the sensor configuration on the motherboard to the system software. The types of information that SDRs can store configuration on are: CPU sensors, chassis intrusion, power supply monitoring, fan speeds, fan presence, board voltages, board temperatures, bus errors, memory errors, and even possibly ASF progress codes. [5] It also contains information about the event generation capabilities, type and number of sensors in the platform, sensor threshold support, and information on what types of readings the sensor provides. The primary purpose of Sensor Data Records is to describe the sensor configuration of the platform management system. Sensor Data Records describe sensors but it does not instantiate sensors. Sensor Data Records also include records describing the number and type of devices that are connected to the system's IPMB, records that describe the location and type of FRU Devices (devices that contain field replaceable unit information). SDR of the particular sensor need to change whenever the configuration of the sensor required to makes change.

This paper describes the implementation of SDR generation automation tool which is currently generated by the engineers manually. The automation tool also provides other features to modify, view and the delete the existing SDR.

II. THE NEED FOR SENSOR DATA RECORDS GENERATION

2.1 Overview of the traditional SDR generation and difficulties

Traditional SDR generation needs an editor to create a SDR for different sensors that build upon the system. IPMI have twelve different types of SDRs. All different SDRs have different format, length, size that need to be filled by different hexadecimal value. Depending upon the filled hexadecimal values the sensors are set to it respective threshold values. This hexadecimal values are provided by the hardware engineers and maintained in an excel files for different platforms. The system administrator needs to calculate the values from the excel sheets and create the SDR manually. The sensors are identified by their respective sensor numbers. Whenever the system administration needs to modified the threshold value or disable or enable the sensors, the SDR needs to make changes. When the values in the spread sheets get updated for different configuration, then also the SDR needs make changes. The traditional way of generating SDR takes a long process and time consuming process. Changes make to the SDR also takes a long process. The enable or disable sensors number can be obtained by using impitool through the EFI shell which gives only sensor number. By using the obtained sensor number, the system administrator has to analyse the whole SDR of the particular system and find out the details of sensors.

2.2 Configuration files and difficulties

The file describing sensors contains details configuration of the sensors along with the various threshold. These details are entered by the engineers manually using editors. The SDR needs to maintain the syntax, format, and spacing. The various fields of the SDR are calculated form the spreadsheets and entered manually by the users. It's a hard task to maintain by human and it may leads to error prone while handling huge amount of data. Manually configure the data take lot of time to analyze, detect and fix all errors.

III. WORKFLOW AND MODELLING FRAMEWORK

The tool consists of two main components. One component handles the task of bridging the SDR file to the user while providing a number of services for modifying, generating, viewing interface. The other components provide a model-driven approach to defining sensor data record format define by IPMI, mapping to configuration sheet and creates the SDR files. The tool required the sensor number to perform all the necessary action that the user need. The IPMI spec provides the SDR format and it is a must to adhere the format set by the specs.

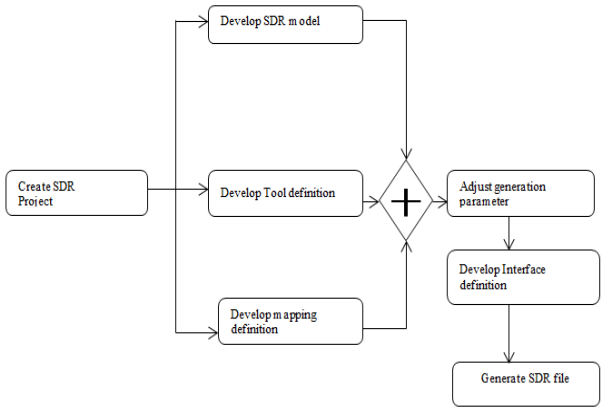

Fig 2 Modelling framework

The fig 2 illustrates the main components and model used during SDR generation. SDR model is the sensor data format defined by IPMI and implemented according to IPMI spec SDR format. This model contains the constraints set by IPMI so that the SDR value can get fit to this model following the IPMI format. The general Sensor Data Record format consists of three major components, the Record Header, Record Key fields, and the Record Body. The Tool definition defined the set of operation that can be done by the tool. The mapping definition provides the mapping of SDR values to configuration sheets and extraction of SDR value from the available Configuration sheets. The adjustment generation would provide the syntax in generating the SDR files. All the operations create, view, modify would done by the user through the graphical interface provided. The tool provides the GUI for interacting with the user. GUI provided the options for creating, deleting, viewing and modifying the SDR. Except for creating a new SDR, the location for SDR needs to be provided while starting the process of deleting, viewing and modifying. Once the location of SDR has provided, the tool access the SDR from the particular location and provided to the users.

3.1 Sensor Data Model:

The general Sensor Data Record format consists of three major components, the Record Header, Record Key fields, and the Record Body.

