





**Project Report**

**ON**

**STT\_GW\_NG**

**FOR**

**Siemens Technology and Services**

**BY**

**Prankul Devi**

**SAVITRIBAI PHULE PUNE UNIVERSITY**

**MASTER IN COMPUTER APPLICATION**

**MAHARASHTRA EDUCATION SOCIETY'S**

**INSTITUTE OF MANAGEMENT AND CAREER COURSES**

**(IMCC), PUNE-411038**

**2016-17**



Maharashtra Education Society's

## INSTITUTE OF MANAGEMENT AND CAREER COURSES (IMCC)

(Recognized by Savitribai Phule Pune University & Approved by  
AICTE)

131, Mayur Colony, Kothrud, Pune 411 038

Telefax . +91-20-25466271, 25463453 • E-mail: [directorimcc@vsni.net](mailto:directorimcc@vsni.net)

**R. V. H. INAMDAR** Director

☎: +91-20-25445067 (Director) Telefax: +91-20-25466271, 25463453 (Registrar)

T: +91-20-25440198 (TPO) Web Site: <[www.mesimcc.org](http://www.mesimcc.org)>

Ref. No : MCA-MCM/Project/328/2016-17

Date : 05/4/17

## CERTIFICATE

This is to certify that the Project Report entitled "*STT\_GW\_NG*" is prepared by *Prankul Devi* a student of *M.C.A.* Course for the Academic Year 2016-17 at M.E.Society's Institute of Management & Career Courses (IMCC), Pune - 411038. M.C.A Course is affiliated to Savitribai Phule Pune University.

To the best of our knowledge, this is original study done by the said student and important sources used by him/her have been duly acknowledged in this report.

The report is submitted in partial fulfillment of M.C.A Course for the Academic Year 2016-17 as per the rules & prescribed guidelines of Savitribai Phule Pune University.

*Dr. Santosh Deshpande*  
HOD, Computer Dept. IMCC

*Dr. Vikas Inamdar*  
Director, IMCC

*External Examiner*

*Internal Examiner*

**SIEMENS**

Siemens Technology and Services Pvt. Ltd.

Date: 31.03.2017

To,

Institute of Management and Career Courses,  
Pune (IMCC).

Dear Sir,

This is to certify that the following student has worked on the project on "STT\_GW\_NG" from 22-08-2016 to 17-03-2017 for which he had been guided by Mr. Shivkumar Phiske.

Name of Student

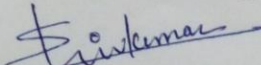
1. Prankul Devi

The student has successfully completed the project and also given a demonstration of this to the satisfaction of the guide.

We wish the student all the very best.

Sincerely,

For Siemens Technology and Services (P) Ltd

  
Authorized Signatory

Unrestricted  
Siemens Technology and Services Private Limited  
Management: Klaus Trescher

Amar Apex, 2<sup>nd</sup> & 3<sup>rd</sup> Floor,  
SL No-1/3A, Hissa No-1/3/A/3-14,  
Baner, Pune - 411 045

Tel.: +91 (020) 30929139  
Fax :  
Web: www.siemens.co.in/sts  
E-mail: contact.sts.in@siemens.com

Registered Office: 130, Pandurang Budhkar Marg, Worli, Mumbai 400018. Telephone +91 22 39677000. Fax +91 22 39677075.  
Corporate Identity Number: U99999MH1986PLC093854. Former name: Siemens Information Systems Ltd.  
Other Offices: Bengaluru, Chennai, Gurgaon, Kolkata, Noida, Pune.

## CERTIFICATE

This is to certify that **Prankul Devi** has completed the project work entitled "**STT\_GW\_NG**" under my guidance. The report is submitted in partial fulfillment of M.C.A. Course for the Academic Year 2016-2017 as per the rules & prescribed guidelines of Savitribai Phule Pune University.

His/Her work is found to be satisfactory and complete in all respects.

**Darshana Yadav**  
**(Internal Project Guide)**

## **Acknowledgement**

I wish to express deep sense of gratitude towards “Siemens Technology and Services” for providing me the opportunity to work with the company and to provide a wonderful and challenging environment for the development of “STT\_GW\_NG”.

I would like to thank Dr. V. H. Inamdar, Director, IMCC, Dr. Santosh Deshpande, HOD of computer department, IMCC,

Dr. Manasi Bhate, TPO of IMCC, Prof. Darshana Yadav, Mr. Abhay Damle for their valuable guidance and kind co-operation throughout the period of the work undertaken

Prankul Devi

MCA-III(Sem-VI)

## Index

### Table of Contents

<b>Chapter 1:Introduction .....</b>	
1.1 Company Profile .....	1
1.2 Existing System and Need for System.....	5
1.3 Scope of Work .....	10
1.4 Operating Environment.....	17
1.5 Detailed Description of Technology Used.....	19
<b>Chapter 2:Proposed System.....</b>	
2.1 Proposed System.....	22
2.2 Objective of System.....	25
2.3 User Requirements.....	26
<b>Chapter 3:Analysys and Design .....</b>	
3.1 Object Diagram .....	28
3.2 Class Diagram .....	29
3.3 Use Case Diagram.....	29
3.4 Activity Diagram.....	34
3.5 Sequence Diagram .....	38
3.6 Entity Relationship Diagram.....	47
3.7 Module Hierarchy Diagram .....	48
3.8 Component Diagram.....	49
3.9 Deployment Diagram.....	50
3.10 Module Specification .....	51
3.11 User Interface Design .....	54
3.12 Data Dictionary .....	63



3.13 Table Specification .....	68
3.14 Test Procedures and Implementation.....	73
<b>Chapter 4:User Manual.....</b>	
4.1 User Manual.....	77
4.2 Operations Manual.....	78
4.2 Program Specification.....	79
<b>Drawbacks and Limitations .....</b>	<b>82</b>
<b>Proposed Enhancements .....</b>	<b>83</b>
<b>Conclusions .....</b>	<b>84</b>
<b>Bibliography .....</b>	<b>85</b>
<b>ANNEXURE .....</b>	
ANNEXURE 1: User Interface Screens .....	
ANNEXURE 2: Reports with Data .....	
ANNEXURE 3: Sample Program Code .....	

# **1. Introduction**

## 1.1 Company Profile



Siemens India is a technology powerhouse that has stood for engineering excellence, innovation, quality and reliability. The company focuses on the areas of electrification, automation and digitalization. It is one of the leading producers of energy-efficient, resource-saving technologies; combined cycle turbines for power generation; and power transmission solutions. Siemens is a pioneer in infrastructure solutions and automation and software solutions for industry. The company is also a leading supplier of medical imaging equipment – such as computed tomography and magnetic resonance imaging systems – and a leader in laboratory diagnostics as well as clinical IT. At the end of September 2015, Siemens India

had around 16,000 employees (Number of employees is 16,000 including all group companies of Siemens in India). Siemens Limited, in which Siemens AG holds 75% of the capital, is the flagship listed company of Siemens AG in India.

The history of Siemens in India dates back to 1867, when Werner von Siemens personally supervised the laying of the first telegraph line between London and Calcutta. The first company office was founded in 1922. In 1957, Siemens was incorporated as a company under the Indian Companies Act.

Siemens has 22 factories located across India, 8 Centres of Competence and 11 R&D Centres and a nation-wide sales and service network. The factories manufacture Steam Turbines, Switchgear, Remote Monitoring Systems (RMS), Motors and Generators, Relays and Smart Grid Systems, Transformers, Railway Bogies and X-ray

Machines. These have been set up replicating global, best-in-class manufacturing systems and practices.

Globally, Siemens is a leader in providing technology solutions for setting up intelligent (smart), sustainable cities. With solutions for Smart Grid, Building Technologies, Mobility and Power Distribution, Siemens has successfully set up smart cities in Vienna and New York, and is already involved in the Restructured Accelerated Power Development and Reforms Programme (R-APDRP) Program of the Government of India for installing Smart Grid solutions in multiple cities in India.

Siemens has been a preferred technology solution provider for the Indian Railways for close to six decades. Over the years, it has further built capabilities in the areas of Metros, Integrated Mobility Platforms, Airport Links, Passenger Coaches, Rail Services and Maintenance,

Urban Traffic Control, Rail Signalling and other state-of-the-art Transportation Solutions.

Siemens India is supporting skill development through Corporate Citizenship initiatives such as ITI up gradation, faculty development and Technical Skills Development Centre. Siemens has also signed Memorandums of Understanding with the Government of Gujarat, and with Steel Authority of India Ltd. Additionally, Siemens has partnered with industrial bodies such as Automation Industry Association, Indian Machine Tool Manufacturers Association and CII to impart technical education.

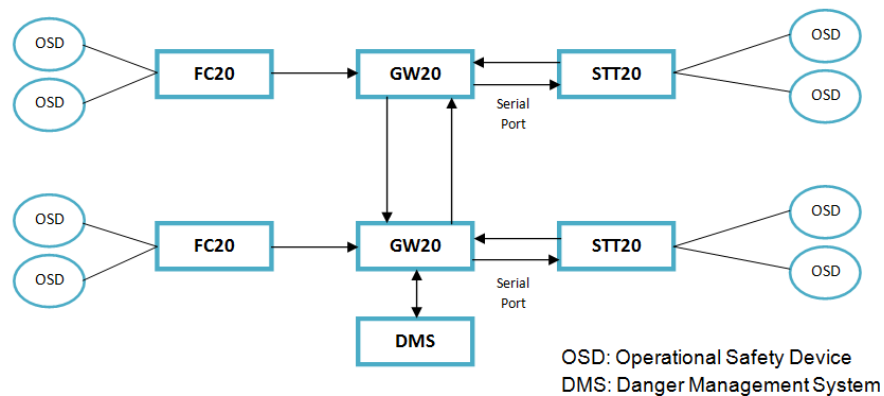
It has been a reliable partner for customers across verticals such as airports, automotive, buildings, cement, chemicals, healthcare, food & beverage, marine, minerals, oil & gas, renewable energy, petrochemicals, railways, textiles, urban infrastructure, pharmaceuticals, ports, power generation and power transmission & distribution.

## 1.2 Existing System and Need for System

The project is based on fire and safety system.

This system is divided into sub systems as:-

- Protection System (STT20)
- Detection System (FC20)
- Gateway (GW20)



In this system as per the client requirement we kept Protection and Detection system separate so that if one system fails another system will remain working. Currently Detection System, DMS and Gateways are communicating over BACNet protocol and Gateway is communicating with Protection System over Serial connection.

OSD are nothing but the detectors in case of FC20 and Protectors in case of STT20. To understand the flow and working of this system lets consider the simple scenario,

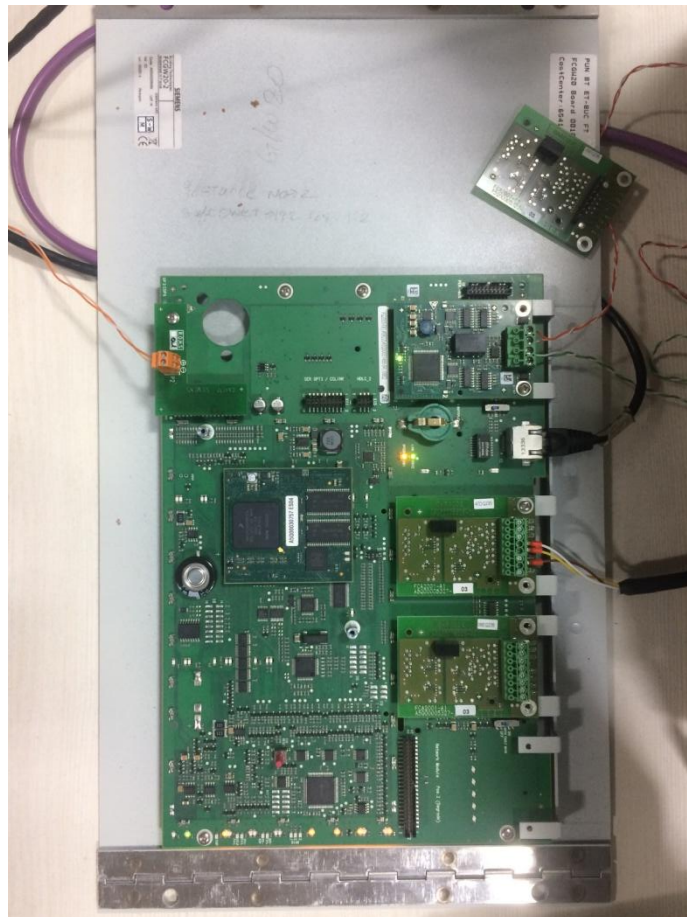
If there is fire or smoke then OSD will detect that incidence and FC20 will notify this incidence to Gateway, Gateway will process that packet and it will forward that packet to STT20 panel, STT20 panel will take action on that incidence like triggering alarm, sprinklers, opening vents, etc. so all these STT20 side devices also known as DAS.

Currently all these sub systems are communicating with each other over BACNet protocol. Whole system is also known as STT20 system. Current STT20 system has some characteristics that should be improved performance wise. To improve the communication between central STT20 panel and Physical Connection to DAS/OSD we have decided to implement a new system which is known as STT NG system. In this system we will use STT20 New Generation panels and we are using FCNet protocol instead of BACNet protocol.



To achieve this we are Re-using the existing STT20 architecture by developing an interface between STT20 and FS20.

### 1. GW20 Panel



## 2. FC20 Panel



## 3. Protection and Detection Devices



#### 4. STT20 Panel



### **1.3 Scope of Work**

I am currently working on Gateway part which is responsible for transmitting the data from Protection system to Detection system and vice a versa. Now only Gateway and DMS will communicate with each other over BACNet and Protection system work on FCNet protocol.

List of modules which I will cover:-

- [1] Detection Panel and Gateway Subscription
- [2] Protection Devices Subscription
- [3] Commanding Protection Devices (STT20 to STT20)
- [4] Monitoring Protection Devices (STT20 to STT20)
- [5] Fire Detected Situation
- [6] Monitor Detection Panel Command (STT20 to STT20)

**[1] Detection Panel and Gateway Subscription:-**

As FC20 and GW20 works on FCNet protocol they are known as FCNet devices. When system starts first we fetch the entire device and network related data from database (XML file).

One GW20 can connect with maximum 20 FC20 and 19 GW20 over the network.

If FC20 and GW20 device comes over the network or get disconnected from the network GW20 receives their connection status as connected or disconnected. If that device status is connected then we subscribe that device. In FCNet protocol any device (ex. Device A) want information of any kind of incidence happened on another device (ex. Device B) then **Device A** should subscribe **Device B** to get all the information from **Device B**.

**[2] Protection Devices Subscription:-**

One STT20 hardware is connected to One GW20 over serial port. In a same way there could be total 20 GW20-STT20 pairs available in network. One STT20 is connected with multiple protection devices (OSDs). Consider there is one GW20 (G1) is connected with one STT20 (S1) and another (G2) is connected with (S2). When (S1) wants the status of protection devices which comes under another (S2), then for STT20 (S1) GW20 (G1) subscribes another GW20 (G2) where (G2) will take Protection Device status from STT20 (S2) and send to GW20 (G1). As STT20 share their protection devices with other STT20 this technique is used so that the status of only subscribed device is shared with another STT20. GW20 can fetch data of Protection Device from database and after their presence GW20 mark it as present.

### **[3] Commanding Protection Devices (STT20 to STT20)**

If there is any incidence detected then to take action on that incidence STT20 (S1) starts their protection devices but sometimes STT20 (S1) needs the help of another STT20 (S2) to control the situation and for that it take help of another STT20 (s2). When any STT20 (s1) wants to give any kind of instruction to Protection Devices which are under another STT20 (s2) then STT20 (S1) can send command for specific Protection Device and in return another STT20 (S2) gives the Protection Device status to commanding STT20 (S1). As there are 20 STT20-GW20 pairs are present in network then one GW20 can send command to another 19 GW20 for Protection Device. GW20 can get Protection Device data from its database and after fetching the data from database GW20 sends command to that devices only.

#### **[4] Monitoring Protection Devices (STT20 to STT20)**

Protection Device maintains their current status like Protection Device is in waiting/ in command/ non-waiting/ fault etc State.

If STT20 sends command to Protection Device then Protection Device state changed to in command.

All these states are shared with STT20 panels so that STT20 can identify which Protection Device is in which state and according to that STT20 takes actions. In database some Protection devices are marked as Common Devices that means all these devices are shared with another STT20. Only state of these devices is shared and there can be maximum 1024 devices which are common.



### **[5] Fire Detected Situation**

Infrastructure is divided into zones like one building is divided into sectors and one sector can have multiple zones. To identifying the exact location of incidence this subdivision is made. Even Zones can have a type like Fire Zone, Smoke Zone etc. And in each zone there are multiple detectors and protection devices are mapped. If there is fire in any zone then detection panel notifies with the zone identification number and state of that zone and detection panel send this information to GW20. When this information receives at Gateway side it transfers it to protection panel means STT20 so that STT20 will active their protection device.

**[6] Monitor Detection Panel Command (STT20 to STT20)**

There are multiple STT20 panel present in network which are working independently as they are configured. But sometime when one STT20 panel receives alarm from FC20 and to take action and to resolve the issue it takes the help of another STT20 panel so that they both can work on same issue to resolve it. To notify this STT20 sends the intercom message to gateway and then gateway forward this message to STT20. In this command STT20 sends Detection Panel Identification Number and another GW20 Identification number to which this packet need to be send.

## **1.4 Operating Environment – Hardware and Software**

### 1.4.1 Software Requirement

- Operating System: Linux
- Programming Language: C

### 1.4.2 Hardware Requirement

- STT20 Board
- Serial Cable
- Ethernet Cable
- Ethernet to USB converter
- Power Supply (24 Volt)

1.4.3 Back end (Data Base)

- We are using XML file which contains

SR No.	Table Name	SR No.	Table Name
1	Project Information	6	Function Information
2	Building Information	7	OSD Information
3	Sector Information	8	MEA20_Sub Panel Information
4	Zone Information	9	MD20_Sub Panel Type Information
5	Detector Information	10	MC Information

## **1.5 Detail Description of Technology Used**

In this project we are using C language to develop our Gateway Application.

C is a general-purpose high level language that was originally developed by Dennis Ritchie for the UNIX operating system. It was first implemented on the Digital Equipment Corporation PDP-11 computer in 1972.

The UNIX operating system and virtually all UNIX applications are written in the C language. C has now become a widely used professional language for various reasons.

C language is one of the powerful languages.

Below are some of the features of C language.

- Reliability
- Portability
- Flexibility
- Interactivity

- Modularity
- Efficiency and Effectiveness
- Structured language
- It produces efficient programs.
- It can handle low-level activities.
- It can be compiled on a variety of computers.

**USES OF C PROGRAMMING LANGUAGE:**

The C programming language is used for developing system applications that forms a major portion of operating systems such as Windows, UNIX and Linux.

Below are some examples of C being used.

- Database systems
- Graphics packages
- Word processors
- Spreadsheets

- Operating system development
- Compilers and Assemblers
- Network drivers
- Interpreters

Some C language concepts which we are using:-

1. OS Signals
2. OS Timers
3. OS queues
4. Threads
5. Sockets
6. Mutex

## **2. Proposed System**



## 2.1 Proposed System

- In STT NG panels FC20 detection panels and different FC20-parts communicate to each other via FCNet.
- The communication between STT20 and STTNG will be FCNet based and two STT20 panels integrated can also communicate to each other via FCNet.
- Having FCNet as the only communication type between all panels will open possibility for more advanced network topologies.

### FCNet

- The term FCNet doesn't represent just a communication protocol, but it is kind of middleware concept in FS20 platform, which enables for the Elements to publish their state or be commanded in a network.

- FCNet is implemented in Java, realized in several components located in EPOS (Embedded Parallel Operating System) layer.

### **Element Identification**

- FC20 and STT20 use same technique to identify elements over network. Each element is identified by its HostID or Sequence Number.
- In STT20 panel network it uses MC20 number or Sequence Number for element identification where MC20 number is unique identification number. Special elements which belong to site are Locks, Lock identification is by Lock number (in range from 1 to 255)
- In FC20 panel network each panel has HostID/PanelID/SequenceNumber for unique identification of element. Usually HostID and PanelID are same but when multiple FC20 panels are connected to FS20 Gateway then PanelID get change

- Element identification is compatible between FC20 and STT20 but FC20 uses 4 byte sequence number for identification but at STT20 side it uses 2 bytes of sequence number so FC20 number cannot be used inside STT20 as it is.

## **2.2 Objectives of System**

In order to replace the current STT20 system, a new system, named STT NG, will be the output of a next R&D project.

To achieve this central panel for this STT NG system is the one used for FS20 system.

Based on this assumption, there is one solution which is possible to connect peripherals:

- Re-use the existing STT20 architecture by developing an interface board.

The goal of this feasibility study is to analyze the selected option prior to the STT NG project launch. The analysis shall cover from the output of the central panel, up to the physical connection to DAS.

## 2.3 User Requirements

### 1) Want Protection and Detection system separate:-

Client wants to keep both the protection and detection systems separate because if any one system out of them fails then another should not get affected.

### 2) Communication with DMS over BACNet:-

Monitor DMS over BACNet only and establish the connection and share element states with DMS.

### 3) DMS can command to STT20 panel.

As DMS is known as Danger Management System, DMS can also take actions any time to resolve the problem by sending commands to STT20 panel to activate protection devices.

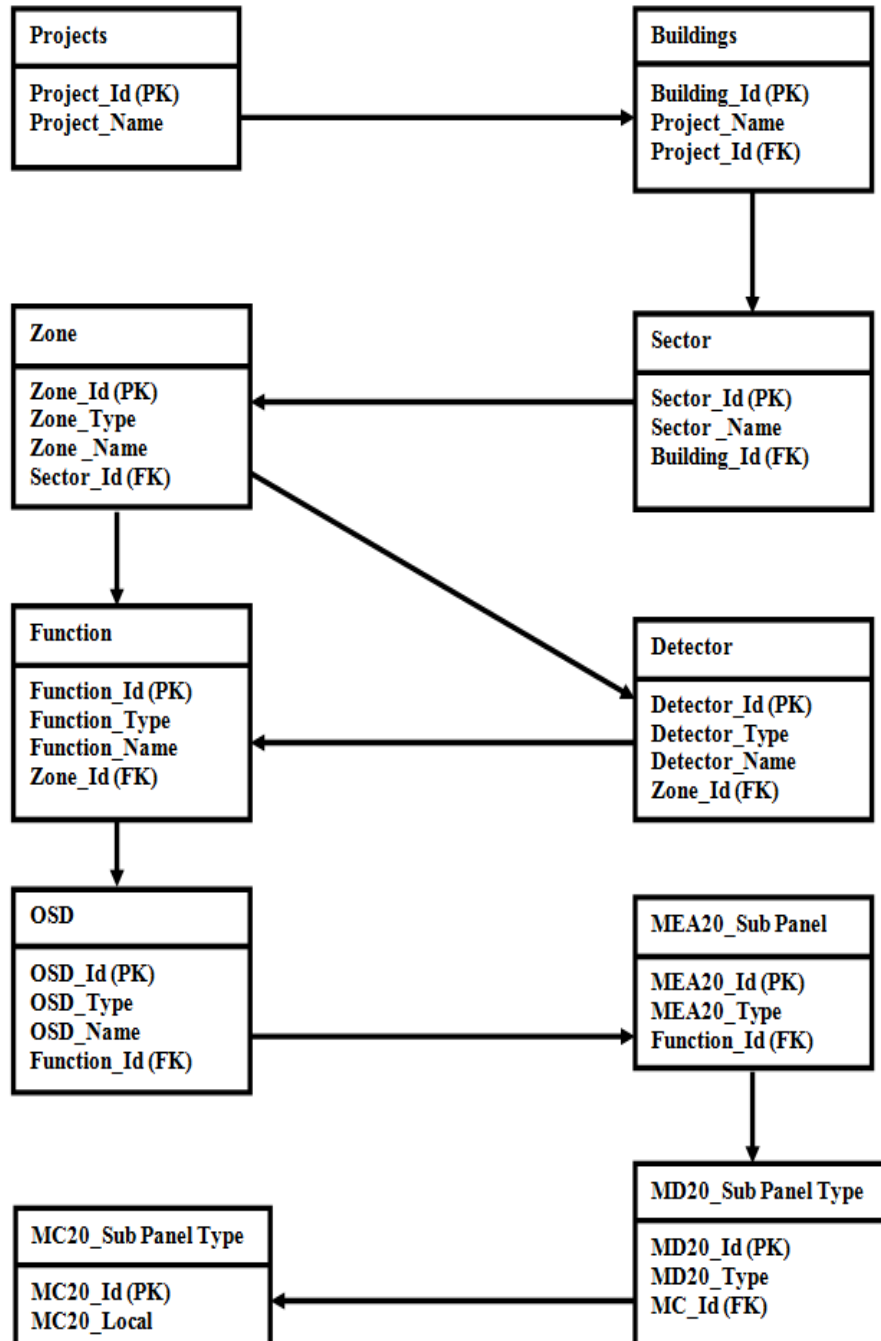
### 4) Alarm priority:-

When there is alarm at detection panel side that Alarm should be sent to STT20 panel from detection panel within 10 seconds.

