Universidad Nacional de Mar del Plata - Facultad de Ingeniería
Departamento de Electrónica

Gestor de Nodo Basado en Interfaz Web

(WBNMS –Web Based Node Manager System)

Informe de Proyecto Final

Autor: Courett Jorge Esteban - Matrícula: 8043
Directora: Ing. Liberatori Mónica Cristina

Universidad Nacional de Mar del Plata - Facultad de Ingeniería
Departamento de Electrónica

Gestor de Nodo Basado en Interfaz Web

(WBNMS –Web Based Node Manager System)

Informe de Proyecto Final

Autor: Courett Jorge Esteban - Matrícula: 8043
Directora: Ing. Liberatori Mónica Cristina

Índice de Contenidos

1 Prologo

El presente proyecto surge a partir de una experiencia laboral en el desarrollo de software de gestión de telecomunicaciones, sumado a un interés personal en las tecnologías de información.

El objetivo inicial fue la creación de una aplicación que permitiera gestionar equipos conectados a una red mediante protocolos estándares. Esta premisa se apoya en cuestiones de carácter práctico. La historia de gestión de Telecomunicaciones demuestra que el caos de protocolos ha sido durante mucho tiempo casi la única norma vigente. Esto obedece en parte a una estrategia comercial a corto plazo de promoción del diseño de protocolos propietarios para captura de clientes y su posterior dependencia, no sólo en cuanto a software sino también en cuanto a equipos se refiere. Pese a esto es posible vislumbrar una tendencia de cambio al respecto, debido en gran medida a la situación de gran competencia que existe hoy en día en el mercado de las telecomunicaciones y a la acción integradora resultante de la explosión en el crecimiento de la Internet.

Otras consideraciones de diseño no menos importantes que se han tenido en cuenta para el desarrollo del presente trabajo, se refieren a la propia Internet y su creciente influencia en nuestras vidas. Esta gran red ha generado nuevas formas de tratar aspectos de la vida cotidiana, muchas de ellas representan nuevas oportunidades de negocios: fuentes de información interactiva en línea, servicios de comercio electrónico, mensajería, video conferencias, video por demanda, comunidades virtuales, son algunas de las nuevas posibilidades que se ofrecen. Por este motivo, se ha pensado en diseñar un sistema de gestión no aplicable exclusivamente a equipos de telecomunicaciones sino también a entidades genéricas que en el futuro pudieran estar conectadas en red y precisen reportarse o cambien sus estados de gestión. El ejemplo más típico lo constituirían los componentes de una casa inteligente. Al respecto, estas perspectivas, aunque un poco atemorizantes en sus implicaciones, no dejan de presentar un costado fascinante. ***Nicholas Negroponte***[1] en su libro ***"Being Digital***[2]***"*** expone claramente estos pensamientos:

-Los bits, "el ADN de la información" se esta transformando en el elemento básico de la interacción humana. ... La revolución digital convertirá a las computadoras en objetos con los que hablaremos, conduciremos e incluso usaremos como vestimenta. Estos cambios alterarán, fundamentalmente, nuestra forma de aprender, de trabajar, de divertirnos...en fin, toda nuestra forma de vida.-

[1] ***Nicholas Negroponte*** fundo y dirige desde 1985 el Laboratorio de Medios del MIT, el mayor y más importante instituto de estudios e investigación interdisciplinaria de futuras formas de comunicación.

[2] ***Being Digital*** fue publicado en su primera edición en 1995.

2 Introducción

Es interesante destacar en primer término los criterios utilizados para la elaboración del presente proyecto y la expectativas de performance relevantes de cada uno de sus componentes. El conjunto de criterios y expectativas es lo que define la estructura de este informe.

El trabajo en sí se basa en los fundamentos teóricos de TMN (Ver Anexo 9.1 - Gestión de redes de telecomunicaciones). Como se explicó anteriormente se ha intentado ampliar su alcance en el sentido no sólo gestionar equipos de Telecomunicaciones, sino cualquier tipo de entidades.

La secuencia del presente informe se apoya en la división planteada por la Teoría de Modelado y Diseño de Sistemas orientado a Objetos (Ver Anexo 9.4 - Análisis, Diseño e Implementación de Sistemas). Los conceptos de dicha teoría son los que se utilizan más comúnmente en el desarrollo de software de aplicación.

A continuación se detalla el propósito y contenido de cada uno de los puntos principales de este Proyecto:

Planteo del Problema : Definición y alcances del problema a resolver. En esta etapa se desarrolla principalmente:

- La declaración del problema.
- La lista de requisitos que se espera deberá cumplir la solución.

Análisis del Sistema : Se analiza el problema y su dominio de aplicación. Se trabaja en esta etapa sobre diversos modelos y diagramas sobre los que, en una etapa posterior, se apoya el diseño de la solución, a saber:

- Modelo de la Arquitectura del Sistema de cual deriva el Diagrama Básico del mismo.
- Diagrama en Bloques Funcionales para reconocimiento de las funciones básicas de cada sistema y la posible subdivisión del problema en bloques funcionales más pequeños o subsistemas.
- Modelo de Objetos (Estático). Diagrama de los Objetos que componen el sistema, resaltando las funciones propias de cada uno y sus interrelaciones.
- Modelo Dinámico. Diagramas de los Estados básicos del sistema, destacándose en este caso las razones por las que se producen los cambios de estado.

Diseño de la solución : Se describen los pasos de diseño de la solución ideada partiendo de las pautas fijadas en el punto anterior. Esta etapa se caracteriza por una mayor granularidad en el tratado de solución:

- Modelo de Datos. Detalla la estructura básica de los datos del sistema.
- Diseño Arquitectónico. Basado en la división en subsistemas realizada en el análisis, se amplían y detallan los Módulos Funcionales.

- Diseño de Plataforma. Elección de los requisitos de Hardware y Software para la solución.
- Diseño de la Interfaz. Estructura básica de la Interfaz de Usuario.

Implementación : En este punto se aplican los resultados de la evaluación de los puntos anteriores para la construcción del Código fuente de la Aplicación, obteniéndose de esta manera:

- La estructura y código de la implementación. Se detallan las clases y librerías utilizadas así como la descripción básica de sus funciones.

El capítulo final del informe detalla (Conclusión) :

- Evaluación del trabajo.
- Posibles mejoras futuras. Pasos a seguir en el desarrollo de futuras versiones.

Anexos: En los Anexos se encuentran desarrollados en detalle los temas que se relacionan con el conocimiento previo necesario para el entendimiento del problema. Se han estructurado como:

- TMN. Desarrollo de los conceptos básicos de la Gestión de Redes de Telecomunicaciones.
- SNMP. Descripción del protocolo de Gestión de Redes Utilizado por la Aplicación.
- Perl. Características del lenguaje de programación utilizado.
- Análisis, Diseño e Implementación de Sistemas. Breve descripción de los pasos requeridos en el desarrollo de Sistemas.
- ASN.1. Descripción del lenguaje de notación abstracto.

3 Planteo del Problema

3.1 Definición del Problema:

" Se plantea la necesidad de solucionar el problema de gestión de entidades en forma remota, cumpliendo con estándares preestablecidos, de aplicación masiva, y teniendo presente la minimización de los costos de implementación. "

Se entenderá por sistema de gestión a aquel que presente no sólo la habilidad de conocer y modificar los distintos estados de un sistema, sino también que brinde la posibilidad de clasificarlos y presentarlos al usuario del sistema en forma conveniente, proveyendo a su vez herramientas adecuadas para su configuración. Además, el sistema deberá ser capaz de reportar en forma rápida y confiable los eventos de gestión que las entidades remotas notifiquen al gestor.

3.2 Relevancia del Problema

La necesidad planteada surge del crecimiento actual y sostenido que significa la existencia de redes de telecomunicaciones de escala mundial respecto de la oportunidad de desarrollar negocios. Se ha tratado de aprovechar la capacidad de propagación de información de gestión que estas redes ofrecen, particularmente Internet. En definitiva, el objetivo del trabajo se traduce en implementar aplicaciones que posibiliten la creación de nuevos servicios a partir de esta información.

.

3.3 Análisis Preliminar

Uno de los requisitos necesarios respecto de la solución es que la misma sea compatible con los protocolos de amplio uso existentes en Internet. El hecho de buscar una solución estandarizada promueve la aplicación sobre entornos multivendedor-mutiusuario.

Por otro lado, un producto que se precie de tener características de acceso masivo, debe ser posible de desarrollar al más bajo costo. Esto no implica baja calidad, sino una amortización lo más efectiva posible. Los grandes recursos disponibles en la Internet para el desarrollo "***open source***[3]", ya sea documentación o código fuente, determinan que desarrollar una aplicación sobre las bases de los mismos se convierta en una condición más que deseable.

La necesidad de proveer un servicio de gestión remota significa que el usuario del mismo, por lo general, no compartirá el espacio físico con el elemento

[3] Open Source: traducción aproximada del inglés: "código abierto", palabras que identifican una tendencia ya arraigada en Internet, en la que los beneficios de "abrir" el código fuente de las aplicaciones, esto es hacerlo público, significa mayores beneficios para la comunidad que pretender un monopolio intelectual.

gestor. Además se deberá proveer este servicio a más de un cliente concurrente. Una buena elección para este tipo de prestaciones lo constituye el modelo Cliente-Servidor, más precisamente aquel con capacidad de proveer servicio a múltiples clientes.

Por otro lado, la aplicación deberá ser capaz de reportar los eventos de gestión que se produzcan en las entidades gestionadas (equipos), buscando multiplicar o más precisamente paralelizar los posibles caminos de los reportes, para así poder aumentar la performance de confiabilidad del sistema.

3.4 Requisitos de la Solución

Una vez planteado el problema y comprendidos los aspectos sobresalientes del mismo, se hacen explícitos a continuación una serie de requisitos necesarios para la solución:

- Proveer Gestión Remota de entidades conectadas a Internet y/o redes privadas. Incluyendo las funciones de Consulta de Estados de gestión, Reporte automático de Estados de Gestión y modificación de variables de Gestión.

- Cumplir con los estándares existentes para Gestión.

- Proveer compatibilidad con protocolos de Internet.

- Ser configurable.

- Presentar un entorno amigable para el usuario.

- Ser del tipo cliente-servidor con capacidad de servir a múltiples clientes simultáneos.

- Proveer servicios de reporte forzado de eventos de gestión, es decir no solamente por demanda del cliente.

- Ser desarrollado en el marco de tecnología Open Source. Aunque este requisito se relaciona más con una decisión de implementación, en el caso del presente trabajo se convierte en requisito al ser adoptado como estrategia para la reducción del costo.

4 Análisis del Sistema

Se tomará como punto de partida los requisitos exigidos para la solución, anteriormente planteados. Se buscará ampliarlos para poder especificar el Sistema lo más detalladamente posible.
Se entiende por Sistema no sólo la aplicación a desarrollar, sino también el dominio en el que la misma interactúa con el resto de los componentes. No se tratará de especificar detalles de diseño ya que esto correspondería a una etapa posterior.

4.1 Arquitectura del Sistema.

Un posible modelo del sistema planteado, queda graficado en la Fig. 4-1.

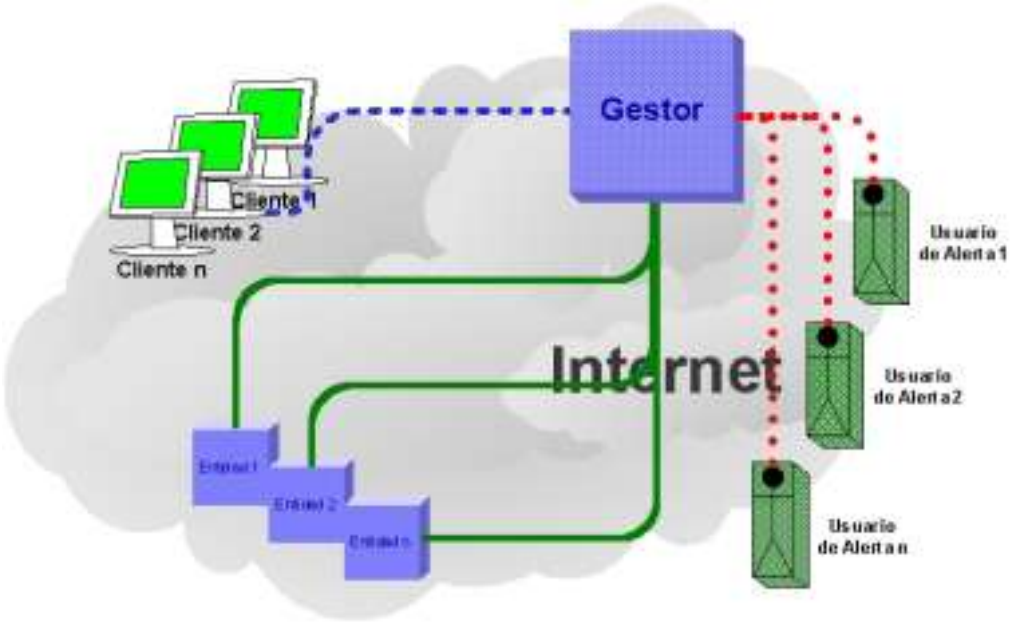


Figura 4-1 Modelo básico del sistema

Como se especifica en los requisitos de la solución, las *entidades* que se pretende gestionar son de cualquier tipo, con la condición de que debe ser factible conectarlas a una red. Los *clientes*, por su parte, tienen la capacidad de conectarse al *Gestor* para requerir sus servicios. Los *usuarios de alerta* son las entidades destinatarias de las notificaciones de eventos realizadas por el Gestor.
El *Gestor* centraliza todas las tareas de gestión, presentación de datos y alerta de eventos. En ningún caso delega estas tareas a los clientes. Es decir, toda acción que se ejecute en las entidades, se realiza por intermedio del gestor.
Las funciones de administración del sistema de gestión comprenden todas aquellas tareas no incluidas en el caso anterior que sea necesario administrarse, por ejemplo la administración de usuarios y permisos. Estas funciones, conjuntamente con las de gestión serán posibles de configurar desde los clientes de manera remota, o en el mismo gestor, localmente.

El ámbito en el que debe funcionar la aplicación quedará acotado por los protocolos de la Internet o de la red funcional privada, no debiendo necesariamente existir una red dedicada a la gestión. Dicho en otras palabras, la aplicación debe convivir con múltiples aplicaciones, por lo que se buscará utilizar protocolos estandarizados para lograr su correcto funcionamiento.

4.2 Bloques funcionales.

El sistema planteado se puede dividir en los varios bloques funcionales básicos, de manera de poder cumplir con los requisitos de la solución. La Fig. 4-2 presenta un gráfico de dichas funciones. A continuación se especifica la funcionalidad de cada bloque.

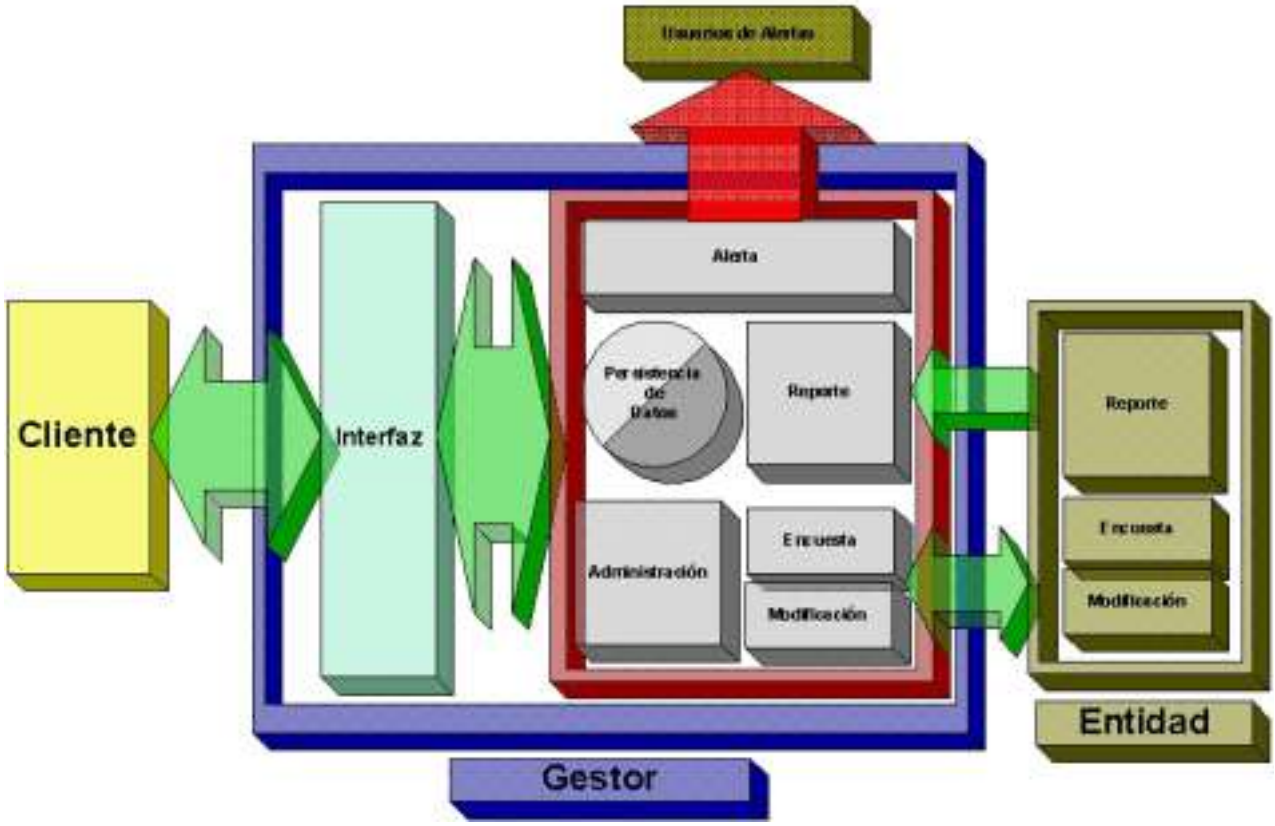


Figura 4-2 Bloque funcionales

4.2.1 Cliente

Define a la entidad usuaria del sistema. Un Cliente del Sistema de Gestión es aquel que requiere al Gestor funciones de gestión sobre las entidades y de administración del sistema. El sistema posee múltiples clientes por gestor.

4.2.2 Entidad

Se conoce con este nombre al bloque gestionado. Se pueden distinguir tres bloques funcionales básicos.

4.2.2.1 Reporte

Cumple la función de reportar el cambio de estado de variables de gestión preestablecidas en la entidad gestionada.

4.2.2.2 Encuesta

Cumple la función de responder a encuestas externas sobre variables de gestión de la entidad.

4.2.2.3 Modificación

Cumple la función de validar y modificar variables de gestión a pedido de un gestor autorizado en la entidad gestionada.

4.2.3 Usuarios de Alertas

Definidos en el Gestor como las entidades que recibirán información de eventos reportados.

4.2.4 Gestor

Brinda servicio de gestión a los clientes del sistema y envía información de alerta a los usuarios de alertas. Constituye un único bloque en el que se distinguen siete bloques funcionales.

4.2.4.1 Interfaz

Provee un medio por el cual el cliente interactúa con el gestor. Realiza tareas requeridas por el Cliente, tales como la configuración de la aplicación, consulta de datos de gestión y de administración, encuestas bajo demanda y modificación variables en las entidades gestionadas por la aplicación.

4.2.4.2 Persistencia de Datos

Provee un repositorio común para almacenar los datos de gestión y administración. Debe facilitar el manejo de los datos del sistema y proveer un adecuado control respecto de la consistencia de los mismos. Soporta consultas externas, carga masiva de datos mediante una interfaz adecuada y arquitectura escalable.

4.2.4.3 Administración

Realiza la alta, baja y modificación de los datos de gestión y configuración del sistema. Provee acceso al estado operativo del sistema y los registros históricos de funcionamiento (Funcionalidad Log del sistema). La interfaz toma este bloque como medio para realizar, a través de ella, las tareas de administración.

4.2.4.4 Encuesta

Se encarga de realizar encuestas, programadas o por demanda del cliente, a las entidades gestionadas. La interfaz utiliza este bloque para realizar dichas funciones sobre las entidades.

4.2.4.5 Modificación

Se encarga de realizar modificaciones en las variables de gestión de las entidades en forma remota. La interfaz requiriere del servicio de este bloque para cumplir con la orden del cliente de realizar la modificación de las variables de gestión en las entidades gestionadas.

4.2.4.6 Reporte

Este bloque es el encargado de recibir los reportes de falla de las entidades. Procesa la información y la almacena en el bloque de persistencia de datos.

4.2.4.7 Alerta

Este bloque es el encargado de procesar la información de eventos reportados al bloque de reportes. La información así procesada es utilizada por la interfaz para notificar los eventos de gestión reportados. También se encarga de tomar acciones de notificación externa hacia los usuarios de alertas. De este modo se multiplican los caminos de reporte, reforzando de así la tarea de gestión en el aspecto de confiabilidad.

4.3 Modelado de Objetos (Estático).

Se analiza a continuación un modelo de clases. Se presenta en la Fig. 4.3 un esquema con notación UML correspondiente al sistema planteado. Se trata de un modelo simplificado basado en el diagrama funcional del punto anterior. Se desarrolla con la complejidad necesaria de tal manera de poder expresar los atributos y funciones de cada clase de objeto, así como también sus interrelaciones.

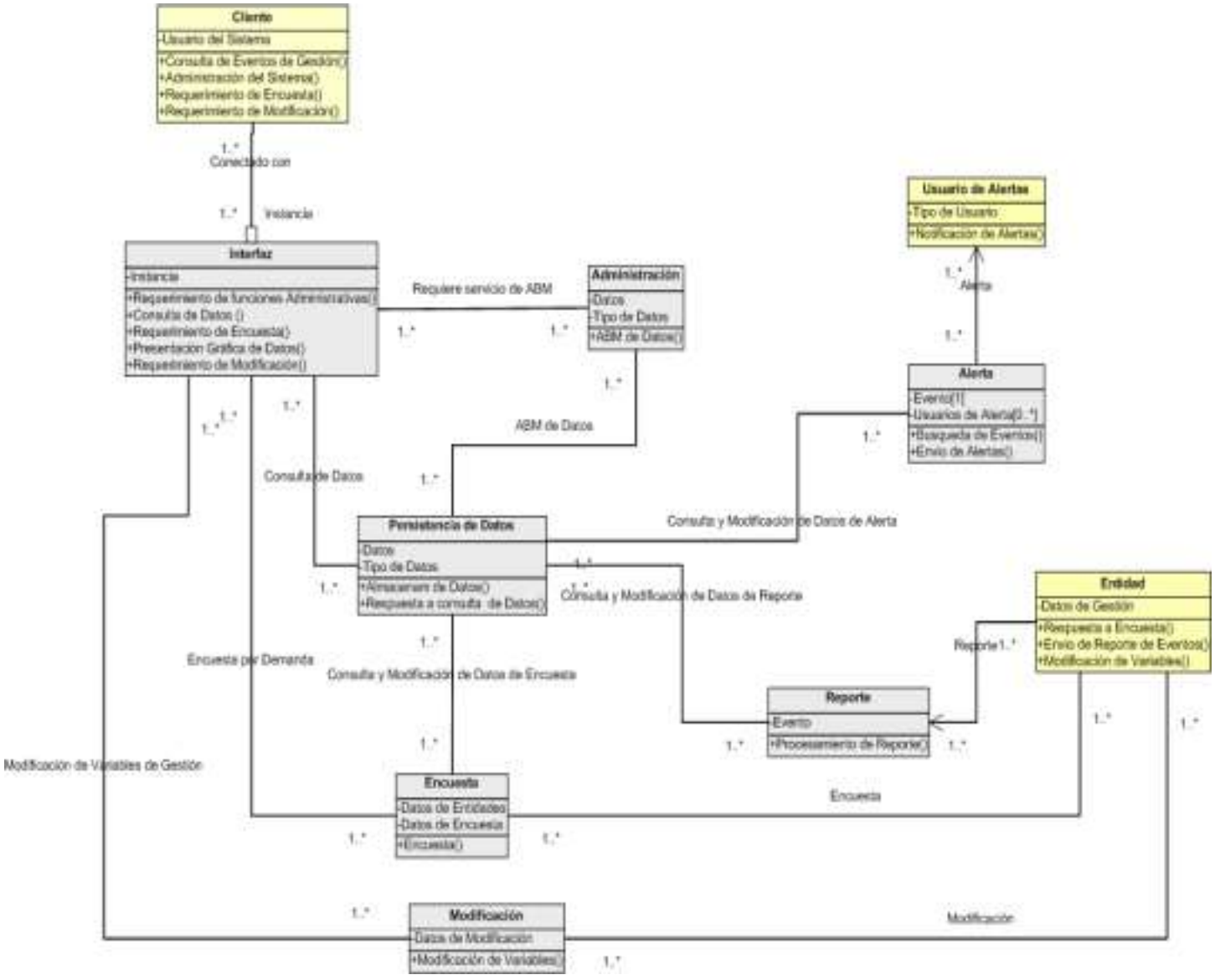


Figura 4-3 Diagrama simplificado de clases del sistema

Este gráfico pretende describir el comportamiento estático del sistema, donde cada uno de los recuadros representa una clase surgida de uno de los bloques funcionales anteriormente descriptos. Cada una de ella posee atributos definidos en la división intermedia del recuadro que representa la clase. Dichos atributos definen, en

conjunto, una instancia definida de la clase, es decir diferencian un Objeto de otro de la misma clase.
La descripción de las acciones que es capaz de realizar un objeto de la clase en cuestión están definidas en la división inferior del recuadro de clase por las operaciones.
Uno de los principales usos de este tipo de gráficos es la posibilidad de ver la interrelación entre las distintas clases. Para ello es necesario especificar como se encuentran asociadas las distintas clases. Esto se realiza mediante líneas llenas que interconectan las clases asociadas entre si, especificando un título que describa cada asociación. Este tipo de relaciones entre clases puede o no tener una dirección de interpretación única, ello es realizado agregando flechas a las líneas de asociación (de no especificarse, las asociación es vi direccional). Para graficar la posibilidad de multiplicidad entre instancias de clases asociadas se utiliza una referencia al número de mínimo de relaciones y el numero máximo de las mismas. Por ejemplo para definir de una a infinitas asociaciones, la notación es 1 ..*.