Record Header

Is the same for all records, consisting of:

Record ID**:** A value that's used for accessing Sensor Data Records

SDR Version**:** The version number of the SDR specification. Used in conjunction with Record Type

Record Type: A number representing the type of the record

Record Length: Number of bytes of data following the Record Length field.

Record Key Fields

The Record 'Key' Fields are a set of fields that together are unique amongst instances of a given record type. For example, for 'sensor' records, these fields specify the location (e.g. slave address, LUN, and Bus ID) and sensor number of the sensor. The Record Key bytes shall be contiguous and follow the Record Header. The number of bytes that make up the Record Key field may vary according to record type.

Record Body

The remaining Record Type specific information for the particular sensor data record.

The details of the format are provided in the IPMI specification. This specification makes the Sensor data model.

As shown in fig 3, the tool processed the inputs provided by the user and the input extracted from the available XLS Configuration sheet. The accepted inputs are filled in the SDRs. The IPMI SDRs are then updated. Once it gets updated successfully, the different sensors that reports to the BMC get initialized and get configured to enable/disable the sensors. The pre-existing SDR can get modified when the spread sheets get updated by providing the sensor number and location. SDRs can be analysed through the display option of the tool by providing the sensor number and location. The user input is either capture from the GUI interface or from the XLS sheet provided**.**

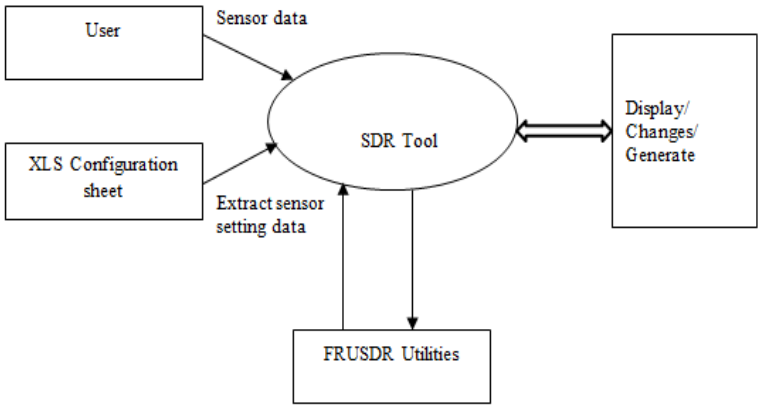

Fig 3 Workflow diagram of the automation tool

IV. IMPLEMENTATION OF AUTOMATION TOOL

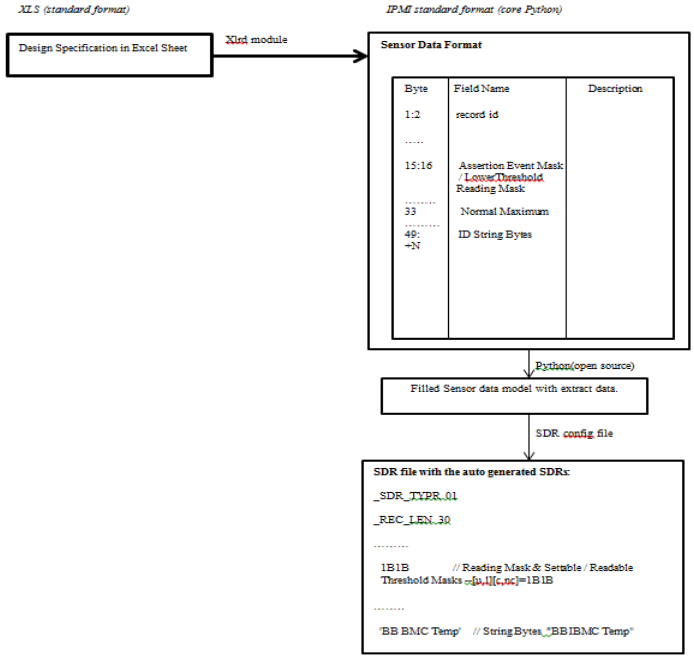

Fig 4 Design Layout of the automation tool

Fig 4 shows the design layout of the automation tool. Editors are the primary mechanism for users to create and modify resources (eg. files). Python provides basic editors such as text along with some more complex byte operation and graphical user interface such as wxPython. Xlrd package were used to extract the sensor data from the available spread sheets. The spread sheets provide the necessary threshold value for different sensors. The different SDR have different format of record. The value entered in the SDR fields must

satisfy the constraints of the format. The constraints are set by the core python. The user needs to provide only the sensor number. Depending upon the sensor number, the necessary corresponding field gets extract from the spread sheets. The configuration data extracted from the spread sheet get written in a SDR that maintained separately. SDR follow a format and syntax that take care by python script. The generated SDR gets update to the BMC by rebooting the system through the EFI shell. Fig 5 shows the tool GUI where user would interact with the application.