- 5) GW20 and FC20 communication over FCNet: -  
Communication between STT20, Gateway and FC20 should be done over FCNet network.
- 6) GW20 should support maximum 20 FC20 panels and 19 GW20 panels.
- 7) GW20 should support Intercom, Command Protection Devices, Device subscription, and Protection Device State monitoring and Alarm packets.

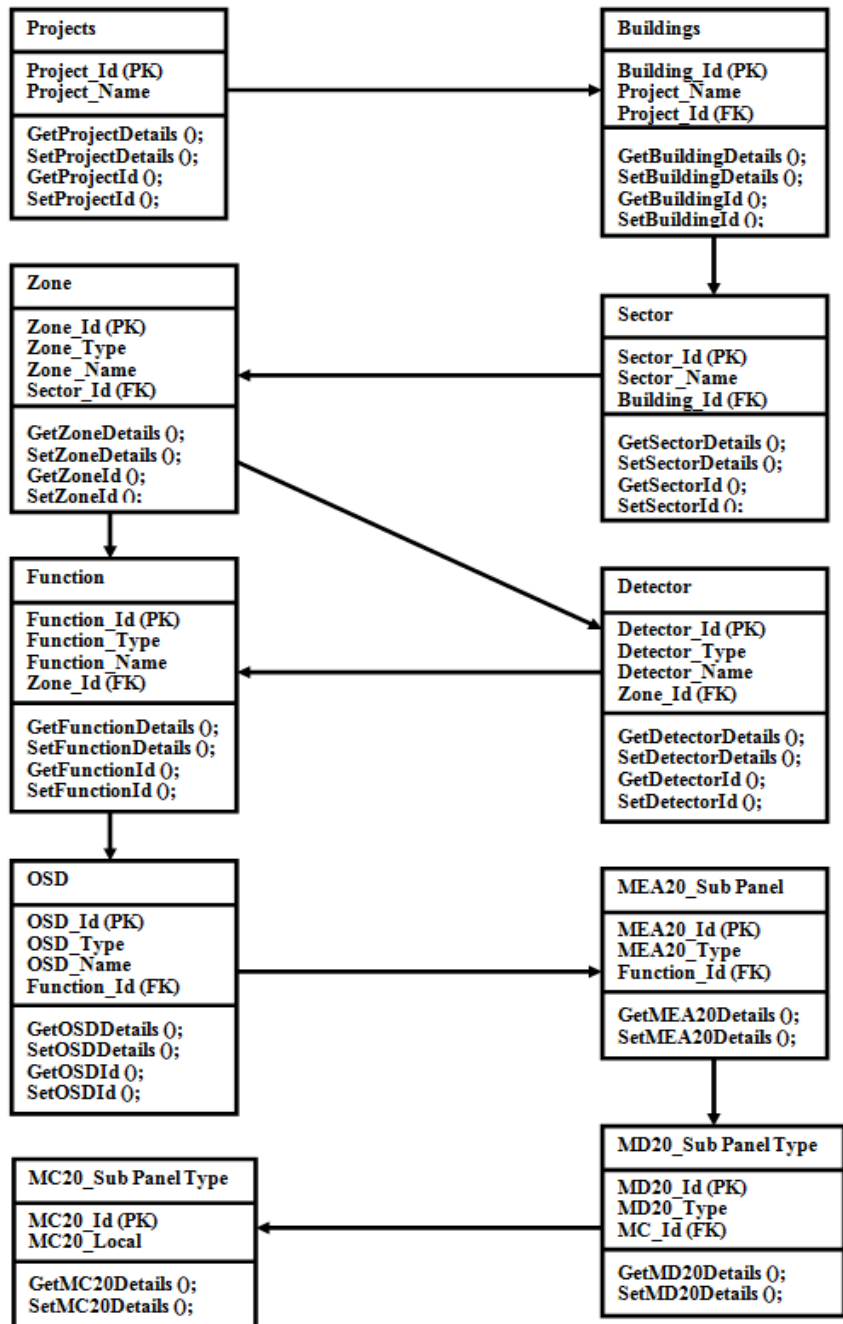
### **3. Analysis and Design**

### 3.1 Object Diagram



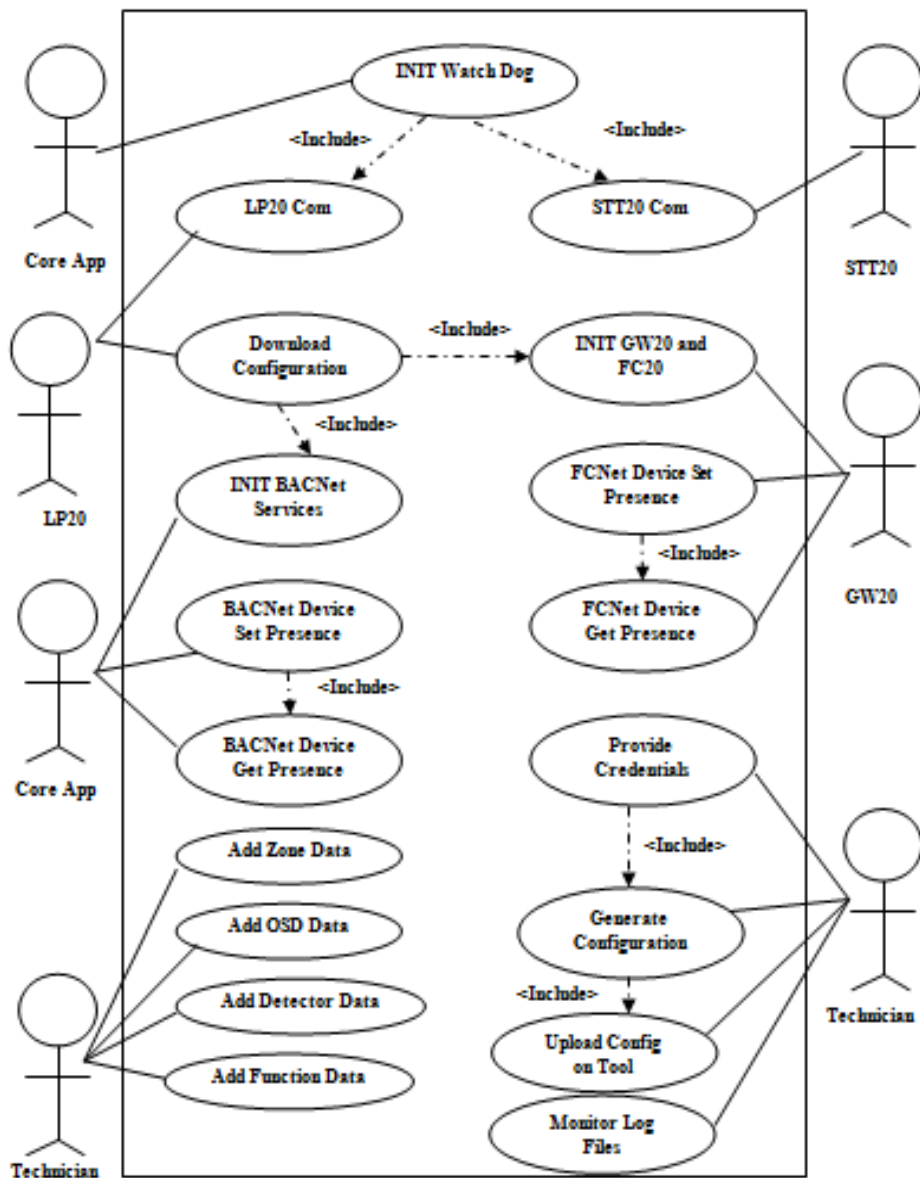


### 3.2 Class Diagram

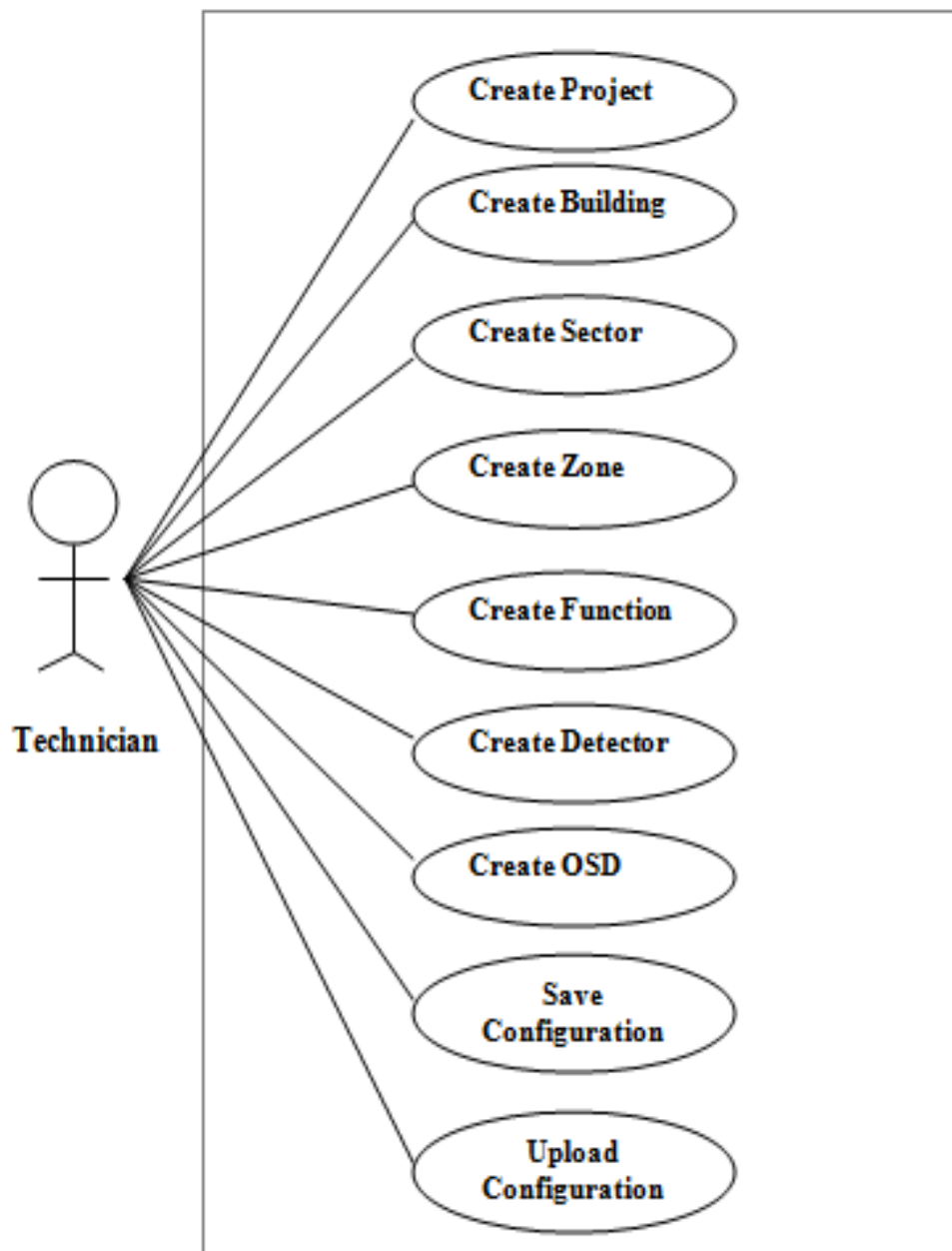


### 3.3 Use Case Diagram

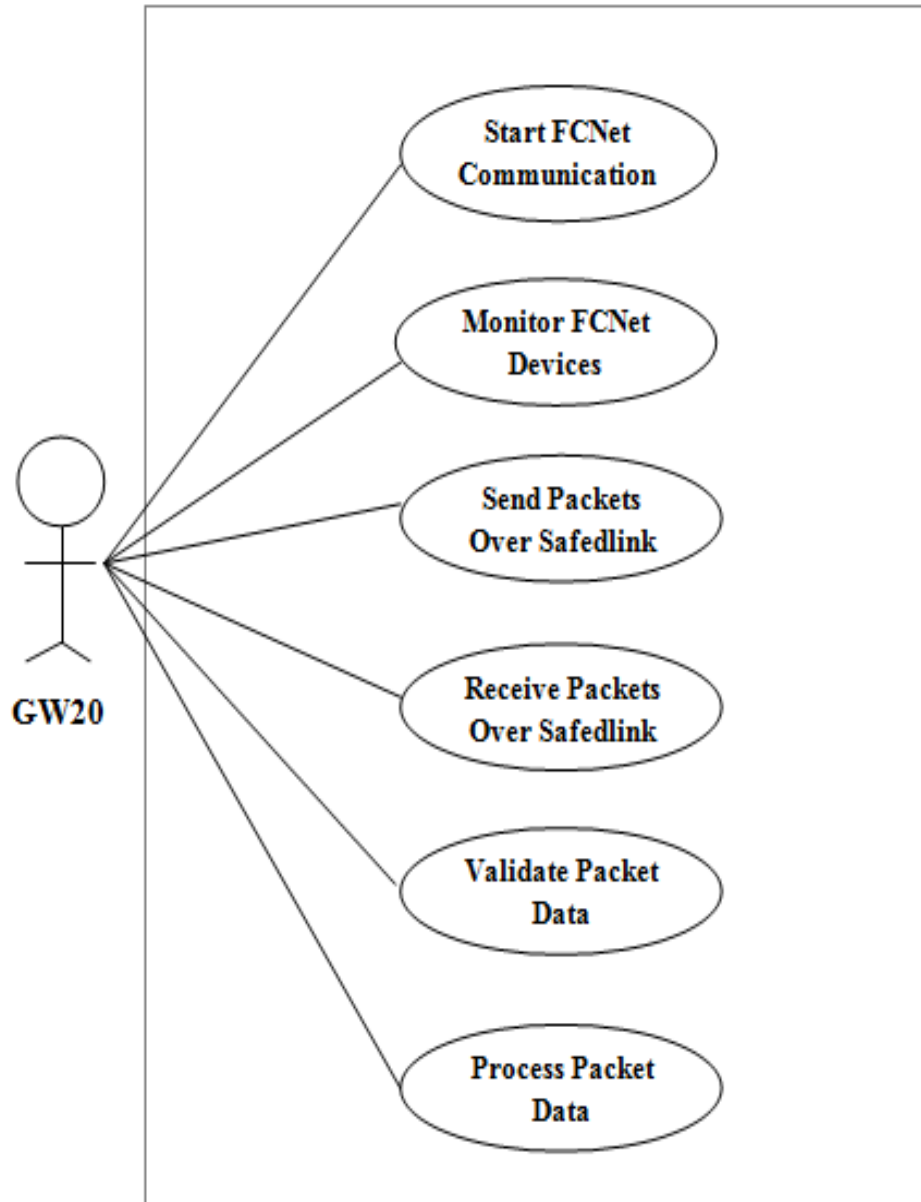
#### 3.3.1 System Use Case Diagram



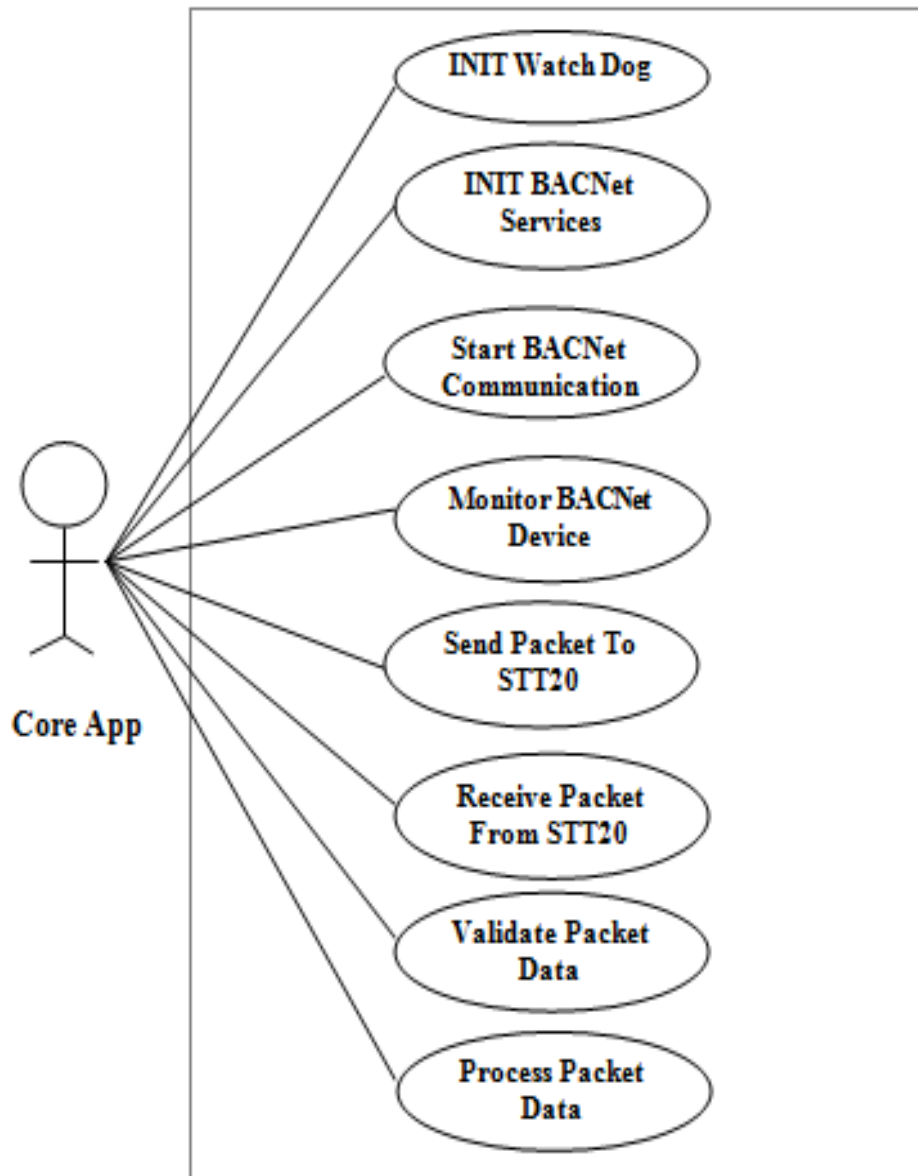
### 3.3.2 Technician Use Case Diagram



3.3.3 GW20 Use Case Diagram

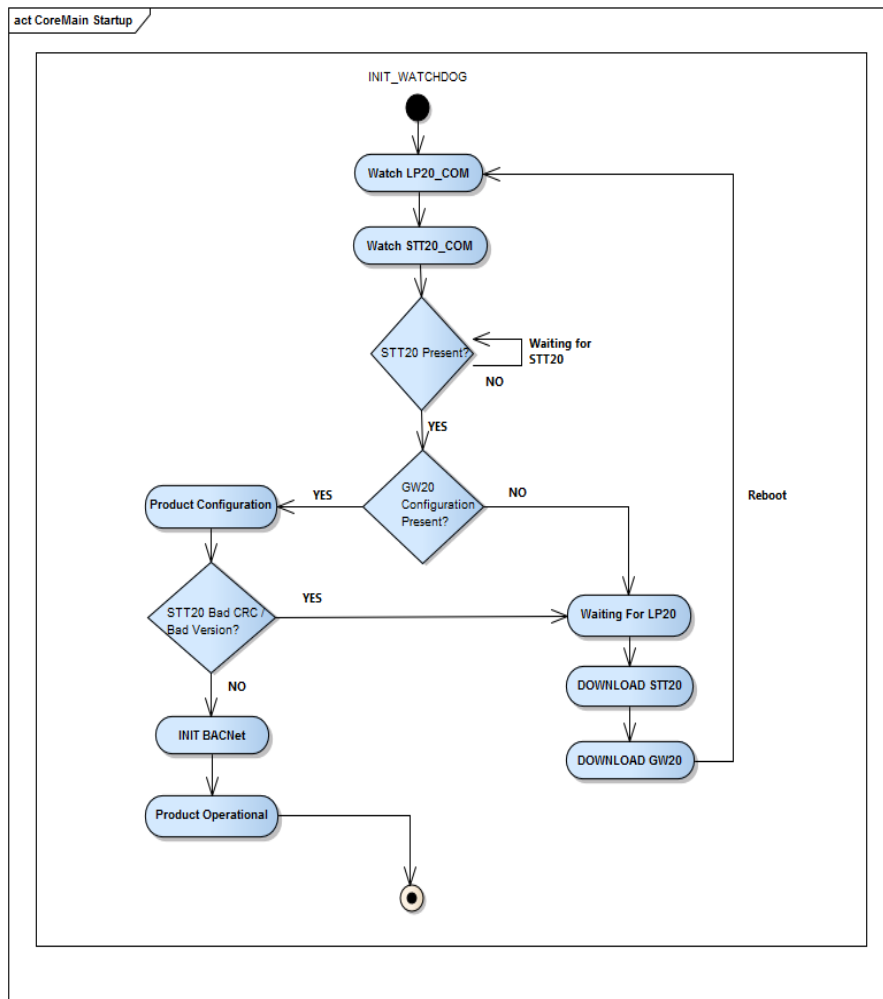


### 3.3.4 CoreApp Use Case Diagram

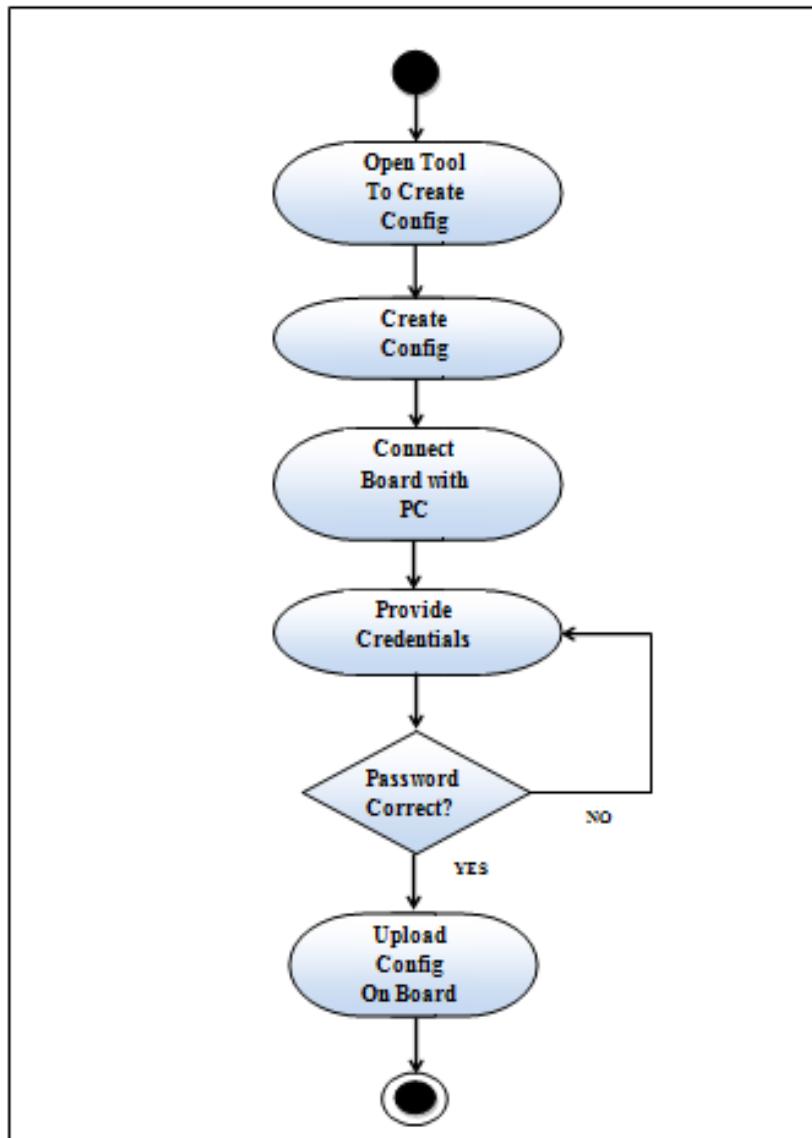


### 3.4 Activity Diagram

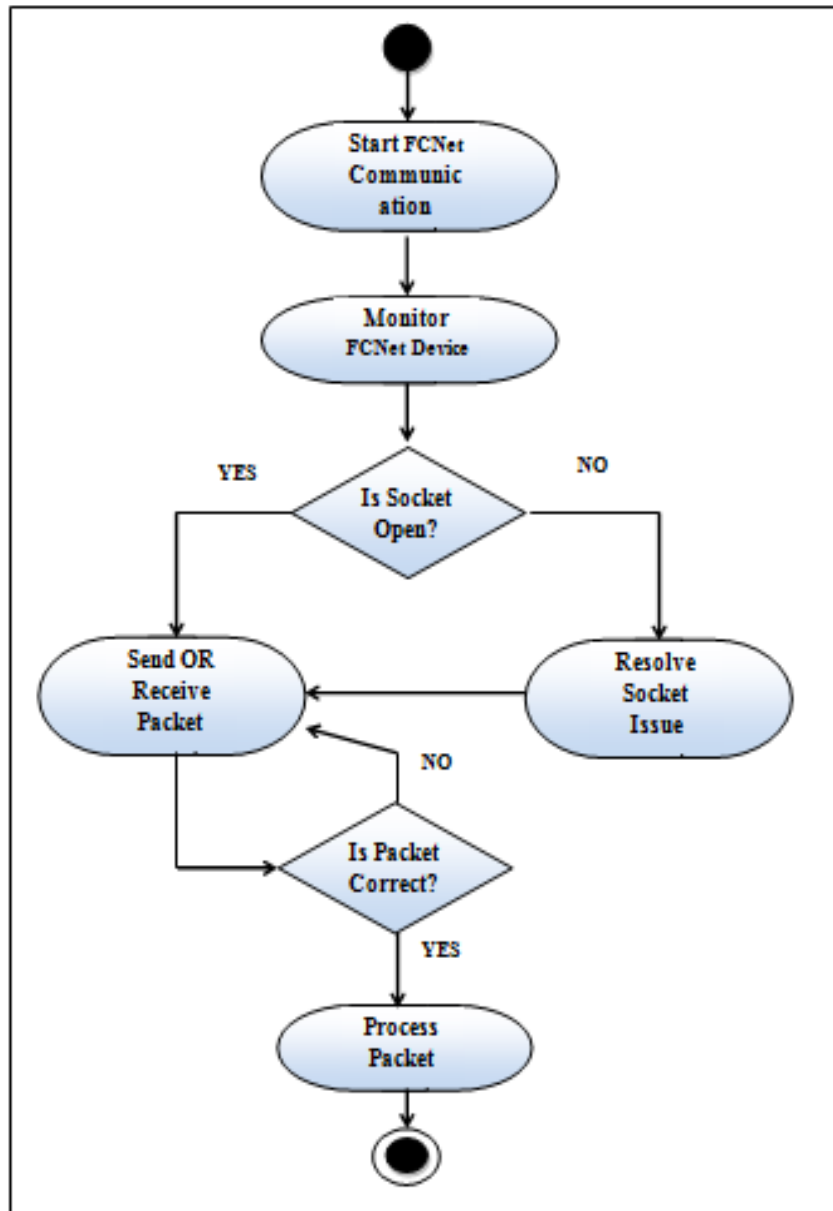
#### 3.4.1 System Activity Diagram



### 3.4.2 Technician Activity Diagram

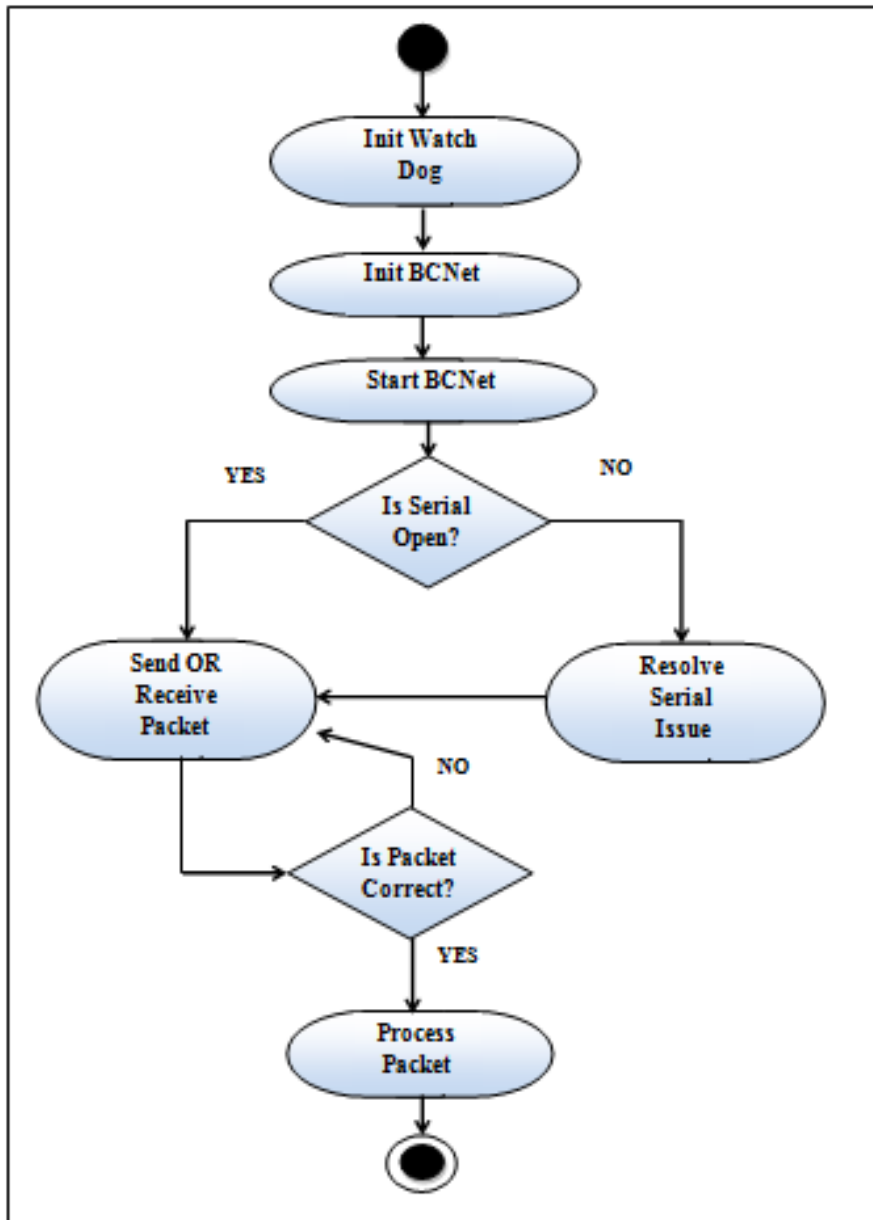


### 3.4.3 GW20 Activity Diagram





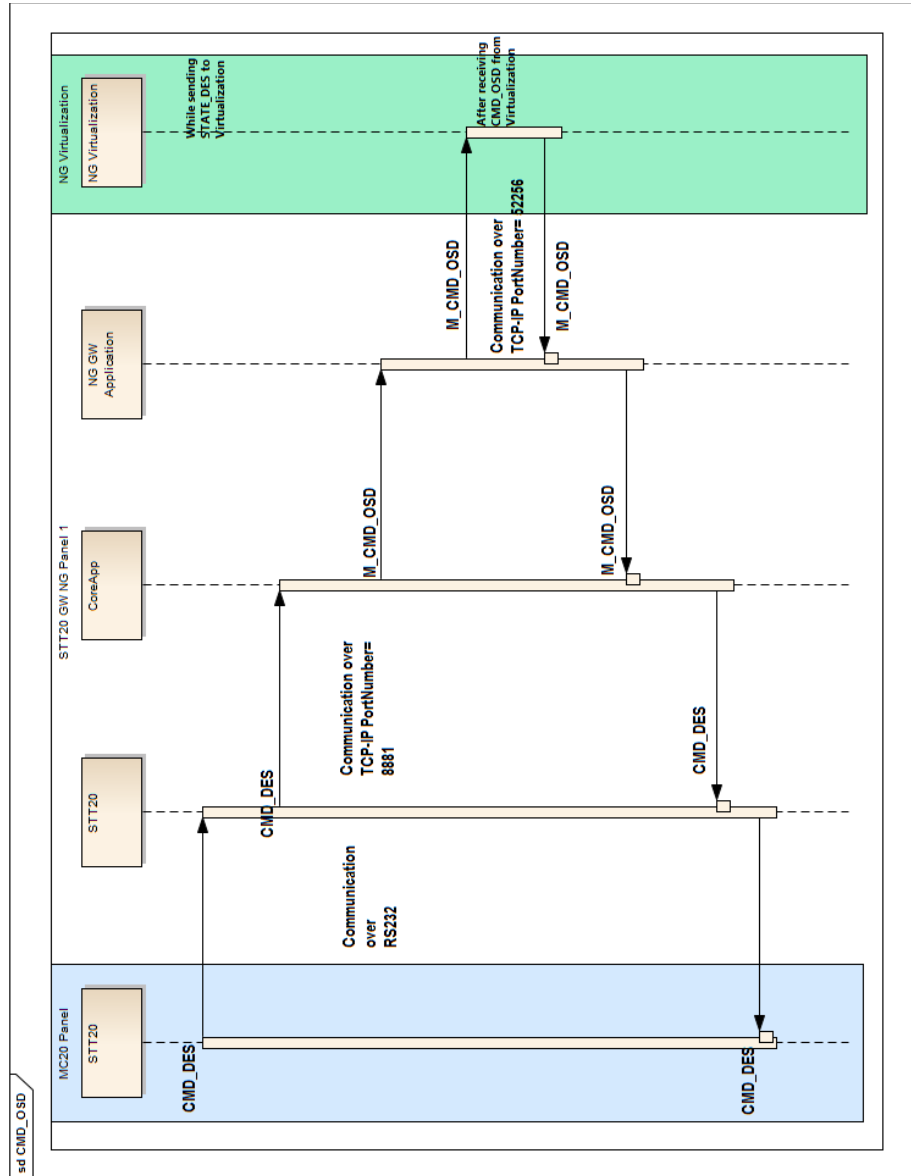
### 3.4.4 CoreApp Activity Diagram





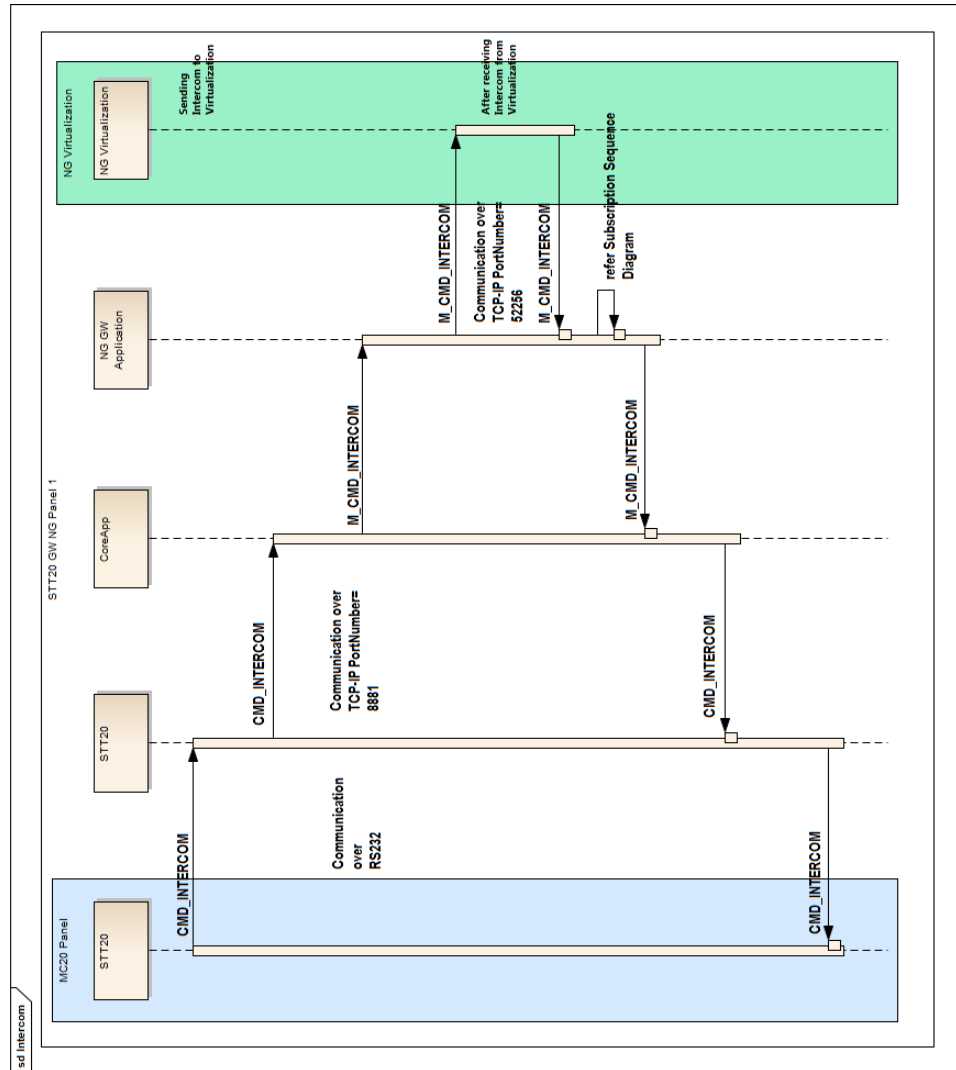
- FC20 panel initiate this process, FC20 panel sends ALARM state to NG Virtualization.
- FC20 panel will send M\_STATE\_ELEMENT packet to GWApp through NG Virtualization component.
- GWApp will extract the data from that packet and add it to Coreapp Normal message queue.
- Coreapp creates MC20 panel compatible packet and send to MC20 through STT20 component.
- If there are multiple zones under linked FC20 are in alarm (New) then that packet is transferred to MC20 panel. When we receive alarm (Removed) for all activated alarms then and only then GW App will send one alarm (Removed) packet to MC20 panel. To achieve this GW App need to keep track of zones which are in alarm?