A continuación se presenta una descripción de las diversas clases y sus métodos asociados:

4.3.1 Cliente

La **Clase Cliente** queda definida por el atributo de ser ***Usuario del Sistema*** de Gestión. Es capaz de realizar las operaciones de ***Consulta de Eventos de Gestión***, ***Requerimiento Modificación***, ***Administración del Sistema*** y de ***Requerimiento de Encuesta*** a través de la ***Clase Interfaz***.

- ***Consulta de Eventos de Gestión:*** El Cliente realiza consultas del estado de las variables de gestión de las *n* entidades gestionadas por el sistema.

- ***Administración del Sistema:*** El Cliente realiza múltiples actividades de Administración del Sistema, entre ellas, creación de usuarios y verificación del sistema.

- ***Requerimiento de Encuesta:*** El Cliente solicita al sistema que se realicen operaciones de encuesta de gestión a determinada/s entidad/es.

- ***Requerimiento Modificación:*** El Cliente solicita al sistema se realice la modificación de variable de gestión en las entidades.

La **Clase Cliente** se encuentra asociada con **Clase Interfaz** a través del calificador de ***Instancia***, ya que el sistema deberá permitir la conexión de múltiples clientes y mantener unívocamente identificada dicha conexión.
Este es un tipo especial de asociación que permite identificarla por un atributo determinado, en este caso la ***Instancia***[4] de conexión.

[4] Instancia: se define para las aplicaciones Cliente-Servidor como la sesión de aplicación establecida entre el cliente y el servidor, al solicitar el primero la conexión.

4.3.2 Interfaz

Definido por el atributo de ***Instancia*** proveniente del identificador de conexión con un cliente determinado. Es capaz de realizar las operaciones de ***Requerimiento de funciones de Administración, Consulta de Datos, Requerimiento de Encuesta y Presentación Gráfica de Datos.***

- ***Requerimiento de funciones de Administración:*** Con esta operación la clase Interfaz comanda sentencias administrativas por orden del cliente. Es una operación realizada en asociación con la **Clase Administración.** Dicha asociación se realiza en carácter de servicio requerido de ABM (Alta, Baja y Modificación de Datos).

- ***Consulta de Datos:*** La Interfaz es capaz de realizar consultas de los datos de gestión y administración por orden del cliente. Esta operación es realizada en asociación con la **Clase Persistencia de Datos.**

- ***Requerimiento de Encuesta:*** La Interfaz solicita por orden del cliente un inicio de encuesta de gestión. Esta Operación es realizada en asociación con la **Clase Encuesta.**

- ***Presentación Gráfica de Datos:*** Esta operación realiza el procesamiento de los datos necesarios para poder presentar en forma gráfica los datos de Gestión. Esta operación atiende al requisito fundamental de obtener una interfaz de usuario amigable (user friendly).

- ***Requerimiento de Modificación:*** La Interfaz solicita por orden del cliente una modificación de determinada variable de gestión. Esta operación es realizada en asociación con la **Clase Modificación**.

4.3.3 Persistencia de Datos

Se definen dos atributos, ***Datos*** y ***Tipos de Datos***, según sean datos de gestión o de administración del sistema. Las operaciones realizadas por esta clase cubren básicamente la tarea de ***Almacenamiento de Datos*** y ***Respuesta a consultas de Datos.***

- ***Almacenamiento de Datos:*** esta operación será realizada de forma de lograr la consistencia de los datos al momento de su almacenamiento. Es requerida por las otras clases asociadas.

- ***Respuesta a consultas de Datos:*** La respuesta a distintas consultas requeridas, por clases asociadas, se realizará por medio de esta operación.

4.3.4 Administración

Posee como atributos los ***Datos*** y el ***Tipo de Datos***, definiéndose como Datos de Gestión y Datos de Administración. Esta clase es la encargada de realizar las tareas de administración a través de la operación de ***ABM de Datos.***

- ***ABM de Datos:*** esta operación cumple la misión de realizar la alta, baja y modificación de los datos de gestión y administración por el requerimiento del servicio de ABM realizado por la **Clase Interfaz** Es realizada en asociación con la **Clase Persistencia de Datos.**

4.3.5 Encuesta

Esta clase maneja dos atributos, los ***Datos de Entidades*** que identifican a las entidades gestionadas factibles de ser encuestadas, y los ***Datos de Encuesta*** que constituyen los datos resultantes de encuestar dichas entidades. La operación de ***Encuesta*** representa la operación principal de esta clase.

- ***Encuesta:*** Esta operación comprende las acciones de encuesta y procesamiento de datos y reporte de la información de gestión de las entidades. Esta operación es realizada en asociación con la **Clase Entidad** que es la encargada de responder a la encuesta.

4.3.6 Reporte

El Atributo principal de esta clase es el ***Evento***. Es decir toda la información asociada a un evento de gestión reportado por una entidad. La Operación de ***Reporte*** es realizada en asociación con la **Clase Persistencia de Datos.**

- ***Reporte***: Cumple la función de recoger los datos de los reportes enviados por las entidades, procesarlos y en asociación con la **Clase Persistencia de Datos** almacenarlos.

4.3.7 Modificación

El atributo correspondiente a esta clase son los ***Datos de Modificación.*** Estos son todos los datos involucrados en el proceso de modificación de variables de gestión. La Operación de ***Modificación*** se realiza en asociación con la **Clase Entidad**.

- ***Modificación:*** Cumple la función de solicitar a la Entidad la modificación de alguna de sus variables de gestión.

4.3.8 Entidad

Esta clase define a los objetos sujetos de gestión por el sistema. En los mismos el principal atributo son los ***Datos de Gestión*** además de la identificación unívoca de los mismos (este tipo de información se da como obvia dentro de un diagrama de

clases). Las operaciones que realiza esta clase son tres, ***Respuesta a Encuesta, Modificación de Variables*** y ***Envío de Reporte de Eventos.***

- ***Respuesta a Encuesta:*** Brinda la capacidad de responder con los datos de gestión a las encuestas realizadas.

- ***Envío de Reporte de Eventos:*** Es la operación que posibilita el informe de ciertos cambios de estado de gestión en la Entidad a Gestores externos. En este caso a la **Clase Reporte.**

- ***Modificación de Variables:*** Esta operación valida y modifica los valores de las variables de gestión en la entidad. En asociación con la **Clase Modificación** se conduce todo el proceso de modificación.

4.3.9 Alerta

Esta clase maneja atributos de ***Evento***, referido a eventos de reporte ya procesados y ***Usuarios de Alerta*** definidos por los identificadores de objetos de la **Clase Usuario de Alerta** asociados a ***Eventos*** específicos.
Realiza funciones de ***Búsqueda de Eventos*** y ***Envío de Alertas***.

- ***Búsqueda de Eventos:*** esta operación es realizada con el fin de identificar eventos factibles de convertirse en Alerta. Esto dependerá de la configuración previa y de los eventos ya procesados por la **Clase Reporte** y almacenada por la **Clase Persistencia de Datos.** Es con esta última con la que se opera en asociación.

- ***Envío de Alertas:*** Por medio de esta función se envían la Alertas a los Usuarios de Alertas. Reportando de esta manera, los eventos de gestión enviados por las Entidades.

4.3.10 Usuario de Alertas

Esta clase define los objetos que son destinatarios de los reportes externos del gestor. Define como atributo el ***Tipo de Usuario*** para multiplicar las formas de Alerta. Y la operación principal es la ***Notificación de Alertas***.

- ***Notificación de Alertas:*** Básicamente se describe como la acción de notificarse de una alerta determinada por parte de un Usuario de Alertas.

4.4 Modelado Dinámico

Un diagrama de estados es una herramienta del UML que permite modelizar los distintos cambios de estados que sufren cada una de las partes del sistema estudiado ante eventos determinados. Estas gráficas están compuestas por recuadros que representan cada uno de los estados unidos por líneas que representan el cambio entre ellos. Estas líneas poseen un único sentido de

interpretación, especificado por flechas, y están calificadas por un título que especifica el evento que produce el cambio de estado.
Se analiza a continuación el modelo dinámico de las operaciones básicas del sistema. Se encontraran dos tipos de situaciones, una en la que el estado inicial esta definido por un evento externo representado por una circunferencia, y otra en la que existe un estado estable en la

4.4.1 Interfaz

En la siguiente figura representa el comportamiento dinámico del subsistema Interfaz. Comenzamos el análisis suponiendo que el subsistema se encuentra en estado latente, ***Esperar Conexión***, en espera de que un cliente se conecte y requiera el servicio del gestor.

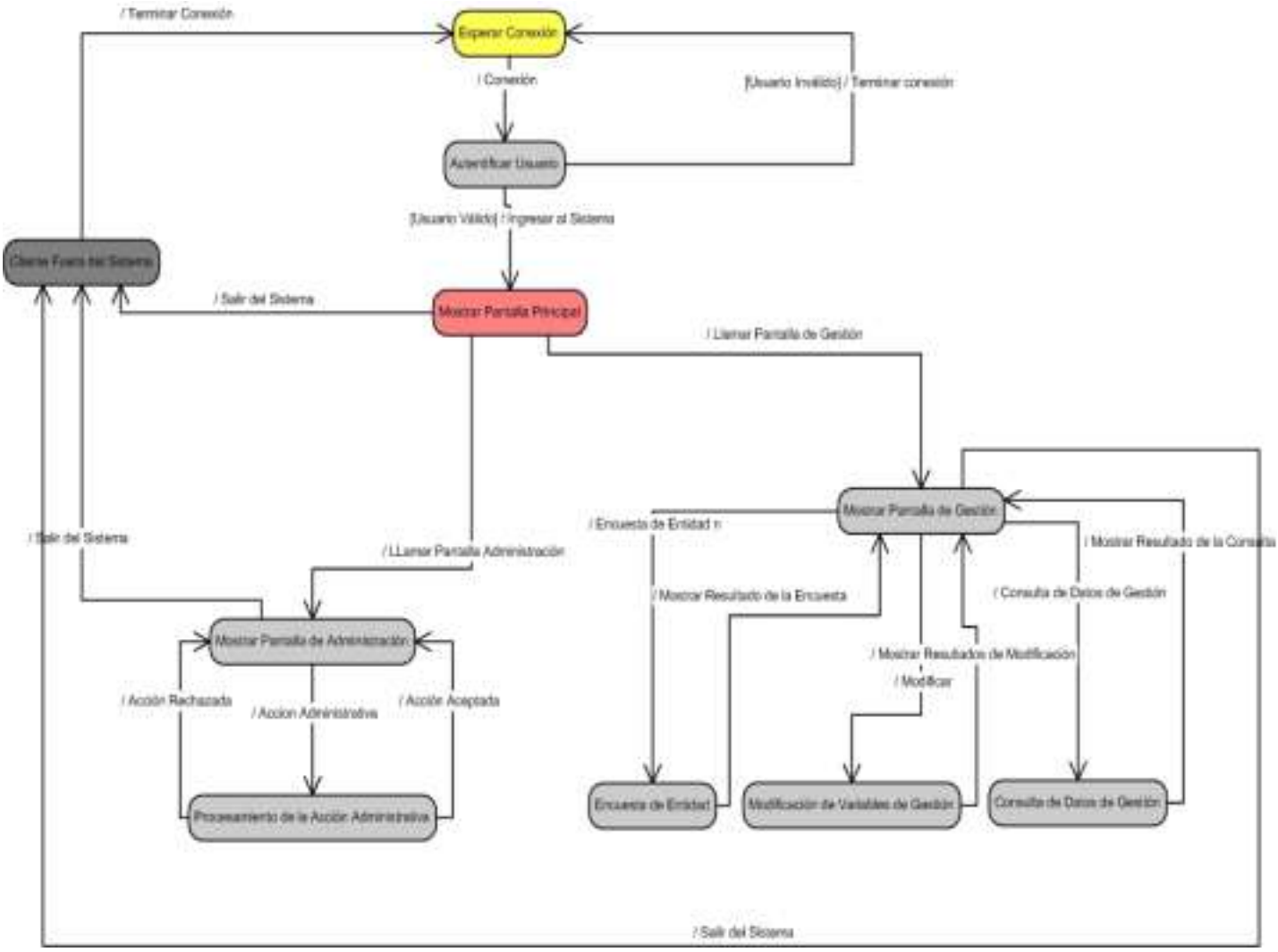


Figura 4-4 Diagrama de estados del subsistema Interfaz

- ***Esperar Conexión:*** Estado inicial que describe el estado de espera en el que se encuentra la Interfaz, cuando no existe una tentativa de conexión por parte de un Cliente. La Interfaz sale de este estado cuando se produce el intento de conexión de un Cliente. Esta acción lleva a la Interfaz al estado de ***Autentificar Usuario.***

- ***Autentificar Usuario:*** Estado en el cual la Interfaz se encuentra en proceso de validación del usuario del sistema que intenta la conexión a través del Cliente. Existirán dos posibles transiciones, una al estado anterior de ***Esperar Conexión*** en el caso de que el usuario sea inválido con la finalización de la conexión antes de poder ser iniciada. Y la otra, al estado de ***Mostrar Pantalla Principal*** en el caso de que la validación del usuario sea exitosa, por lo que se ingresará al sistema.

- ***Mostrar Pantalla Principal:*** Estado en el cual se le da la bienvenida al usuario autenticado o válido del sistema y se le ofrecen los menúes de operación relevantes. Mediante la selección de los mismos el usuario puede requerir un nuevo cambio de estado, que puede ser una entre tres posibilidades. Tomar la acción de salir del sistema, lleva a la Interfaz al estado de ***Cliente Fuera del Sistema***. El pasaje al estado ***Mostrar la Pantalla de Administración*** se produce al tomar la acción de llamar esta pantalla. Vale la pena aclarar que no se fija de esta manera ninguna decisión acerca de cómo se implementará la Interfaz, simplemente se menciona la separación funcional requerida. Por último, al llamar la pantalla de Gestión se pasa al estado ***Mostrar Pantalla de Gestión.***.

- ***Mostrar la Pantalla de Administración:*** En este estado se le ofrece al usuario la posibilidad de tomar acciones administrativas, las cuales conducen al estado de ***Procesamiento de la Acción Administrativa.*** Otra opción puede ser la de salir del sistema, que pasa la Interfaz al estado de ***Cliente Fuera del Sistema***.

- ***Procesamiento de la Acción Administrativa:*** Es el estado en el cual se procesa el requerimiento realizado por el usuario. Este estado concluye con dos posibles acciones que llevan nuevamente la Interfaz al estado de ***Mostrar Pantalla de Administración*** con diferentes implicaciones: aceptación y su realización en forma correcta o rechazo de la acción.

- ***Mostrar Pantalla de Gestión:*** Este es otro de los estados posibles de llegada a partir del estado ***Mostrar Pantalla Principal***. El usuario del sistema podrá decidir la acción a seguir, eligiendo entre cuatro posibles acciones. La Encuesta de Entidad, una acción que llevará a la interfaz al estado ***Encuesta de Entidad***. La ***Modificación***, conduce al estado ***Modificación de Variables de Gestión.*** La acción de Consulta de datos de gestión produce la transición al estado ***Consulta de Datos de Gestión***. La última acción llevará la Interfaz al estado de ***Cliente Fuera del Sistema***.

- ***Encuesta de Entidad:*** Durante este estado la Interfaz realiza la tarea de encuesta de una entidad específica, tomando la acción de mostrar el resultado de la encuesta, provocando la transición al estado de ***Mostrar Pantalla de Gestión.***

- ***Modificación de Variables de Gestión:*** Durante este estado se realiza el requerimiento de modificación de las variables de gestión a la entidad. Al obtenerse la respuesta de la entidad, con el resultado de este requerimiento,

se produce la transición al estado ***Mostrar Pantalla de Gestión*** desplegando dicha respuesta.

- ***Consulta de Datos de Gestión:*** Durante este estado la Interfaz realiza la tarea de consulta a una entidad específica, tomando la acción de mostrar el resultado de esta consulta, lo que provoca la transición al estado de ***Mostrar Pantalla de Gestión.***

4.4.2 Encuesta

La Fig. 4.5 representa el comportamiento dinámico del subsistema de Encuesta. Se comienza a analizar a partir de un estado Inicial en el que es requerida una encuesta. Esta acción provoca el disparo del mecanismo de Encuesta y la transición del Bloque de Encuesta al estado de ***Preparación de Encuesta***.

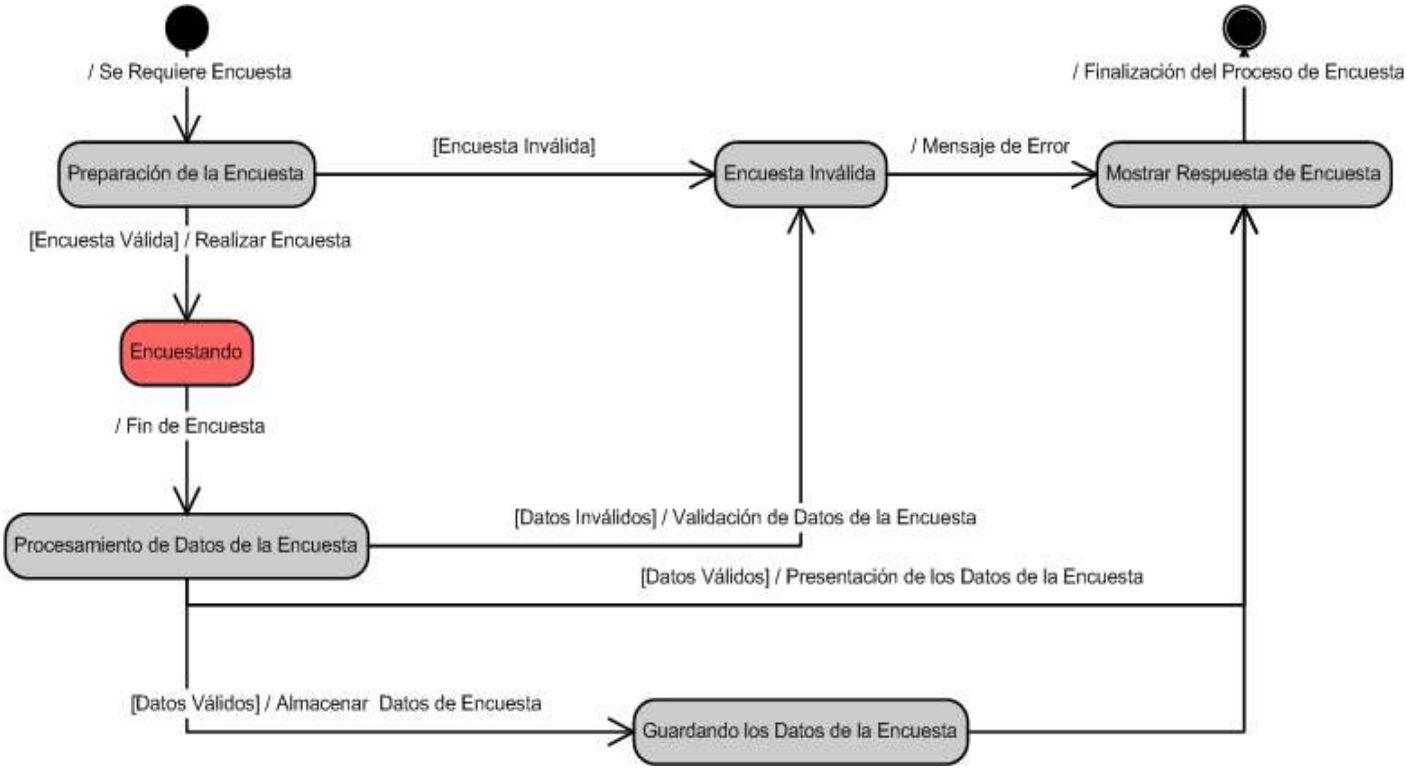


Figura 4-5 Diagrama de estados del subsistema de Encuesta

- ***Preparación de Encuesta:*** Durante este estado el Bloque de Encuesta procesa los datos necesarios para poder realizar la encuesta requerida. El resultado de esta preparación puede concluir en dos cambios de estado posibles. Si se obtiene una encuesta inválida se produce la transición al estado ***Encuesta Inválida***. Si en cambio, se obtiene una encuesta válida se produce la transición al estado ***Encuestando*** al ejecutarse la acción de realizar la encuesta

- ***Encuesta Inválida:*** Durante este estado, el Bloque de Encuesta genera un Mensaje de Error destacando el carácter de la invalidez de la encuesta. Esta acción produce una transición al estado ***Mostrar Respuesta de Encuesta***, en este caso un error.

- ***Encuestando:*** Durante este estado, el Bloque de Encuesta realiza el Proceso de Encuesta. Al terminar la encuesta se produce un cambio de estado a ***Procesamiento de Datos de Encuesta.***

- ***Procesamiento de Datos de Encuesta:*** Se procesan los datos generados durante la encuesta. Luego de esto, se pueden tomar dos posibles cursos de acción. Uno en el que los datos resultado de la encuesta son inválidos, hecho que produce una transición al estado de ***Encuesta Inválida***. En el otro curso de acción los datos son válidos, hecho que produce el almacenamiento de los mismos y el cambio al estado ***Guardando los Datos de la Encuesta,*** en el caso que sea requerido el almacenamiento, o la transición directa al estado ***Mostrar Respuesta de Encuesta*** en caso contrario.

- ***Guardando los Datos de la Encuesta:*** Durante este estado se almacenan los datos producto de la encuesta. Al terminar este proceso se produce una transición al estado ***Mostrar Respuesta de Encuesta.***

- ***Mostrar Respuesta de Encuesta:*** Durante este estado se muestra el resultado de la encuesta, esto puede ser tanto un mensaje de error como también los datos correctos provenientes de la entidad. Al finalizar este proceso se lleva al Bloque de Encuesta al estado final de ***Finalización del Proceso de Encuesta***.

4.4.3 Reporte

La Fig. 4.6 representa el comportamiento dinámico del subsistema de Reporte. Este subsistema responde al siguiente comportamiento básico. Se parte del estado estable ***Esperando Reporte***.

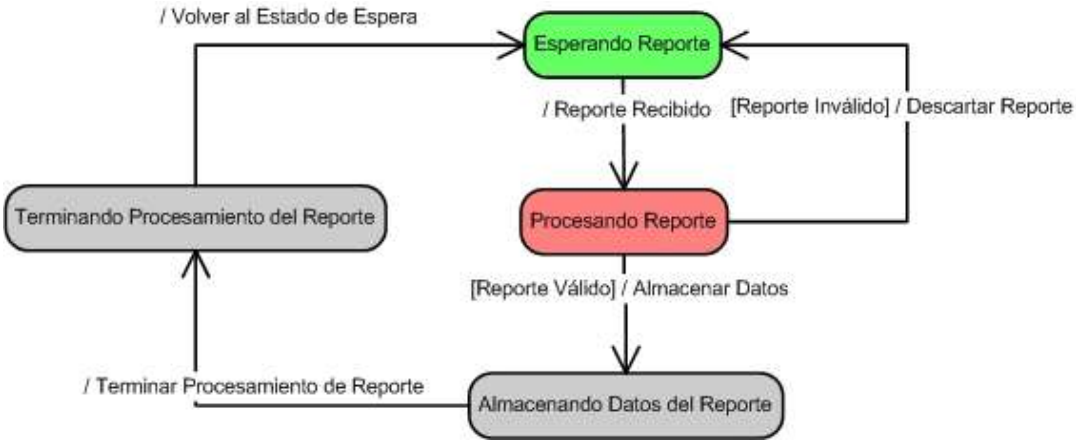


Figura 4-6 Diagrama de estados del subsistema de Reporte

- ***Esperando Reporte:*** Este es el estado estable del Bloque de Reporte. Mientras que no sea recibido un reporte, el subsistema no cambia de estado. Al producirse la recepción de un reporte se produce la transición al estado ***Procesando Reporte.***

- ***Procesando Reporte:*** Durante este estado se procesa el reporte recibido lo que produce dos cursos de acción posible. En el caso de ser inválido el

reporte se produce el cambio al estado estable ***Esperando Reporte.*** Ante un reporte válido se produce la transición al estado ***Almacenando Datos del Reporte*** al iniciarse este almacenamiento.

- ***Almacenando Datos del Reporte:*** Durante este estado se produce el almacenamiento de los datos del Reporte. Concluido este proceso se produce la transición al estado ***Terminando Procesamiento del Reporte.***

- ***Terminando Procesamiento del Reporte:*** durante este estado el subsistema de Reporte concluye el procesamiento del reporte recibido y luego se produce la transición al estado inicial ***Esperando Reporte***.

4.4.4 Modificación

La Fig. 4.7 representa el comportamiento dinámico del subsistema de Modificación. Se comienza a analizar a partir de un estado inicial en el que es requerido un proceso de Modificación de variables de gestión en una entidad determinada. Esta acción produce una transición al estado ***Preparación del Proceso de Modificación***.