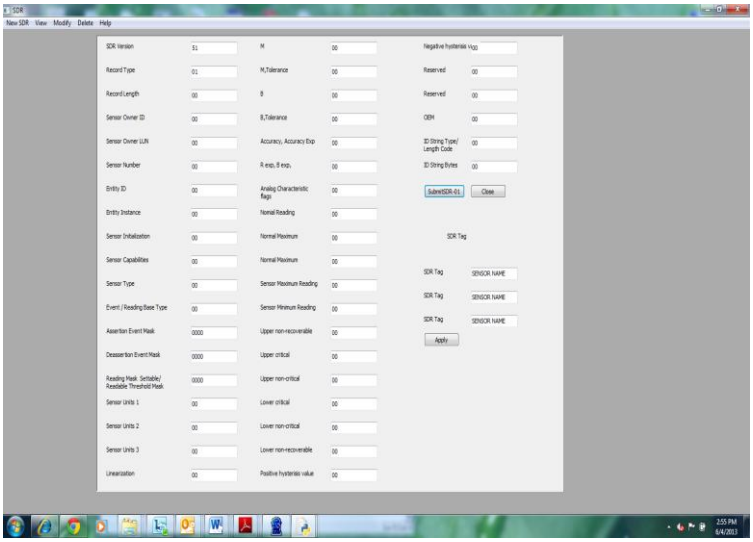

Fig 5. Automation tool GUI

System Engineers ensuring that the sensors are recognizable and the SDR values are usable by the related firmware. Hardware and System engineers update the SDR value calculation spread sheet for any newly enable or supported sensors. SDR creation tool was made as the result of the need to develop in an easier way to generate the SDR file, graphical interface which would save lot of time and error free sensor data records. Whenever sensor values need to be modified, the necessary sensor number can be obtained by using ipmi tool. The obtained sensor number applied to the SDR tool to extract the sensor details. The details get changes and the follows the updating process.

V. CONCLUSION

This paper explains approach for implementation of automation tool for SDR generation that is used by BMC. The tool is implemented by using Python scripts language for implementing the sensor data model. The widgets wxPython is used for the GUI to provide the interface to the users. The xlrd python module is used for extracting the data from the configuration sheets. The methodology is composed of:

- A front-end part, which provide the interface to the users and provide the entry point of sensor details for each sensors.
- A back end part which relies on sensor data model ie the IPMI sensor data format for the twelve sensors types provided in the IPMI specification.
- The automation tool relies on the configuration sheets for different sensors provided by the Thermal Engineers.

The advantages of implementing this tool are primarily reflected in generating the SDR which overcomes the manual tasks. The tool provides the data validation at the input to ensure only the clean data enters the repository. It ensures field level and record level (independence between the different fields in the same) data integrity. It has the ability to generate the SDR file in a single click just by providing the Sensor number. The previous SDR file can be regenerated and order the records for human readability. The SDR generated by this automation tool can be used for different platforms. It also indicated using this tool saved significant time and increase in the quality of the SDR over traditional process SDR creation.

REFERENCES

[1]. Zhilou Yu and Hua Ji , "Research of IPMI Management based on BMC SOC", *978-1-4244-5326-9/10/ ©2010 IEEE.*

[2]. Intelligent Platform Management Specification v1.5 Document Revision 1.1, 20 February 2002.

[3]. Chokchai Leangsuksun, Tirumala Rao, Anand Tikotekar,Stephen L.Scott, Richard libby,Jeffrey S.Vetter, Yung-Chin Fang,Hong Ong," IPMI-based Efficient Notification Framework for Large Scale

Cluster Computing", *IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06) 0-7695-2585-7/06 © 2006 IEEE*

[4]. Corey Minyard, Montavista,Software, "IPMI – A Gentle Introduction with Open IPMI", February 10, 2006

[5]. Richard Libby "Effective HPC Hardware Management and Failure Prediction Strategy Using IPMI".

[6]. Alexandre Chagoya-Garzon, Nicolas Poste, Frederic Rousseau "Semi-Automation of Configuration Files Generation for Heterogeneous Multi-Tile Systems*", 0730-3157/11 © 2011 IEEE DOI 10.1109/COMPSAC.2011.28*

[7]. Marko Krnjetin and Lana Misirlis," Graphical tool for generating linker configuration files in embedded systems", *978-1-4577-1500-6/11/ ©2011 IEEE*

[8]. Marko Krnjetin and Lana Misirlis," Graphical tool for generating linker configuration files in embedded systems", *978-1-4577-1500-6/11/ ©2011 IEEE*

[9]. Lars Nagel, "Autonomous Sensor Data Processing", 15. January 2006

[10]. Bei Yu and Hans Jargen Skovgaard," A Configuration tool to increase Product Competitiveness", *1094-7167/98 © 1998 IEEE, IEEE Intelligent System*

[11]. Mark Pilgrim, "Dive Into Python", 28 July 2002

[12]. Chris Withers, John Machin," Working with Excel files in Python", Euro Python 2009, Birmingham