- If there are multiple zones under remote FC20 are in alarm (New) then that packet is not transferred to MC20 panel. When we receive alarm (Removed) for at least single zone we send it to MC20 panel.

Commanding Protection Devices:-



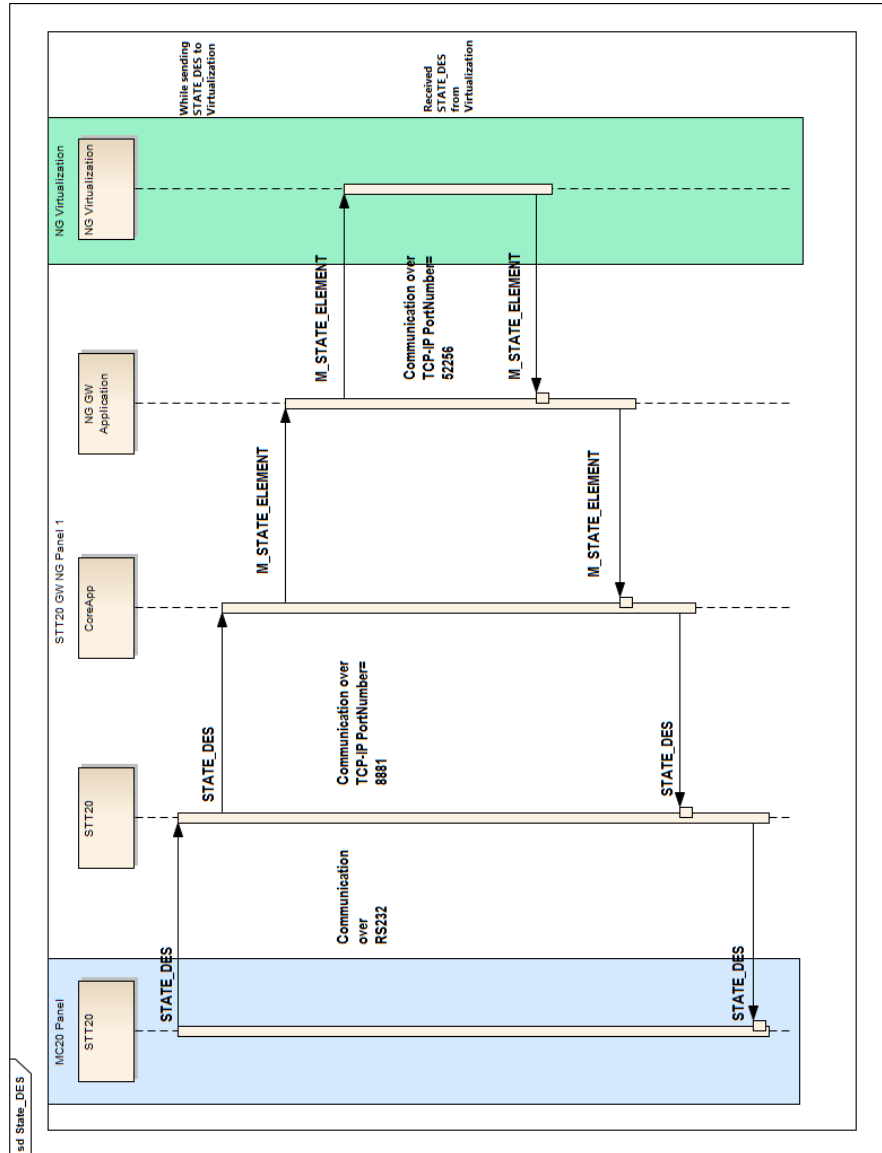
- MC20 panel initiate this process, MC20 panel sends DES command to another MC20 panel.
- MC20 panel will send CMD\_DES packet to Coreapp through STT20 component.
- Coreapp will extract the data from that packet and add it to GW application TX queue.
- GW Application creates FCNet compatible packet and send it to NG Virtualization.
- Vice-versa CMD\_OSD packet received at GW Application side from NG Virtualization and gets transferred to MC20 panel.

Monitor Detection Panel:-



- MC20 panel initiate this process, MC20 panel sends INTERCOM command to another MC20 panel.
- MC20 panel will send CMD\_INTERCOM packet to Coreapp through STT20 component.
- Coreapp will extract the data from that packet and add it to GW application TX queue.
- GW Application creates FCNet compatible packet and send it to NG Virtualization.
- When GW App receives Intercom command from NG Virtualization it sends it to MC20 panel and subscription create packet to NG Virtualization for that remote FC20.

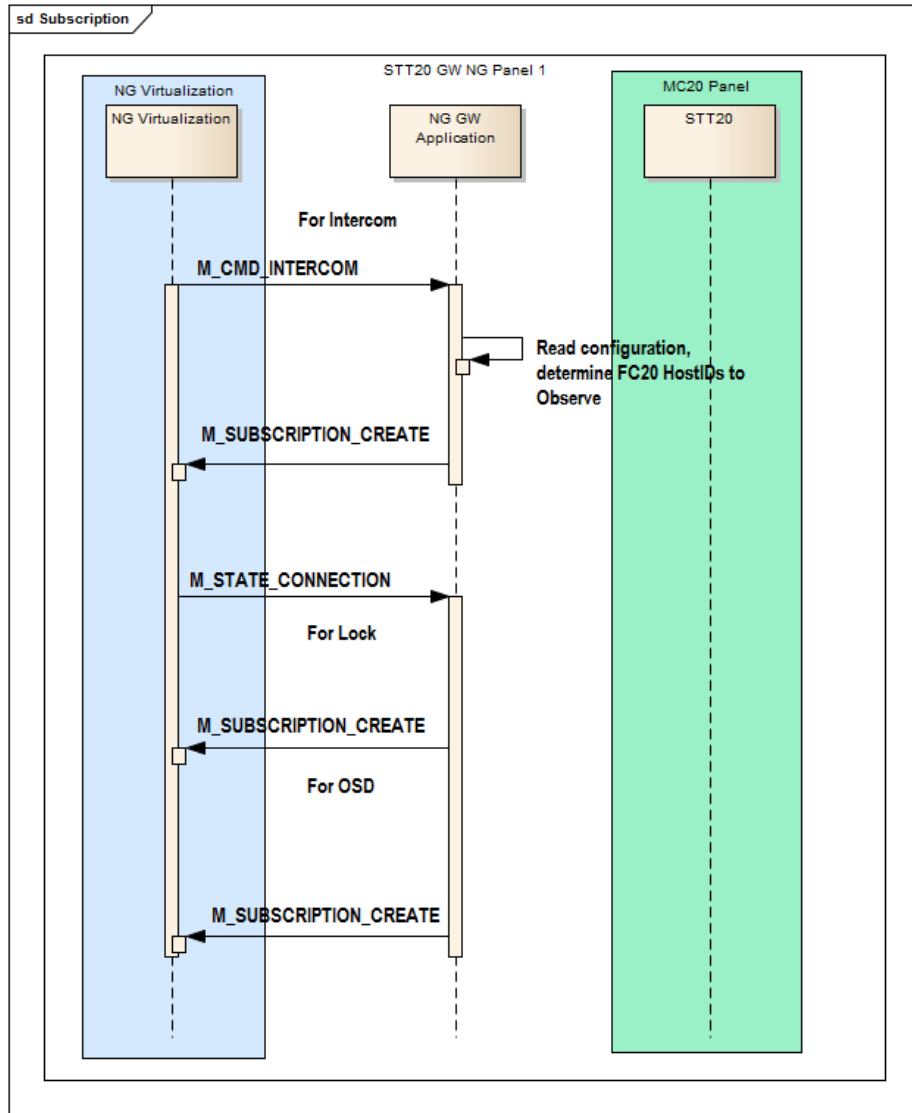
Monitor Detection Devices:-



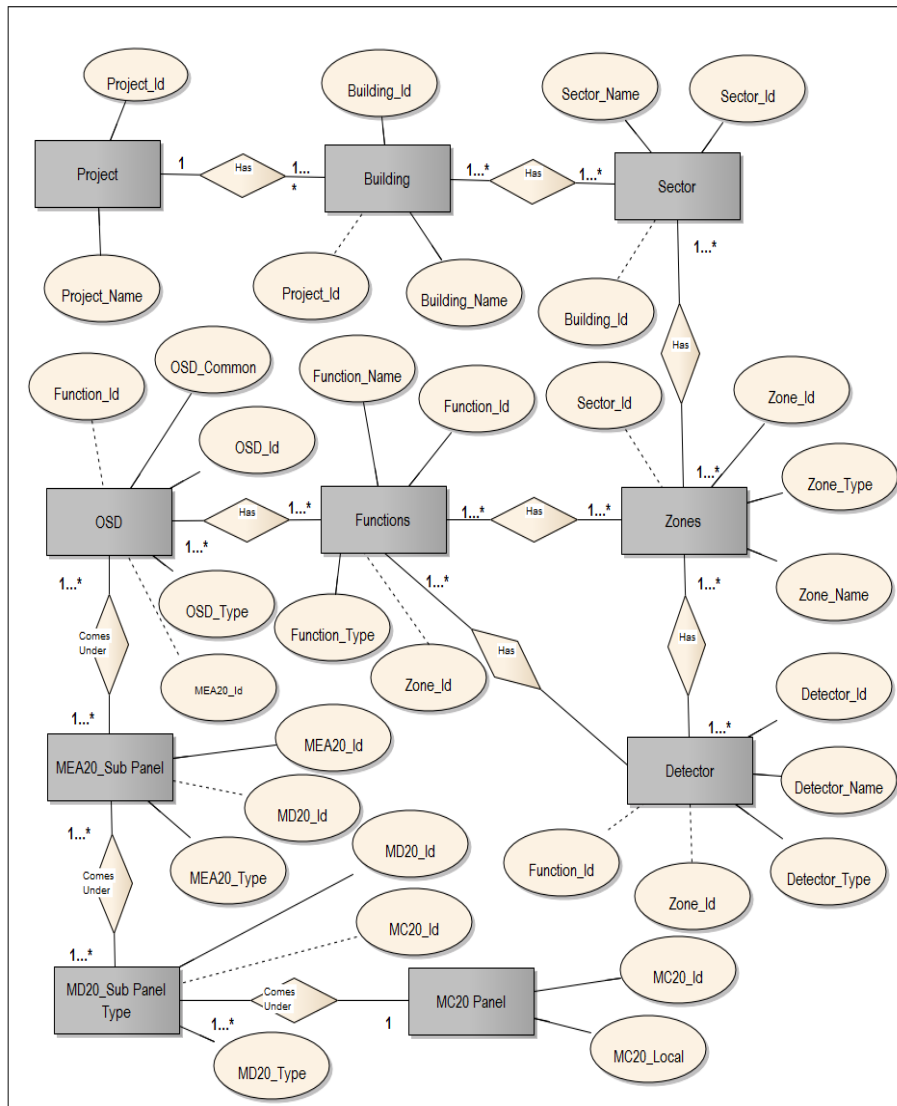


- MC20 panel initiate this process, MC20 panel sends DES state to another MC20 panel.
- MC20 panel will send STATE\_DES packet to Coreapp through STT20 component.
- Coreapp will extract the data from that packet and add it to GW application TX queue.
- GW Application creates FCNet compatible packet and send it to NG Virtualization.
- Vice-versa STATE\_DES packet received at GW Application side from NG Virtualization and gets transferred to MC20 panel.
- Here we need to map OSD states as representation of states over FCNet is different and over STT20 protocol is different.