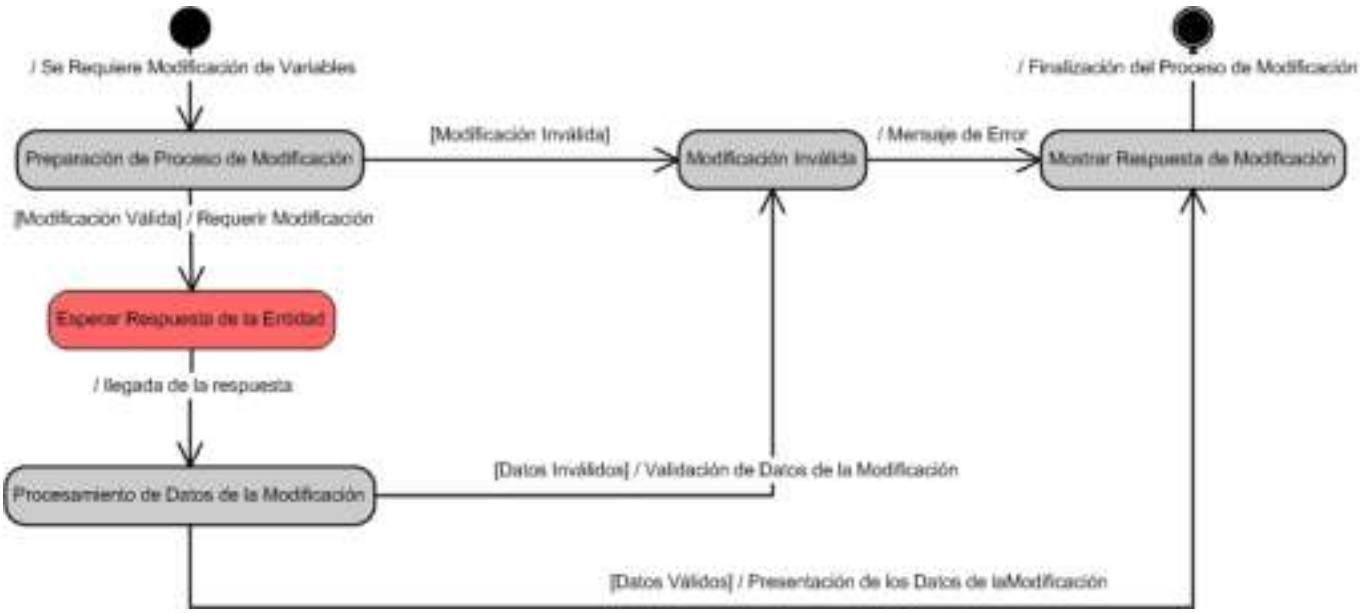


Figura 4-7 Diagrama de estados del subsistema de Modificación

- ***Preparación del Proceso de Modificación:*** Durante este estado, el Bloque de Modificación realiza las operaciones necesarias de validación de los datos para realizar la acción correspondiente. Del resultado de estas validaciones surgen dos posibles transiciones de estado. La primera, en el caso de una modificación inválida, es el paso al estado de ***Modificación Inválida***. En el caso de una modificación válida, es el paso al estado de ***Espera de Respuesta de la Entidad***, al requerir a la entidad que se realice la Modificación.

- ***Espera de Respuesta de la Entidad:*** Durante este estado el Bloque de Modificación espera una respuesta de la entidad al requerimiento realizado. Al obtener respuesta, se produce la transición al estado de ***Procesamiento de los datos de Modificación***.

- ***Procesamiento de los datos de Modificación*:** Se procesa la respuesta recibida de la entidad. Esto puede concluir, que los datos son inválidos y pasar al estado de ***Modificación Inválida*** o que los datos son válidos y presentar este resultado al pasar al estado ***Mostrar Respuesta de Modificación***.

- ***Modificación Inválida:*** Durante este estado, se procesa el mensaje de error y se pasa al estado ***Mostrar Respuesta de Modificación***.

- ***Mostrar Respuesta de Modificación:*** Durante este estado se procesa y se muestra la respuesta al proceso de Modificación, y luego se da por terminado dicho proceso.

4.4.5 Alerta

La Fig. 4.8 representa el comportamiento dinámico del subsistema de Alerta. Este subsistema posee un estado "estable" denominado ***Esperando Eventos Nuevos***.

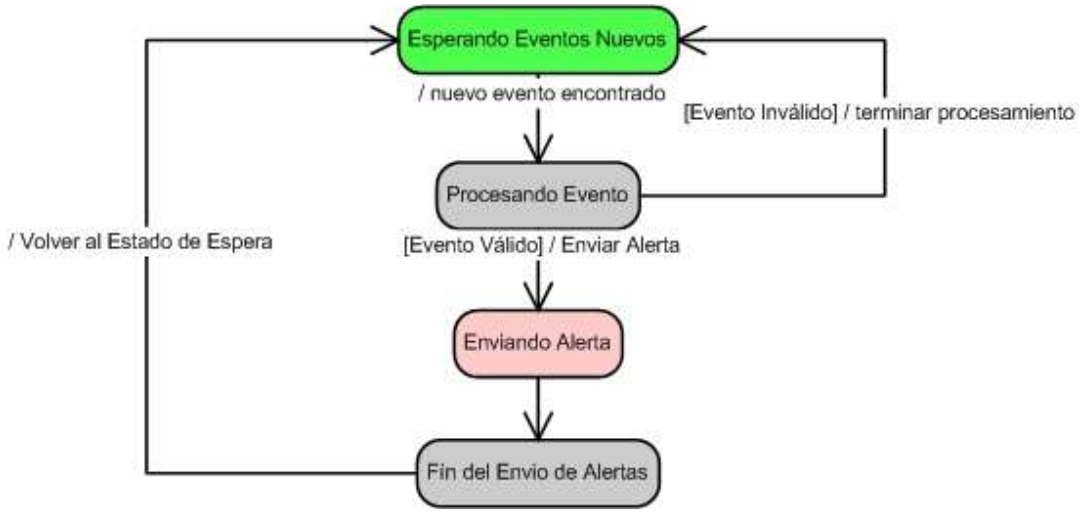


Figura 4-8 Diagrama de estados del subsistema de Alerta

- ***Esperando Eventos Nuevos:*** El Bloque de alerta se encuentra en este estado hasta que un nuevo evento reportado y procesado sea hallado. Al producirse esto, se dispara la transición al estado ***Procesando Evento***.

- ***Procesando Evento:*** Durante este estado se procesa la información del evento encontrado. El resultado de este procesamiento deriva en dos acciones posibles. La primera, en el caso de ser un evento Inválido, se termina el procesamiento de la acción y se vuelve al estado estable ***Esperando Eventos Nuevos***. La segunda, en el caso de un evento válido, se produce la transición al estado ***Enviando Alerta***.

- ***Enviando Alerta:*** Durante este estado, el Bloque de alerta envía a los Usuarios de Alertas configurados las alertas derivadas de los eventos encontrados. Al terminar este proceso se produce el cambio al estado ***Fin del Envio de Alertas.***

- ***Fin del Envio de Alertas:*** Este es un estado de transición de Bloque antes de pasar el estado estable luego del procesamiento de alerta realizado.

4.5 Conclusión de la etapa de Análisis

Al concluir esta etapa, se encuentra totalmente definido el sistema de gestión, con la profundidad que se consideró necesaria según cuestiones de practicidad.
A partir de los datos del sistema y de la declaración de requisitos de la solución, se diseñará una solución que abarque todas las necesidades planteadas.

5 Diseño de la solución

De acuerdo a las necesidades planteadas en la etapa correspondiente al análisis del problema y atendiendo a los criterios preestablecidos, en este capítulo se presenta el diseño de un sistema que contemple todos los aspectos mencionados previamente e incorpore las siguientes premisas globales de diseño:

- Utilizar plataformas estándares para garantizar la facilidad de mantenimiento posterior y no desviar el esfuerzo de desarrollo en problemas de diseño ya solucionados (ventaja del Open Source).

- Lograr un equilibrio entre la complejidad del diseño y las prestaciones del software.

- Basar el diseño en el concepto de ciclos de mejora continua de la aplicación. Con este objetivo presente se diseñará teniendo en cuenta la modularidad del software y hardware involucrados.

- Diseñar interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior

El producto final se denominará ***WBNMS***, Web Based Node Manager System, explicitando de esta manera la funcionalidad y utilidad del sistema final.
El diseño en sí se divide en cuatro etapas. En primer lugar, se plantea el diseño de un Modelo de Datos que especifique la forma en que será tratada la información de administración/gestión por el ***WBNMS***. Luego de este primer paso, se trabaja sobre el diseño de la arquitectura de ***WBNMS***. A continuación se diseña la plataforma de hardware y software. Por último, se encara el diseño de la Interfaz.
Aunque el diseño pueda ser dividido de la manera mencionada, es conveniente realizar cada una de las etapas teniendo presente en simultáneo aspectos claves de las mismas

5.1 Modelo de Datos

En esta etapa se define un modelo de la estructura organizativa básica de los datos del sistema. El criterio utilizado para desarrollar dicho modelo, es la agrupación de datos en categorías. Esto contribuye a la simplificación de su manejo.

5.1.1 Usuarios del Sistema

Las acciones que serán capaces de realizar los usuarios del sistema surgirán a lo largo del desarrollo del diseño del sistema. Pero en referencia a la Figura 5.1 se especifica que un usuario del sistema ***WBNMS*** es de dos tipos posibles:

- ***Operador:*** Se trata de un usuario con atribuciones limitadas, que realiza operaciones de usuario propiamente dichas y no posee permisos de acceso a las funciones de configuración.

- ***Administrador:*** Este usuario posee acceso a todas las funciones de ***WBNMS,*** típicamente sus funciones se relacionan con la administración. En este sentido, es el usuario encargado de configurar el sistema.

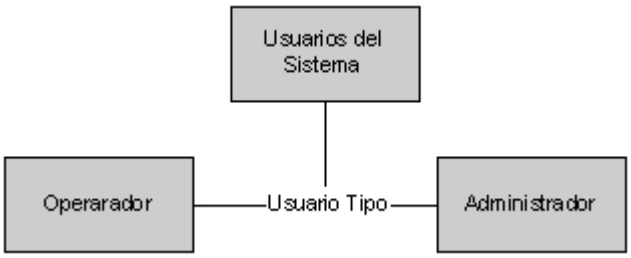


Figura 5-1 Categorización de los usuarios del sistema

5.1.2 Entidades

Cada entidad gestionada esta asociada a un ***tipo de entidad***, como se presenta en la Fig. 5.2. El tipo agrupa lógicamente entidades de características comunes, otorgando de este modo atribuciones operativas características, tales como representación iconográfica y asociación a ***perfiles de acción***. En punto siguiente se aclara mejor este concepto.

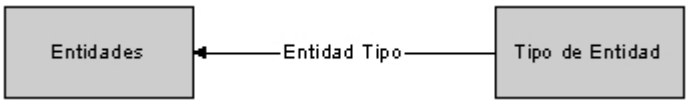


Figura 5-2 Categorización de las entidades

5.1.3 Perfiles de Acción

Las acciones de encuesta y modificación de variables de gestión se agrupan en perfiles de acción. De este modo se permite dichas variables puedan ser seleccionadas y agrupadas en una fase previa a las tareas de gestión, facilitándolas y flexibilizando su accionar.
Los perfiles de Acción son del tipo ***Encuesta*** o del tipo ***Modificación*** y están asociados a un ***Tipo de Entidad*** creado con el fin de clasificar las entidades a gestionar.

- ***Encuesta:*** Perfiles de acción que permiten la realización de encuestas a la las entidades de un ***grupo n,*** definido por el ***Tipo de Entidad*** asociado.

- ***Modificación:*** Perfiles de acción que permiten la modificación de variables de gestión en las entidades de un ***grupo n,*** definido por el ***Tipo de Entidad*** asociado.

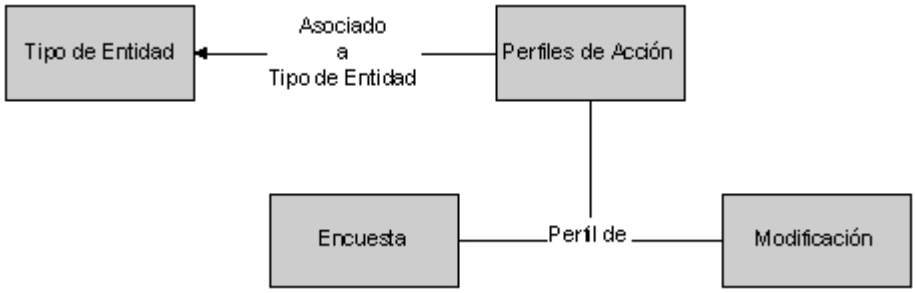


Figura 5-3 Categorización de los perfiles de acción

5.1.4 Reportes y Alertas

Un evento reportado por una entidad gestionada, puede cambiar de categoría dentro del sistema a medida que es procesado por los distintos bloques del sistema. De esta forma, según el gráfico de la Figura 5.4, cuando un evento es enviado por una entidad gestionada, si corresponde a alguno de los eventos configurados para ser reportados, se transforma en reporte y es tratado de acuerdo a dicha configuración. En un paso posterior, si un reporte corresponde a alguna de las alertas configuradas, entonces, el reporte se transforman en una o varias alertas y estas son enviadas a los usuarios de alerta que se encuentren previamente configurados a tal fin.
En resumen se definen:

- ***Reportes (Alarmas):*** Se configuran de manera que los eventos enviados por las entidades puedan ser procesados y reportados de acuerdo con esta configuración.

- ***Alertas:*** Se configuran para responder a reportes específicos. Estos reportes son procesados y enviados a los ***Usuarios de Alerta*****s** según sea especificado.

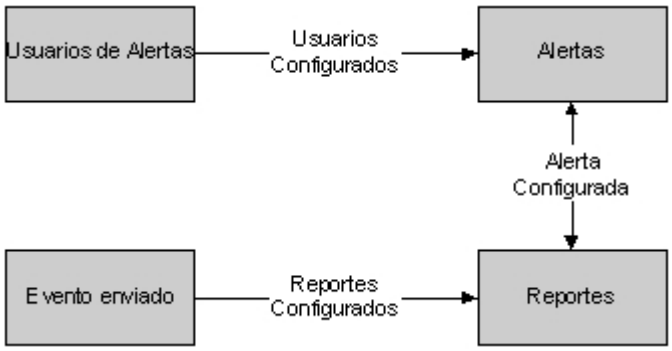


Figura 5-4 Tratamiento de eventos enviados por las entidades gestionadas al gestor

5.1.5 Presentación de la Información de gestión

Se define un sistema gráfico de localización de las entidades-equipos en un área geográfica determinada por un mapa, por ejemplo en el área definida por una

ciudad. Relacionar en la presentación gráfica la entidad-equipo con su ubicación geográfica permite organizar la gestión de entidades de manera más sencilla. Puede ser más representativa esta visión del sistema que un mapa de red por ejemplo y es la que suelen utilizar los sistemas de gestión más conocidos. A su vez, se define una la relación entre un mapa y otro diferente para permitir la navegación de datos a través de distintos niveles jerárquicos (mapas). Siguiendo con el ejemplo anterior, el icono de la ciudad dentro de un mapa de la provincia permitiría este tipo de cambio de nivel, lo mismo sucedería con el icono de una provincia dentro del mapa de una país. . Esta estructura de mapas y submapas, permite visualizar con el nivel de granularidad necesario la información gestión según sea el requerimiento del caso. Por ejemplo, tanto si el operador como el administrador desean ver el estado de todos las entidades-equipos dentro de una ciudad en comparación con los estados de entidades-equipos en otra ciudad de la provincia, solo tendrán que visualizar el mapa de la provincia que contenga los íconos de las ciudades que correspondan para realizar dicha comparación.
En resumen, una manera de obtener la información de configuración necesaria para soportar la organización lógica antes detallada, puede lograrse mediante la configuración localización geográficamente dentro de un mapa, según se detalla en el gráfico de la Fig. 5.5. Asimismo un mapa (submapa) puede configurarse para estar localizado dentro de otro mapa (mapa).

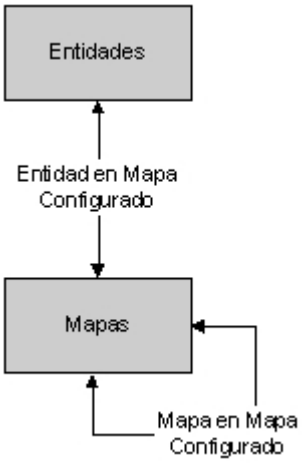


Figura 5-5 Relaciones entre Mapas, Submapas y entidades gestionadas

5.2 Diseño Arquitectónico

El diseño de la arquitectura de ***WBNMS*** se plantea desde una división en subsistemas del modelo funcional referenciado en la etapa de análisis del Gestor. Como consecuencia de ello, se consideran inicialmente los siete bloques funcionales como subsistemas de gestión.

- ***Administración:*** Se requiere que este subsistema realice todas las funciones administrativas. Las mismas implican el diálogo con el subsistema de ***Persistencia de Datos***, el monitoreo del estado operativo y el manejo de la autentificación del sistema.

- ***Persistencia de Datos:*** Este subsistema surge de la necesidad de mantener en forma persistente los datos de administración y gestión. Se pueden presentar tres posibles alternativas de diseño: utilización de archivos de ***texto plano***, interrelacionar el sistema con una ***Base de Datos*** de algún tipo (Relacional u Objetos), o una combinación de las anteriores. Aunque podría considerarse que esta es una decisión de implementación más que de diseño, influirá en el diseño de los demás subsistemas, por tratarse este del subsistema que centraliza todos los datos.

- ***Encuesta:*** Este subsistema será el encargado de realizar los procesos de encuesta a las entidades. Habrá que definir que protocolos utilizará en esta operación. También deberá realizar encuestas en forma automática de manera de "controlar" el estado de las entidades gestionadas (***keepalive***).

- ***Reporte:*** Este subsistema será el encargado de recibir y procesar los ***eventos*** enviados por las entidades gestionadas. Se deberá también definir en este caso cuál es el protocolo a utilizar.

- ***Modificación:*** Este subsistema será el encargado de "requerir" a las entidades la modificación de sus variables de gestión. Para ello también se deberá definir el protocolo a utilizar.

- ***Alerta:*** Consultará al subsistema de ***Persistencia de Datos*** en busca de ***reportes*** (alarmas), para luego procesar las ***alertas*** y enviarlas a los ***Usuarios de Alerta*** correspondientes (según el modelo de datos planteado). Atendiendo al requisito de "centralización" de los datos, esta operación se plantea en forma independiente del subsistema de ***Reporte***, ligándola a él únicamente por medio del subsistema de ***Persistencia de Datos*** (DB).

Uno de los principales puntos, que quedaron sin definición en la fase de análisis, es la definición de protocolos para el intercambio de la información en la red. Como condición de diseño se destaca la utilización de protocolos estándar en un entorno compatible con la Internet, con lo cual es inmediata la decisión de utilizar la pila de protocolos ***TCP/IP***. El protocolo de gestión que provee este conjunto se conoce como ***SNMP*** (Simple Network Management Protocol, ver Anexo - ***Protocolo de Gestión SNMP.***). Se trata de un protocolo de gestión que provee facilidades de reporte, encuesta y modificación remota de variables, todas requeridas para la solución ***WBNMS***. El protocolo SNMP utiliza mensajes básicos para lograr su funcionalidad: ***get – encuesta***, ***set – modificación*** y ***trap – reporte***. Los mensajes get permiten a la estación de mangement obtener el valor de los objetos en el agente. Los mensajes set, a su vez, permiten que el administrador configure valores de objetos en los agentes. Para permitir que el agente notifique al sistema de gestión eventos significativos se utilizan los mensajes trap. La estrategia más utilizada en este sentido es que la central encueste a los agentes una vez por día o en el momento de la configuración como para establecer una línea de partida , a partir de la cual los agentes se vuelven los responsables de notificar eventos inusuales. Una vez que sucede esta condición de excepción, uno de los posibles cursos de acción consiste en dirigir encuestas al agente que se ha notificado con el

mensaje de trap o a alguno cercano, de tal manera de obtener mayor información respecto del evento.
Como se puede observar, no es posible cambiar la estructura de la base de datos por el agregado o borrado de objetos, ni editar comandos para que se lleve a cabo una acción. Esto limita la capacidad del sistema de gestión de red, pero permite simplificar la implementación del protocolo.
Por otro lado, el protocolo SNMP fue diseñado a nivel de capa de aplicación en el modelo TCP/IP para ser operado sobre el protocolo UDP. La ventaja de esta elección es que no existen conexiones entre la estación central y los agentes que se están monitoreando, sino que cada intercambio entre ellos es una transacción separada.
La información de gestión manejada por este protocolo se encuentra almacenada en la ***Base de Información de Gestión*** (MIB). Se trata de la colección de objetos que representan recursos a ser gestionados en el elemento agente. Cada nodo en la red mantiene una MIB que representa el estado de los recursos gestionados en ese nodo, de tal manera que los mismos pueden leerse para monitoreo o modificarse para control. Se utiliza un esquema común de representación para que el sistema soporte interoperabilidad.
Por ello el diseño deberá incluir un nuevo subsistema denominado ***Compilación de MIBs***; que será el encargado de interpretar los datos de las MIBs incorporadas, específicamente a la MIB del sistema.

- ***Compilación de MIBs:*** Este subsistema se encargará de compilar la información residente en las MIBs necesaria para cumplir con las tareas de gestión. Se buscará que los resultados queden almacenados dentro del subsistema de ***Persistencia de Datos***, conformándose así la MIB del sistema.

- ***Interfaz:*** Este subsistema se encargará de brindar "servicios" a la Interfaz de Usuario propiamente dicha. Dentro de estos servicios, principalmente se encuentran los de consulta al subsistema de ***Persistencia de Datos***, llamada a funciones administrativas, manejo de errores y tareas de preprocesamiento de datos ingresados por el Usuario. Además deberá proveer una forma de visualización estándar de la MIB del sistema.

5.2.1 Módulos Funcionales

Se dividirá al sistema en módulos funcionales, de manera de cumplir con las funciones planteadas para cada uno de los subsistemas. Esto tiene como objetivo acercar los modelos de análisis a la realidad "práctica" de la implementación. La Fig. 5.6 representa la arquitectura completa y resalta el contraste entre los subsistemas y los módulos funcionales que a continuación se detallan.
Se resaltan al frente del gráfico los ocho subsistemas en colores y en el fondo los distintos módulos funcionales. Es importante destacar que existen módulos funcionales compartidos con el subsistema de Administración.

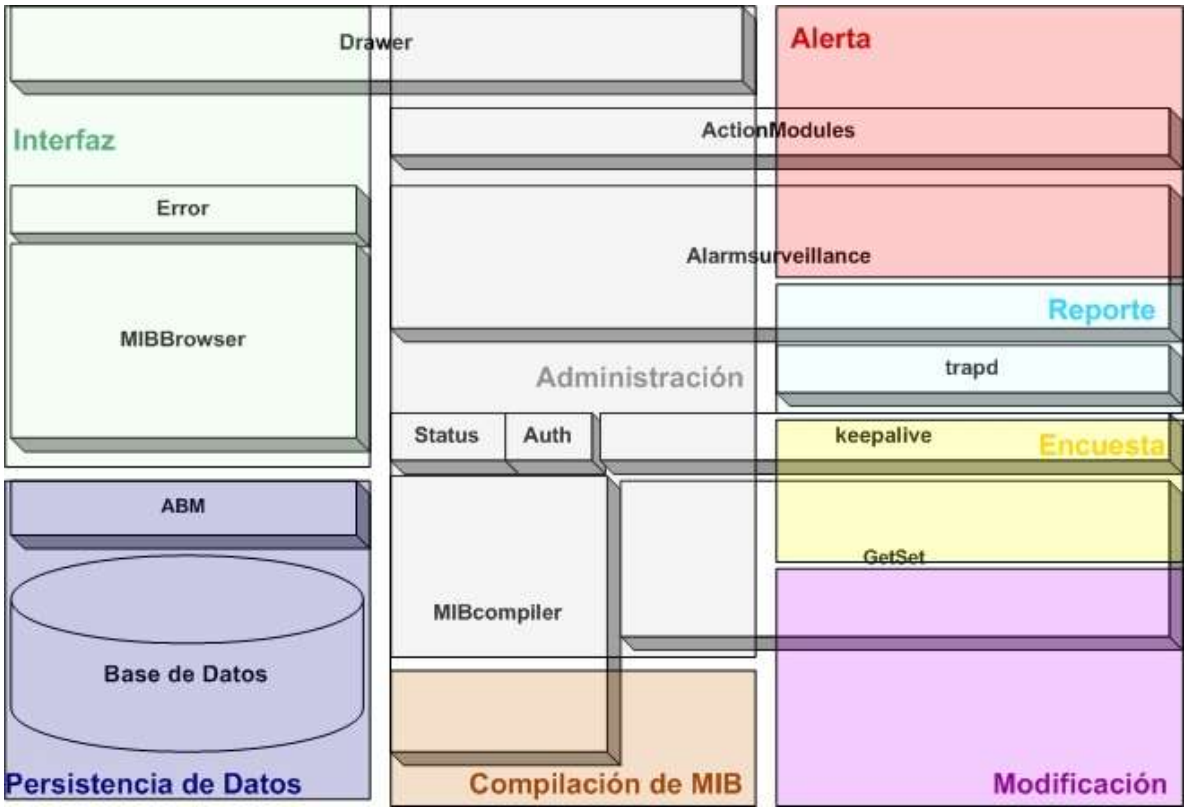


Figura 5-6 Módulos funcionales por subsistema

5.2.1.1 Módulos del subsistema de Encuesta

Una vez definido el protocolo a utilizar, ***SNMP***, el proceso de encuesta se realizará utilizando los mensajes estándares para encuesta de dicho protocolo: ***GetRequest*** y ***GetResponse***. Para poder construir la unidad de datos de protocolo (PDU) ***GetRequest PDU*** y transmitirlo posteriormente, el subsistema deberá conocer como primera cuestión el nombre de la ***comunidad***, este término se relaciona con un concepto de "password" para la gestión en SNMP, más específicamente con una política de control de acceso. También deberá conocer la dirección IP de la entidad gestionada y los ***OID*** (Object Identifier) que identifican a cada una de las variables-objetos que desean ser encuestados en esa entidad. El subsistema de ***Persistencia de Datos*** será consultado por el de ***Encuesta*** para obtener estos datos, relativos a cada una de las entidades gestionadas. El mensaje ***GetRequest*** será respondido por la entidad con el mensaje ***GetResponse***, con el mismo identificador de requerimiento, para match de requerimiento-respuesta. Se empaquetarán en dicha respuesta los datos requeridos o el mensaje de error que corresponda. Luego, el subsistema ***Encuesta*** procesará la respuesta de la entidad y reportará el resultado al subsistema que haya requerido el proceso de encuesta.
Según el modelo de datos planteado, se definirán ***Perfiles de Encuesta***, de manera de ordenar el funcionamiento del subsistema ***Encuesta*** en forma lógica. Cada uno de estos perfiles agruparan ***n variables-objetos*** de gestión, de manera que al seleccionar uno de ellos, la entidad responda por las ***n variables*** del perfil que se encuentra definido en el modelo de datos.