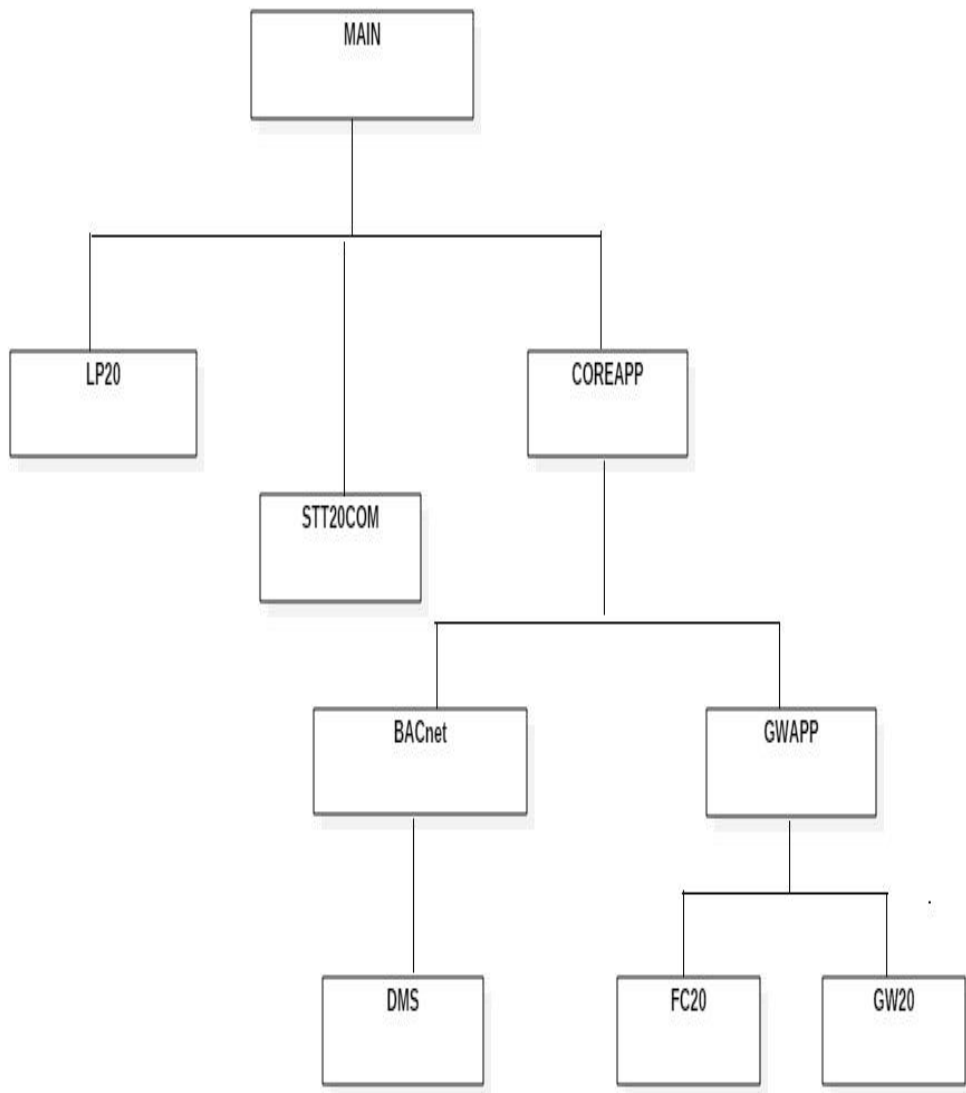
**Sending Subscription:-**



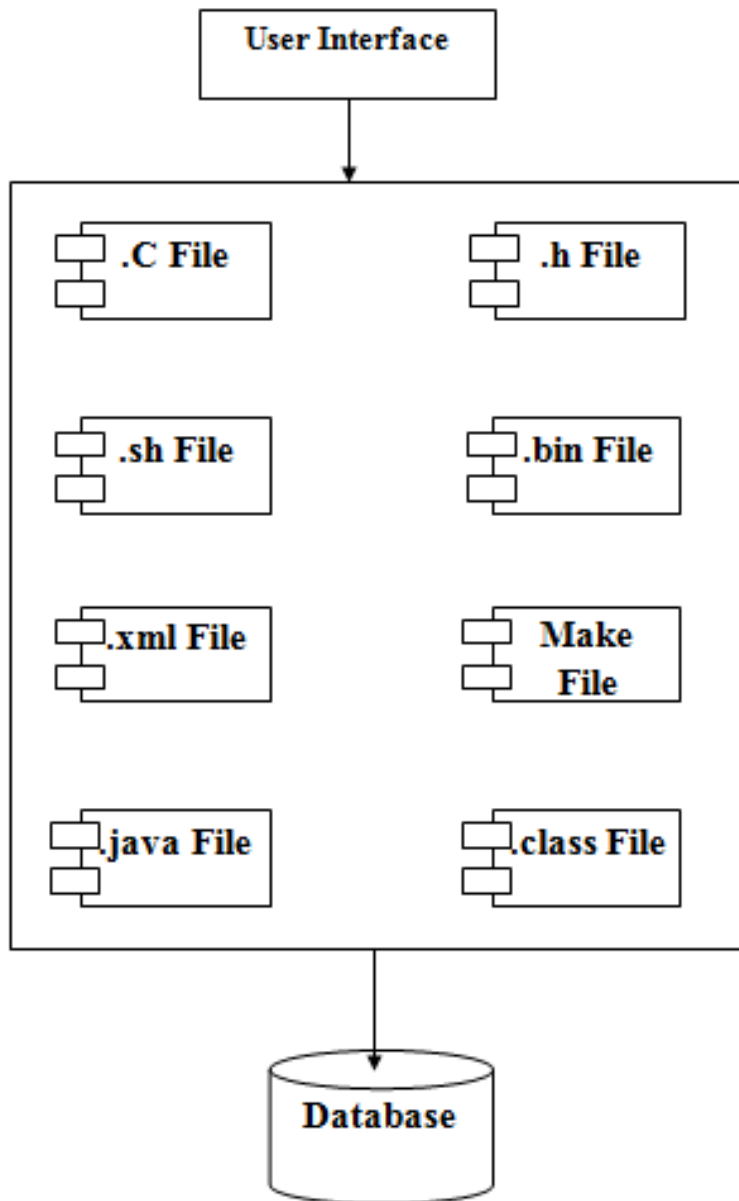
### 3.6 Entity Relationship Diagram



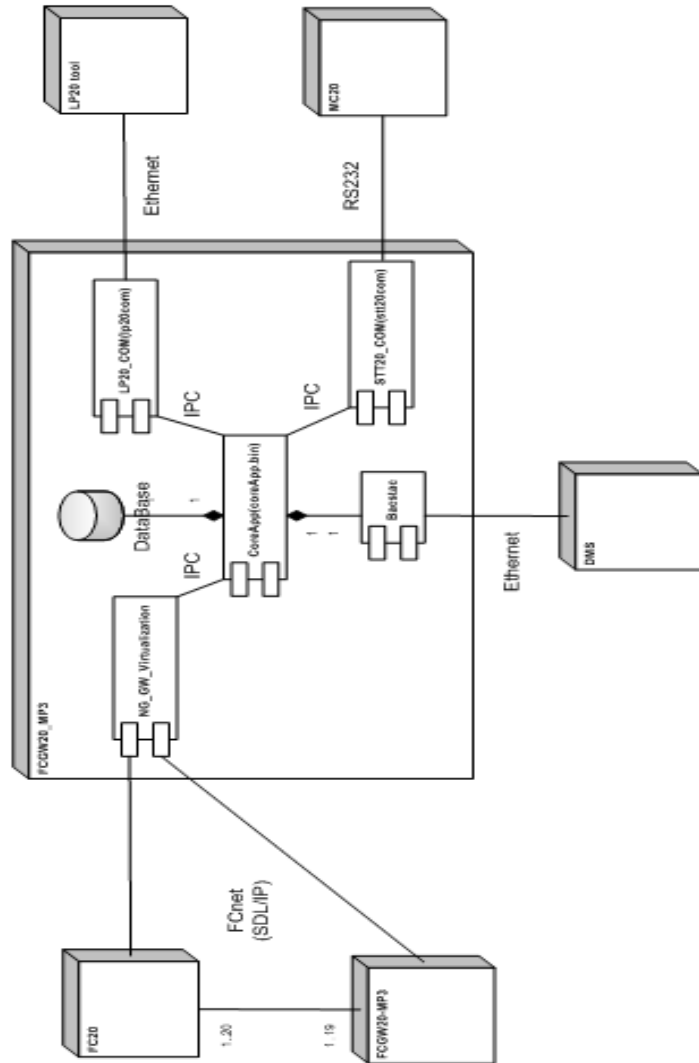
### 3.7 Module Hierarchy Diagram



### 3.8 Component Diagram



### 3.9 Deployment Diagram



### **3.10 Module Specification**

#### **[1] Core Application:-**

Core Application is main component of the system. As the name indicates core application has main responsibility. Core application initializes the watch dog which is used to monitor the other processes. Watch dog start looking for LP20 com driver and STT20 com driver. When these two processes become operational core app initialize the GW Application and BACNet service.

#### **[2] STT20**

STT20 communicates with STT20 com driver and Core application. STT20 communicate with STT20 com over serial communication means RS232 and with core application over TCP IP communication using 8881 port. STT20 transfer configuration file to actual STT20 hardware.

### **[3] LP20**

LP20 communicates with LP20 com driver over TCP IP communication using 2000 and 2001 port. After that LP20 starts downloading configuration file for STT20 and GW20 module.

LP20 can communicate with STT20 hardware directly over RS232 communication. It generates .XML file which provides database for GW20 and STT20 in which Project, Building, Sector, Zone etc. details are provided.

### **[4] GW20**

GW20 is new module introduced to establish communication between detection panel and protection panel. GW20 extract data from database and initialize the FC20 and GW20 device details. After that GW20 start monitoring all BACNet and FCNet devices to start communication with them.



### **[5] Technician**

Technician is responsible for creating database and configuration file. Technician uploads that file on board using package importing tool.

Technician then start SAT tool for monitoring the system and sniffing the packets. It starts SAT tool and provide IP address and port number 5004 then that tool connects with hardware and on that tool technician start monitoring all the logs and LED light.

## 3.11 User Interface Design

### 3.11.1 Create New Project

BDV: F-FXS2004-US\_en\_1-60.20.1\_01.eBDV, Standard BDV

File name: \* C:\Siemens\F-FXS2004-US\_en\_1-V7.0\Sites\CCL4\_Testing

Site name: \* CCL4\_Testing

Site no.:  
Site ID:

Metadata version: 60.20.1  
Metadata version integrated Voice: 60.17.0  
Voice audio library version: 60.8.0  
Creation date: 2017-03-28T17:08:15  
Date of last transfer to panel(s):

Creator

First name:  
Last name: \* Sharad Wagh  
Function:  
Organisation:  
Sales region:  
Company:  
Zip code:  
City:  
Country:  
Phone no.:  
License no.:  
E-mail: sharad.wagh@siemens.com

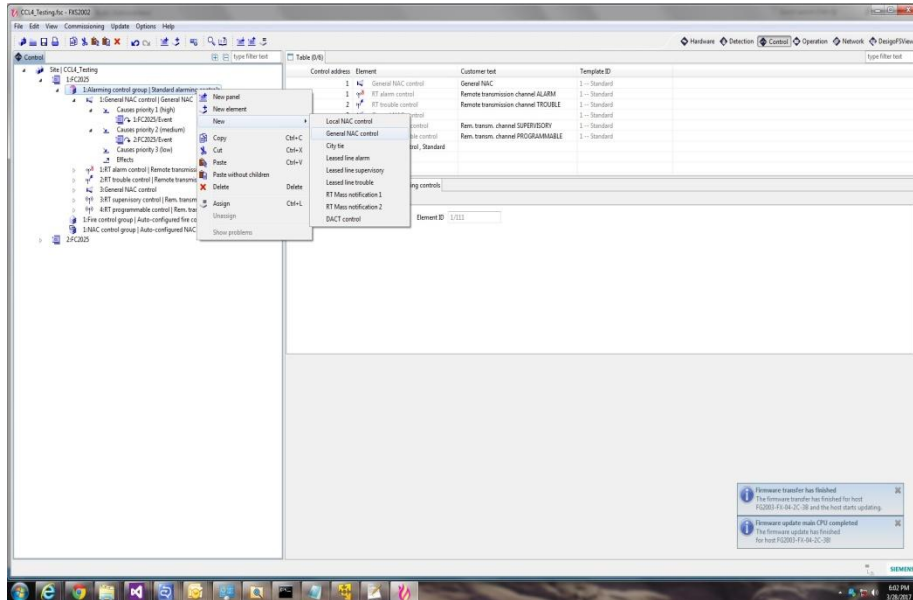
Customer

Name:  
Street:  
Street no.:  
Zip code:  
City:

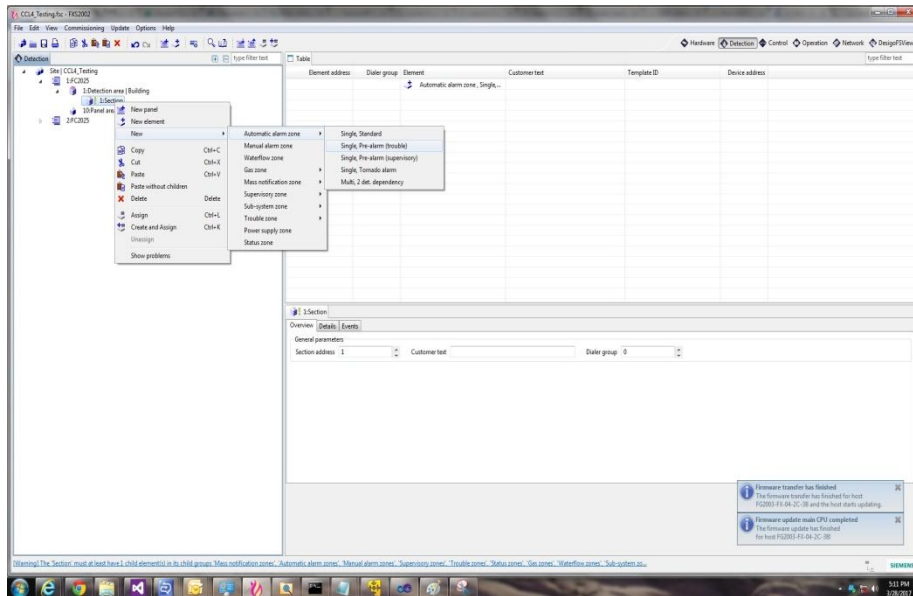
Clear all creator values Set creator values as default

Create Cancel

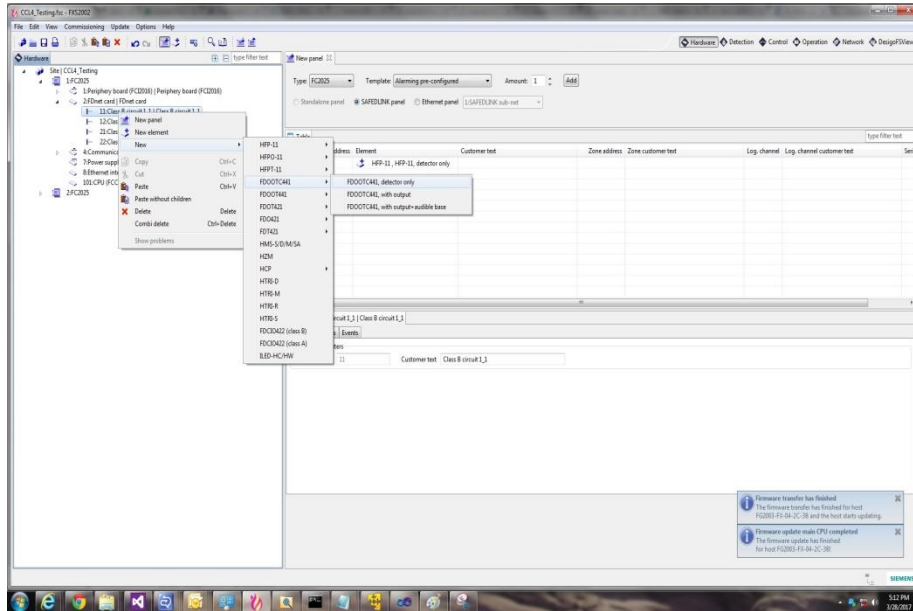
### 3.11.2 Add New Sector



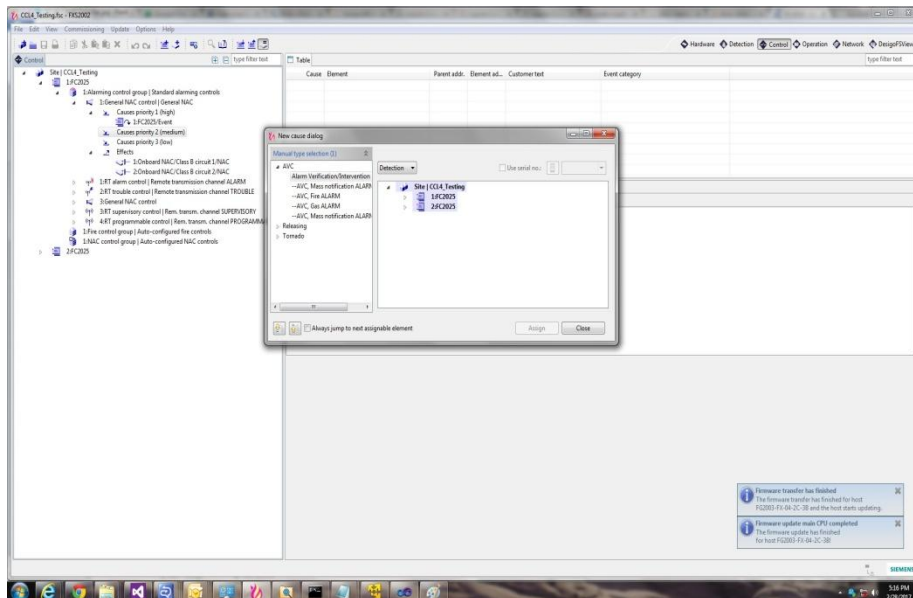
### 3.11.3 Add New Zone



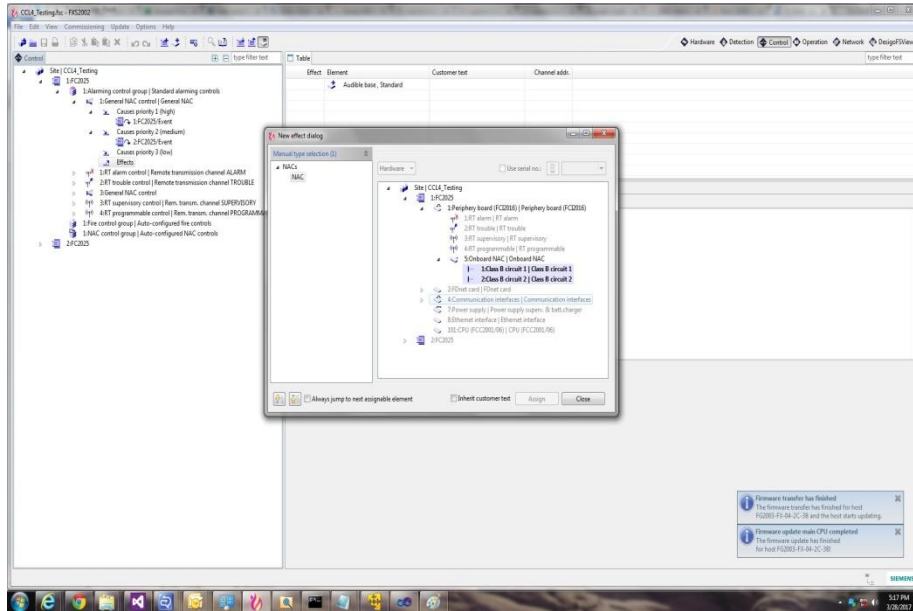
### 3.11.4 Add New Detector



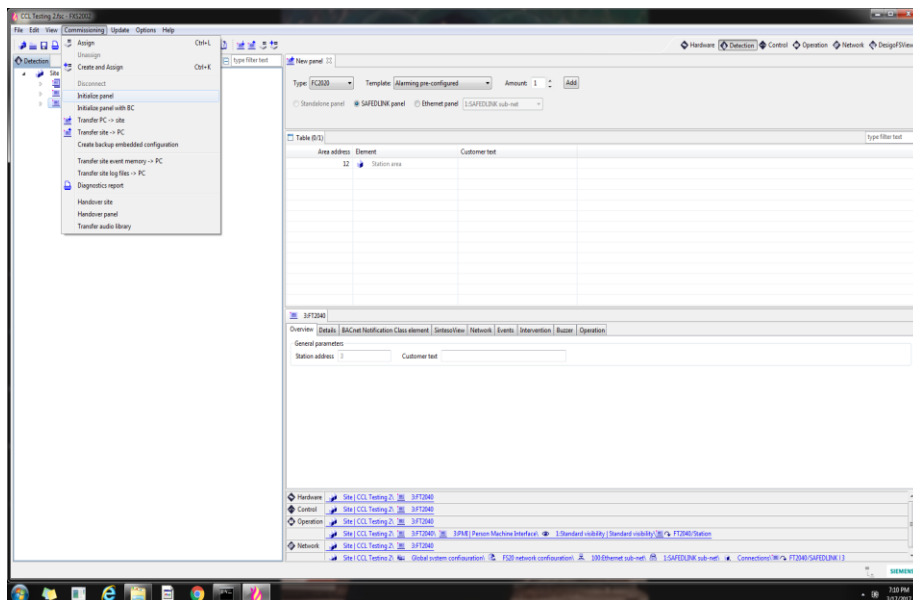
### 3.11.5 Add New Function



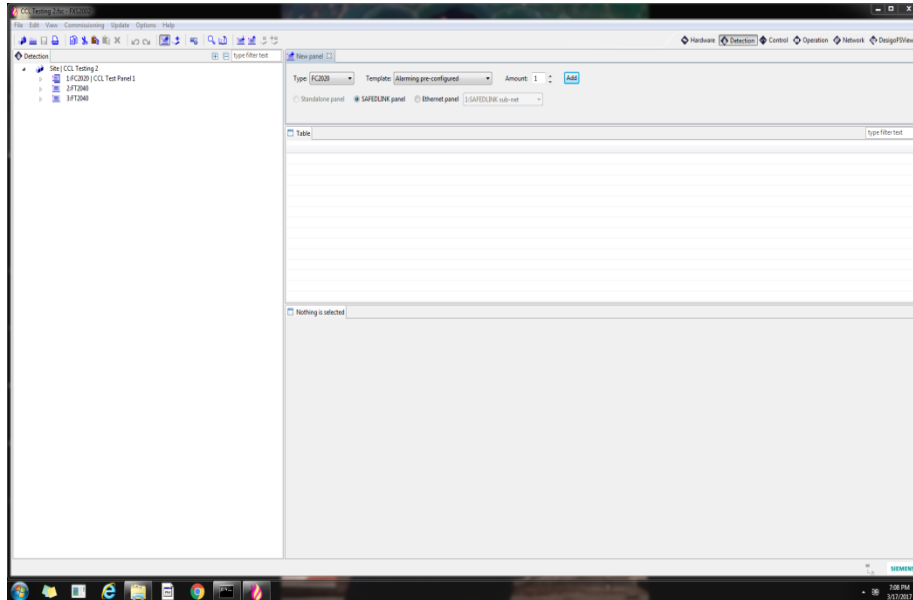
### 3.11.6 Add New Action (Protection Device)