Esta función del subsistema ***Encuesta*** será cubierta por parte del módulo ***GetSet***, aquella relacionada con Get – encuesta. La ***encuesta automática*** también planteada para este subsistema será realizada en cambio por el módulo funcional ***keepalive***

5.2.1.1.1 Keepalive

Este módulo cumple la función de monitorear a intervalos regulares el estado de las entidades gestionadas y descubrir posibles nuevas entidades (***autodiscovery***) en un entorno prefijado, como podría ser un rango de direcciones IP o subredes. Los resultados de este proceso se deben almacenar para que el ***usuario-cliente*** considere el curso de acción a seguir: incorporarlos a la gestión o descartarlos. Este módulo operará en forma "independiente" del resto del sistema, pero su interfaz con el mismo son los reportes de sus resultados almacenados en la Base de Datos.

5.2.1.1.2 GetSet

El módulo ***GetSet*** realiza la función de "manejar" los mensajes PDU de encuesta asociados a SNMP, mencionados en puntos anteriores. Para ello procesa los mensajes de requerimiento y respuesta de encuesta, verificando su validez contra la MIB del sistema. Las respuestas de estas operaciones de encuesta son reportadas a la interfaz para ser dirigidas (forward) al cliente que las haya requerido. No se realizará un almacenamiento en forma persistente de los resultados de encuestas realizadas, ya que no es un requisito planteado para la solución, siendo las encuestas un recurso auxiliar a la tarea de gestión.

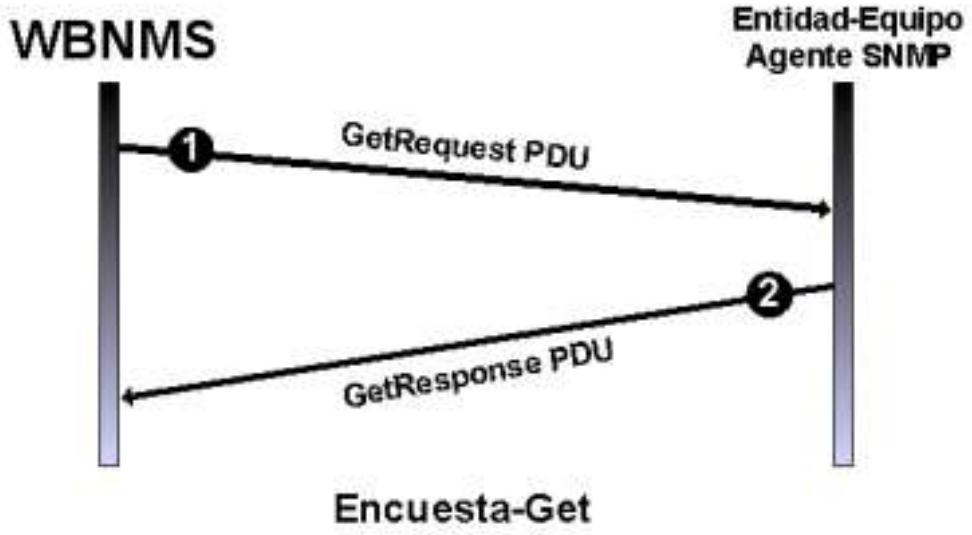


Figura 5-7 Intercambio de Mensajes de Encuesta

La Fig. 5.7 representa el intercambio de mensajes de encuesta. En este punto es conveniente recordar que la operación de encuesta es atómica: o se contesta información sobre todos los valores encuestados o no se contesta ninguno. Esto genera condiciones de error que aparecen en las respuestas cuando, por ejemplo, el nombre de la variable encuestada no coincide con ningún identificador de la MIB, el tamaño de la respuesta es muy grande para que el agente pueda manejarlo o por

algún otro motivo un valor de variable no puede ser presentado. También vale la pena recordar que SNMP sólo permite la encuesta sobre objetos que son "hojas" del "árbol". Por ejemplo, para encuestar los valores de una tabla hay que identificarlos de manera individual, no se puede solicitar una fila de la tabla con una sola referencia utilizando el mensaje GetRequest, para ello es necesario implementar un mensaje del tipo GetNextRequest, no implementado para WBNMS. Este tipo de mensaje permite leer tablas dinámicamente.

5.2.1.2 Módulos del subsistema de Modificación

Los mensajes definidos por el protocolo ***SNMP*** para la modificación de variables-objetos de gestión son: ***SetRequest*** y ***GetResponse***. El primero de ellos tiene el mismo formato PDU que ***GetRequest***, la diferencia es que se utiliza para modificar valores de los objetos, no para leerlos. El subsistema deberá contar con los datos de las variables a modificar, esto son: ***OIDs*** y tipos de las variables (definidas por el protocolo). Así como también los datos de la entidad (IP, comunidad).Estos datos se consultarán al subsistema de ***Persistencia de Datos***. Los nuevos valores de las variables a modificar son suministrados por el cliente que solicita el proceso de modificación. La operación, como sucede en el caso de encuesta, es atómica y, por tanto, sujeta a las mismas condiciones de error.
Cuando el cliente solicita el proceso de modificación de una entidad específica, selecciona las variables a modificar y sus nuevos valores. El subsistema ***Modificación*** construye y envía, a la entidad específica, el mensaje ***SetRequest.*** La entidad responde con el mensaje ***GetResponse,*** en el cual declara las variables-objetos modificadas o el error producido. Luego, el subsistema ***Modificación*** procesa esta respuesta y la reporta al cliente. Según el modelo de datos planteado, se definen ***Perfiles de Modificación*** para agrupar las variables-objetos de entidades gestionas que son plausibles de modificación, es decir que tienen permiso de escritura. Las funciones de este subsistema serán cumplidas por el módulo funcional ***GetSet*** (específicamente la parte relacionada con Set - modificación).

5.2.1.2.1 GetSet

Realiza la función manejar los mensajes PDU de modificación (Set) del protocolo SNMP. Procesa los mensajes de requerimiento y respuesta verificando su validez contra la MIB del sistema. Las respuestas de estas operaciones de encuesta son reportadas a la interfaz para ser dirigidas luego al cliente que requiriera de ellas. Al igual que con los resultados de las encuestas realizadas, no se realizará un almacenamiento persistente de los resultados del proceso de modificación.
La siguiente figura representa el intercambio de mensajes de modificación de variables en las entidades (Set).

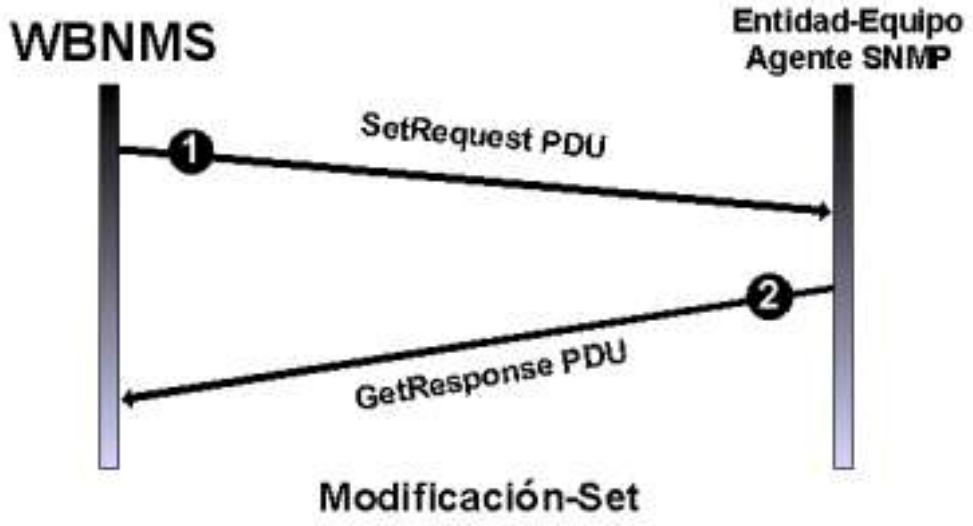


Figura 5-8 Intercambio de Mensajes de Modificación

5.2.1.3 Módulos del subsistema de Reporte

El mensaje ***PDU*** definido por ***SNMP*** para el envío de reportes es ***Trap.*** Se utiliza para proveer al sistema de gestión de un medio de notificación asincrónica de algún evento significativo. El formato de mensaje varia de una versión de protocolo a otra. En rasgos generales, la información que la entidad envía en estos mensajes es la misma, cambiando principalmente el formato. Suponiendo un mensaje genérico ***Trap-SNMPvn (versión n)***, se puede decir que la entidad envía el mensaje para reportar un evento de gestión predefinido, ocurrido o reportado en la misma, este último en el caso que la entidad actúe como Proxy. El subsistema de ***Reporte*** procesará el mensaje recibido y almacenará el resultado del mismo a través del subsistema de ***Persistencia de Datos***. Durante el procesamiento del mensaje, este será reconocido como un reporte configurado o ***alarma***, según el modelo de datos de Reportes. Esta diferenciación se hace con el fin de descartar mensajes indeseados, tales como mensajes erróneos, redundantes, no importantes, etc.. Tanto los mensajes deseados como los no deseados, son reportados para no perder la ocurrencia del evento. El evento "siempre tendrá valor", y corresponde al cliente decidir cuando eliminarlo del sistema.
Las funciones de este subsistema serán cumplidas por el módulo funcional ***trapd*** y por parte del módulo ***Alarmsurveillance***.

5.2.1.3.1 Trapd

Cumple la función de recibir y preprocesar los reportes envidados por las entidades, cumpliendo con el protocolo SNMP. Interpreta el mensaje ***trap PDU*** y verifica su validez con la MIB del sistema. Almacenará el resultado para su posterior procesamiento por otro módulo del sistema. La naturaleza del modo de operación de este módulo sugiere que actúa en forma aislada del resto del sistema, utilizando como interfaz la Base de Datos.
La Figura 5.9 describe el camino que recorren los mensajes de reporte (traps) desde la entidades gestionadas hacia el gestor.

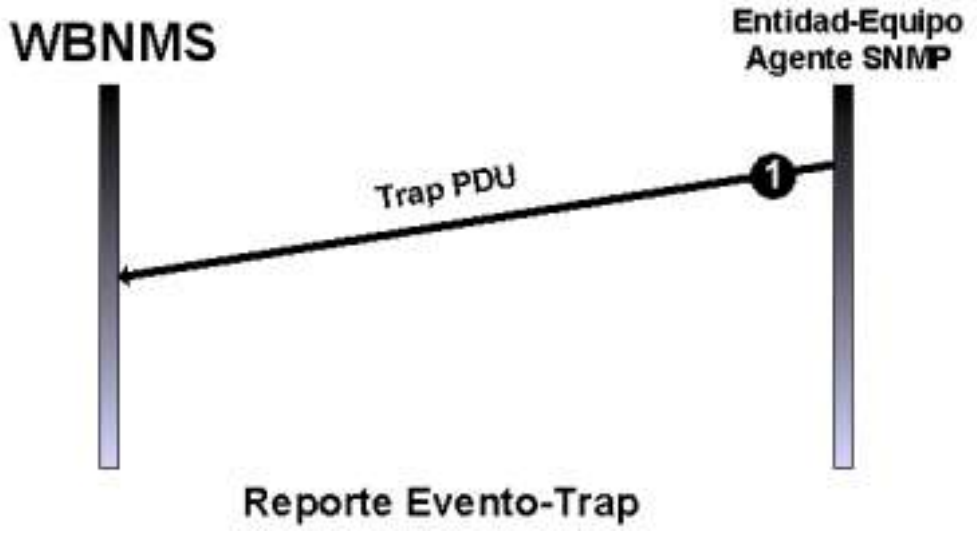


Figura 5-9 Intercambio de Mensajes de Reporte de Eventos

5.2.1.3.2 Alarmsurveillance

Este módulo vigila la aparición de nuevos eventos procesados por el módulo ***trapd*** y los procesa como reportes válidos (alarmas) o los descarta. El procesamiento incluye análisis de severidad del reporte, según la configuración establecida, y cambio de estado, en las entidades y mapas, relacionado en el sistema según la severidad. Este procesamiento asegura el reporte gráfico coherente del estado de las entidades según los reportes configurados (alarmas) en el sistema.

5.2.1.4 Módulos del subsistema de Alerta

El subsistema de Alerta constituye una "extensión" del sistema de gestión planteado para una red gestora basada en ***SNMP***. Es función de este subsistema actuar de "Proxy", de intermediario que maneja el flujo de eventos reportados por las entidades al gestor, para ser enviados a los ***Usuarios de Alertas .*** Así, se definirán distintos ***Modos de Alerta*** asociados a la forma de cómo serán enviadas dichas alertas.
Para la solución ***WBNMS***, se definen entonces dos ***Modos de Alerta***:

- ***Proxy SNMP:*** Los mensajes reportados por las entidades, eventos que sean procesados, alarmas y aquellos que sean reconocidos como ***Alertas*** válidas asociadas a este ***modo de alerta***, serán reenviados vía SNMP a los ***Usuarios de Alerta*** (forwarding de alarmas). En este caso, dichos Usuarios serán otros sistemas gestores, identificados por su nombre, número IP y comunidad.
- ***Proxy SMTP:*** Los mensajes reportados por las entidades que sean procesados y reconocidos como alertas válidas asociadas a este ***modo de alerta***, serán enviados a los ***Usuarios de Alerta*** vía e-mail. Cada uno de estos usuarios es identificado por su nombre y casilla de e-mail.

Este subsistema debe ser capaz de adaptarse fácilmente a la incorporación de nuevos modos de alerta. Las funciones del mismo serán cumplidas por el módulo funcional ***ActionsModules*** y por parte del módulo ***Alarmsurveillance***.

5.2.1.4.1 ActionsModules

Cubre las funcionalidades de los distintos ***modos de alerta***. Esto es, procesar los datos de las alertas y enviarlas a los ***Usuarios de Alertas*** configurados. Posee, por lo tanto, la capacidad de manejar todos los protocolos involucrados.

5.2.1.4.2 Alarmsurveillance

Este módulo no sólo cumple funciones para el subsistema de ***Reporte***, sino que además una parte importante del mismo se encarga de procesar los reportes, convirtiéndolos en alertas según la configuración, verificar los ***usuarios de alertas*** asociados y requerirle al módulo ***ActionsModules*** el envío de las alertas a esos usuarios.

5.2.1.5 Módulos del subsistema de Compilación de MIB

Al tomarse la decisión de utilizar ***SNMP*** como protocolo de Gestión, se plantea la necesidad de poder codificar las variables-objetos de gestión de acuerdo con lo especificado para dicho protocolo. Esta información de gestión se encuentra definida en las MIB's "genéricas" que describen variables-objetos estándares y en las MIB's "privadas", relacionadas a las entidades-equipos a gestionar. La definición de MIB ha sido diseñada para acomodar la posibilidad de crecimiento y, en este sentido, flexibilidad para agregar nuevos objetos. Debido a la estandarización de la información de gestión estructurada (SMI) y del esquema de identificación de objetos, es posible gestionar objetos desde estaciones de gestión bajo el concepto de interoperabilidad. El formato definido para las MIB's es el de Archivos de Texto codificado en ***ASN.1*** (ver Anexo - ASN.1_Abstract Syntax Notation One). Para que una estación de gestión pueda gestionar objetos MIB privados, la estructura MIB privada debe ser cargada en la misma antes de poder ofrecerla a los usuarios. Para que el sistema pueda interpretar la información de gestión, habrá que poder compilarla previamente en un formato reconocido por el sistema. Se define un módulo único para este propósito, el subsistema ***MIBCompiler.***

5.2.1.5.1 MIBCompiler

Este módulo compila la información de las MIB's y la almacena en la Base de datos. La declaración de Objetos de gestión definidos en una MIB varía según la versión de SNMP utilizado. Se tratará de lograr un subsistema de Compilación de MIB que sea escalable, es decir que puedan incorporarse nuevas variables de

compilación sin modificar las anteriores. Se plantean seis tipos de objetos de MIB a ser compilables por esta versión de ***WBNMS***. Se trata de:

- ***OBJECT IDENTIFIER:*** define el orden jerárquico dentro de la MIB.
- ***OBJECT-TYPE:*** identifica y define a un objeto dentro de la MIB.
- ***TRAP-TYPE:*** define e identifica mensajes PDU trap para SNMP v1.
- ***NOTIFICATION-TYPE:*** define e identifica mensajes PDU trap para SNMP v2.
- ***SEQUENCE:*** define una secuencia de objetos dentro de la MIB.
- ***TEXTUAL-CONVENTIONS:*** define convenciones textuales dentro de la MIB para SNMPv2.En el caso de SNMPv1, este tipo de declaraciones se realizan implícitamente en la definición de objetos.

La compilación de una MIB se hará una sola vez, en el momento de su incorporación a la gestión, o sea a la MIB del sistema. Luego los datos de la misma serán consultados desde la Base de Datos.

5.2.1.6 Módulos del subsistema de Interfaz

El subsistema Interfaz está dedicado a la mediación entre los clientes y el resto del sistema. Esta característica de cliente-servidor junto con el requisito de acceso masivo planteado en los requisitos de la solución, determinan la elección de una ***interfaz Web*** como la más adecuada. Para poder ofrecer los servicios que de él se esperan, este subsistema tendrá como base un ***servidor Web*** capaz de manejar aplicaciones del lado servidor. De esta manera podrá interactuar con el resto del sistema ***WBNMS*** y presentar los resultados pertinentes al cliente. Este deberá contar con un navegador Web estándar como único requisito de compatibilidad para acceder al sistema. De esta manera se simplifica el diseño ya que no se precisa un software diseñado especialmente para una funcionalidad y, por lo tanto, costoso. Más adelante se detalla el flujo de navegación con el que deberá cumplir la aplicación para satisfacer los requisitos operacionales.
Según lo planteado, este subsistema deberá interpretar el lenguaje ***HTML*** y manejar una interfaz CGI. Teniendo en cuenta esto, los módulos funcionales asociados a este subsistema se reducen a: un módulo de presentación de errores, ***Error***; otro de Navegación de MIB's, ***MIBBrowser***; y un tercero de manejo del entorno gráfico de gestión, ***Drawer.***

5.2.1.6.1 Error

Centraliza las funciones de reporte de error del sistema. Formatea en lenguaje HTML el mensaje de error a ser presentado al usuario. El objetivo es que todo mensaje de error, tanto sea de administración , como ser la carga de MIBs , creación de usuarios, etc. o mensajes de error de la gestión derivados del intercambio SNMP sean tratados por este módulo funcional.

5.2.1.6.2 MIBBrowser

Interpreta los datos de la MIB del sistema almacenada en la Base de Datos y los formatea para su presentación a través de la interfaz Web. El formato elegido, por ser el formato clásico, es el de un árbol de objetos de MIBs jerárquicamente ordenados, siendo de esta forma posible su navegación en busca de detalles acerca de cada uno de los objetos de la misma.

5.2.1.6.3 Drawer

Interpreta la información de las entidades-equipos y mapas, así como sus estados operativos reportados (eventos recibidos), y crea una representación grafica conjunta, que describe el estado "actualizado" del universo de gestión. Además, construye una representación HTML que facilita la navegación jerárquica de mapas (definida en el modelo de datos planteado) y consulta de datos de las entidades-equipos.
Este módulo funcional actuará a intervalos regulares preestablecidos, dependiendo de la frecuencia de actualización de presentación requerida por el cliente.

5.2.1.7 Módulos del subsistema de Persistencia de Datos

Dado el requisito de modularidad y escalabilidad en el modelo de datos, se recurrirá a la utilización de una Base de Datos como repositorio "principal" de los datos de administración y gestión del sistema ***WBNMS***. Es necesario tomar una decisión de implementación referida al tipo de base de datos a utilizar: Relacional u Objetos. Esta decisión afectará significativamente el tipo de diseño de la base de datos y por lo tanto a su resultado final. Además este subsistema deberá proveer un módulo funcional que centralice el almacenamiento y consulta en dicha Base de Datos. Se plantea entonces el módulo ***ABM***.

5.2.1.7.1 Base de Datos

Tomando como premisa en nuestro diseño del subsistema de persistencia de datos la utilización de una ***Base de Datos***, se planteará a continuación los criterios adoptados para su diseño, así como también sus componentes más sobresalientes.

5.2.1.7.1.1 Diseño de la Base de Datos

Para el diseño de la estructura de la base de datos se siguió y amplió el modelo de datos planteado en la primer parte del diseño. Se cubren, de esta manera los requisitos de manejo de datos de todos los subsistemas que componen ***WBNMS***. La siguiente figura contiene una representación gráfica de las diferentes tablas que componen la Base de Datos ***WBNMSDB***, expresando mediante flechas las relaciones que existen entre ellas.

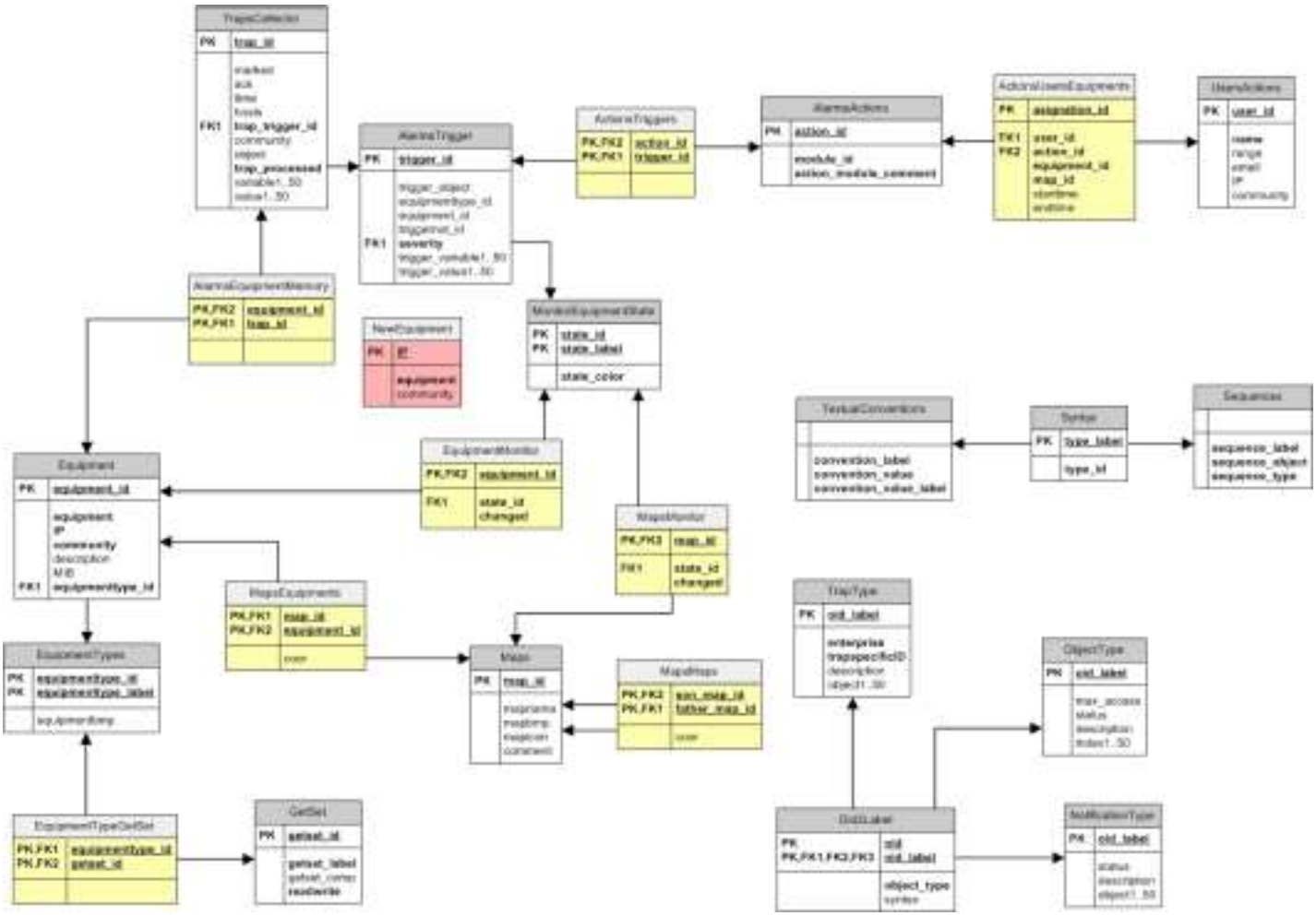


Figura 5-10 Diagrama de la Base de Datos *WBNMSDB*

Para facilitar las visualización de las tablas es que se divide el gráfico anterior y se listan a las distintas tablas que constituyen la Base de Datos ***WBNMSDB*** del sistema ***WBNMS***:

- ***AlarmsTrigger:*** Esta tabla posee los datos de configuración de reportes, denominados ***triggers***. Las columnas de la misma representan:
  - ***trigger_id :*** Es el identificador de cada fila de datos. Representa la ***llave principal de la tabla***[5].
  - ***trigger_object:*** representa el nombre de reporte configurado.
  - ***equipmenttype_id:*** relacionada con los tipos de entidad-equipo.
  - ***equipment_id:*** relacionada con las entidades-equipos.
  - ***triggernot_id:*** identifica que otra fila de la tabla reconoce (ACK) el reporte definido en esta fila.
  - ***severity:*** identifica la severidad asociada al evento representado en cada fila. La definición de las severidades se encuentra en la tabla ***MonitorEquipmentState***.
  - ***trigger_variable1..50***[6]***:*** representa 50 columnas asociadas a posibles variables de gestión asociadas al reporte.

[5] Identifica unívocamente cada línea de la tabla (objeto).

- ***trigger_values1..50:*** representa 50 columnas asociadas a los posibles valores de las variables de gestión asociadas al reporte.

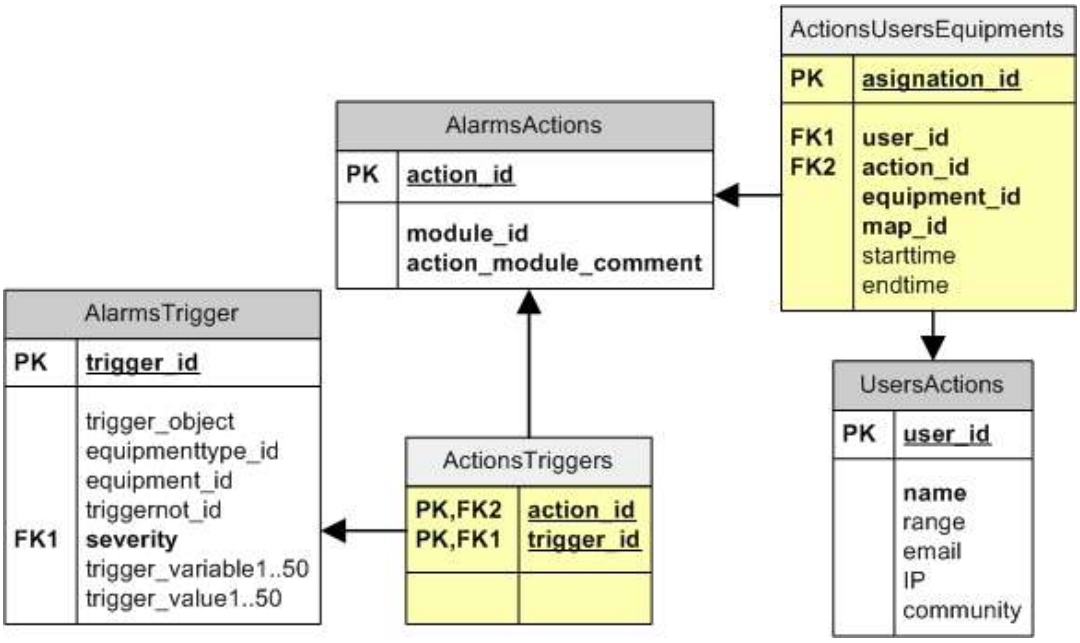


Figura 5-11 Tablas AlamsTrigger,AlarmsActions,ActionsTriggers,ActionsUsersEquipmentes y UsersActions

- ***AlarmsActions:*** Esta tabla define los tipos de alerta definidos en el sistema (SNMP, SMTP,etc.). Las columnas de la misma representan:
  - ***action_id:*** define unívocamente al tipo de alerta. Es la llave principal de la tabla.
  - ***module_id:*** identifica al módulo que realiza la acción de alerta.
  - ***action_module_comment:*** comentario respecto del modulo.

- ***ActionsTriggers:*** Es la Tabla que relaciona un tipo de alerta con una configuración de reporte, ***trigger***. Las columnas de la misma representan:
  - ***action_id:*** asociada con la columna del mismo nombre de la tabla ***AlarmsActions***. Es parte de la llave principal.
  - ***trigger_id:*** asociada con la columna del mismo nombre de la tabla ***AlarmsTrigger***. Es parte de la llave principal.

- ***UsersActions:*** Esta tabla define los usuarios de alerta. Las columnas de esta tabla representan:
  - ***user_id:*** define unívocamente al usuario de alertas. Es la llave principal de la tabla.
  - ***name:*** representa el nombre del usuario.
  - ***range:*** rango del usuario, usuario común o administrador.
  - ***email:*** casilla de email del usuario.

[6] Por razones de espacio se adopto esta notación (***1..n***) para representar una serie de n columnas.

- ***IP:*** número de IP del usuario.
- ***community:*** comunidad del usuario.

- ***ActionsUsersEquipments:*** Esta tabla asocia usuarios y tipos de alertas, permitiendo configurar por entidad-equipo, mapa, turno(starttime-endtime). Sus columnas definen:

  - ***asignation_id:*** define unívocamente la asociación realizada. Es la llave principal de la tabla.
  - ***user_id:*** asociada con la columna del mismo nombre de la tabla ***UsersActions***.
  - ***action_id:*** asociada con la columna del mismo nombre de la tabla ***AlarmsActions***.
  - ***equipment_id:*** relacionada con una entidad-equipo. De valor 0 si no esta relacionada. Indica que el usuario es responsable de dicha entidad-equipo.
  - ***map_id:*** relacionada con un mapa. De valor 0 si no esta relacionada. Indica que el usuario es responsable de dicha zona (mapa).
  - ***starttime:*** inicio (hora:minutos) del turno asociado al usuario de alerta.
  - ***endtime:*** finalización (hora:minutos) del turno asociado al usuario de alerta.

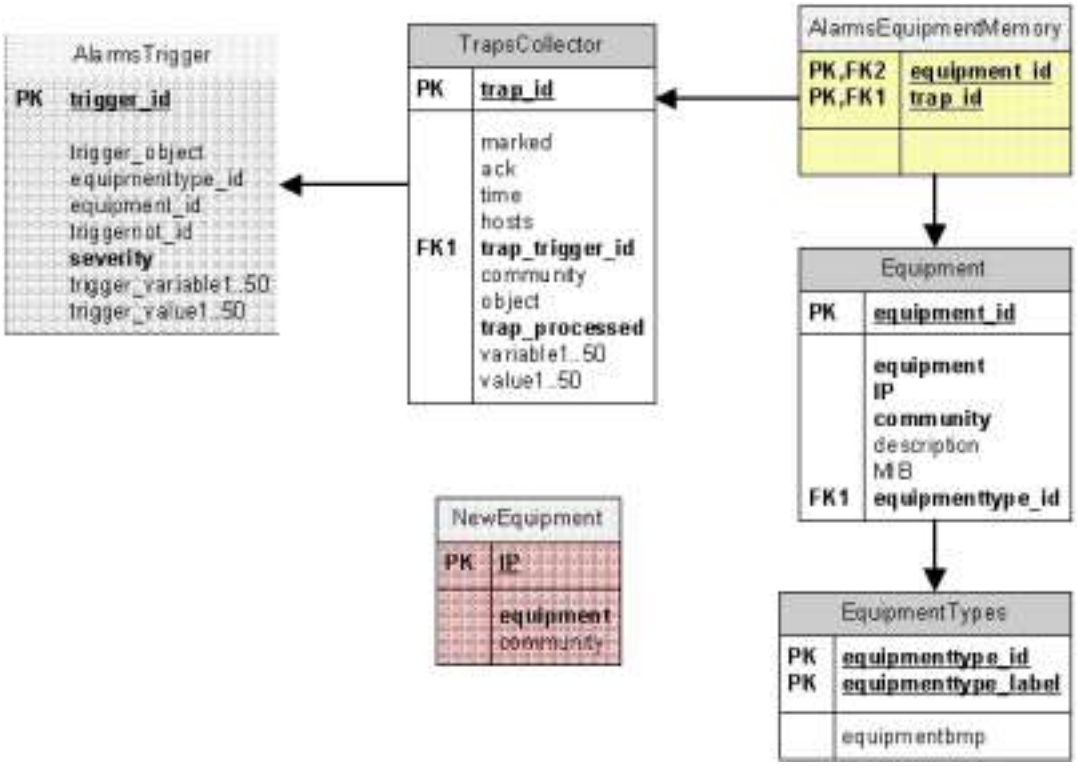


Figura 5-12 Tablas TrapsCollector,AlarmsEquipmentMemory,Equipment,EquipmentTypes y NewEquipment

- ***TrapsCollector:*** Los eventos reportados por las entidades-equipos son almacenados luego de ser procesados por el subsistema de Reporte en esta tabla. Las columnas representan:

  - ***trap_id:*** identifica unívocamente al evento reportado. Es la llave principal de la tabla.
  - ***marked:*** indica con un valor distinto de *Null, ACK,* que el evento ha sido reconocido por un usuario-cliente.
  - ***ack:*** será el <trap_id> que realizo *el ACK* (según marked) del evento (acción realizada por el sistema) o el <Usuario> (especificado para un usuario-cliente en particular) que realizo la acción de *ACK* manualmente.
  - ***time:*** indica el tiempo en el que fue recibido el evento-trap.
  - ***hosts:*** indica la dirección IP de la entidad-equipo que reportó el evento-trap.
  - ***community:*** indica la comunidad (SNMP) con el que fue reportado el evento-trap.
  - ***object:*** indica el objeto de gestión que fue reportado por la entidad-equipo.
  - ***variable1..50:*** 50 posibles variables definidas para el objeto de gestión reportado.
  - ***value1..50:*** valor de las 50 variables de gestión.
  - ***trap_trigger_id:*** asociado a un reporte configurado (trigger) de la tabla ***AlarmsTriggers*** por índice ***trigger_id*** o null si no existe un reporte compatible.
  - ***trap_processed:*** indicación interna del sistema de si el evento-trap ha sido procesado como reporte (1) o no (0).

- ***Equipment:*** Esta tabla contiene las entidades-equipos gestionadas por ***WBNMS*** . Sus columnas definen:

  - ***equipment_id:*** define unívocamente a la entidad-equipo. Es la llave principal de la tabla.
  - ***equipment:*** nombre de la entidad-equipo.
  - ***IP:*** dirección IP de la entidad-equipo.
  - ***community:*** comunidad SNMP de la entidad equipo.
  - ***description:*** descripción acerca de la entidad-equipo.
  - ***MIB:*** MIB de referencia de entidad-equipo.
  - ***equipmenttype_id:*** define a de cuál de los tipos definidos en la tabla ***EquipmentTypes*** es la entidad-equipo.

- ***EquipmentTypes:*** En esta tabla se encuentran definidos los tipos de entidades-equipo. Sus columnas definen:

  - ***equipmenttype_id:*** define unívocamente al tipo de entidad-equipo. Es la llave principal de la tabla.
  - ***equipmenttype_label:*** es el nombre asociado al tipo de entidad-equipo.

- ***equipmentbmp:*** asigna una imagen para representar a las entidades-equipos de este tipo.

- ***AlarmsEquipmentMemory:*** Esta Tabla asocia una entidad-equipo con un evento-trap recibido. Las columnas de esta tabla definen:

  - ***equipment_id:*** asociada con la columna del mismo nombre de la tabla ***Equipments***. Es parte de la llave principal.
  - ***trap_id:*** asociada con la columna del mismo nombre de la tabla ***TrapsCollectos***. Es parte de la llave principal.

- ***NewEquipment:*** En esta tabla quedan almacenadas las entidades-equipos nuevas que han sido descubiertas por el sistema mediante un proceso de encuesta automática. Las columnas definen:

  - ***IP:*** dirección IP de la entidad-equipo encontrada. Es la llave principal de la tabla.
  - ***community:*** comunidad de la entidad-equipo encontrada.

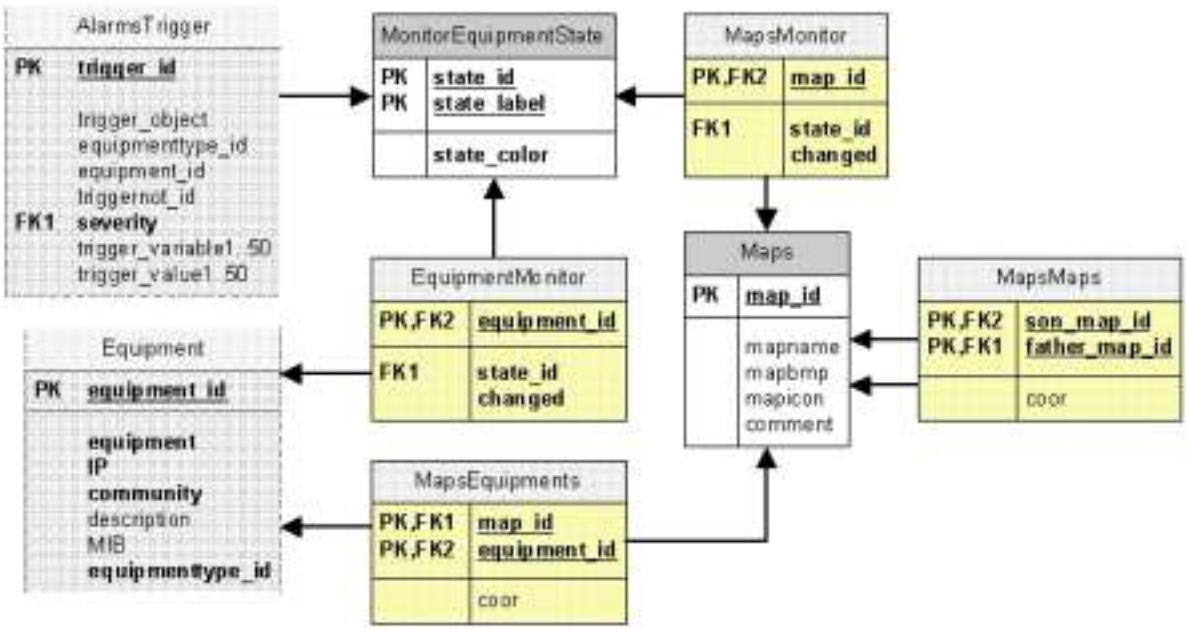


Figura 5-13 Tablas MonitorEquipmentState,EquipmentMonitor,MapsEquipments,Maps,MapsMonitor y MapsMaps

- ***MonitorEquipmentState:*** En esta tabla se definen los estados posibles que pueden tener las entidades-equipos referidas a los reportes recibidos. Las columnas de la misma definen:

  - ***state_id:*** define numéricamente al estado. Es parte de la llave principal de la tabla.
  - ***state_label:*** define el nombre del estado. Es la otra parte de llave principal de la tabla.

- ***state_color:*** define el color que representa al estado.

- ***Maps:*** En esta tabla se encuentran definidos los mapas configurados del sistema. Las columnas de esta tabla definen:

  - ***map_id:*** define unívocamente al mapa. Es la llave principal de la tabla.
  - ***mapname:*** nombre del mapa.
  - ***mapbmp:*** vista ampliada del mapa.
  - ***mapicon:*** icono que representa al mapa.
  - ***comment:*** comentario asociado al mapa.

- ***EquipmentMonitor:*** En esta tabla se monitorea el estado de las entidades-equipos gestionadas. Las columnas definen:

  - ***equipment_id:*** asociada a la columna del mismo nombre de la tabla ***Equipment***. Es la llave principal de la tabla.
  - ***state_id:*** asociada a la columna del mismo nombre de la tabla ***MonitorEquipmentState***.
  - ***changed:*** indicador del sistema que señala con un "1" que se ha modificado el estado de la entidad-equipo y con un "0" que sigue estable.

- ***MapsMonitor: :*** En esta tabla se monitorea el estado de los mapas. Sus columnas definen:

  - ***map_id:*** asociada a la columna del mismo nombre de la tabla ***Maps***. Es la llave principal de la tabla.
  - ***state_id:*** asociada a la columna del mismo nombre de la tabla ***MonitorEquipmentState***.
  - ***changed:*** indicador del sistema que señala con un "1" que se ha modificado el estado del mapa y con un "0" que sigue estable.

- ***MapsEquipments:*** Esta tabla asocia los mapas definidos con las entidades-equipos. Las columnas de esta tabla definen:

  - ***map_id:*** asociada a la columna del mismo nombre de la tabla ***Maps***. Es parte de la llave principal de la tabla.
  - ***equipment_id:*** asociada a la columna del mismo nombre de la tabla ***Equipment***. Es parte de la llave principal de la tabla.
  - ***coor:*** define la posición en coordenadas (x:y) de la entidad-equipo respecto del mapa.

- ***MapsMaps:*** Esta tabla asocia los mapas (hijos) con otros mapas (padres). Las columnas de esta tabla definen:

- ***son_map_id:*** asociada a la columna ***map_id*** de la tabla ***Maps***. Representa al mapa hijo relativamente hablando. Es parte de la llave principal de la tabla.
- ***father_map_id:*** asociada a la columna ***map_id*** de la tabla ***Maps***. Representa al mapa raíz o padre relativamente hablando. Es parte de la llave principal de la tabla.
- ***coor:*** define la posición en coordenadas (x:y) del mapa hijo respecto del mapa padre.

Figura 5-14 Tablas EquipmentTypeGetSet y Getset

- ***Getset:*** En esta tabla se encuentran definidos los perfiles de Acción, tanto los de Modificación (Set) y los de Encuesta (Get). Las columnas de esta tabla representan:

  - ***getset_id:*** define unívocamente al perfil de acción. Es la llave principal de la tabla.
  - ***getset_label:*** nombre asociado al perfil de acción.
  - ***getset_comp:*** objetos de gestión que son componentes del perfil de acción.
  - ***readwrite:*** identifica al tipo de perfil con un "1" es de lectura-escritura (Modificación) y con un "0" si es de solo lectura (Encuesta).

- ***EquipmentTypeGetSet:*** Esta tabla asocia los perfiles de acción definidos en la tabla ***GetSet*** y los tipos de entidades-equipos definidos en la tabla ***EquipmentTypes***. Las columnas de esta tabla representan:

  - ***equipmenttype_id:*** asociada a la columna del mismo nombre de la tabla ***EquipmentTypes***. Es parte de la llave principal.
  - ***getset_id:*** asociada a la columna del mismo nombre de la tabla ***GetSet***. Es parte de la llave principal.

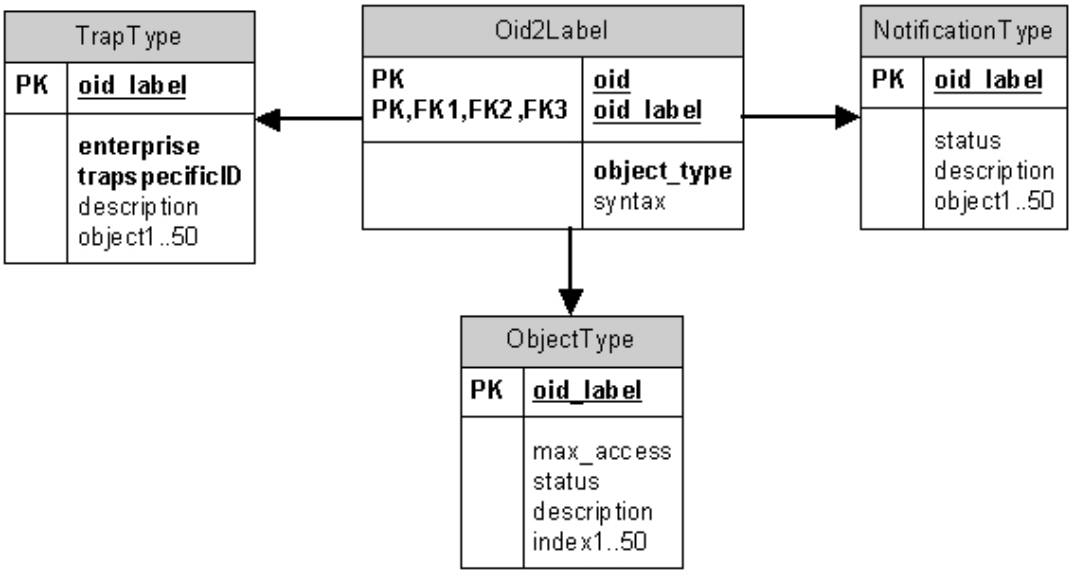


Figura 5-15 Tablas TrapType, Oid2Label,ObjectType y NotificationType

- ***Oid2Label:*** Es tabla asocia la notación OID con su equivalente en texto. Además muestra el tipo y la sintaxis del objeto definida para MIBs SNMP. Las columnas de esta tabla definen:

  - ***oid:*** identificador de objeto de gestión en formato OID. Es parte de la llave principal.
  - ***oid_label:*** identificador de objeto de gestión en formato texto. Es parte de la llave principal.
  - ***object_type:*** declara el tipo de objeto de gestión (Object-Type, Notification-Type o Trap-Type).
  - ***sintax:*** define la sintaxis del objeto.

- ***ObjectType:*** Esta tabla describe a la clase de objeto de gestión del tipo OBJECT-TYPE. Las columnas definen:

  - ***oid_label:*** asociada con la columna del mismo nombre de la tabla ***Oid2Label***. Es la llave principal de la Tabla.
  - ***max-access:*** corresponde al valor MAX-ACCESS (SNMPv2) y al valor ACCESS (SNMPv1) definidos como el modo de acceso al objeto (solo lectura, lectura escritura).
  - ***status:*** corresponde al valor de STATUS. Indica el estado de situación del objeto en la MIB (obligatorio, opcional).
  - ***description:*** corresponde al valor DESCRIPTION. Descripción del objeto.
  - ***index1..50:*** en el caso de ser una tabla, indica los 50 posibles índices de la misma.

- ***NotificationType:*** Esta tabla representa al tipo de objeto NOTIFICATION-TYPE definido para SNMPv2 como objeto de reporte. Las columnas definen:

- ***oid_label:*** asociada con la columna del mismo nombre de la tabla ***Oid2Label***. Es la llave principal de la Tabla.
- ***status:*** corresponde al valor de STATUS. Indica el estado de situación del objeto en la MIB (obligatorio, opcional).
- ***description:*** corresponde al valor DESCRIPTION. Descripción del objeto.
- ***object1..50:*** el objeto de reporte puede contener hasta 50 objetos reportados.

- ***TrapType*** Esta tabla representa al tipo de objeto TRAP-TYPE definido para SNMPv1 como objeto de reporte. Las columnas definen:

  - ***oid_label:*** asociada con la columna del mismo nombre de la tabla ***Oid2Label***. Es la llave principal de la Tabla.
  - ***enterprise:*** corresponde al identificador (enterprise number) del trap.
  - ***description:*** corresponde al valor DESCRIPTION. Descripción del objeto.
  - ***object1..50:*** el objeto de reporte puede contener hasta 50 objetos reportados.

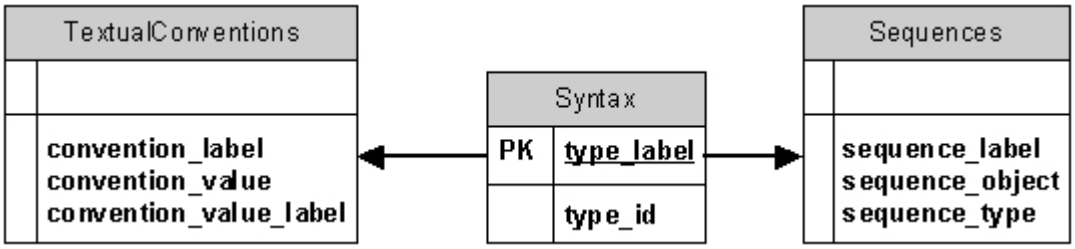


Figura 5-16 Tablas TextualConventions, Syntax y Sequences

- ***Syntax:*** Esta tabla contiene la referencia para los tipos de SEQUENCE y TEXTUAL-CONVENTION de la MIB. Las columnas describen:

  - ***type_label:*** nombre de la secuencia o convención textual. Es la llave primaria de la tabla.
  - ***type_id:*** identifica entre SEQUENCE (2) y TEXTUAL-CONVENTION (1).

- ***TextualConventions:*** Esta tabla contiene los objetos definidos como TEXTUAL-CONVENTION. Las columnas de la tabla representan:

  - ***convention_label:*** nombre del objeto de este tipo.
  - ***convention_value:*** valor entero de la convención.
  - ***convention_value_label:*** valor textual de la convención.

- ***Sequences:*** Esta tabla contiene los objetos definidos como SEQUENCES. Las columnas de la tabla representan:

  - ***sequence _label:*** nombre del objeto de este tipo.

- ***sequence _object:*** nombre del objeto perteneciente a la secuencia.
- ***sequence _type:*** tipo del objeto perteneciente a la secuencia.

5.2.1.7.2 ABM

Centraliza las funciones de Alta, Baja y Modificación de la Base de Datos ***WBNMSDB*** del sistema ***WBNMS***. Además, especifica los distintos modos de consulta que son requeridos por el resto del sistema en su interrelación con la Base de Datos.

5.2.1.8 Módulos del subsistema de Administración

Este subsistema actúa a pedido del cliente, a través de la Interfaz, realizando funciones administrativas que incluyen la gestión de mapas, de entidades-equipos, de tipos de entidades-equipos, de usuarios de alerta, de usuarios del sistema, de perfiles de acción, de MIB's, de reportes, de alertas y del estado del sistema. Por lo que casi todos los módulos funcionales anteriormente descriptos, también cumplen funciones administrativas, como se recalca en la Figura 5.6 que grafica la relación entre los distintos módulos funcionales y los subsistemas de ***WBNMS***. Además, este subsistema de Administración posee dos módulos funcionales que actúan exclusivamente bajo su ámbito, ***Auth*** y ***Status***.

5.2.1.8.1 Auth

Encargado de crear, borrar y modificar usuarios del sistema, de acuerdo con los tipos de privilegios descriptos en el modelo de datos (Operador/administrador). La autentificación de usuarios será delegada al servidor Web, por lo que el módulo ***Auth*** tendrá que ser compatible con la filosofía de control de acceso por él utilizada.

5.2.1.8.2 Status

Es el encargado de consultar el estado del sistema y de administrar los ***archivos logs***[7] del sistema.