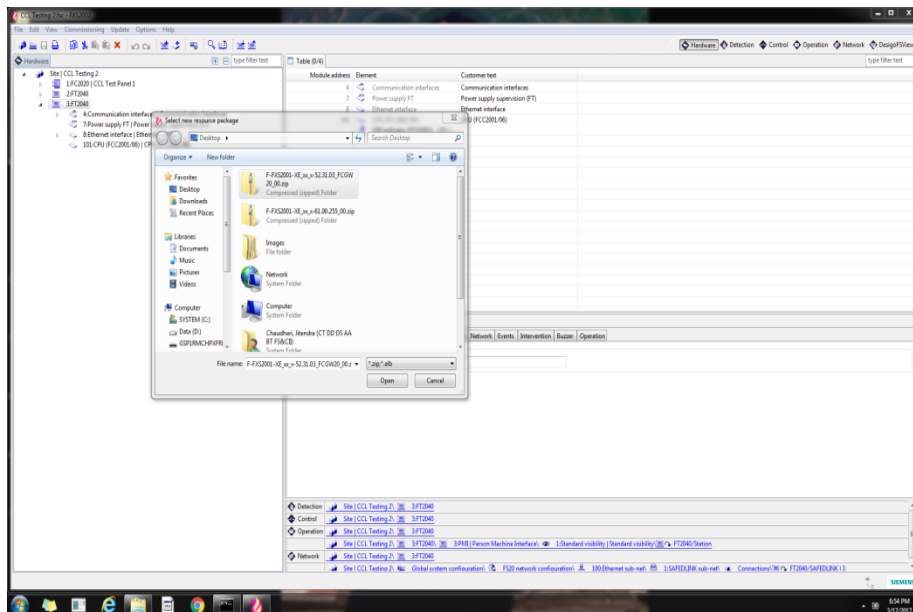
### 3.11.7 Upload Site



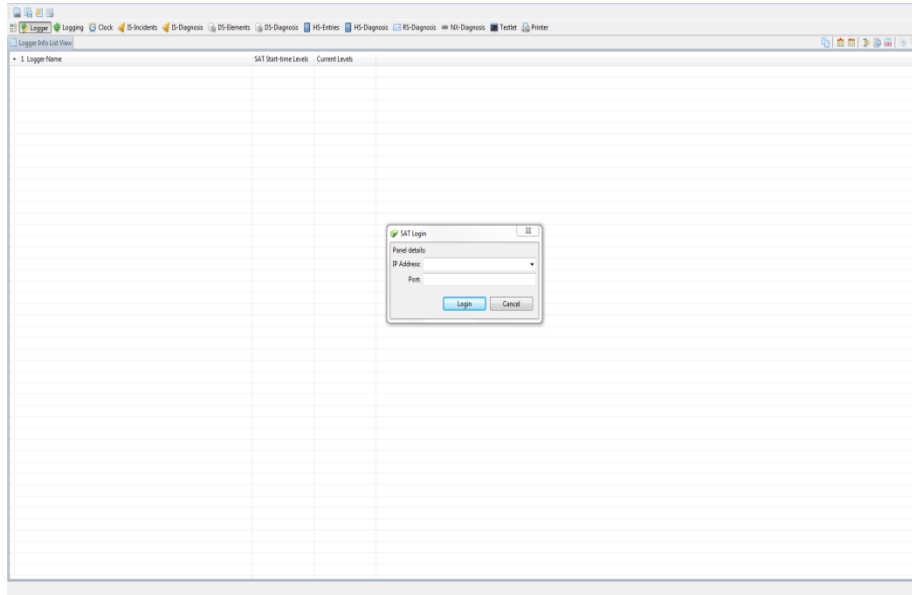
### 3.11.8 Create New Panel



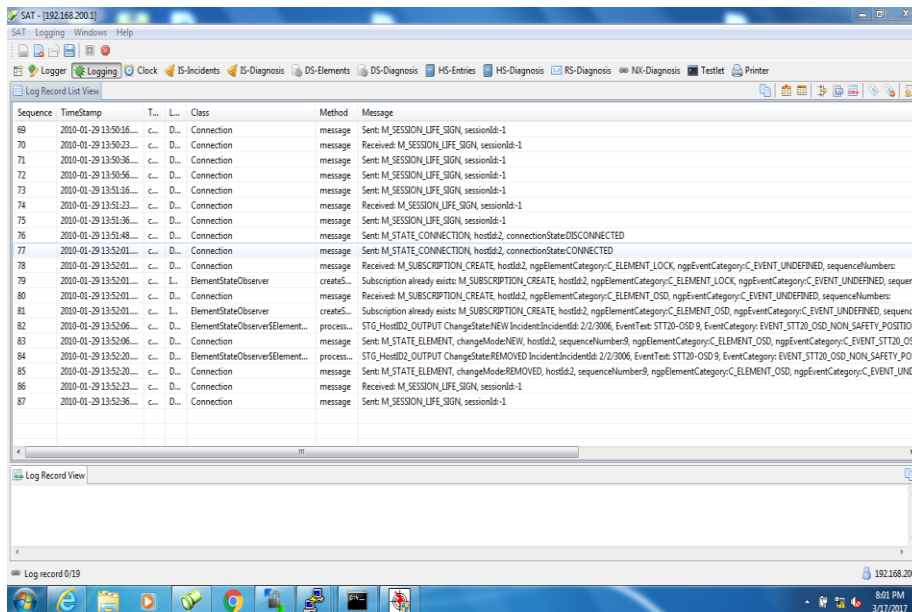
### 3.11.9 Import Resource Package



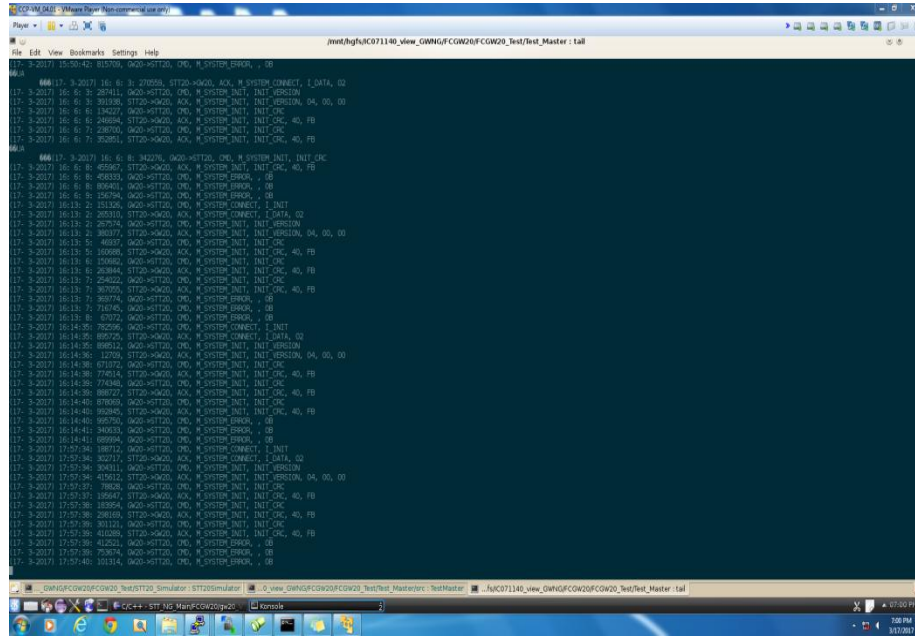
### 3.11.10 SAT Report Tool Login



### 3.11.11 SAT Logger



### 3.11.12 STT20 Simulator



### 3.11.13 FC20 LCD Display





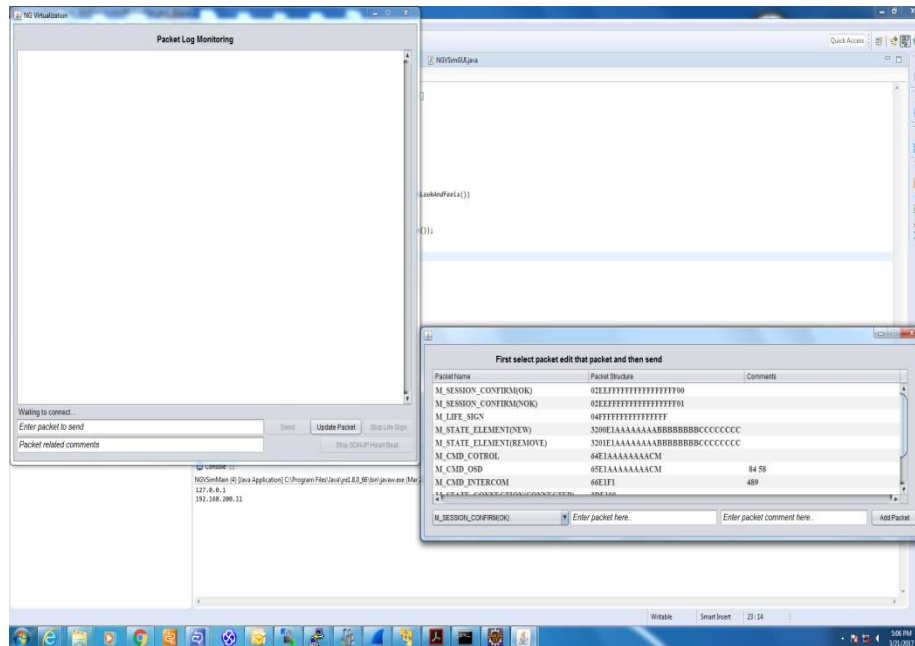
### 3.11.14 GW20 LCD Display



### 3.11.15 STT20 LCD Display



### 3.11.16 NG Virtualization Simulator



## 3.12 Data Dictionary

### 3.10.1 Project Table

Serial Number	Field Name	Data Type	Size	Description
1	Project_Id	Unsigned int	4 Bytes	Primary key of the table
2	Project_Name	Char[]	1 Byte/char	Project Name

### 3.10.2 Building Table

Serial Number	Field Name	Data Type	Size	Description
1	Building_Id	Unsigned int	4 Bytes	Primary key of the table
2	Building_Name	Char[]	1 Byte/char	Building Name
3	Project_Id	Unsigned int	4 Bytes	Foreign key of the table

### 3.10.3 Sector Table

Serial Number	Field Name	Data Type	Size	Description
1	Sector_Id	Unsigned int	4 Bytes	Primary key of the table
2	Sector_Name	Char[]	1 Byte/char	Sector Name
3	Building_Id	Unsigned int	4 Bytes	Foreign key of the table

### 3.10.4 Zones Table

Serial Number	Field Name	Data Type	Size	Description
1	Zone_Id	Unsigned int	4 Bytes	Primary key of the table
2	Zone_Name	Char[]	1 Byte/char	Zone Name
3	Zone_Type	Unsigned Char	1 Byte	It holds Zone types
4	Sector_Id	Unsigned int	4 Bytes	Foreign key of the table

### 3.10.5 Detector Table

Serial Number	Field Name	Data Type	Size	Description
1	Detector_Id	Unsigned int	4 Bytes	Primary key of the table
2	Detector_Name	Char[]	1 Byte/char	Detector Name
3	Detector_Type	Unsigned Char	1 Byte	It holds detector type
4	Zone_Id	Unsigned int	4 Bytes	Foreign key of the table

### 3.10.6 Functions Table

Serial Number	Field Name	Data Type	Size	Description
1	Function_Id	Unsigned int	4 Bytes	Primary key of the table
2	Function_Name	Char[]	1 Byte/char	Function Name
3	Function_Type	Unsigned Char	1 Byte	It holds Function type
4	Zone_Id	Unsigned int	4 Bytes	Foreign key of the table

### 3.10.7 OSD Table

Serial Number	Field Name	Data Type	Size	Description
1	OSD_Id	Unsigned int	4 Bytes	Primary key of the table
2	OSD_Type	Unsigned Char	1 Byte	It holds OSD type
3	OSD_Common	Unsigned Char	1 Byte	It shows whether OSD is common or not
4	Function_Id	Unsigned int	4 Bytes	Foreign key of the table
5	MEA20_Id	Unsigned int	4 Bytes	Foreign key of the table

### 3.10.8 MEA20\_Sub\_Panel Table

Serial Number	Field Name	Data Type	Size	Description
1	MEA20_Id	Unsigned int	4 Bytes	Primary key of the table
2	MEA20_Type	Unsigned Char	1 Byte	
3	MD20_Id	Unsigned int	4 Bytes	Foreign key of the table

### 3.10.9 MD20\_Sub\_Panel\_Type Table

Serial Number	Field Name	Data Type	Size	Description
1	MD20_Id	Unsigned int	4 Bytes	Primary key of the table
2	MD20_Type	Unsigned char	1 Byte	MD20 panel type
3	MC20_Id	Unsigned int	4 Bytes	Foreign key of the table

### 3.10.10 MC20\_Panel Table

Serial Number	Field Name	Data Type	Size	Description
1	MC20_Id	Unsigned int	4 Bytes	Primary key of the table
2	MC20_Local	Unsigned char	1 Byte	It shows whether the MC20 panel is Local or Remote

### 3.13 Table Specification

#### 3.11.1 Project Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	Project_Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	Project_Name	Char[]	1 Byte/ char	-	Project Name

#### 3.11.2 Building Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	Building_Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	Building_Name	Char[]	1 Byte /char	-	Building Name
3	Project_Id	Unsigned int	4 Bytes	FK	Foreign key of the table



### 3.11.3 Sector Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	Sector_Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	Sector_Name	Char[]	1 Byte /char	-	Sector Name
3	Building_Id	Unsigned int	4 Bytes	FK	Foreign key of the table

### 3.11.4 Zones Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	Zone_Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	Zone_Name	Char[]	1 Byte/char	-	Zone Name
3	Zone_Type	Unsigned Char	1 Byte	-	It holds Zone types
4	Sector_Id	Unsigned int	4 Bytes	FK	Foreign key of the table

### 3.11.5 Detector Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	Detector_Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	Detector_Name	Char[]	1 Byte /char	-	Detector Name
3	Detector_Type	Unsigned Char	1 Byte	-	It holds detector type
4	Zone_Id	Unsigned int	4 Bytes	FK	Foreign key of the table

### 3.11.6 Functions Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	Function_Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	Function_Name	Char[]	1 Byte /char	-	Function Name
3	Function_Type	Unsigned Char	1 Byte	-	It holds Function type
4	Zone_Id	Unsigned int	4 Bytes	FK	Foreign key in table

### 3.11.7 OSD Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	OSD_Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	OSD_Type	Unsigned Char	1 Byte	-	It holds OSD type
3	OSD_Common	Unsigned Char	1 Byte	-	It shows whether OSD is common or not
4	Function_Id	Unsigned int	4 Bytes	FK	Foreign key of the table
5	MEA20_Id	Unsigned int	4 Bytes	FK	Foreign key of the table

### 3.11.8 MEA20\_Sub\_Panel Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	MEA20_Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	MEA20_Type	Unsigned Char	1 Byte	-	
3	MD20_Id	Unsigned int	4 Bytes	FK	Foreign key of the table

### 3.11.9 MD20\_Sub\_Panel\_Type Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	MD20_ Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	MD20_ Type	Unsigned char	1 Byte	-	MD20 panel type
3	MC20_ Id	Unsigned int	4 Bytes	FK	Foreign key of the table

### 3.11.10 MC20\_Panel Table

Serial No	Field Name	Data Type	Size	Constraints	Description
1	MC20_ Id	Unsigned int	4 Bytes	PK	Primary key of the table
2	MC20_ Local	Unsigned char	1 Byte	-	It shows whether the MC20 panel is Local or Remote

### **3.14 Test Procedures and Implementation**

#### **➤ Unit Testing :**

A Unit testing focuses verification efforts on the smallest unit of software design the module. The unit test is normally white box oriented and the step can be conducted in parallel for multiple modules. In unit testing the module interface was tested to ensure that information properly flows into and out of a program unit under test. On testing individual units results were satisfactory and thus the accuracy and reliability of the unit was tested.

#### **➤ Integration Testing:**

Although each module is verified individually during modules testing, it is important to determine if the modules are working properly when linked together. This is also referred to as integration testing or also as interfacing. In this step output is compared with the manually calculated output. This comparison gives the result of system testing. This finally ensures that the system is functioning properly. On testing the results were

satisfactory and thus the accuracy and reliability of the integrated software was tested.

➤ **System Testing :**

It is the testing of the whole system prior to delivery. The purpose of system testing is to identify defects that will only surface when a complete system is assembled. i.e. defects that cannot be attributed to individual components or the interaction between two components. System testing includes testing of performance, security, configuration sensitivity, start up and recovery from failure modes. On testing the results were satisfactory and thus the accuracy and reliability of the system was tested.

➤ **GUI Testing :**

Graphical user interface testing is the process of testing the user interface of the application to ensure it meets its specifications. In addition to functionality, GUI testing evaluates design elements such as layout, colours, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links and content. On

testing the results were satisfactory and thus the accuracy of UI was tested.

➤ **Platform Testing :**

For web-application, platform testing means four main points, viz. Web forms display correctly on all supported browsers and supported versions of those browsers. The web-application appropriately handles unsupported browser versions, such as by displaying instructions for downloading the required version. The web application has acceptable performance over slower forms of network connections such as modems. On testing the results were satisfactory and thus the accuracy and reliability of the system was tested.

➤ **Performance Testing :**

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

➤ **Load testing :**

It is the simplest form of testing conducted to understand the behaviour of the system under a specific load on the database, application server, etc.

➤ **Stress testing :**

It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.

➤ **Security Testing :**

Security testing is a process intended to reveal flaws in the security mechanisms of a system that protect data and maintain functionality as intended. Typical security requirements may include specific elements of confidentiality, integrity, authentication, availability, authorization and non-repudiation. On testing the results were satisfactory and thus the security of the system was tested.



### 3.14.1 Alarm signal from remote FC20

Step Name	Description	Expected
<b>step 1</b>	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
<b>step 2</b>	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
<b>step 3</b>	Run the coreapp application on both the panels. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the GWNG Virtualization simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERVISION to the GWNG Virtualization simulator. GWAPP sends subscription create packet to FC20.
<b>step 4</b>	Send Alarm packet (Alarm ON) from FC20	State Zone packet with Alarm on is received on local and remote STT20 Simulator.
<b>step 5</b>	Disconnect the FC20 from network	Both the STT20 simulator should display - FC20 not connected.

### 3.14.2 Alarm signal from remote FC20 to GW20 (3 panels, 1FC20, 2 GW20s)

Step Name	Description	Expected
step 1	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
step 2	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
step 3	Run the coreapp application. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the GWNG Virtualization simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERVISION to the GWNG Virtualization simulator.
step 4	Send M_STATE_CONNECTION packet from the GWNG Virtualization simulator	GWAPP receives M_STATE_CONNECTION packets and displays their connection states in the log file of coreapp and STT20 simulator. GWAPP sends M_SUBSCRIPTION_CREATE packet to the GWNG Virtualization simulator.
step 5	Send M_STATE_ELEMENT packet with host ID, sequence number,	Coreapp displays M_STATE_ELEMENT packet received in the log file.

	<p>ELEMENT_CATEGORY as C_ELEMENT_ZONE and EVENT_CATEGORY as C_EVENT_ALARM and state as NEW/REMOVED from the GWNG Virtualization simulator. Sequence number field contains zone number. Send M_STATE_ELEMENT packets for all zone numbers.</p>	<p>Coreapp sends STATE_ZONE packet with the same host ID, sequence number and state (Active/Deactivate) as received in the M_STATE_ELEMENT packet to the STT20 simulator.</p> <p>Coreapp does not send STATE_ZONE packet until all the zones are deactivated i.e. M_STATE_ELEMENT packet for all the zone numbers is received.</p>
<b>step 6</b>	View the log file of STT20 simulator	Log file displays STATE_ZONE packet received.

### 3.14.3 Multiple Alarm signals from local FC20

Step Name	Description	Expected
<b>step 1</b>	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
<b>step 2</b>	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
<b>step 3</b>	Run the coreapp application. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the GWNG Virtualization

		<p>simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERVISION to the GWNG Virtualization simulator.</p>
<b>step 4</b>	<p>Send M_STATE_CONNECTION packet from the GWNG Virtualization simulator</p>	<p>GWAPP receives M_STATE_CONNECTION packets and displays their connection states in the log file of coreapp and STT20 simulator. GWAPP sends M_SUBSCRIPTION_CREATE packet to the GWNG Virtualization simulator.</p>
<b>step 5</b>	<p>Send M_STATE_ELEMENT packet with host ID, sequence number, ELEMENT_CATEGORY as C_ELEMENT_ZONE and EVENT_CATEGORY as C_EVENT_ALARM and state as NEW from the GWNG Virtualization simulator. Sequence number field contains zone number. Send M_STATE_ELEMENT packets for all zone numbers.</p>	<p>Coreapp displays M_STATE_ELEMENT packet received in the log file. Coreapp sends STATE_ZONE packet coreapp displays M_STATE_ELEMENT packet received in the log file. Coreapp sends STATE_ZONE packet with the same host ID, sequence number and state (Active/Deactivate) as received in the M_STATE_ELEMENT packet to the STT20 simulator.</p>
<b>step 6</b>	<p>Send M_STATE_ELEMENT packet with state as removed from GWNG Virtualization simulator.</p>	<p>Coreapp displays M_STATE_ELEMENT packet received in the log file. Coreapp sends STATE_ZONE packet with</p>

		the state as Deactivate to the STT20 simulator only when it receives M_STATE_ELEMENT packet for all the zones which were previously in Alarm state.
<b>step 7</b>	View the log file of STT20 simulator.	Log file displays STATE_ZONE packet received.

### 3.14.4 Intercom Control from MC20 to STT

Step Name	Description	Expected
<b>step 1</b>	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
<b>step 2</b>	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
<b>step 3</b>	Run the coreapp application. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the GWNG Virtualization simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERVISION to the GWNG Virtualization simulator.
<b>step 4</b>	Send M_STATE_CONNECTION packet from the GWNG	GWAPP receives M_STATE_CONNECTION packets and displays their

	Virtualization simulator	connection states in the log file of coreapp and STT20 simulator. GWAPP sends M_SUBSCRIPTION_CREATE packet to the GWNG Virtualization simulator.
<b>step 5</b>	Send CMD_INTERCOM packet with target MC20 number and SDI number(to which MC20 should monitor) from STT20 simulator	Log file of coreapp displays CMD_INTERCOM packet received. GWAPP sends M_CMD_INTERCOM with the same MC20 number and SDI number as received in the CMD_INETRCOM packet to the GWNG Virtualization simulator.
<b>step 6</b>	View the log file of GWNG Virtualization simulator	Log file displays M_CMD_INTERCOM packet received.

### 3.14.5 Intercom Control from STT to MC20

Step Name	Description	Expected
<b>step 1</b>	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
<b>step 2</b>	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
<b>step 3</b>	Run the coreapp application. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the

		GWNG Virtualization simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERVISION to the GWNG Virtualization simulator.
<b>step 4</b>	Send M_STATE_CONNECTION packet from the GWNG Virtualization simulator	GWAPP receives M_STATE_CONNECTION packets and displays their connection states in the log file of coreapp and STT20 simulator. GWAPP sends M_SUBSCRIPTION_CREATE packet to the GWNG Virtualization simulator.
<b>step 5</b>	Send M_CMD_INTERCOM packet with host ID of MC20 and host ID of remote FC20 from the GWNG Virtualization simulator.	Coreapp displays M_CMD_INTERCOM packet received in the log file. Coreapp sends CMD_INETRCOM packet to the STT20 simulator. Coreapp sends M_SYUBSCRIPTION_CREATE packet to the remote FC20. GWNG Virtualization simulator displays M_SUBSCRIPTION_PACKET received.
<b>step 6</b>	View the log file of STT20 simulator	Log file displays CMD_INTERCOM received.