5.2.1.8.3 Alarmsurveillance,Drawer,GetSet,keepalive,ActionModules y MIBCompiler

Como ya se mencionó, los módulos antes descriptos, también realizan las tareas de administración necesarias para el funcionamiento del módulo respectivo. En general, son tareas de la configuración del sistema de gestión ***WBNMS***.

[7] Archivos que contienen la información de operación de un computador.

5.3 Diseño de Plataforma

Se establece a continuación los aspectos referidos a la plataforma del sistema que no han sido definidos aún. Para ello se ha dividido el problema en dos partes, agrupando a las plataformas de software por un lado y al hardware por otro.

5.3.1 Software

La elección de la plataforma de software para la aplicación ***WBNMS*** se se refiere a la elección de cuatro ítems fundamentales: Sistema Operativo, Servidor Web, Base de Datos y lenguaje de implementación.

5.3.1.1 Sistema Operativo

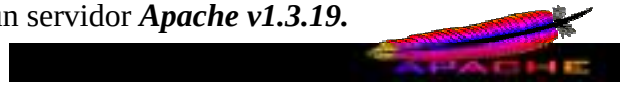
La elección del Sistema Operativo es una de las principales consideraciones a realizar. En este caso se opta por una distribución del SO Linux. Linux es un SO desarrollado bajo GNU, General Public License, y su código fuente es gratuito y puede usarse para una gran variedad de propósitos que incluyen networking, desarrollo de software o su utilización como plataforma de usuario final. Por esta versatilidad se lo considera una alternativa excelente y de bajo costo respecto de otros sistemas operativos más caros. Debido a su funcionalidad y disponibilidad su uso se ha vuelto popular en todo el mundo e inclusive muchos programadores de soft han tomado su código fuente para adaptarlo a sus propias necesidades. Cada vez son más los proyectos relacionados con portar el SO a diversas configuraciones de hardware y diferentes propósitos.

La distribución elegida para este proyecto en particular es ***Mandrake 8.0***. Mandrake Linux fue creado en 1998 con el propósito de hacer que Linux fuese más sencillos de utilizar para cualquier usuario. Hasta ese momento Linux se presentaba como un SO poderoso y estable pero que demandaba un gran conocimiento técnico para su uso y se basaba en la filosofía de línea de comando. Mandrake tuvo su oportunidad de integrar los mejores entornos gráficos de escritorio y contribuir con sus propias utilidades de configuración gráfica. El desarrollo y distribución de Mandrake se encuentra bajo la licencia GPL (General Public License) que provee a cualquiera el derecho de copiar, distribuir, examinar, modificar y mejorar el sistema siempre y cuando los resultados sean retornados a la comunidad. Es decir que una computadora corriendo este SO puede ser tan sencilla de usar como aquella que corra Windows o Mac OS gracias a su interfaz gráfica pero con todo poder de un sistema Unix en su Core. El Mandrake Linux PowerPack, por su parte contiene más de aplicaciones de alta calidad que incluyen un Office Suite completo, más soporte de instalación a un costo de aproximadamente 75 Euros ($69 US) o gratis según sea el caso y soporte virtualmente infinito en la Internet para manos expertas, mientras que el SO equivalente Microsoft Windows + MS Office cuesta aproximadamente 750 Euros ($685 US) sin soporte técnico.
Debido a la fortaleza de Linux, un sistema típico Mandrake Linux puede correr por meses sin degradar su performance. Por otro lado, la versión i586 está optimizada

para la arquitectura Pentium de procesadores y compatibles, para proveer la mejor performance sobre este hardware.

5.3.1.2 Servidor Web

El servidor Web utilizado debe ser compatible con el sistema operativo Linux. Además debe soportar las aplicaciones CGI y manejar la autentificación de Usuarios. El servidor Web por excelencia en esta plataforma es el ***Apache***. Este servidor ha sido el más popular en la Internet desde abril de 1996. El Netcraft Web Server Survey de Octubre de 2003 encontró que más del 64% de los sitios web en Internet están usando Apache. Claro está, es ***Open Source***. Cumple perfectamente los requisitos planteados ya que es un servidor poderoso y flexible que se ajusta a HTTP/1.1 e implementa los últimos protocolos. Es altamente configurable y extensible con módulos de terceras partes que pueden usar el módulo API de Apache. Puede correr sobre Windows NT/9x, Netware 5.x, OS/2, y la mayoría de las versiones de Unix. Permite el seteo de páginas protegidas por passwords y gran cantidad de usuarios por página sin que se degrade la performance del servidor. Se pueden mencionar entre sus caracterísicas más sobresalientes: respuestas adaptables a problemas y errores mediante seteo de archivos o scripts CGI, directivas de índice de directorios, reescritura de URL y aliasing, negociación de contenidos, virtual hosting y registros en formatos configurables o enviables a pipes en el caso de Linux, permitiendo su procesamiento en tiempo real.
En este caso se utilizará un servidor ***Apache v1.3.19.***



5.3.1.3 Servidor de Base de Datos

Se utilizara un servidor de Base de Datos SQL ***MySQL,*** ya que es la más popular, rápida y confiable en el entorno ***Open Source.*** Mysql es un servidor de Base de datos relaciónales. Las bases de datos relacionales permiten guardar datos en tablas separadas agregando velocidad y flexibilidad al proceso de consulta y actualización de los datos. Es posible realizar consultas combinando los datos de distintas tablas. Mysql utiliza el lenguaje SQL (Structured Query Language) que es el lenguaje comúnmente utilizado para acceder a las base de datos. MySQL es un software Open Source, siendo posible gracias a ello su uso y modificación, bajo las reglas de GPL (GNU General Public License). Su arquitectura lo hace extremadamente rápido y fácil de adaptar a diversas necesidades. También se trata de un soft compacto, estable y fácil de desarrollar. La separación entre el núcleo del servidor y la máquina de almacenamiento hace posible que corra con control de transacción estricto o con acceso a disco ultra rápido sin transacción, lo que sea más apropiado a la situación. Específicamente, para ***WBNMS*** se utilizara un servidor de Bases de Datos ***MySQL v3.23.52.***

5.3.1.4 Lenguaje de Implementación

El lenguaje en el que se desarrollará la aplicación ***WBNMS*** deberá ser compatible con toda la plataforma software (OS, DB, WEBServer). En este contexto, las dos principales opciones para el lenguaje de programación son el ***Perl***, el ***PHP*** y adecuando la plataforma, también ***Java*** podría servir. Las tres opciones planteadas, son lenguajes independientes de la plataforma que poseen la capacidad de programación orientada a objetos (u de objetos) y proveen una adecuada integración con el entorno Web. Java será descartado por los mayores requisitos de Hardware para su implementación, tanto en la fase de desarrollo como en la fase operativa. El PHP (Personal Home Page Tool) permite el diseño del entorno gráfico web por separado de la lógica de scrips siendo esta su principal ventaja frente al Perl , que en cambio, genera el entorno gráfico desde los scrips por lo que su diseño es mas complicado. De las dos opciones se eligió el ***Perl*** (ver Anexo – 9.3 Perl.) ya que es de distribución estándar con los sistemas Linux y posee mayor fortaleza en el desarrollo de aplicaciones orientadas a servidores mas que a al desarrollo de la interfaz, caso del PHP, aunque cumple perfectamente con los requisitos de la misma.

Se utilizará ***Perl v5.6.0***.

5.3.2 Hardware

La plataforma de Hardware será simplemente aquella que soporte la aplicación en su conjunto. Además deberá poseer una interfaz de Red. Se utilizará una plataforma:

- ***Servidor:*** *Intel Pentium 200 MHz*
- ***Memoria RAM:*** *64 MB RAM*
- ***Disco Duro:*** *8 GB*
- ***Placa de Red****: FastEthernet 10/100 Mbps – 10BASE-T_100BASE-TX*

5.4 Diseño de la Interfaz

Se describe en esta etapa el proceso de diseño de la interfaz de usuario. Para ello se divide este diseño en dos fases separadas, según sea el punto de ingreso al sistema, vía Interfaz Web o vía Interfaz Local (consola).

5.4.1 Interfaz Web.

La interfaz Web se encuentra configurada de manera de responder como un sitio común de Internet. Como muestra la siguiente figura el Usuario, luego de validarse en el sistema (como Administrador u Operador), accede a una página principal

WBNMS, que es la puerta de acceso a la zona de Gestión (Operador y Administrador) y la zona de Administración (solo Administrador).

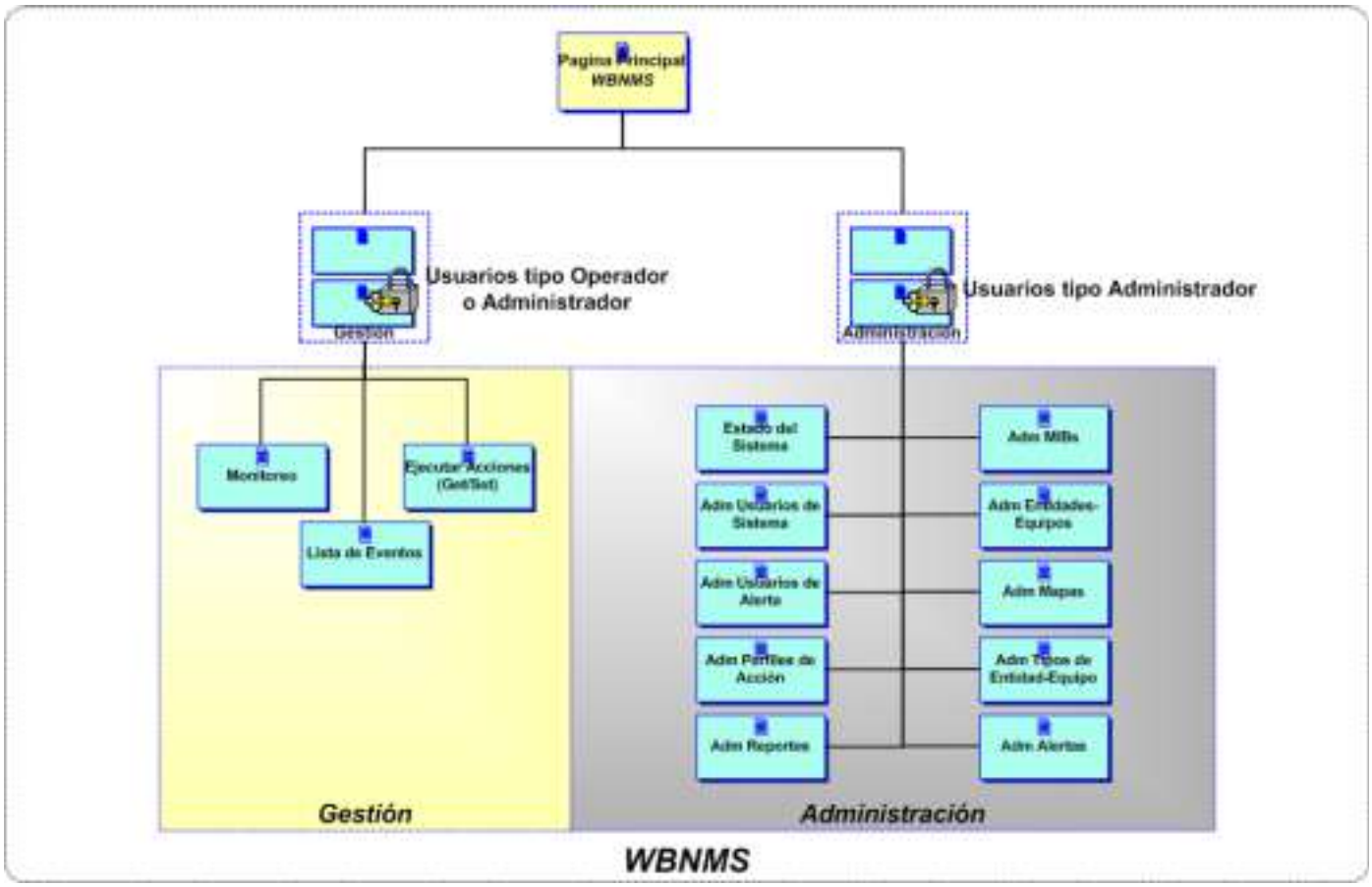


Figura 5-17 Esquema de Pantallas de la Interfaz Gráfica *WBNMS*

Zona de Gestión: Brinda el acceso a las páginas de Monitoreo, de Ejecución de Acciones de Encuesta y Modificación (Get/Set) y de Lista de Eventos Recibidos. Se describe estas páginas y sus componentes:

- **Monitoreo:**
  - Selección de Mapas.
  - Navegación de Mapas por jerarquía.
  - Consulta de Reportes por Entidad-Equipo componente de Mapa.
  - Ejecución de Acciones Get/Set sobre Entidad-Equipo componente de Mapa.
  - Estado general del sistema.
  - Monitor de Reportes Recibidos.

- **Ejecutar Acciones:**
  - Ejecución de Acciones Get y Set por Entidad.

- **Lista de Eventos:**
  - Filtrar Eventos.
  - Mostrar Eventos-Reportes-Alertas.

- Borrar Eventos.
- Reconocer (ACK) Eventos-Reportes.

Zona de Administración: Brinda el acceso a las páginas administrativas de Estado del Sistema, Usuarios del Sistema, Usuarios de Alerta, Perfiles de Acción, Reportes, MIBs, Entidades-Equipos, Mapas, Tipos de Entidad-Equipo, Alertas. Se describe estas páginas y sus componentes:

- **Estado del Sistema:**
  - Consulta del estado del sistema y sus componentes.
  - Visualización y eliminación de los Logs del sistema (de cada uno de los componente del mismo).

- **Usuarios del Sistema (WBNMS):**
  - Creación de Usuarios del Sistema. Selección de tipo de Usuario y Password.
  - Eliminación de Usuarios del Sistema.

- **Usuarios de Alertas:**
  - Listado de Todos los Usuarios de Alertas Configurados.
  - Creación y Modificación de Usuarios destinatarios de Alertas. Selección de nombre, rango, número de teléfono móvil, IP y comunidad.
  - Eliminación de Usuarios destinatarios de Alertas.

- **Perfiles de Acción:**
  - Listado de todos los perfiles definidos.
  - Creación de Perfiles de Acción. Selección de nombre y tipo (Solo Lectura, Lectura/Escritura) del Perfil.
  - Eliminación de Perfiles de acción.

- **Reportes:**
  - Listado de Reportes definidos por Tipo, Entidad-Equipo, o general.
  - Creación, Modificación y Eliminación de Reportes.
  - Selección de componentes del reporte y configuración de filtrado por variables.

- **Alertas:**
  - Listado de Alertas Configuradas.
  - Creación, Modificación y Eliminación de Asociaciones entre Reportes y usuarios de Alerta (=Alerta).

- **Tipo de Entidad-Equipo:**
  - Listado de los Tipos Configurados.
  - Creación, Modificación y Eliminación de Tipos.
  - Asociación de Perfiles de Acción con Tipos de Entidad-Equipo.

- **Mapas:**
  - Listado de Mapas Configurados.

- Creación, Modificación y Eliminación de Mapas.
- Asociación con Entidades-Equipos y otros Mapas.
- Administración Gráfica de la posición de las Entidades y Mapas dentro del Mapa. Selección de coordenadas e íconos.

- **Entidades-Equipos:**
  - Listado de Entidades-Equipos Configurados.
  - Creación, Modificación y Eliminación de Entidades-Equipos.

- **MIBs:**
  - Listado de MIBs Compiladas.
  - Listado de MIBs disponibles en el Sistemas a ser compiladas.
  - Incorporar MIBs al Sistema.
  - Borrar MIBs.
  - Navegación de la MIB del Sistema (Todas las MIBs Compiladas).

5.4.2 Interfaz Local

Son Accesibles localmente las funciones de Backup de la Base de Datos (***WBNMSDB***) y de cambio de Estado del Sistema (Start/Stop/Status).

- **Backup:** Dado por la interfaz provista por ***MySQL*** y el comando:

```
mysqldump WBNMSDB
```

- **Estado del Sistema:** Cada uno de los componentes del sistema que requieran una interfaz de activación, recurrirán al siguiente formato:

```
'DaemonWBNMS'  Start/Stop/Status
```

5.5 Conclusión de la etapa de Diseño

El siguiente es un esquema de red que intenta resumir el funcionamiento general del sistema de Gestión de nodo ***WBNMS***.

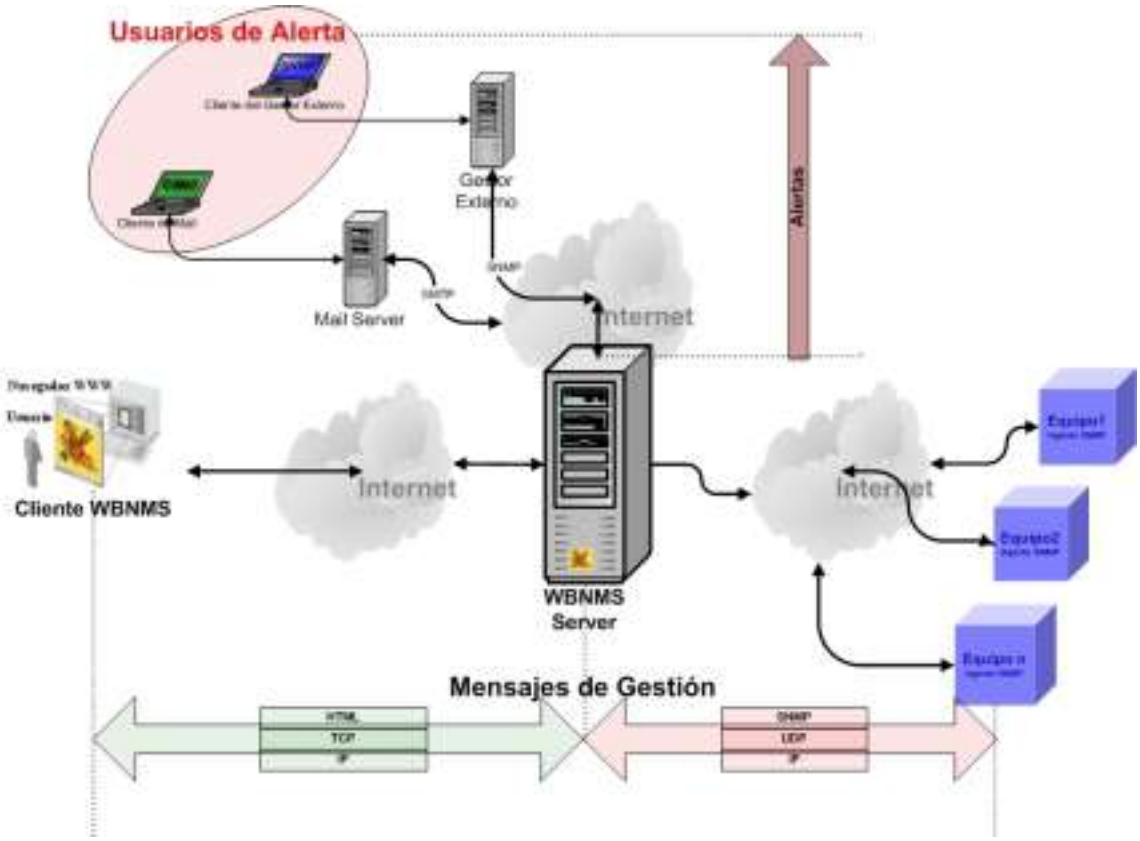


Figura 5-18 Esquema de Funcionamiento del Sistema

El Usuario se conecta usando un Navegador WWW como cliente ***WBNMS***, a través de la Internet o red privada, con el servidor ***WBNMS***. Esta comunicación se realiza vía IP/TCP/HTML.
El flujo de mensajes de Gestión (Traps, Gets y Sets) entre las Entidades-Equipos y el Sistema de Gestión ***WBNMS***, a través de la Internet o red privada, es realizado vía IP/UDP/SNMPv1.
Esta información de Gestión es procesada por el Sistema para su presentación a requerimiento del Usuario del Sistema o para su envío, a través de la Internet o red privada, vía mensajes de alerta a los Usuarios de Alertas. Los modos de Alerta son: SNMP o E-MAIL(SMTP). La comunicación del sistema ***WBNMS*** con los Usuario de Alertas esta manejada por sistemas externos, que actúan de ***Gateways*** entre ellos.
La interacción cliente-servidor se realiza utilizando la interfaz CGI. Es así como el usuario requiere servicios al sistema de Gestión ***WBNMS*** y este responde a ellos.
El siguiente gráfico, esquematiza el flujo de mensajes entre el usuario y la interfaz CGI.

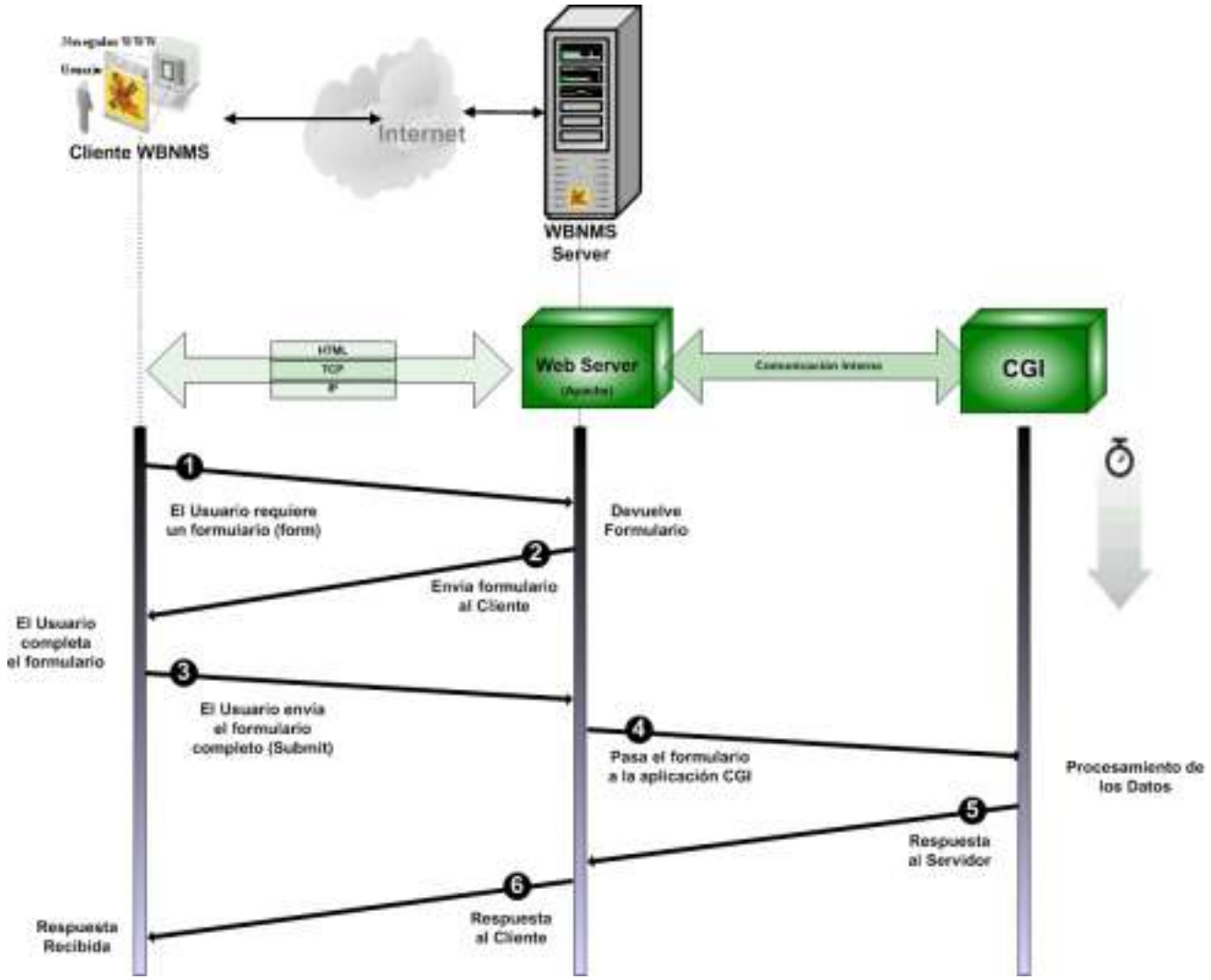


Figura 5-19 Intercambio de Mensajes entre Cliente y Servidor

Toda aplicación cliente-servidor requiere de una solicitud expresa del usuario hacia el servidor como punto de partida. El servidor envía los requisitos en forma de formulario. El Usuario lo completa y lo envía de vuelta al servidor. Este lo pasa a la aplicación CGI que lo procesa y devuelve la respuesta al servidor, que la reenvía al cliente. Donde el Usuario obtiene la respuesta a su requerimiento inicial.

Este proceso se repite en toda interacción entre el cliente y el sistema ***WBNMS***. Por lo que la respuesta del sistema esta sujeta a retardos de procesamiento interno, retardos de intrínsecos de la red y retardos producidos por el handshake de la interfaz CGI.

El procesamiento la información de Gestión y Administración es realizada por los módulos funcionales descriptos en el Diseño del Sistema ***WBNMS***. En el siguiente gráfico se pretende esquematizar el resultado del proceso de diseño y su interacción con el sistema de gestión.

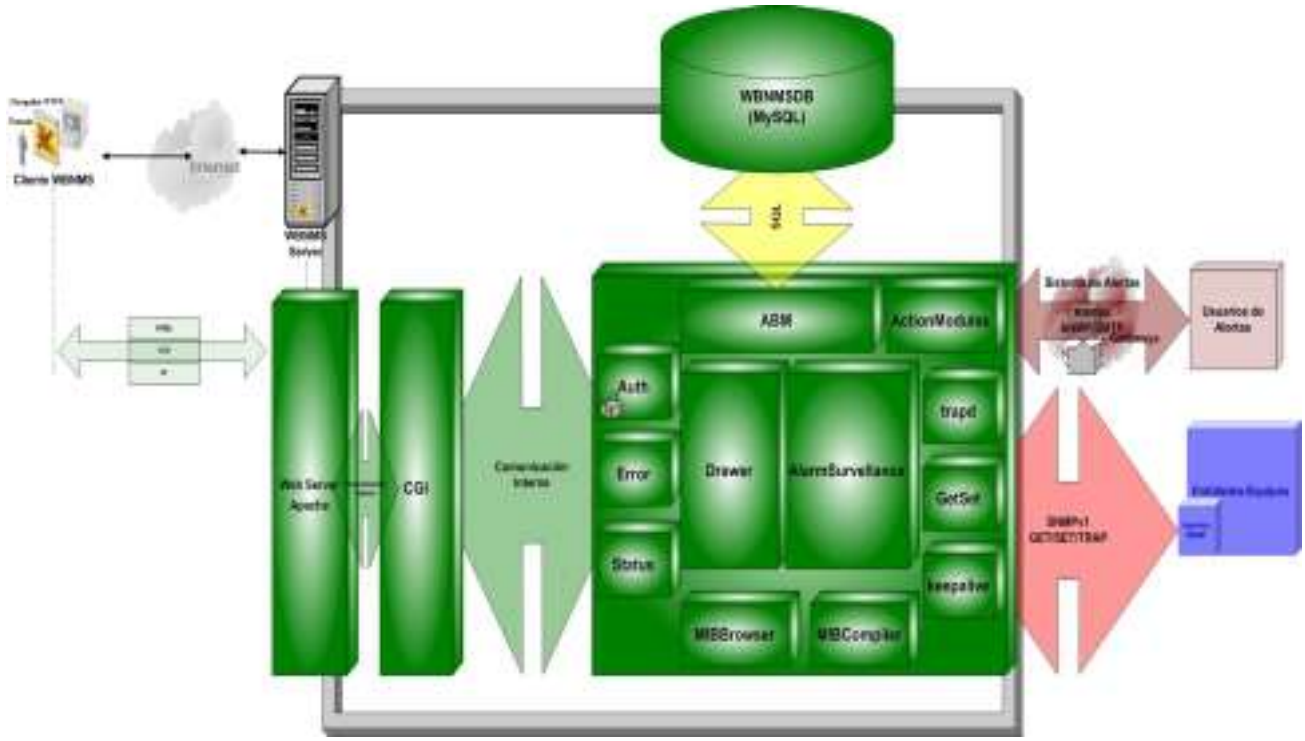


Figura 5-20 Esquema de los componentes del sistema diseñado

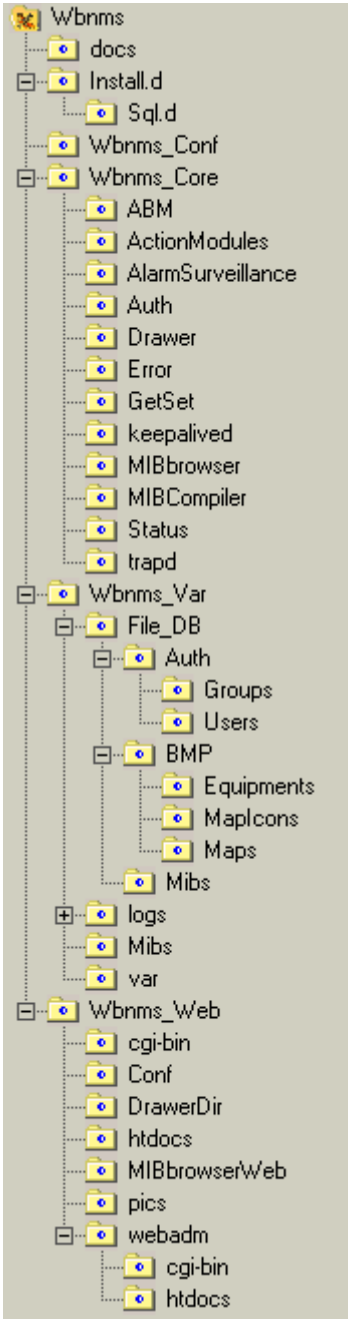
Este gráfico es una extensión del gráfico de la figura 5.16 que detalla los distintos módulos funcionales diseñados en este capítulo y su interrelación con los componentes externos del gestor, como ser el cliente, la entidad gestionada y el usuario de alerta, destacando también la posibilidad de ubicar la base de datos en uno o varios servidores Mysql externos en lugar de uno local. Se especifica además el tipo de mensajes que fluyen entre ellos:

- SNMP, Get/Set/Trap entre módulos trapd, GetSet y keepalive y Agentes SNMP de las entidades gestionadas.
- SNMP traps y SMTP entre los ActionModules y los Usuarios de alerta correspondientes.
- HTML/TCP/IP entre el cliente y el servidor Web Apache.
- Mensajes Internos entre la API CGI del servidor Apache y módulos Perl del sistema.
- SQL entre el módulo ABM y el/los servidor/es Mysql.

6 Implementación

La etapa de implementación tiene por objetivo la transformación de los aspectos sobresalientes y objetivos planteados en las etapas previas de Análisis y Diseño. En el desarrollo de la misma se deben tomar decisiones que condicionarán el funcionamiento pero intentan respetar todas las especulaciones previa. El resultado es un sistema real llamado ***WBNMS v1.0***.

6.1 Estructura de directorios *WBNMS*



A continuación se presenta la estructura de directorios implementada para la aplicación ***WBNMSv1.0***. Existen 6 subdirectorios principales:

- ***Docs:*** Contiene la documentación disponible del sistema ***WBNMS***.
- ***Install.d:*** Directorio que contiene archivos de Instalación y archivos de instalación SQL (Sql.d).
- ***Wbnms_Conf:*** Único repositorio de archivos de Configuración del Sistema
- ***Wbnms_Core:*** Núcleo funcional del sistema ***WBNMS***. Cada uno de sus subdirectorios especifica uno de los 12 Módulo Funcionales.
- ***Wbnms_Var:*** Agrupa Variables del Sistema (var), MIBs disponibles (Mibs), archivos gráficos(FileDB/BMP), de autentificación de usuarios y grupos (FileDB/Auth) y Mibs compiladas (FileDB/Mibs).
- ***Wbnms_Web:*** Directorio raíz de la interfaz Web. Aplicaciones CGI de operador (cgi-bin) y administrador (webadm/cgi-bin), archivos html operador (htdocs) y administrador (webadm/htdocs), archivos de configuración Web(Conf), archivos gráficos generados por del módulo Drawer (DrawerDir), archivos Web del navegador de MIBS (MIBbrowserWeb) y archivos gráficos (pics).

6.2 Archivos de Configuración *WBNMS*

La configuración del sistema se define en un único archivo, '***WbnmsConf.pm***'. En el mismo se encuentran especificados los paths[8] y variables de configuración de todos los módulos del sistema. Se ha escrito de acuerdo al formato de Perl relativo a módulos, para poder ser cargado con el comando *'**use WbnmsConf** '*, lo que simplifica dicha operación. En el desarrollo de cada módulo funcional se detallan las entradas a este archivo de configuración.

6.3 Módulos Funcionales *WBNMS*

Se recorren a continuación cada uno de los módulos funcionales, especificando el conjunto de archivos que les dan forma. Se utiliza con el mayor grado conveniente, la orientación a objetos en el estilo de programación.

6.3.1 Módulo ABM

Los archivos que conforman este Módulo se encuentran dentro del subdirectorio ***Wbnms_Core/ABM***. Esta conformado por tres archivos:

- ***Directorio Wbnms_Core/ABM:***
  - ***ABMManager.pm :*** Clase principal para el manejo de datos. (ver Anexo Código - ABMManager.pm)
  - ***ABMFile.pm :*** Clase de acceso a Archivos de texto. (ver Anexo - Código ABMFile.pm)
  - ***ABMDB.pm :*** Clase de acceso a la Base de Datos MySql.(ver Anexo - Código ABMDB.pm).

En la siguiente figura se detallan en un gráfico UML estas tres clases y su interrelación, destacando sus atributos y las funciones que implementan. A continuación se describe cada una de ellas:

[8] Mas allá de la traducción tradicional "camino", indica en informática al trayecto de búsqueda de una aplicación determinada dentro del sistema. En Linux/UNIX desde root **/.**

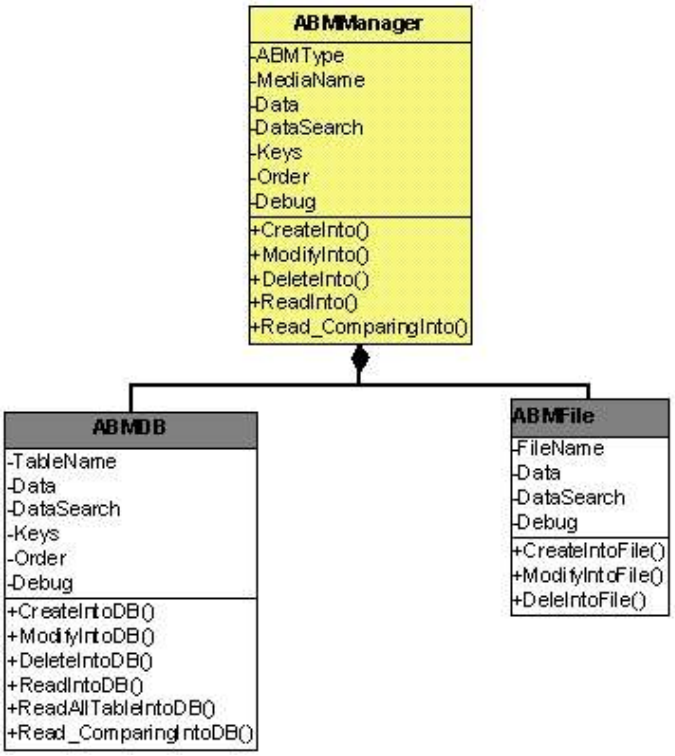


Figura 6-1 Clases del Módulo ABM

6.3.1.1 ABMManager.pm

Esta clase es la encargada de manejar el flujo de Datos en los procesos de Alta, Baja, Modificación y Lectura para el sistema ***WBNMS*** . Tiene como agregadas las clases ABMFile y ABMDB (relación de rombo en el; gráfico UML).

Los atributos de esta clase son :

- ***ABMType:*** Define el tipo de acceso, a DB (***ABMType=1***) o a File (***ABMType=2***).
- ***MediaName:*** Define el nombre del medio de almacenamiento (DB: nombre de la tabla, File: nombre del archivo).
- ***Data:*** Datos a almacenar o Modificar.
- ***DataSearch:*** Datos como concepto de búsqueda para Modificación o Eliminación.
- ***Keys:*** Definición de variables a Mostrar.
- ***Order:*** Criterio de orden de los datos a presentar.
- ***Debug:*** Modo de debugging.

El formato de los atributos con que se crea el objeto ABMManager dependerá del tipo de acceso que se requiera. Ya que esto define la clase a la que se llama para llevar a cabo la acción (ABMDB o ABMFile).

Las funciones que implementa son:

- ***CreateInto***: Crea una nueva entrada en medio definido por MediaName. Son requeridos los atributos MediaName, ABMType y Data.
- ***ModifyInto:*** Modifica un dato en MediaName según búsqueda DataSearch. Son requeridos los atributos MediaName, ABMType, Data y DataSearch.
- ***DeleteInto:*** Borra un dato en MediaName según la búsqueda DataSearch. Son requeridos solo estos dos atributos.
- ***ReadInto***: Lee datos desde MediaName. Son opcionales la búsqueda según DataSearch , Order y la presentación según Keys.
- ***Read_ComparingInto:*** Lee datos de la combinación de varios Medios. Exclusiva para Base de Datos.

6.3.1.2 ABMFile.pm

Esta clase actúa por pedido de la clase ABMManager sobre medios del tipo archivo (***ABMType=2***). Es requerido el archivo de configuración ***WbnmsConf*** para obtener el Path hacia el directorio destino de los medios de este tipo en ***WBNMS*** (File_DB).Por lo tanto ***WbnmsConf*** deberá registrar un elemento de configuración del tipo:

```
$WbnmsConf::CONFPATH{'File_DB'}
```

Los atributos de esta clase son :

- ***FileName:*** Define el nombre del archivo sobre el que se actúa.
- ***Data:*** Datos a almacenar o Modificar.
- ***DataSearch:*** Datos como concepto de búsqueda para Modificación o Eliminación.
- ***Debug:*** Modo de debugging.

Las funciones que implementa son:

- ***CreateIntoFile***: Crea una nueva entrada en el archivo definido por FileName. Son requeridos los atributos FileName y Data.
- ***ModifyIntoFile:*** Modifica un dato en el archivo FileName según búsqueda DataSearch. Son requeridos los atributos FileName, Data y DataSearch.
- ***DeleteIntoFile:*** Borra un dato en el archivo FileName según la búsqueda DataSearch. Son requeridos solo estos dos atributos.

6.3.1.3 ABMDB.pm

Esta clase actúa por pedido de la clase ABMManager sobre los medios del tipo DB (***ABMType=1***).
Es requerido el archivo de configuración ***WbnmsConf***, de donde se obtienen los siguientes datos:

```
$WbnmsConf::CONFDB{'DataBaseUserName'} = Nombre del Usuario de la DB
$WbnmsConf::CONFDB{'DataBasePassword'}= Password de Usuario de la DB
$WbnmsConf::CONFDB{'DataBaseName'}= Nombre de la Base de Datos
$WbnmsConf::CONFDB{'DataBaseHost'}= Host de la Db
```

Clase Utilizada:

- DBI para la interfaz a la Base de Datos Mysql.

Los atributos de esta clase son :

- ***TableName:*** Define el nombre de la Tabla de la Base de Datos.
- ***Data:*** Datos a almacenar o Modificar.
- ***DataSearch:*** Datos como concepto de búsqueda para Modificación o Eliminación.
- ***Keys:*** Definición de variables(columnas de la tabla) a Mostrar.
- ***Order:*** Criterio de orden de los datos a presentar.
- ***Debug:*** Modo de debugging.

Las funciones que implementa son:

- ***CreateIntoDB***: Crea una nueva entrada en la Tabla definida por TableName. Son requeridos los atributos TableName y Data.
- ***ModifyIntoDB:*** Modifica un dato en la tabla definida por TableName según búsqueda DataSearch. Son requeridos los atributos TableName, Data y DataSearch.
- ***DeleteIntoDB:*** Borra un dato en la tabla definida por TableName según la búsqueda DataSearch. Son requeridos solo estos dos atributos.
- ***ReadIntoDB***: Lee datos desde la tabla TableName. Son opcionales la búsqueda según DataSearch , Order y la presentación según Keys.
- ***ReadAllTableIntoDB:*** Lee los datos de toda la tabla definida por TableName. Son opcionales los modificadores Keys y Order para la presentación.
- ***Read_ComparingInto:*** Lee datos de la combinación de varias Tablas. Son opcionales los modificadores Keys y Order para la presentación. TableName y DataSearch deberán ser de la siguiente forma:

```
$ABMDBObj->TableName("Tabla1,Tabla2,...Tabla n");
$ ABMDBObj ->DataSearch(formato lógico Sql*);
```

*Ej: "columna1 = columna2 and columna3 != columna2".

6.3.2 Módulo trapd

Los archivos que conforman este Módulo se encuentran dentro del subdirectorio ***Wbnms_Core/trapd***. Esta conformado por un archivo principal y cuatro secundarios:

- ***Directorio Wbnms_Core/trapd:***

  - ***trapd.pl:*** Demonio de Traps. (ver Anexo Código - trapd.pl)
    - ***trapd.rc:*** Shell script de control del demonio de traps. (ver Anexo – Código trapd.rc)

- ***start:*** script de arranque del demonio de traps.(ver Anexo - Código start)
- ***stop:*** script de stop del demonio de traps.(ver Anexo - Código stop)
- ***renametraplog:*** script de renombrado de log.(ver Anexo – Código renametraplog.pl)

6.3.2.1 trapd.pl - Trap Daemon

Se encarga de recibir los eventos en formato de traps SNMPv1, procesarlos y almacenarlos en la base de datos.

Es requerido el archivo de configuración ***WbnmsConf***, del cual se extraen los siguientes datos:

$WbnmsConf::CONFTRAP{'logpath'} = Path del log del trap Daemon
$WbnmsConf::CONFTRAP{'trapfile.log' } = archivo de log
$WbnmsConf::CONFTRAP{'port' } = puerto de traps (162)
$WbnmsConf::CONFTRAP{'pidfile' } = archivo de PID
$WbnmsConf::CONFTRAP{'community' } = comunidad SNMP
$WbnmsConf::CONFTRAP{'Debug' }= 1 Debug ON,0 Debug OFF

Clases y librerías Utilizadas:

- ***ABMManager*** para almacenar los eventos recibidos.
- ***BER*** para decodificar la información contenida en los traps.
- ***SNMP_Session*** y ***SNMP_util*** para manejar los PDU traps (SNMPv1).
- ***POSIX*** para manejar los procesos (PIDs)

El funcionamiento del demonio de traps (trapd.pl) detallado en la Figura 6-2 es el siguiente:

El demonio es arrancado con el levantamiento del sistema operativo a través del script de control, trapd.rc. Luego el demonio se encuentra en escucha en un puerto UDP a la espera de recibir un Trap válido. Cuando es recibido este trap, se crea un nuevo proceso que lo interpreta y almacena el evento en la base de datos. Además ingresa el evento en el trapdaemonlog. Luego, este proceso muere.

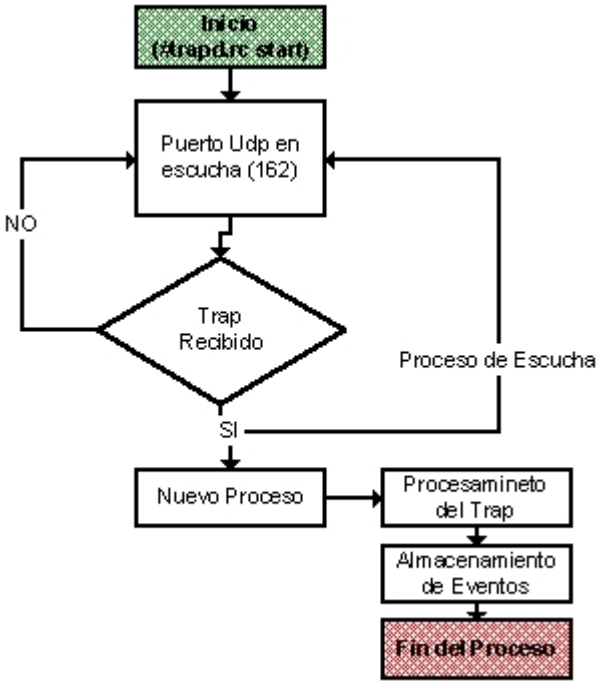


Figura 6-2 Diagrama de flujo del demonio de traps

6.3.2.2 trapd.rc

Este es el script de control del demonio trapd.pl. A través de él, se inicia, se apaga y se consulta el estado del demonio. Para ello se vale de los scripts de Start, Stop y renametraplog.pl.

trapd.rc { start | stop | status | restart }

6.3.3 Módulo keepalive

Los archivos que conforman este Módulo se encuentran dentro del subdirectorio ***Wbnms_Core/keepalive***. Esta conformado por cuatro archivos principales y cuatro secundarios:

- ***Directorio Wbnms_Core/keepalive:***

  - ***keepalive.pl:*** Demonio de keepalive. (ver Anexo Código - keepalive.pl)
  - ***keepalive.rc:*** Shell script de control del demonio keepalive. (ver Anexo – Código keepalive.rc)
  - ***start:*** script de arranque del demonio keepalive.(ver Anexo - Código start)
  - ***stop:*** script de stop del demonio keepalive.(ver Anexo - Código stop)
  - ***renamekeepalivelog:*** script de renombrado de log.(ver Anexo – Código renamekeepalivelog.pl)

- ***PollManager:*** Clase de administración e Interfaz del Módulo(ver Anexo – Código PollManager.pm)
- ***snmppoller:*** Clase de que implementa la encuesta automática (ver Anexo – Código snmppoller.pm)
- ***snmppollerManager:*** Clase de control de la encuesta automática. (ver Anexo – Código snmppollerManager.pm)

6.3.3.1 keepalive.pl

Se encarga de verificar la conexión SNMP entre el gestor y las entidades-equipos, y de detectar nuevas entidades en un rango IP preestablecido. Los resultados de este proceso son almacenados en la base de datos.

Es requerido el archivo ***WbnmsConf***, del cual se extraen los siguientes datos:

```
%WbnmsConf::KEEPALIVE =
  (
'logpath' => "Path del log del demonio",
'keepalivefile.log' => "keepalivefile.log",
'startscan' => 'IP inicial',
'endscan' => 'IP final',
'port'  => 'Puerto SNMP(161)',
'timeout' => 'timeout en la conexión de prueba en segundos',
'retries' => 'número de reintentos',
'pidfile' => "keepalive.pid",
'community' => "comunidad SNMP (public)",
'checkinterval' =>'intervalo de verificación en segundos ( 900)',
'Debug' => debug mode ON (1) u OFF(0),
);
```

Clases y librerías Utilizadas:

- ***snmppollerManager*** para manejar el proceso de encuesta automática.
- ***ABMManager*** para manejar el acceso a la Base de Datos.
- ***SNMP_util*** para manejar los PDU Get.
- ***POSIX*** para manejar los procesos (PID).

El funcionamiento del demonio detallado en la Figura 6-3 es el siguiente:

Se inicia con el sistema y realiza una consulta a la base de datos para reconocer las entidades-equipos gestionadas. Luego crea un proceso al llamar al ***snmppollerManager***, y este se encarga de manejar el proceso de encuesta automática. Repite la misma acción para el rango IP predefinida como de auto-descubrimiento. Luego espera un intervalo hasta volver a repetir la secuencia.

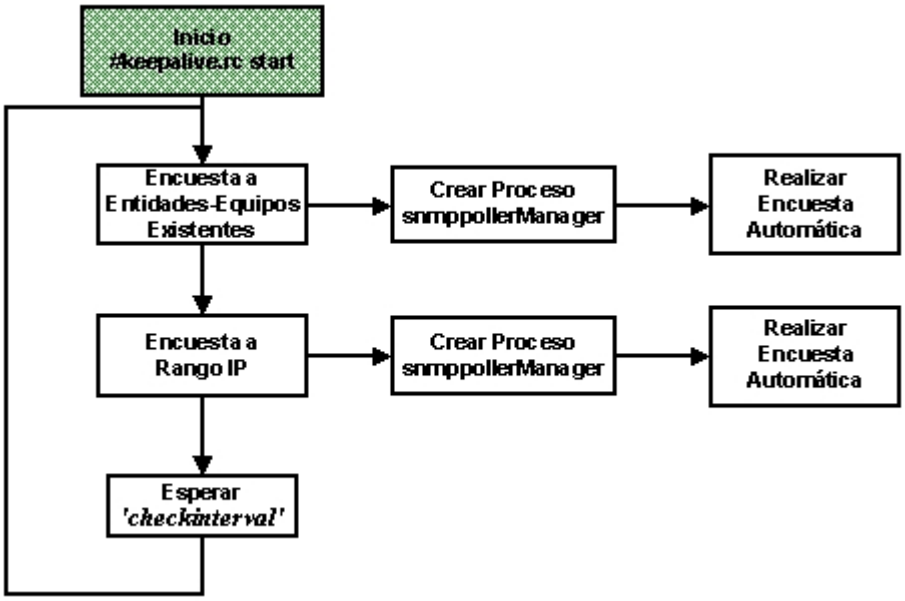


Figura 6-3 Diagrama de Flujo del demonio keepalive

6.3.3.2 snmppollerManager.pm

Es la clase encargada de manejar el proceso de encuesta automática, a pedido del demonio ***keepalive***.

Clase Utilizada:

- ***snmppoller*** para realizar el proceso de encuesta automática.

La relación con la clase anterior esta detallada en el siguiente gráfico UML:

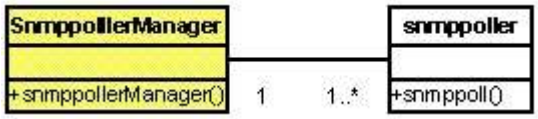


Figura 6-4 Relación de Clases SnmppollerMannager y Snmppoller

A continuación se detalla cada componente:

6.3.3.3 snmppoller.pm

Este Objeto es creado desde snmppollerManager para cada entidad-equipo o IP a encuestar. Implementa la función ***snmpoll*** de encuesta. Según el tipo de encuesta requerido, de verificación de entidades-equipos existentes o de descubrimiento de nuevas entidades-equipos, actúa enviándoles mensajes ***"sysName"*** en Get PDU (pregunta el nombre del sistema). Luego almacena en la base de datos el resultado.

Si es una encuesta de verificación, los resultados negativos son almacenados como eventos ***"linkdown"*** en la tabla ***TrapsCollector***. En cambio se es del otro tipo, los resultados, IP y comunidad descubiertos, son almacenados en la tabla ***NewEquipment***.

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***SNMP_util*** para manejar los mensajes SNMP.
- ***POSIXm*** para el manejo de variables temporales.

6.3.3.4 PollManager.pm

Esta Clase es instanciada desde la Interfaz Web (programas CGI) para realizar tareas de administración del módulo ***keepalive***.

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***ErrorManager*** para manejar la presentación de los mensajes de Error.

Las funciones que implementa son:

- ***ReadEquipments***: Lee de la Base de Datos la tabla ***NewEquipment*** para presentar los datos de las entidades-equipos descubiertos en el proceso de descubrimiento**.** En este proceso también consulta la tabla ***EquipmentTypes*** para presentar los tipos disponibles.
- ***AddEquipment:*** Traslada una entidad-equipo descubierta a la tabla ***Equipment***, creado una nueva entidad de gestión. En este proceso la borra de la tabla ***NewEquipment***.
- ***DelEquipment:*** Borra una entidad-equipo descubierta de la tabla ***NewEquipment.***

6.3.3.5 keepalive.rc

Este es el script de control del demonio keepalive.pl. A través de él, se inicia, se apaga y se consulta el estado del demonio. Para ello se vale de los scripts de Start, Stop y renamekeepalivelog.pl.

keepalive.rc { start | stop | status | restart }

6.3.4 Módulo GetSet

Este Módulo se encuentran conformado por un único archivo, que se encuentra dentro del subdirectorio ***Wbnms_Core/GetSet***.