## 3.14.6 OSD Control from MC20 to STT

Step Name	Description	Expected
<b>step 1</b>	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
<b>step 2</b>	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
<b>step 3</b>	Run the coreapp application. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the GWNG Virtualization simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERVISION to the GWNG Virtualization simulator.
<b>step 4</b>	Send M_STATE_CONNECTION packet from the GWNG Virtualization simulator	GWAPP receives M_STATE_CONNECTION packets and displays their connection states in the log file of coreapp and STT20 simulator. GWAPP sends M_SUBSCRIPTION_CREATE packet to the GWNG Virtualization simulator.
<b>step 5</b>	Send CMD_DES packet with host ID, Sequence number and state as Activate/Deactivate from STT20 simulator	Log file of coreapp displays CMD_DES packet received. GWAPP sends M_CMD_OSD packet with same host ID, sequence number and state



		Activate/Deactivate as received in CMD_DES packet to the GWNG Virtualization simulator.
<b>step 6</b>	View the log file of GWNG Virtualization simulator	Log file displays M_CMD_OSD packet received.

### 3.14.7 OSD Control from STT to MC20

<b>Step Name</b>	<b>Description</b>	<b>Expected</b>
<b>step 1</b>	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
<b>step 2</b>	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
<b>step 3</b>	Run the coreapp application. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the GWNG Virtualization simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERVISION to the GWNG Virtualization simulator.
<b>step 4</b>	Send M_STATE_CONNECTION packet from the GWNG Virtualization simulator	GWAPP receives M_STATE_CONNECTION packets and displays their connection states in the log file of coreapp and STT20 simulator.

		GWAPP sends M_SUBSCRIPTION_CREATE packet to the GWNG Virtualization simulator.
<b>step 5</b>	Send M_CMD_OSD packet with target host ID, sequence number and command as Activate/Deactivate from GWNG Virtualization simulator.	Coreapp displays M_CMD_OSD packet received in the log file Coreapp sends CMD_DES packet with same host ID, sequence number and command as received in the M_CMD_OSD packet to the STT20 simulator.
<b>step 6</b>	View the log file of STT20 simulator	Log file displays CMD_DES packet received.

### 3.14.8 OSD Monitoring from MC20 to STT

Step Name	Description	Expected
<b>step 1</b>	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
<b>step 2</b>	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
<b>step 3</b>	Run the coreapp application. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the GWNG Virtualization simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERV

		ISION to the GWNG Virtualization simulator.
<b>step 4</b>	Send M_STATE_CONNECTION packet from the GWNG Virtualization simulator	GWAPP receives M_STATE_CONNECTION packets and displays their connection states in the log file of coreapp and STT20 simulator. GWAPP sends M_SUBSCRIPTION_CREATE packet to the GWNG Virtualization simulator.
<b>step 5</b>	Send STATE_DES packet with host ID, sequence number and state from STT20 simulator. State could be one of Wait_Position, Non_Wait_Position, Fault, Non_Safety_Position, Safety_Position, InCommand, Fault_InCommand, Fault_Safety_Position and Fault_Non_Safety_Position	Log file of coreapp displays STATE_DES received. GWAPP sends M_STATE_ELEMENT with host ID, sequence number, state as new/removed, ELEMENT_CATEGORY OSD, EVENT_CATEGORY same as received in STATE_DES packet to the GWNG Virtualization simulator.
<b>step 6</b>	View the log file of GWNG Virtualization simulator	Log file displays M_STATE_ELEMENT packet received.

## 3.14.9 OSD Monitoring from STT to MC20

Step Name	Description	Expected
<b>step 1</b>	Power on GW20. Wait about 30 sec Run the STT20 Simulator first and then TestMaster	Log in to the board through PuTTY is possible.
<b>step 2</b>	Run the GWNG virtualization simulator.	Simulator waits until the GWAPP is started.
<b>step 3</b>	Run the coreapp application. Wait about 15 sec	GWAPP connects with the GWNG virtualization simulator. GWAPP sends M_SESSION_INIT to the GWNG Virtualization simulator and receives M_SESSION_CONFIRM. GWAPP sends M_CONNECTION_SUPERVISION to the GWNG Virtualization simulator.
<b>step 4</b>	Send M_STATE_CONNECTION packet from the GWNG Virtualization simulator	GWAPP receives M_STATE_CONNECTION packets and displays their connection states in the log file of coreapp and STT20 simulator. GWAPP sends M_SUBSCRIPTION_CREATE packet to the GWNG Virtualization simulator.
<b>step 5</b>	Send M_STATE_ELEMENT packet with element category C_ELEMENT_OSD and event category as any one of	Coreapp displays M_STATE_ELEMENT packet received in the log file Coreapp sends STATE_DESCRIPTOR packet to the STT20 simulator.

	OSD_STATES from GWNG Virtualization simulator	
<b>step 6</b>	View the log file of STT20 simulator	Log file displays STATE_DES packet received.

**3.14.10 OSD monitoring between two GW20s and STT20 simulators**

<b>Step Name</b>	<b>Description</b>	<b>Expected</b>
<b>step 1</b>	Power on both GW20 boards. Wait about 30 sec Run STT20 simulator and Test Master for both the boards	Login to the board through PuTTY is possible.
<b>step 2</b>	Run coreapp application on both the boards. Wait about 15 sec View the system log file.	GWAPP connects with GWNG Virtualization. GWAPP sends M_SESSION_INIT and receives M_SESSION_CONFIRM packet form GWNG Virtualization. GWAPP sends M_CONNECTION_SUPERVISION packet to the GWNG Virtualization. Both the coreapp applications start displaying connection states (connected or disconnected) of remote GW20s and FC20s every 20 sec. Both the GWAPP send M_SUSBCRIPTION_CREATE packet to GWNG Virtualization for GW20s and

		FC20s if their connection states are 'connected'
<b>step 3</b>	Send STATE_DES packet from local STT20 simulator with Sequence number of OSD	Local coreapp receives lock STATE_DES packet. Local GWAPP sends M_STATE_ELEMENT packet with host ID = Instance number of local GW20 and sequence number = sequence number of OSD as received in STATE_DES packet to the GWNG Virtualization Remote GWAPP receives M_STATE_ELEMENT packet from GWNG Virtualization with same host ID and sequence number as sent by local GWAPP. Remote GWAPP should send STATE_DES packet to remote STT20 simulator with sequence number same as sent by local STT20 simulator.
<b>step 4</b>	View log file of remote STT20 simulator.	Remote STT20 simulator should display STATE_DES packet received with the sequence number same as received from local STT20 simulator.

## **4. User Manual**

## **4.1 User Manual**

### **[1] Uploading Resource Package on Board**

- Login to Jenkins website and download latest Resource package.
- Import that Zip file into Tool then click on update CPU Board.
- Enter password and reboot the board
- After that upload site on that board.
- Reboot the board and GW20 application will become operational.

### **[2] Starting SAT Tool**

- Open SAT tool.
- Enter IP Address and Port Number
- Select which log you want to see.
- After selecting Logs start Logging screen.



## **4.2 Operations Manual**

### **[1] Upload Database**

- There is tool called as LP20 tool which is used to upload database (.xml) file to GW20 and to STT20 via GW20.
- Open that tool then select that file and then click on Telecharge so that file uploading will start.
- After that board will reboot and GW20 and STT20 will start communicating.

### **[2] Generate Configuration File**

- There is one tool called as F-FS20X which is use to create configuration file.
- In configuration file you can add Detector, Zones, OSDs, and Functions.
- After that you have to add Network related data like IP-Address, Port Number and other details.

### 4.3 Program Specification

#### ➤ Core Application Program

Serial No.	Program Name	Constraints	Description
1	Core Application	Packets received from STT20 and FC20 should be correct (Valid). Socket and Serial communication should be active.	This is the core part of the system as it communicates with each and every module

#### ➤ GW20 Program

Serial No.	Program Name	Constraints	Description
1	GW20	Socket Communication should be active. FCNet communication should be initialized.	This program communicates with FC20 and GW20 over FCNet protocol.

➤ **STT20 Program**

Serial No.	Program Name	Constraints	Description
1	GW20	Socket Communication should be active. FCNet communication should be initialized.	This program communicates with FC20 and GW20 over FCNet protocol.

➤ **LP20 Program**

Serial No.	Program Name	Constraints	Description
1	LP20	Socket communication should be active. Packets received from LP20 tool should be valid.	This program communicates with LP20 tool to download configuration file.

➤ **Technician**

Serial No.	Program Name	Constraints	Description
1	Technician	While uploading configuration file password should be correct.	This is the responsible person who creates the configuration file and uploads it to GW20 and STT20.

## **1. Drawbacks and Limitations**

- In current system we have Danger Management System (DMS) which works on BACNet. Because of this implementation we have to support both the protocols BACNet and FCNet.
- FS20 and GW20 using different panels and because of that field work is extra over head.

## **2. Proposed Enhancements**

- Danger Management System (DMS) will be moved from BACNet protocol to FCNet protocol
- Instead of using different panels for FS20 system and GW20 system use one panel and keep all other requirements as it is.

### **3. Conclusions**

As the current system implemented is working as per the mark and all the requirements are fulfilled.

FCNet and BACNet protocol can work with each other but if that BACNet part is moved to FCNet then it will also work properly.

In future we can cover all the drawbacks and we can implement proposed system so that performance of this system will increase.

## 4. Bibliography

- **Books:-**

1. Linux Programming
2. BACNet: The Global Standard for Building Automation and Control Networks

- **Linux MAN Page**

- **Web biography**

[www.javatpoint.com](http://www.javatpoint.com)

<http://www.tutorialspoint.com>

[www.stackoverflow.com](http://www.stackoverflow.com)



## **5. ANNEXURES**

# [1] ANNEXURES 1: User Interface

- Create New Project

Create new site

General Comment History

BDV: F-FXS2004-US\_en\_1-60.20.1\_01.eBDV, Standard BDV

File name: \* C:\Siemens\F-FX2030\US\_en\_1-V7.0\Sites\CCL4\_Testing Browse

Site name: \* CCL4\_Testing

Site no.:

Site ID:

Metadata version: 60.20.1

Metadata version integrated Voice: 60.17.0

Voice audio library version: 60.8.0

Creation date: 2017-03-28T17:08:15

Date of last transfer to panel(s):

Creator

First name:

Last name: \* Sharad Wagh

Function:

Organisation:

Sales region:

Company:

Zip code:

City:

Country:

Phone no.:

License no.:

E-mail: sharad.wagh@siemens.com

Customer

Name:

Street:

Street no.:

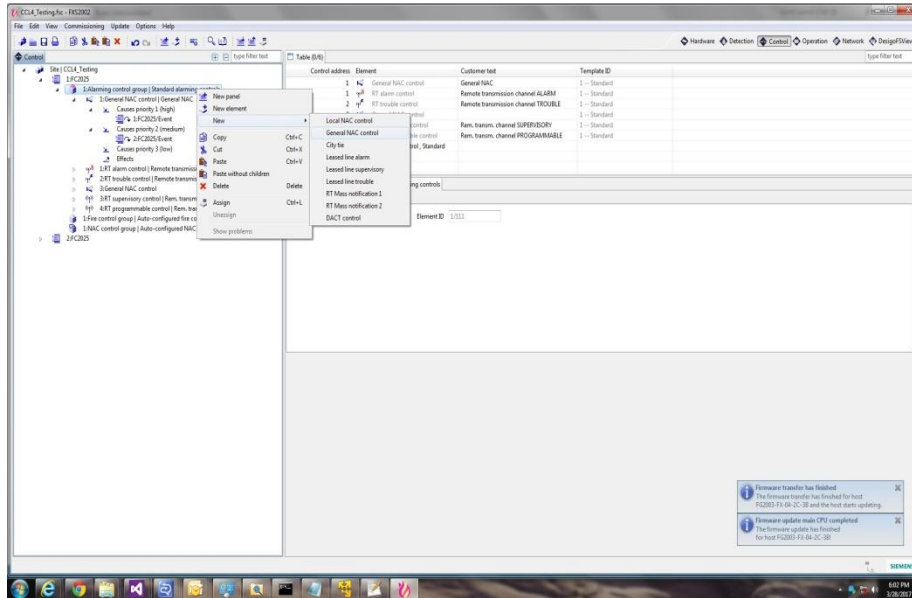
Zip code:

City:

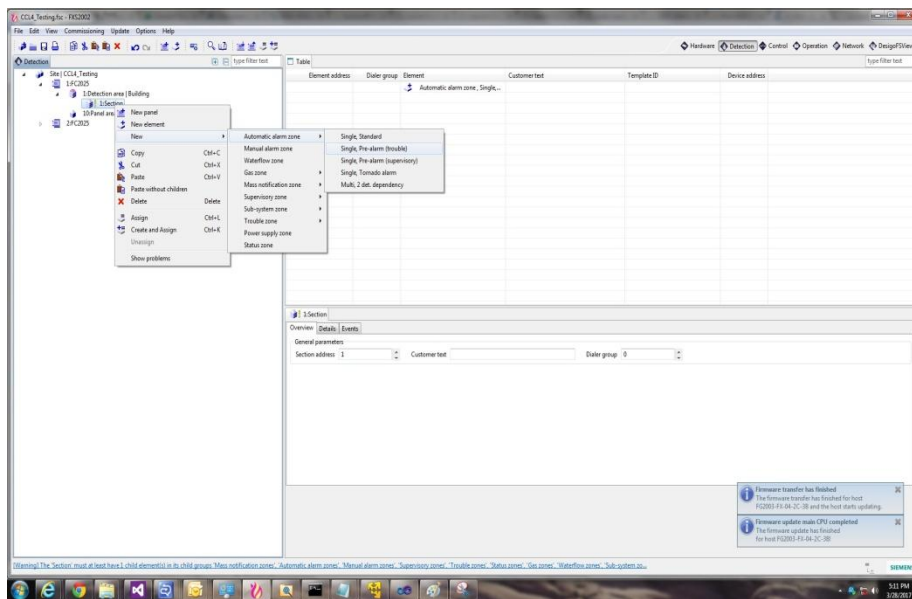
Clear all creator values Set creator values as default

Create Cancel

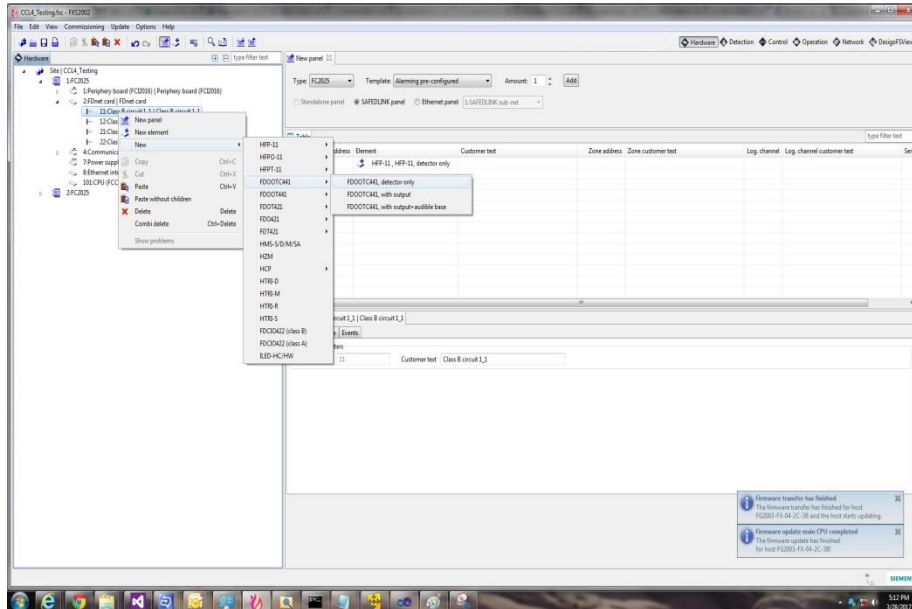
- Add New Sector



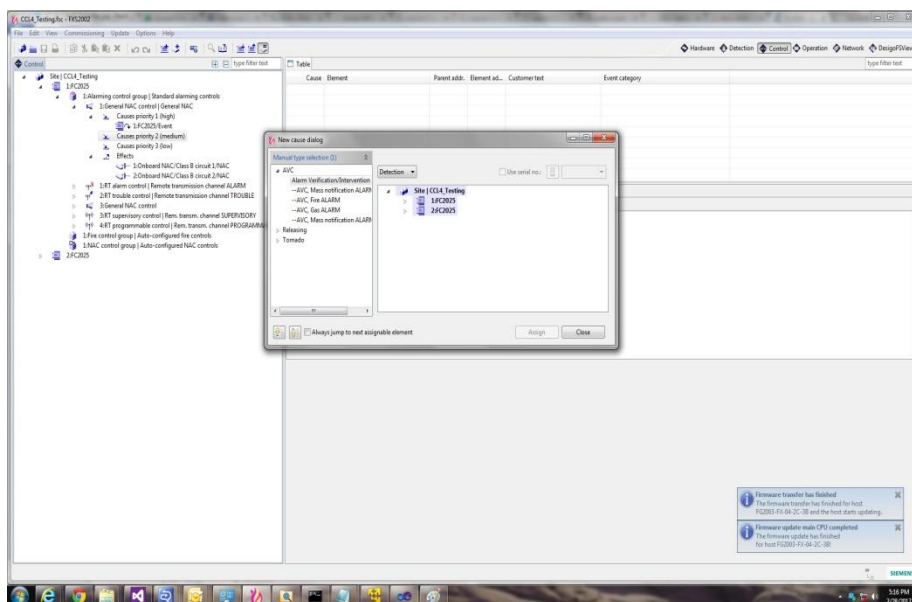
- Add New Zone



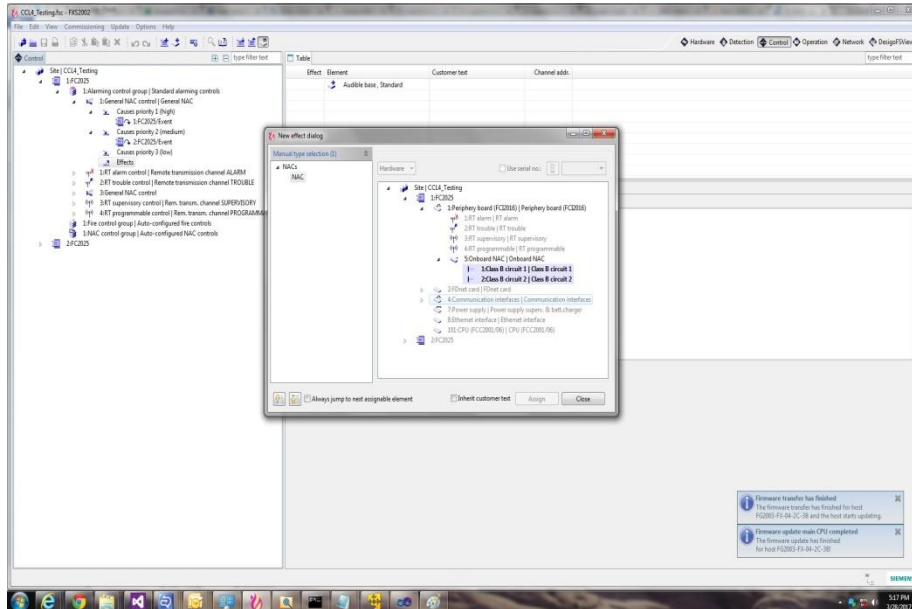
- Add New Detector



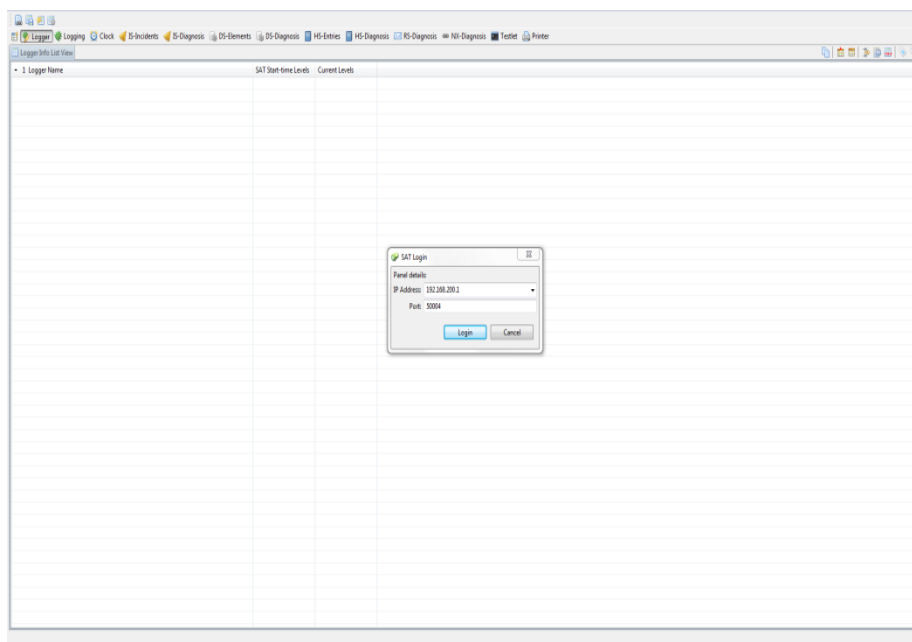
- Add New Function



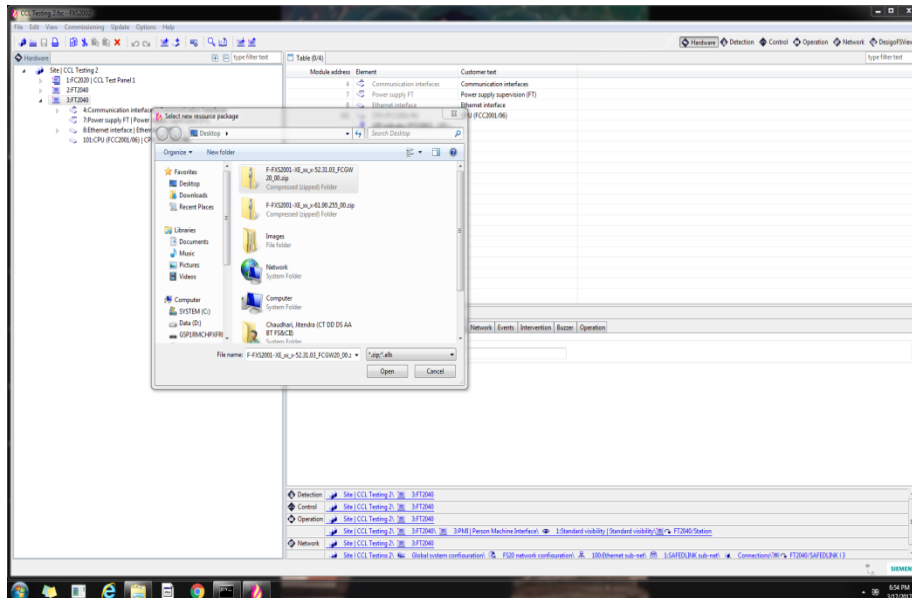
- Add New Action



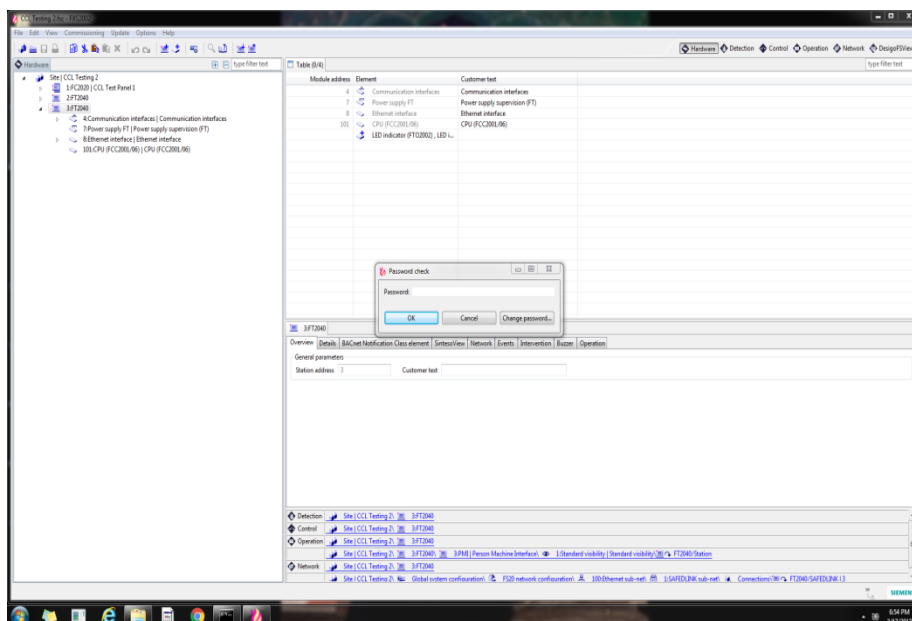
- SAT Login



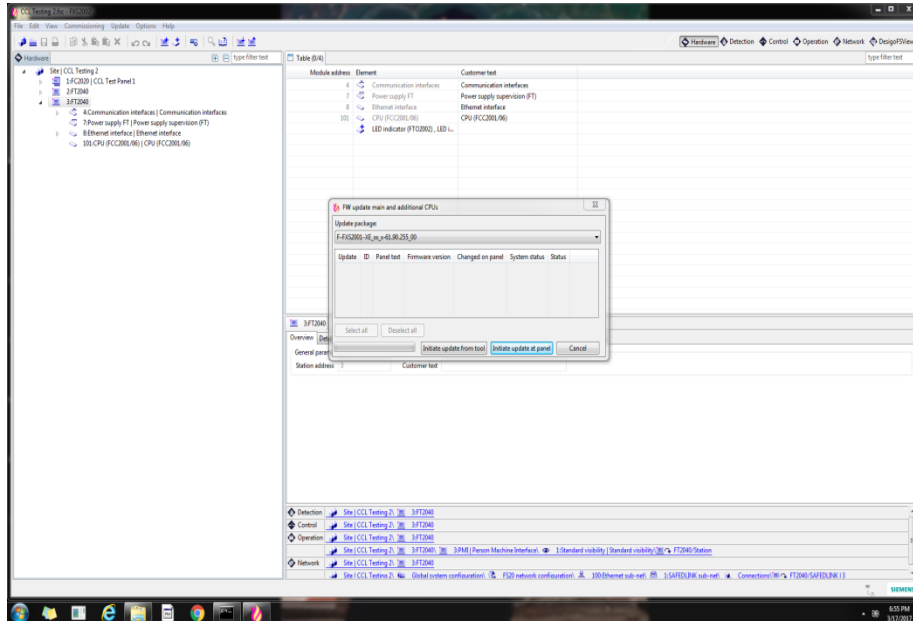
- **Import Package**



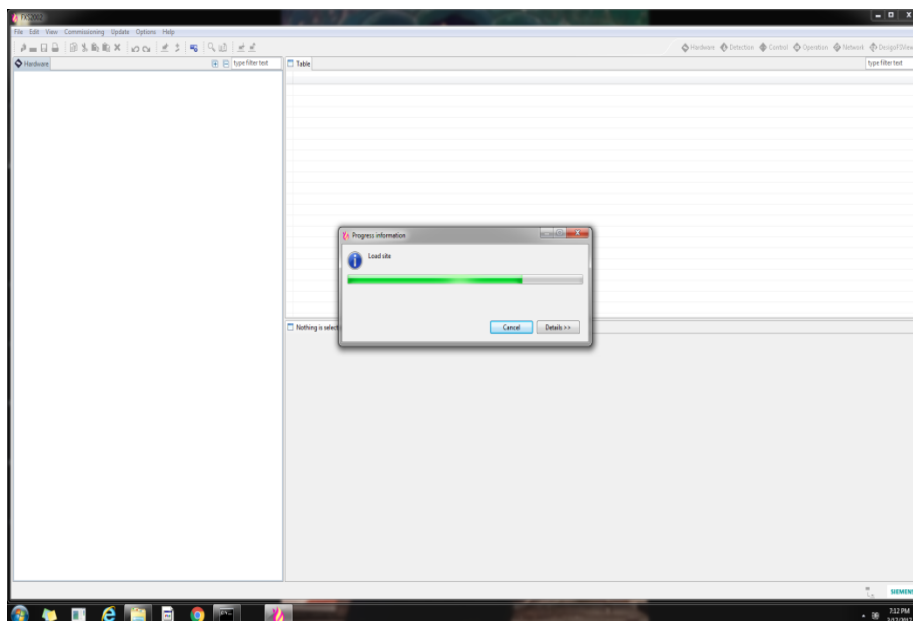
- **Enter Password**



- **Initiate Update**



- **Site Loading**



- Packet Transmission

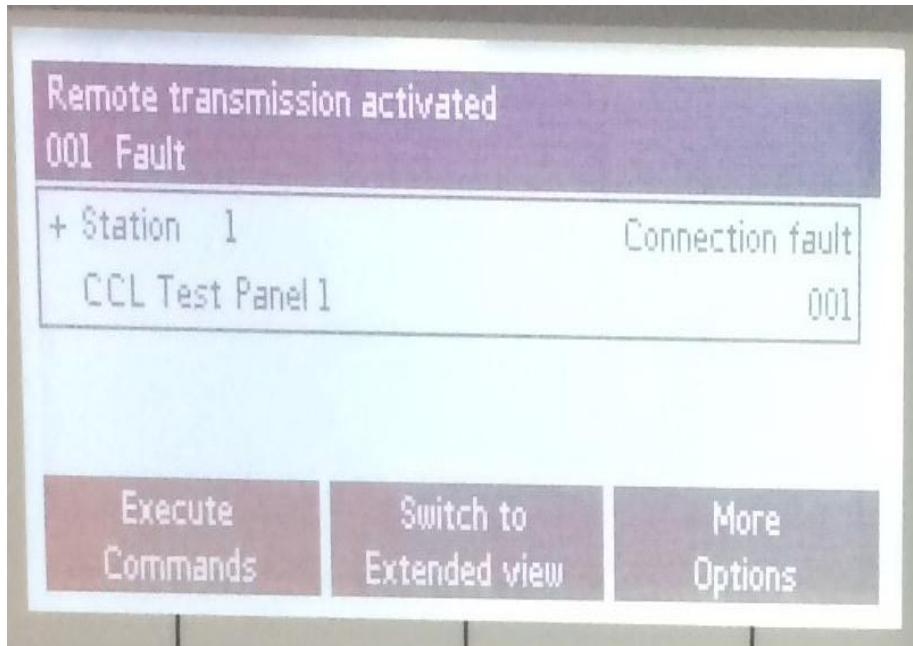
Sequence	TimeStamp	Thread	Level	Class	Method	Message
107	2013-01-29 14:00:14:808	com.siemens.fsp.fsp.g...	DEBUG	Connection	message	Received M_STATE_ELEMENT_changeNodeMER_hotd62, sequenceNumber30, ngfElementCategory_C_ELEMENT_LOCK, ngfEventCategory_C_EVENT_ACTIVE
108	2013-01-29 14:00:20:448	com.siemens.fsp.fsp.g...	DEBUG	Connection	message	Sent M_SESSION_LIFE_SIGN, unserialid=1
109	2013-01-29 14:00:29:833	com.siemens.fsp.fsp.g...	DEBUG	Connection	message	Received M_STATE_ELEMENT_changeNodeREMOVED_hotd62, sequenceNumber30, ngfElementCategory_C_ELEMENT_LOCK, ngfEventCategory_C_EVENT_ACTIVE
110	2013-01-29 14:00:29:837	com.siemens.fsp.fsp.g...	DEBUG	LockSTZConnector	removeIncident	Incident removed IncidentId: 2/2/3007, EventFeet: STZD-Lock-30, EventCategory: ACTIVE, ProcessingRate: 6346, ElementId: itemType: LockSTZDConnectorDem...

- Select Type of Message

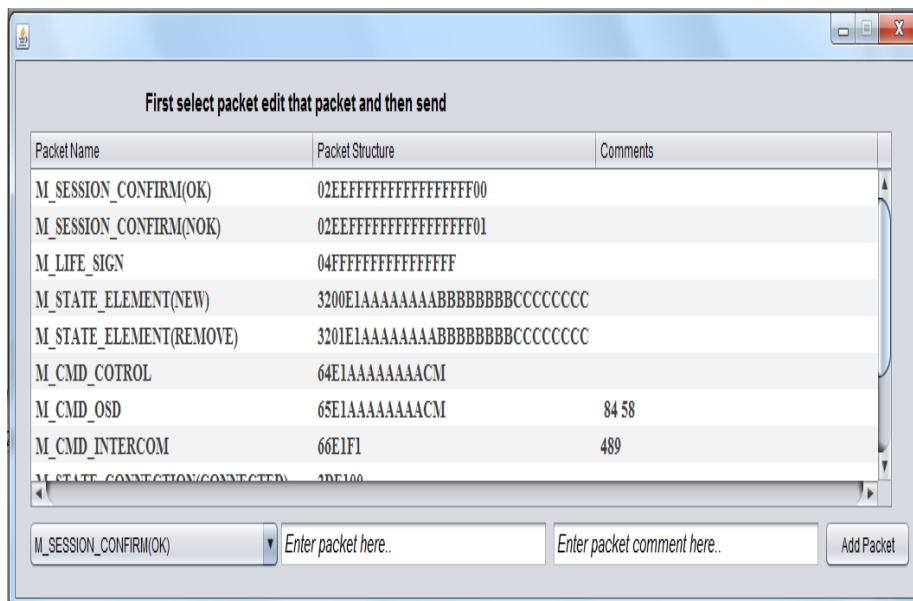
Logger Name	S4T Start-Time Levels	Current Levels
com.siemens.fsp.fsp.g...	WARN	WARN
com.siemens.fsp.fsp.s...	WARN	WARN
com.siemens.fsp.fsp.l...	WARN	WARN
com.siemens.fsp.fsp.u...	WARN	WARN
com.siemens.fsp.fsp.m...	WARN	WARN
com.siemens.fsp.fsp.a...	WARN	WARN
com.siemens.fsp.fsp.i...	WARN	WARN
com.siemens.fsp.fsp.o...	WARN	WARN
com.siemens.fsp.fsp.e...	WARN	WARN
com.siemens.fsp.fsp.c...	WARN	WARN
com.siemens.fsp.fsp.p...	WARN	WARN
com.siemens.fsp.fsp.d...	WARN	WARN
com.siemens.fsp.fsp.f...	WARN	WARN
com.siemens.fsp.fsp.t...	WARN	WARN
com.siemens.fsp.fsp.h...	WARN	WARN
com.siemens.fsp.fsp.n...	WARN	WARN
com.siemens.fsp.fsp.r...	WARN	WARN
com.siemens.fsp.fsp.w...	WARN	WARN
com.siemens.fsp.fsp.m...	WARN	WARN
com.siemens.fsp.fsp.s...	WARN	WARN
com.siemens.fsp.fsp.l...	WARN	WARN
com.siemens.fsp.fsp.u...	WARN	WARN
com.siemens.fsp.fsp.m...	WARN	WARN
com.siemens.fsp.fsp.a...	WARN	WARN
com.siemens.fsp.fsp.i...	WARN	WARN
com.siemens.fsp.fsp.o...	WARN	WARN
com.siemens.fsp.fsp.e...	WARN	WARN
com.siemens.fsp.fsp.c...	WARN	WARN
com.siemens.fsp.fsp.p...	WARN	WARN
com.siemens.fsp.fsp.d...	WARN	WARN
com.siemens.fsp.fsp.f...	WARN	WARN
com.siemens.fsp.fsp.t...	WARN	WARN
com.siemens.fsp.fsp.h...	WARN	WARN
com.siemens.fsp.fsp.n...	WARN	WARN
com.siemens.fsp.fsp.r...	WARN	WARN
com.siemens.fsp.fsp.w...	WARN	WARN



- **LCD Display**



- **Packet Generator**

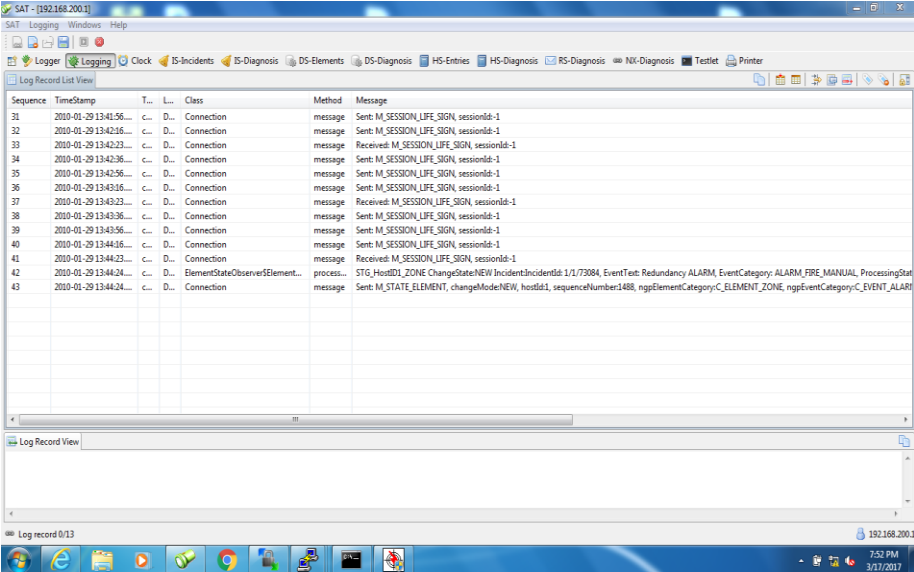


- **GW20 Board LED Bulb Indicator**



## [2] ANNEXURES 2: Reports

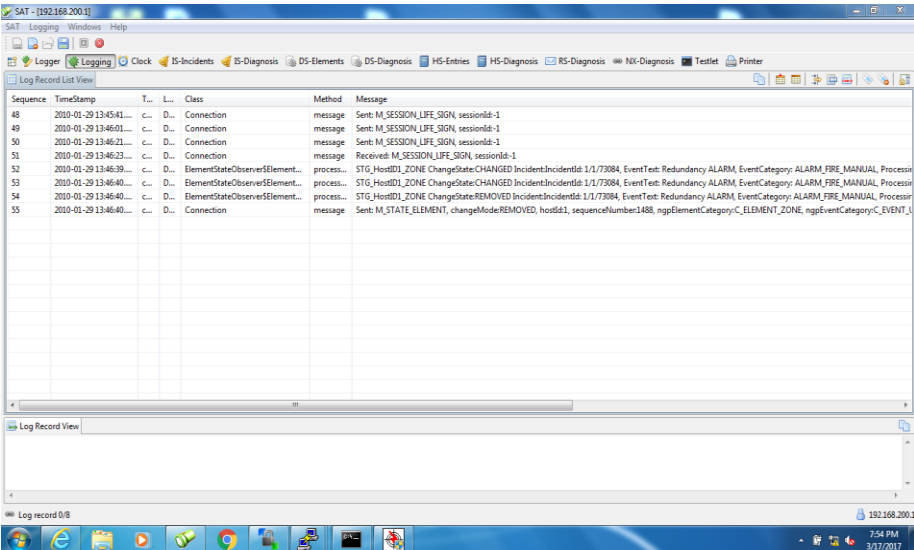
- Alarm Activated



The screenshot shows the SAT Logging application interface. The 'Log Record List View' is active, displaying a table of log entries. The entry at sequence 42 is highlighted, indicating an alarm activation event.

Sequence	TimeStamp	T...	L...	Class	Method	Message
31	2010-01-29 13:41:56...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
32	2010-01-29 13:42:16...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
33	2010-01-29 13:42:23...	c...	D...	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
34	2010-01-29 13:42:36...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
35	2010-01-29 13:42:56...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
36	2010-01-29 13:43:16...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
37	2010-01-29 13:43:23...	c...	D...	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
38	2010-01-29 13:43:36...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
39	2010-01-29 13:43:56...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
40	2010-01-29 13:44:16...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
41	2010-01-29 13:44:23...	c...	D...	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
42	2010-01-29 13:44:24...	c...	D...	ElementStateObserverElement...	process...	STG_HostD1_ZONE ChangeState:NEW IncidentIncidentId: 1/1/73084, EventText: Redundancy ALARM, EventCategory: ALARM_FIRE_MANUAL, ProcessingState...
43	2010-01-29 13:44:24...	c...	D...	Connection	message	Sent: M_STATE_ELEMENT, changeMode:NEW, hostId1, sequenceNumber:1488, ngsElementCategory:C_ELEMENT_ZONE, ngsEventCategory:C_EVENT_ALARM...

- Alarm Deactivate



The screenshot shows the SAT Logging application interface. The 'Log Record List View' is active, displaying a table of log entries. The entry at sequence 52 is highlighted, indicating an alarm deactivation event.

Sequence	TimeStamp	T...	L...	Class	Method	Message
48	2010-01-29 13:45:41...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
49	2010-01-29 13:46:01...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
50	2010-01-29 13:46:21...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
51	2010-01-29 13:46:23...	c...	D...	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
52	2010-01-29 13:46:39...	c...	D...	ElementStateObserverElement...	process...	STG_HostD1_ZONE ChangeState:CHANGED IncidentIncidentId: 1/1/73084, EventText: Redundancy ALARM, EventCategory: ALARM_FIRE_MANUAL, ProcessingState...
53	2010-01-29 13:46:40...	c...	D...	ElementStateObserverElement...	process...	STG_HostD1_ZONE ChangeState:CHANGED IncidentIncidentId: 1/1/73084, EventText: Redundancy ALARM, EventCategory: ALARM_FIRE_MANUAL, ProcessingState...
54	2010-01-29 13:46:40...	c...	D...	ElementStateObserverElement...	process...	STG_HostD1_ZONE ChangeState:REMOVED IncidentIncidentId: 1/1/73084, EventText: Redundancy ALARM, EventCategory: ALARM_FIRE_MANUAL, ProcessingState...
55	2010-01-29 13:46:40...	c...	D...	Connection	message	Sent: M_STATE_ELEMENT, changeMode:REMOVED, hostId1, sequenceNumber:1488, ngsElementCategory:C_ELEMENT_ZONE, ngsEventCategory:C_EVENT_ALARM...

- Life Sign

Sequence	TimeStamp	Thread	Level	Class	Method	Message
138	2010-02-19 14:05:40.489	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
139	2010-02-19 14:05:59.756	com.siemens.fsp.05-p-	DEBUG	Connection	message	Received M_SESSION_LIFE_SIGN, sessionId=1
140	2010-02-19 14:06:06.489	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
141	2010-02-19 14:06:26.489	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
142	2010-02-19 14:06:46.479	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
143	2010-02-19 14:06:56.543	com.siemens.fsp.05-p-	DEBUG	Connection	message	Received M_SESSION_LIFE_SIGN, sessionId=1
144	2010-02-19 14:07:06.471	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
145	2010-02-19 14:07:26.472	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
146	2010-02-19 14:07:46.472	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
147	2010-02-19 14:07:56.539	com.siemens.fsp.05-p-	DEBUG	Connection	message	Received M_SESSION_LIFE_SIGN, sessionId=1
148	2010-02-19 14:08:06.473	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
149	2010-02-19 14:08:26.473	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
150	2010-02-19 14:08:46.474	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
151	2010-02-19 14:08:56.518	com.siemens.fsp.05-p-	DEBUG	Connection	message	Received M_SESSION_LIFE_SIGN, sessionId=1
152	2010-02-19 14:09:06.476	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
153	2010-02-19 14:09:26.477	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
154	2010-02-19 14:09:46.478	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
155	2010-02-19 14:09:56.534	com.siemens.fsp.05-p-	DEBUG	Connection	message	Received M_SESSION_LIFE_SIGN, sessionId=1

- Command Protection device Activate

Sequence	TimeStamp	Thread	Level	Class	Method	Message
93	2010-02-19 13:57:26.437	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
94	2010-02-19 13:57:46.438	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
95	2010-02-19 13:58:06.441	com.siemens.fsp.05-p-	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
96	2010-02-19 13:58:26.471	com.siemens.fsp.05-p-	DEBUG	Connection	message	Received M_CMD_CommandTypeM_CMD_Type_CSD, hostId=1, sequenceNumber=1, command_C_COMMAND_ACTIVATE, detectionPareHostId=9
97	2010-02-19 13:58:46.433	com.siemens.fsp.05-p-	DEBUG	RequestPublisher	win	COMPLETED request:14087

- Command Protection device Deactivate

Sequence	TimeStamp	Thread	Level	Class	Method	Message
93	2010-05-20 11:51:30.437	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
94	2010-05-20 11:51:30.438	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
95	2010-05-20 11:50:06.141	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
96	2010-05-20 11:50:07.721	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
97	2010-05-20 11:50:08.110	com.jametes.hp.OS.p...	DEBUG	RequestPublisher	run	COMPLETED request 24009
98	2010-05-20 11:50:28.142	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
99	2010-05-20 11:50:27.142	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_CMD_CommandTypeM_CMD_TYPE_OSD, hostId1, sequenceNumber=3038, command_C_COMMAND_REACTIVATE_detectorPanelHostId
100	2010-05-20 11:50:27.813	com.jametes.hp.OS.p...	DEBUG	RequestPublisher	run	COMPLETED request 24009
101	2010-05-20 11:50:46.444	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1

- Intercom command

Sequence	TimeStamp	Thread	Level	Class	Method	Message
57	2010-05-20 11:52:09.636	com.jametes.hp.OS.p...	WARN	ConnectionDataConnects	receiveSessionInit	Unexpected call
58	2010-05-20 11:52:01.139	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_CONNECTION_SUPERVISION_startSuperviseIntruder
59	2010-05-20 11:52:01.143	com.jametes.hp.OS.p...	DEBUG	NetworkObserver	startObserving	Send M_CREATE_CONNECTION, hostId1, connectionState=CONNECTED
60	2010-05-20 11:52:01.145	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_STATE_CONNECTION, hostId1, connectionState=CONNECTED
61	2010-05-20 11:52:01.139	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_STATE_CONNECTION, hostId1, connectionState=CONNECTED
62	2010-05-20 11:52:01.146	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_DESCRIPTION_CREATE, hostId1, ngElementCategory_C_ELEMENT_ZONE, ngEventCategory_C_EVENT_ALARM, sequenceNumbers
63	2010-05-20 11:52:01.174	com.jametes.hp.OS.p...	DEBUG	ElementStateObserver	createSubscription	IncidentSubscription(1)1, hostId1_ZONE
64	2010-05-20 11:52:01.212	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_DESCRIPTION_CREATE, hostId1, hostId2, sequenceNumber=8, ngElementCategory_C_ELEMENT_ZONE, ngEventCategory_C_EVENT_UNLOCK, sequenceNumbers
65	2010-05-20 11:52:01.203	com.jametes.hp.OS.p...	DEBUG	ElementStateObserver	createSubscription	IncidentSubscription(1)1, hostId1_LOCK
66	2010-05-20 11:52:01.107	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_DESCRIPTION_CREATE, hostId1, ngElementCategory_C_ELEMENT_ZONE, ngEventCategory_C_EVENT_UNLOCK, sequenceNumbers
67	2010-05-20 11:52:01.252	com.jametes.hp.OS.p...	DEBUG	ElementStateObserver	createSubscription	IncidentSubscription(1)1, hostId1_ZONE
68	2010-05-20 11:52:04.162	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_STATE_ELEMENT_changeOfStateREMOVED, hostId1, sequenceNumber=8, ngElementCategory_C_ELEMENT_ZONE, ngEventCategory_C_EVENT_STTID_OSD_NON, Incident for OSD-8 created. Incident IncidentId= 2-2-2006, EventFeat: STTID-OSD-8, EventCategory: EVENT_STTID_OSD_NON, Safety_Position, ProcessingState
69	2010-05-20 11:52:04.193	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_STATE_ELEMENT_changeOfStateREMOVED, hostId1, sequenceNumber=8, ngElementCategory_C_ELEMENT_ZONE, ngEventCategory_C_EVENT_STTID_OSD_NON, Incident removed IncidentId= 2-2-2006, EventFeat: STTID-OSD-8, EventCategory: EVENT_STTID_OSD_NON, Safety_Position, ProcessingState: SOME, EventMetaData: jean
70	2010-05-20 11:52:29.195	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
71	2010-05-20 11:52:29.192	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
72	2010-05-20 11:52:29.403	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
73	2010-05-20 11:52:40.404	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
74	2010-05-20 11:52:40.405	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
75	2010-05-20 11:52:19.862	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_SESSION_LIFE_SIGN, sessionId=1
76	2010-05-20 11:52:29.408	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
77	2010-05-20 11:52:40.409	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
78	2010-05-20 11:52:40.409	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
79	2010-05-20 11:52:40.410	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
80	2010-05-20 11:52:40.411	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
81	2010-05-20 11:52:40.412	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
82	2010-05-20 11:52:09.612	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
83	2010-05-20 11:52:09.708	com.jametes.hp.OS.p...	DEBUG	Connection	message	Received M_SESSION_LIFE_SIGN, sessionId=1
84	2010-05-20 11:52:29.413	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
85	2010-05-20 11:52:40.415	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
86	2010-05-20 11:52:06.636	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
87	2010-05-20 11:52:06.636	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
88	2010-05-20 11:52:06.636	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
89	2010-05-20 11:52:06.636	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1
90	2010-05-20 11:52:46.406	com.jametes.hp.OS.p...	DEBUG	Connection	message	Send M_SESSION_LIFE_SIGN, sessionId=1

- Protection device monitoring

Sequence	TimeStamp	T.	L.	Class	Method	Message
69	2010-01-29 13:50:15...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
70	2010-01-29 13:50:22...	c.	D.	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
71	2010-01-29 13:50:36...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
72	2010-01-29 13:50:56...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
73	2010-01-29 13:51:16...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
74	2010-01-29 13:51:23...	c.	D.	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
75	2010-01-29 13:51:36...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
76	2010-01-29 13:51:48...	c.	D.	Connection	message	Sent: M_STATE_CONNECTION, hostId=2, connectionState=DISCONNECTED
77	2010-01-29 13:52:01...	c.	D.	Connection	message	Sent: M_STATE_CONNECTION, hostId=2, connectionState=CONNECTED
78	2010-01-29 13:52:01...	c.	D.	Connection	message	Received: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_LOCK, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
79	2010-01-29 13:52:01...	c.	L.	ElementStateObserver	create...	Subscription already exists: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_LOCK, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
80	2010-01-29 13:52:01...	c.	D.	Connection	message	Subscription already exists: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_OSD, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
81	2010-01-29 13:52:01...	c.	L.	ElementStateObserver	create...	Subscription already exists: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_OSD, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
82	2010-01-29 13:52:06...	c.	D.	ElementStateObserverElement...	process...	STG_HostID2_OUTPUT ChangeStateNEW Incident:incidentId= 2/2/2006, EventText: STT20-OSD 9, EventCategory: EVENT_STT20_OSD_NON_SAFETY_POSITION
83	2010-01-29 13:52:06...	c.	D.	Connection	message	Sent: M_STATE_ELEMENT, changeMode=NEW, hostId=2, sequenceNumber=9, nggElementCategory=C_ELEMENT_OSD, nggEventCategory=C_EVENT_STT20_OSD
84	2010-01-29 13:52:20...	c.	D.	ElementStateObserverElement...	process...	STG_HostID2_OUTPUT ChangeStateREMOVED Incident:incidentId= 2/2/2006, EventText: STT20-OSD 9, EventCategory: EVENT_STT20_OSD_NON_SAFETY_POSITION
85	2010-01-29 13:52:20...	c.	D.	Connection	message	Sent: M_STATE_ELEMENT, changeMode=REMOVED, hostId=2, sequenceNumber=9, nggElementCategory=C_ELEMENT_OSD, nggEventCategory=C_EVENT_UNDEFINED
86	2010-01-29 13:52:23...	c.	D.	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
87	2010-01-29 13:52:36...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1

- Device Connection State

Sequence	TimeStamp	T.	L.	Class	Method	Message
69	2010-01-29 13:50:15...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
70	2010-01-29 13:50:22...	c.	D.	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
71	2010-01-29 13:50:36...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
72	2010-01-29 13:50:56...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
73	2010-01-29 13:51:16...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
74	2010-01-29 13:51:23...	c.	D.	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
75	2010-01-29 13:51:36...	c.	D.	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
76	2010-01-29 13:51:48...	c.	D.	Connection	message	Sent: M_STATE_CONNECTION, hostId=2, connectionState=DISCONNECTED
77	2010-01-29 13:52:01...	c.	D.	Connection	message	Sent: M_STATE_CONNECTION, hostId=2, connectionState=CONNECTED
78	2010-01-29 13:52:01...	c.	D.	Connection	message	Received: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_LOCK, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
79	2010-01-29 13:52:01...	c.	L.	ElementStateObserver	create...	Subscription already exists: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_LOCK, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
80	2010-01-29 13:52:01...	c.	D.	Connection	message	Subscription already exists: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_OSD, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
81	2010-01-29 13:52:01...	c.	L.	ElementStateObserver	create...	Subscription already exists: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_OSD, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:

## • Device Subscription

The screenshot shows the SAT Logging application interface. The 'Log Record List View' is active, displaying a table of log entries. The table has columns for Sequence, TimeStamp, T..., L..., Class, Method, and Message. The log entries show a sequence of connection and message events related to a device subscription process.

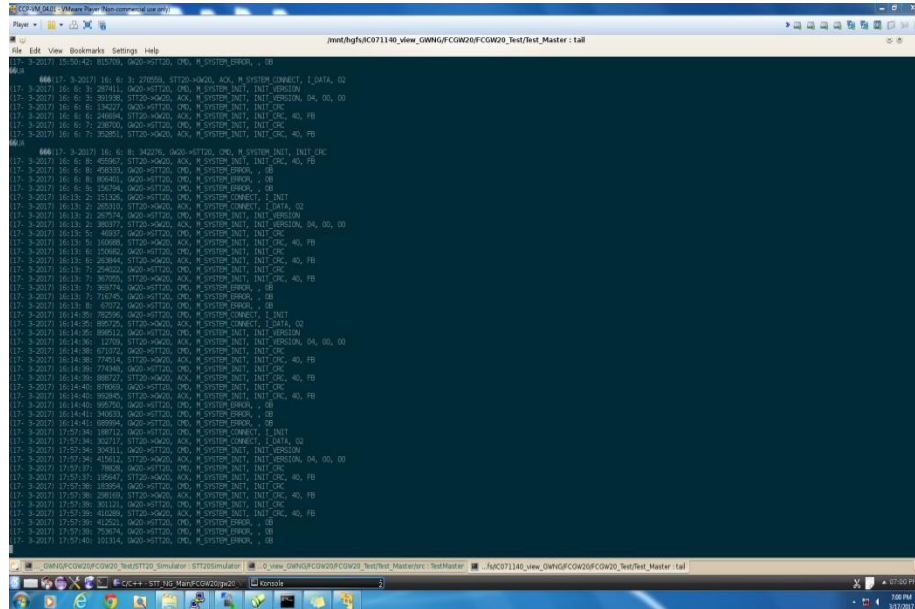
Sequence	TimeStamp	T...	L...	Class	Method	Message
69	2010-01-29 13:50:15...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
70	2010-01-29 13:50:22...	c...	D...	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
71	2010-01-29 13:50:36...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
72	2010-01-29 13:51:06...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
73	2010-01-29 13:51:16...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
74	2010-01-29 13:51:23...	c...	D...	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
75	2010-01-29 13:51:36...	c...	D...	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
76	2010-01-29 13:51:48...	c...	D...	Connection	message	Sent: M_STATE_CONNECTION, hostId=2, connectionState=DISCONNECTED
77	2010-01-29 13:52:01...	c...	D...	Connection	message	Sent: M_STATE_CONNECTION, hostId=2, connectionState=CONNECTED
78	2010-01-29 13:52:01...	c...	D...	Connection	message	Received: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_LOCK, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
79	2010-01-29 13:52:01...	c...	L...	ElementStateObserver	createE...	Subscription already exists: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_LOCK, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
80	2010-01-29 13:52:01...	c...	D...	Connection	message	Received: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_OSD, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:
81	2010-01-29 13:52:01...	c...	L...	ElementStateObserver	createE...	Subscription already exists: M_SUBSCRIPTION_CREATE, hostId=2, nggElementCategory=C_ELEMENT_OSD, nggEventCategory=C_EVENT_UNDEFINED, sequenceNumbers:

## • Function Commanding

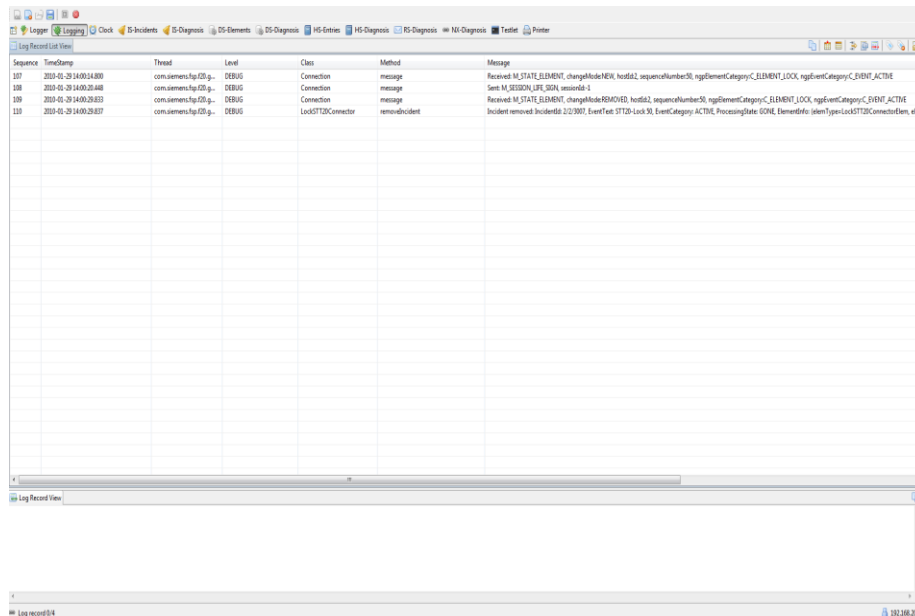
The screenshot shows the SAT Logging application interface. The 'Log Record List View' is active, displaying a table of log entries. The table has columns for Sequence, TimeStamp, Thread, Level, Class, Method, and Message. The log entries show a sequence of connection and message events related to function commanding, including state changes and command execution.

Sequence	TimeStamp	Thread	Level	Class	Method	Message
115	2010-02-19 14:12:38.285	com.siemens.tp.03.p...	DEBUG	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
116	2010-02-19 14:12:38.440	com.siemens.tp.03.p...	DEBUG	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
117	2010-02-19 14:12:48.817	com.siemens.tp.03.p...	DEBUG	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
118	2010-02-19 14:12:54.020	com.siemens.tp.03.p...	DEBUG	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
119	2010-02-19 14:12:58.844	com.siemens.tp.03.p...	DEBUG	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
120	2010-02-19 14:13:14.452	com.siemens.tp.03.p...	DEBUG	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
121	2010-02-19 14:13:14.870	com.siemens.tp.03.p...	DEBUG	Connection	message	Received: M_SESSION_LIFE_SIGN, sessionId=1
122	2010-02-19 14:13:21.274	com.siemens.tp.03.p...	DEBUG	Connection	message	Received: M_STATE_ELEMENT_CHANGE, changeMode=REMOVED, hostId=1, sequenceNumber=0, nggElementCategory=C_ELEMENT_LOCK, nggEventCategory=C_EVENT_ACTIVE
123	2010-02-19 14:13:31.404	com.siemens.tp.03.p...	DEBUG	Connection	message	Received: M_STATE_ELEMENT_CHANGE, changeMode=REMOVED, hostId=2, sequenceNumber=2, nggElementCategory=C_ELEMENT_LOCK, nggEventCategory=C_EVENT_ACTIVE
124	2010-02-19 14:13:38.404	com.siemens.tp.03.p...	DEBUG	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
125	2010-02-19 14:13:40.429	com.siemens.tp.03.p...	DEBUG	Connection	message	Received: M_CMD_CommandTypeM_CMD_TYPE_CONTROL_STD, hostId=1, sequenceNumber=2069, command=COMMAND_ACTIVATE, detectorPanel=hostId=1
126	2010-02-19 14:13:46.840	com.siemens.tp.03.p...	DEBUG	RequestPublisher	run	NEGATIVE request 21011
127	2010-02-19 14:13:54.058	com.siemens.tp.03.p...	DEBUG	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1
128	2010-02-19 14:14:05.459	com.siemens.tp.03.p...	DEBUG	Connection	message	Received: M_CMD_CommandTypeM_CMD_TYPE_CONTROL_TECHNICAL, hostId=1, sequenceNumber=3, command=COMMAND_ACTIVATE, detectorPanel=hostId=1
129	2010-02-19 14:14:05.473	com.siemens.tp.03.p...	DEBUG	RequestPublisher	run	NEGATIVE request 21012
130	2010-02-19 14:14:14.456	com.siemens.tp.03.p...	DEBUG	Connection	message	Sent: M_SESSION_LIFE_SIGN, sessionId=1

## • STT20 Simulator



## • Function Lock Commanding





### [3] ANNEXURES 3: Sample Program Code

- **Array Of Function Pointer**

```
void (*processFCNet_Packets_ForRx[5])(GW_APP_MSG_CREATE_TYPE)=
{
    Received_Session_Lifesign_Packet,
    Received_Element_State_Packet,
    Received_Connection_State_Packet,
    Received_Command_Packet,
    Received_Session_Confirm_Packet
};
```

- **Create Message Queue**

```
int Create_GWapp_Rx_Message_Queue()
{
    int retVal = 0;

    msgQidGwappRx = Create_Message_Queue('R');
    if(-1 == msgQidGwappRx)
    {
        MSG_ERR("GWAPP CORE: Couldn't create Message queue for GwApp-RX: %m");
        retVal = -1;
    }

    else
    {
        //CORE_INFO("Created GWApp Rx message Queue with ID : %d", msgQidGwappRx);
    }
    return retVal;
}
```

- **Create Timer**

```
bool Create_Timer(timer_t *time_id, void (*function)(union signal))
{
    //Create a timer with the specified callback function.
    bool status = true;
    struct sigevent sevent;
    int result;

    sevent.sigev_notify = SIGEV_THREAD;
    sevent.sigev_value.sival_ptr = (void *)time_id;
    sevent.sigev_notify_function = function;
    sevent.sigev_notify_attributes = NULL;
    result = timer_create(CLOCK_REALTIME, &sevent, time_id);
    if(result == -1)
    {
        MSG_ERR("CORE Main: Couldn't create timer: %m");
        setCoreState(CORE_STATE_STARTUP_ERROR);
        status = false;
    }
    return status;
}
```

- **Get Id Function**

```
void Get_Device_NumeroMC_From_InstanceNumber(unsigned char instanceNumber, unsigned char *mcNumber)
{
    int FcnetDeviceIndex;
    // following function will check instance number is present in FcArray or not
    // if instance number is found then it returns index of that instance number.
    FcnetDeviceIndex = Get_FCnet_Device_Position_In_List(((int)instanceNumber));
    if(0 <= FcnetDeviceIndex)
    {
        *mcNumber = FcnetData.FCnetDeviceDataObj[FcnetDeviceIndex].FCnetDeviceSubsetDataObj.numeroMC;
    }
}
```

- Hash Defines

```
//STT20 events
#define C_EVENT_STT20 99
#define C_EVENT_STT20_OSD 199
#define C_EVENT_UNDEFINED 0
#define C_EVENT_STT20_OSD_WAIT_POSITION 10199
#define C_EVENT_STT20_OSD_NON_WAIT_POSITION 20199
#define C_EVENT_STT20_OSD_NON_SAFETY_POSITION 30199
#define C_EVENT_STT20_OSD_SAFETY_POSITION 40199
#define C_EVENT_STT20_OSD_INCOMMAND 50199
#define C_EVENT_STT20_OSD_FAULT 60199
#define C_EVENT_STT20_OSD_FAULT_INCOMMAND 70199
#define C_EVENT_STT20_OSD_FAULT_SAFETY_POSITION 80199
#define C_EVENT_STT20_OSD_FAULT_NON_SAFETY_POSITION 90199
```

- Get Message Queue Id

```
int Get_Message_Queue_Id(char queue)
{
    if(queue == 'C')
        return msgQidCoreappCmd;

    else if(queue == 'R')
        return msgQidGwappRx;

    else if(queue == 'G')
        return msgQidGwappTx;

    else if(queue == 'T')
        return msgQidSDNTx;

    else if(queue == 'N')
        return msgQidCoreapp;

    else
        return -1;
}
```

- **Set Timer**

```
bool Set_Timer(timer_t time_id, int timeInSec)
{
    bool status = true;
    int result;
    struct itimerspec ts;

    ts.it_value.tv_sec = timeInSec;
    ts.it_value.tv_nsec = 0;
    ts.it_interval = ts.it_value;

    //Start the timer
    result = timer_settime(time_id, 0, &ts, 0);
    if(result == -1)
    {
        MSG_ERR("CORE Main: Set_Timer failed: %m");
        status = false;
    }
    return status;
}
```

- **Thread Creation**

```
int Create_Thread(void *(* func)(void *), unsigned char priority, void* arg)
{
    pthread_t thread;
    //For setting the attributes
    pthread_attr_t attr;
    //For setting the priority
    struct sched_param sparam= {priority};
    int retVal = -1;
    //initialize attributes
    pthread_attr_init(&attr);
    //add scheduling info to the attr structure
    pthread_attr_setschedparam(&attr, &sparam);
    //Set the scheduling policy to FIFO
    pthread_attr_setschedpolicy(&attr, SCHED_RR);
    //Start the thread
    retVal = pthread_create(&thread, &attr, func, arg);
    //Don't wait for the thread to finish
    pthread_detach(thread);
    return retVal;
}
```

- **Union Creation**

```
typedef union
{
    SESSION_TYPE_PACKET          sessionPacket;
    SESSION_LIFESIGN_TYPE_PACKET lifesignPacket;
    SUBSCRIPTION_TYPE_PACKET     subscriptionPacket;
    STATE_ELEMENT_TYPE_PACKET    stateElementPacket;
    SUPERVISION_TYPE_PACKET      supervisionPacket;
    CONNECTION_STATE_TYPE_PACKET connectionStatePacket;
    CMD_TYPE_PACKET              cmdPacket;
    CMD_INTERCOM_TYPE_PACKET     intercomPacket;
    BYTE_STREAM                  byteStream;
}GW_NG_PACKET_TYPES;
```

- **Send Command Packet**

```
void Send_Command_Packet(GW_APP_MSG_CREATE_TYPE msgbuf_CMD_CONTROL)
{
    if(Get_NG_Virtualization_State() == VIRTUALIZATION_RUNNING_STATE)
    {
        if( M_CMD_TYPE_INTERCOM != msgbuf_CMD_CONTROL.gwappMsg.packet.cmdPacket.cmdType )
        {
            msgbuf_CMD_CONTROL.gwappMsg.packet.byteStream.messageBodyLength = M_CMD_CONTROL_LENGTH;
        }
        else
        {
            msgbuf_CMD_CONTROL.gwappMsg.packet.byteStream.messageBodyLength = M_CMD_INTERCOM_LENGTH;
        }
        if(Send_To_SDN_IP_Tx_Message_Queue(&msgbuf_CMD_CONTROL) == GW20_FAILURE)
        {
            MSG_TEST(
                "GWAPP: Failed to Send_M_CMD_CONTROL_to_Vir message to SDN_IP Tx queue ");
            Handle_Gwapp_Error();
            // TODO: Return error;
        }
        //CORE_INFO("Sent command packet");
    }
}
```

- **Write Data On Socket**

```
int Send_To_Client(int sockRemote, unsigned char* packet, short packetSize)
{
    //Write packetSize bytes from packet onto the socket, and return error or number of bytes written.
    if(packet!=NULL)
        return write(sockRemote, packet, packetSize);

    else
        return -1;
}
```

- **Send State Element Packet**

```
void Send_Element_State_Packet(GW_APP_MSG_CREATE_TYPE msgbuf_State_Element)
{
    if(Get_NG_Virtualization_State() == VIRTUALIZATION_RUNNING_STATE)
    {
        msgbuf_State_Element.gwappMsg.packet.byteStream.messageBodyLength = M_STATE_ELEMENT_LENGTH;

        if(Send_To_SDN_IP_Tx_Message_Queue(&msgbuf_State_Element) == GW20_FAILURE)
        {
            MSG_TEST("GWAPP: Failed to send M_STATE_ELEMENT message to SDN_IP Tx queue ");
            Handle_Gwapp_Error();
        }
        //CORE_INFO("Sent state element packet");
    }
}
```

- **Set Status of Remote FC20**

```
int Set_Remote_FC20_Monitoring_Status(int instance_number, unsigned char newStatus)
{
    int retVal=-1;

    //Get the index of that FC20 in the device array.
    int FcnetDeviceIndex = Get_FCnet_Device_Position_In_List(instance_number);

    //Check that the FC20 is not Linked. Linked FC20s are always monitored.
    if((FcnetDeviceIndex >= 0)&&
        (FCnetData.FCnetDeviceDataObj[FcnetDeviceIndex].FCnetDeviceSubsetDataObj.monitored != ALWAYS
        {
            //Set new monitoring status
            FCnetData.FCnetDeviceDataObj[FcnetDeviceIndex].FCnetDeviceSubsetDataObj.monitored = newStatus;
            retVal = 0;
        }
    }
    return retVal;
}
```

- **Init Mutex**

```
int Init_GWapp_Mutexes()
{
    if((0 != pthread_mutex_init(&FCnetMutex, NULL)) ||
        (0 != pthread_mutex_init(&gwappStateMutex, NULL)) ||
        (0 != pthread_mutex_init(&sdnIpStateMutex, NULL)))
        return -1;

    return 0;
}
```

## • Send Life Sign Packet

```
void Send_Session_Lifesign_Packet(union signal sv)
{
    if(Get_NG_Virtualization_State() == VIRTUALIZATION_RUNNING_STATE)
    {
        GW_APP_MSG_CREATE_TYPE msgbuf_LifeSign;
        memset(&msgbuf_LifeSign, -1, sizeof(GW_APP_MSG_CREATE_TYPE));
        msgbuf_LifeSign.mtype = MSQ_DEFAULT_MSG_TYPE;

        msgbuf_LifeSign.gwappMsg.msgType = M_SESSION_LIFE_SIGN;
        msgbuf_LifeSign.gwappMsg.packet.lifesignPacket.sessionIdMajor = FCnetData.sessionIdMajor;
        msgbuf_LifeSign.gwappMsg.packet.lifesignPacket.sessionIdMinor = FCnetData.sessionIdMinor;

        msgbuf_LifeSign.gwappMsg.packet.byteStream.messageBodyLength = M_SESSION_LIFE_SIGN_LENGTH;
        if(-1==Send_To_SDN_IP_Tx_Message_Queue(&msgbuf_LifeSign))
        {
            // TODO: Return error;
            MSG_TEST("GWAPP: Failed to send M_SESSION_LIFE_SIGN message message to SDN_IP Tx queue ");
            Handle_Gwapp_Error();
        }
        //CORE_INFO("Sent Lifesign packet");
    }
}
```

## • Push Message Into Queue

```
static int Push_Into_Message_Queue(void* msgBuf, int msgQid, int flags, int msgSize)
{
    int result = -1;

    if(NULL!=msgBuf)
    {
        result = msgsnd(msgQid, msgBuf, msgSize, flags);

        if(-1 == result && (errno != EINTR))
        {
            CORE_ERR("Failed to put the message in the queue: %d with error %m", msgQidCoreapp);
        }
    }

    return result;
}
```



- **Pop Message From Queue**

```
static int Pop_From_Message_Queue(void* msgBuf, int msgQid, int msgType, int flags, int msgSize)
{
    int size = -1;
    if(NULL!=msgBuf)
    {
        size = msgrcv(msgQid, msgBuf, msgSize, msgType, flags);

        if((-1 == size) && (errno != EINTR))
        {
            if(errno != ENOMSG)
            {
                MSG_ERR("Error in retrieving message from %d queue with error set to %m", msgQid);
                size = -2;
            }
        }
    }
    return size;
}
```