Directorio Wbnms_Core/ GetSet:

- ***GetSetManager:*** Clase de encuesta, Modificación y administración del módulo.(Ver Anexo – Código GetSetManager.pm)

6.3.4.1 GetSetManager.pm

Esta clase cumple una doble función, por un lado realiza las encuestas y modificaciones, definidas en los perfiles de acción, sobre las entidades-equipos. Y por otro lado también realiza las funciones administrativas que requiere el módulo.

Clases y librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***SNMP_util*** para manejar los PDU Get y Set.
- ***ErrorManager*** para manejar la presentación de los mensajes de Error.

Las funciones que implementa son:

- ***ReadEquipments:*** Lee de la tabla ***Equipment*** para presentar la lista de las entidades-equipos gestionados.
- ***ReadProf:*** Lee los perfiles de acción definidos para cada uno de los tipos de entidad-equipo y los devuelve. En este proceso lee las tablas ***EquipmentTypeGetSet*** y ***GetSet***.
- ***ReadAllProfiles:*** Lee todos los perfiles de acción definidos de la tabla ***GetSet***.
- ***AddComp:*** Agrega un nuevo componente de gestión a (oid_label) a un perfil de acción. Es ingresado en la tabla ***GetSet***.
- ***DelComp:*** Borra un componente de un perfil de acción. De la tabla ***GetSet***.
- ***CreateProf:*** Crea un nuevo perfil de acción el la tabla ***GetSet***.
- ***UnCreateProf:*** elimina un perfil de acción de la tabla ***GetSet***.
- ***AddProf:*** Asocia un perfil de acción con un tipo de entidad-equipo, en la tabla ***EquipmentTypeGetSet***.
- ***DelProf:*** Borra una asociación de la tabla ***EquipmentTypeGetSet.***
- ***ReadEquipmentData:*** Lee los datos de una entidad-equipo en particular de la tabla ***Equipment*** según el *equipment_id*.
- ***Get:*** Realiza la encuesta según cada uno de los componentes definidos en el perfil de acción y presenta el resultado de la misma. Para ello utiliza la librería SNMP_Util con la que construye el mensaje PDU Get y una función interna para formatear los datos (_Prepare()).
- ***Set:*** Realiza la modificación según cada uno de los componentes definidos en el perfil de acción y presenta el resultado de la misma. Para ello utiliza la librería SNMP_Util con la que construye el mensaje PDU Set y una función interna para formatear los datos (_Prepare()).

6.3.5 Módulo Status

Este Módulo se encuentran conformado por dos archivos, que se encuentran dentro del subdirectorio ***Wbnms_Core/Status***.

Directorio Wbnms_Core/ Status:

- ***StatusChecker:*** Clase de verificación de estado del sistema ***WBNMS***.(Ver Anexo – Código StatusChecker.pm)
- ***DaemonsLogViewer:*** Clase de administración de logs del sistema ***WBNMS*** (Ver Anexo – Código DaemonsLogViewer.pm)

6.3.5.1 StatusChecker.pm

Esta clase es la encargada de verificar el estado del sistema ***WBNMS***. Esto es el estado operativo de todos los demonios que lo componen y de la DB.

Es requerido el archivo de configuración ***WbnmsConf***, de donde se obtienen los siguientes datos:

$WbnmsConf::CONFPATH{'AlarmSurveillance'}=Path del demonio AlarmSurveillance
$WbnmsConf::CONFALARMSUR{'pidfile'}=PID del demonio AlarmSurveillance
$WbnmsConf::CONFPATH{'trapd'}= Path del demonio trapd
$WbnmsConf::CONFTRAP{'pidfile'}=PID del demonio trapd
$WbnmsConf::CONFDRAWER{'DrawerCorepath'}= Path del demonio Drawer
$WbnmsConf::CONFDRAWER{'pidfile'}=PID del demonio Drawer
$WbnmsConf::CONFPATH{'keepalive'}=Path del demonio keepalive
$WbnmsConf::KEEPALIVE{'pidfile'}=PID del demonio keepalive
$WbnmsConf::CONFDB{'DataBaseUserName'} = Nombre del Usuario de la DB
$WbnmsConf::CONFDB{'DataBasePassword'}= Password de Usuario de la DB
$WbnmsConf::CONFDB{'DataBaseName'}= Nombre de la Base de Datos
$WbnmsConf::CONFDB{'DataBaseHost'}= Host de la Db

Clase Utilizada:

- DBI para la interfaz a la Base de Datos Mysql.

Las funciones que implementa son:

- ***CheckStatus:*** Verifica el estado de cada demonio al verificar si existe su PID activo. Para ello se recurre a un truco: Los procesos activos generan un directorio de nombre igual al PID dentro del directorio */proc*. Este directorio es creado, residente en memoria, por el sistema Unix al ser activado. Es así, que esta función lee los nombres de los directorios existentes en /proc y los comparan con los PID asignados a los demonios ***WBNMS***, para conocer su estado. El estado de la Base de Datos es

verificado mediante la función interna ***_connectDB***. Esta prueba una conexión con la DB y devuelve así el estado de la misma.

- ***Start:*** activa un demonio determinado. Esta función no esta implementada en esta versión de ***WBNMS*** v1.0.
- ***Stop:*** detiene un demonio determinado. Esta función no esta implementada en esta versión de ***WBNMS*** v1.0.

6.3.5.2 DaemonsLogViewer.pm

Esta clase implementa funciones de Administración de logs del sistema ***WBNMS***. Además de funciones de Interfaz, formateando datos para la presentación.

Es requerido el archivo de configuración ***WbnmsConf***, de donde se obtienen los siguientes datos:

$WbnmsConf::CONFALARMSUR{'logpath'}=Path al log de AlarmSurveillance
$WbnmsConf::CONFMAILER{'logpath'}=Path al log del Módulo Mailer
$WbnmsConf::CONFSMS{'logpath'}= Path al log del Módulo SMS
$WbnmsConf::CONFTRAP{'logpath'}= Path al log de trapd

Las funciones que implementa son:

- ***IdentifyComponents:*** Relaciona los distintos Módulos o demonios con los respectivos directorios de logs.
- ***ReadLogs:*** Lee del directorio de logs de un demonio o módulo, devolviendo todos los archivos de logs existente.
- ***OpenLog:*** Lee un archivo de log específico.
- ***DelLog:*** Borra un archivo de log determinado

6.3.6 Módulo Auth

Este Módulo se encuentran conformado por un archivo, que se encuentran dentro del subdirectorio ***Wbnms_Core/Auth***.

Directorio Wbnms_Core/ Auth:

- ***Auth:*** Clase de administración de Usuarios del sistema ***WBNMS***. (Ver Anexo – Código Auth.pm)

6.3.6.1 Auth.pm

Esta clase implementa funciones administrativas de usuarios y grupos del sistema ***WBNMS***, cumpliendo con el formato definido por el servidor Web Apache. Este último es el encargado de controlar el acceso al sistema vía la Interfaz Web.

El servidor Apache espera encontrar (si esta configurado para ello) un archivo en cada directorio que especifique el modo de acceso al mismo y los permisos requeridos para ello (de grupo y usuario). Por defecto, el nombre de este archivo es ***.htaccess***. En el caso del sistema ***WBNMS*** existen dos archivos ***.htaccess*** definidos. Uno para nivel de acceso de Operador (incluye Administrador) en el directorio ***Wbnms_Web.***

```
AuthType Basic
AuthName "WBNMS"
AuthUserFile <raiz de Wbnms>/Wbnms_Var/File_DB/Auth/Users/password
AuthGroupFile <raiz de Wbnms>/Wbnms_Var/File_DB/Auth/Groups/groups
Require group Administrador Operador
```

Y otro para el nivel de acceso de Administrador en el directorio ***Wbnms_Web/webadm.***

```
AuthType Basic
AuthName "Se Requiere PassWord de Administrador-WBNMS"
AuthUserFile <raiz de Wbnms>/Wbnms_Var/File_DB/Auth/Users/password
AuthGroupFile <raiz de Wbnms>/Wbnms_Var/File_DB/Auth/Groups/groups
Require group Administrador
```

Al encontrar estos archivos en un directorio, el servidor Web busca los archivos de usuarios (***AuthUserFile***) y grupos (***AuthGroupFile***) para requerirle al Usuario del sistema ***WBNMS*** su autentificación.

El formato del archivo ***password*** es el siguiente:

```
<Nombre del usuario>:<pasword encriptado>
```

El formato del archivo ***groups*** es el siguiente:

```
<Nombre del grupo>:<Nombre del usuario 1> <Nombre del usuario n>
```

Volviendo a la descripción de ***Auth.pm***. Las funciones implementadas cumplen con estos formatos requeridos por el servidor Apache.

Es requerido el archivo de configuración ***WbnmsConf***, de donde se obtienen los siguientes datos:

```
$WbnmsConf::CONFAUTH{'UsersFile'}=Path al archivo de usuario( password)
$WbnmsConf::CONFAUTH{'GroupsFile'}= Path al archivo de grupo( groups)
```

Los atributos de esta clase son :

- ***Name:*** Nombre del usuario.
- ***Password:*** Password del usuario.
- ***CheckPass:*** Password del usuario reingresado para verificación de consistencia.
- ***Group:*** Grupo del Usuario.

- ***Debug:*** Modo de Debugging, 0 OFF y 1 ON.

Las funciones que implementa son:

- ***readGroupsandUsers:*** Lee del archivo de usuario (password) y del archivo de grupo (groups) los usuarios y grupos respectivamente. Devuelve los todos los usuarios configurados especificando a que grupo pertenecen (operador o administrador).
- ***AddUsers:*** Agrega un nuevo usuario al archivo password. Esta función llama internamente a la función _crypt que realiza el encriptado del password (***digest authentication, MD5 – RFC2617***). Se requiere que estén definidos los atributos de ***Name, Password*** y ***CheckPass***.
- ***AddGroupUsers:*** Agrega un nuevo usuario a un grupo determinado en el archivo de ***groups***.
- ***DelUser:*** Borra un usuario del archivo ***password***.
- ***DelGroupUsers:*** Borra un usuario de un grupo determinado en el archivo ***groups***.

6.3.7 Módulo Error

Este Módulo se encuentran conformado por un único archivo, que se encuentra dentro del subdirectorio ***Wbnms_Core/Error***.

Directorio Wbnms_Core/ Error:

- ***ErrorManager:*** Clase de manejo de presentación de errores del sistema ***WBNMS***.(Ver Anexo – Código ErrorManager.pm)

6.3.7.1 ErrorManager.pm

Esta clase formatea el mensaje de error de manera de que sea presentable al usuario (html). Es una clase que forma parte exclusivamente de la Interfaz.

Es requerido el archivo de configuración ***WbnmsConf***, de donde se obtiene el siguiente dato:

$WbnmsConf:: CONFERROR { 'ErrorPath'}= Path al programa CGI de Error

Existe para esta clase un único atributo definido:

- message: Mensaje de texto de error a ser presentado.

También existe solo una función implementada:

- Display: Devuelve una llamada en *JavaScript* a una ventana de error definida por el ***ErrorPath***.

6.3.8 Módulo AlarmSurveillance

Este Módulo se encuentran conformado por un nueve archivos, que se encuentra dentro del subdirectorio ***Wbnms_Core/AlarmSurveillance***.

Directorio Wbnms_Core/ AlarmSurveillance:

- ***AlarmSurd.pl:*** Demonio de procesamiento de Reportes (Ver Anexo – Código AlarmSurd.pl)
  - ***AlarmSur.rc:*** Script de control del demonio AlarmSurd (Ver Anexo – Código AlarmSurd.rc).
  - ***Start:*** Script de inicio del demonio (Ver Anexo – Código start).
  - ***Stop*** Script de stop del demonio (Ver Anexo – Código stop).
  - ***RenameAlarmSurlog.pl***: Script de renombrado de logs del demonio (Ver Anexo – Código renameAlarmSurlog.pl).
- ***ActionManager.pm:*** Clase de selección y disparo de sub-módulos de Alerta del ***Módulo ActionModules*** (Ver Anexo – Código ActionManager.pm).
- ***AlarmTriggersManager.pm:*** Clase de administración de Reportes (Triggers) y Alertas del Módulo (Ver Anexo – Código AlarmTriggersManager.pm).
- ***LogManager.pm:*** Clase de administración de Eventos del Módulo (Ver Anexo – Código LogManager.pm).
- ***StatusChange.pm:*** Clase de manejo de estados de gestión en Entidades-Equipos y Mapas (Ver Anexo – Código StatusChange.pm).

6.3.8.1 AlarmSurd.pl

Este Demonio es iniciado con el sistema y permanece activo a la espera de un evento factible de ser reportado.

Es requerido el archivo de configuración ***WbnmsConf***, del cual se extraen los siguientes datos:

$WbnmsConf:: CONFALARMSUR{'logpath'} = Path del log del AlarmSur Daemon
$WbnmsConf:: CONFALARMSUR{'AlarmSurfile.log' } = archivo de log
$WbnmsConf:: CONFALARMSUR{'pidfile' } = archivo de PID
$WbnmsConf:: CONFALARMSUR{'checkinterval' } = intervalo de verificación
$WbnmsConf:: CONFALARMSUR{'Debug' }= 1 Debug ON,0 Debug OFF

Clases y librerías Utilizadas:

- ***ABMManager*** para interactuar con la DB.
- ***POSIX*** para manejar los procesos (PIDs.
- ***ActionManager*** para manejar las alertas.
- ***StatusChange*** para manejar los cambios de estado en los mapas y entidades-equipos.

El funcionamiento del demonio de alarmas (reportes y alertas) (AlarmSurd.pl) graficado en la Figura 6-5 es el siguiente:

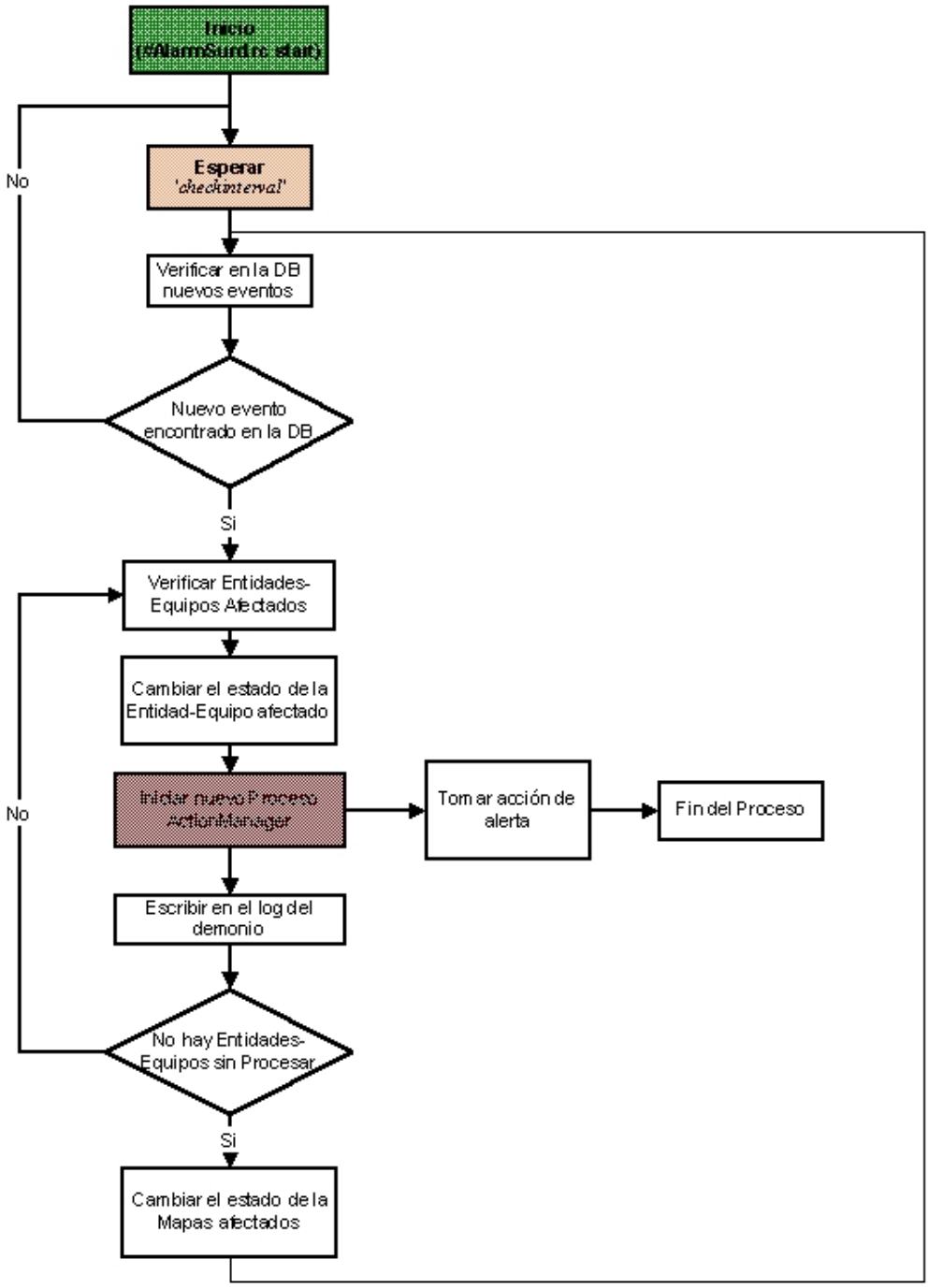


Figura 6-5 Diagrama de Flujo del demonio de Alarmas

El demonio es arrancado con el levantamiento del sistema operativo a través del script de control, AlarmSurd.rc. Luego entra en un ciclo de espera, dado por el ***'checkinterval'***. Terminado este tiempo, verifica en la DB, tabla ***TrapsCollector***, la existencia de nuevos eventos, ya procesados por el ***trapd***. Si existen estos eventos, sigue con el procesamiento de los mismos, de lo contrario comienza un

nuevo ciclo de espera. El procesamiento de los eventos, ahora reconocidos como reportes, comienza con una verificación de las entidades-equipos afectadas. Luego por cada una de ellas, se modifica su estado, y se genera un nuevo proceso, al llamar a un objeto ActionManager, el cual envía alertas cuando sea conveniente. Luego escribe en el archivo de log del demonio lo efectuado. Si no hay mas entidades-equipos para procesar relacionadas con el evento recibido, entonces se cambia el estado de los mapas afectados. Si no existen mas eventos a considerar, entonces se vuelve al ciclo de espera. De lo contrario se procesa el nuevo evento.

6.3.8.2 AlarmSurd.rc

Este es el script de control del demonio AlarmSurd. A través de él, se inicial, se apaga y se consulta el estado del demonio. Para ello se vale de los scripts de Start, Stop y renameAlarmSurlog.pl.

AlarmSurd.rc { start | stop | status | restart }

6.3.8.3 ActionManager.pm

Esta clase maneja el proceso de envío de alertas a los "usuarios de alerta". El código responde a la utilización de los tres sub-módulos de alerta, SMS, SNMP y EMAIL. Es por lo tanto, necesario modificar el código fuente para agregar nuevos sub-módulos de alerta.

Clases y librerías Utilizadas:

- ***Mailer*** , sub-módulo EMAIL.
- ***SMSMessenger***, sub-módulo SMS.
- ***Trapper***, sub-módulo SNMP.
- ***ABMManager*** para manejar el acceso a la Base de Datos.

La relación entre esta clase y las distintas clases asociadas es detallada en el gráfico UML de la siguiente figura:

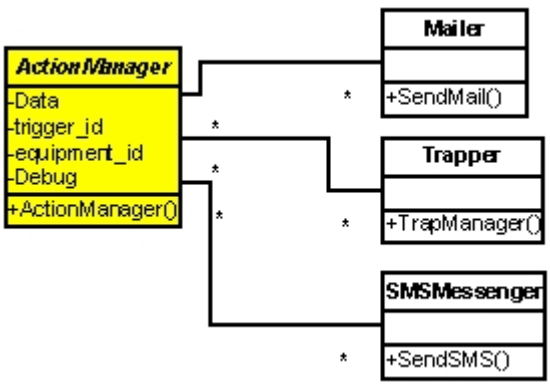


Figura 6-6 Relación de Clases ActionManager ,Mailer, Trapper y SMSMessenger

Los atributos de esta clase son :

- ***Data:*** Datos del evento a ser enviados en la Alerta.
- ***trigger_id:*** Identificador de disparador de evento (reporte configurado).
- ***equipment_id:*** Identificador de entidad-equipo relacionado con evento.
- ***Debug:*** Modo de Debugging, 0 OFF y 1 ON.

Esta Clase implementa un única función:

- ***ActionManager:*** Maneja el envió de alertas a los "usuarios de alerta".Para ello consulta la DB, específicamente la tabla ***ActionsTriggers***, de donde extrae las alertas configuradas. Es decir, que evento es factible de convertirse en una alerta. Además, verifica en la tabla ***ActionsUsersEquipments*** a modo de filtro (entidad-equipo, mapa, horario), cuales eventos se enviaran y cuales no. Luego llama al sub-módulo de alerta que corresponda según la configuración de la alerta y envía a través de él la Alerta propiamente dicha. Para definir los "usuarios de alerta", consulta la tabla ***UsersActions***.

6.3.8.4 AlarmTriggersManager.pm

Esta clase cumple las funciones administrativas del Módulo referidas a la configuración de disparadores de reportes (triggers), configuración de Alertas y presentación de eventos.

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***ErrorManager*** para manejar la presentación de los mensajes de Error.

Las funciones que implementa son:

- ***ReadAlarmTriggers:*** Consulta los disparadores de reportes (triggers) que se encuentran en la tabla ***AlarmsTrigger***. Esta función provee la posibilidad de utilizar distintos filtros en la búsqueda, de acuerdo con los parámetro que se le pasen.
- ***ReadAlarmTriggerfromID:*** Consulta los datos de un trigger especificado por su identificador (columna ***trigger_id***).
- ***ReadActionTriggers:*** Consulta las relaciones entre los triggers activados y las alertas asociadas. Para ello lee la tabla ***ActionsTriggers***.
- ***ReadEquipments:*** Lee las Entidades-Equipos configuradas.
- ***ReadEquipmentTypes:*** Lee los tipos de Entidades-Equipos configurados.
- ***ReadSeverity:*** Lee los estados configurados de la tabla ***MonitorEquipmentState*** (Severidades)**.**
- ***AddAlarmTriggers:*** Agrega en la tabla ***AlarmsTrigger*** un nuevo trigger.
- ***DelAlarmTriggers:*** Borra de la tabla ***AlarmsTrigger*** un trigger específico, según el identificador (trigger_id).

- ***ModAlarmTriggers:*** Modifica de la tabla ***AlarmsTrigger*** los datos de un trigger determinado.
- ***AddActionTrigger:*** Agrega una nueva asociación Alerta-Reporte en la tabla ***ActionsTriggers***.
- ***DelActionTrigger:*** Elimina una asociación Alerta-Reporte de la tabla ***ActionsTriggers***.
- ***ModActionTrigger:*** Modifica una asociación Alerta-Reporte en la tabla ***ActionsTriggers***.
- ***ReadObjects:*** Lee de la tabla ***Oid2label*** los datos de los objetos de gestión de la MIB ***WBNMS*** (Todas las MIBs compiladas en el sistema).
- ***ReadAllAlarms:*** Lee y reconoce los objetos de gestión del tipo "TrapType" y "NotificationType", los cuales son factibles de convertirse en triggers de reportes.

6.3.8.5 LogManager.pm

Esta clase se encarga de realizar funciones de administración de los eventos recibidos y procesados.

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***StatusChange*** para cambiar estados de las entidades-equipos.
- ***ErrorManager*** para manejar la presentación de los mensajes de Error.

Las funciones que implementa son:

- ***ReadAllLog:*** Lee todos los eventos y reportes (eventos procesados) de la tabla ***TrapsCollector***.
- ***ReadProcessedLog:*** Lee los reportes de la tabla ***TrapsCollector***. Y Busca los datos asociados al reporte de las tablas ***AlarmsTrigger***, ***Equipment***, ***EquipmentTypes***.
- ***ReadProcessedLogWUsers:*** Lee las alertas enviadas (evento-reporte-alerta) asociadas a los "usuarios de alerta".
- ***FilterLog:*** Filtra Eventos según criterios. Estos criterios son pasados a la función como parámetro.
- ***ProcessedAlarmStatus:*** Ofrece una traducción entre el formato interno de estados y severidades, con el formato a presentar en la interfaz.
- ***ParseAlarms:*** Utilizando las funciones ***ReadAllLog*** o ***ReadProcessedLog***, según sea el caso, esta función ofrece a la interfaz a posibilidad de realizar filtrados de los eventos-reportes y presentar los resultados.
- ***AckEvent:*** marca como reconocido un evento, columna ***marked = "ACK"*** de la tabla ***TrapsCollector***.

- ***AckAlarm:*** Reconoce un evento manualmente (desde la interfaz) indicando el usuario del sistema que lo ha requerido, columna ***ack="nombre del usuario"*** de la tabla ***TrapsCollector.***
- ***DelEvent:*** Borra un evento-reporte de la tabla ***TrapsCollector***.

6.3.8.6 StatusChange.pm

Esta clase se encarga de realizar los cambios de estado de gestión en las entidades-equipos y mapas relacionados con el evento procesado.

Los atributos de esta clase son :

- ***Data:*** Datos pasados de gestión.
- ***Debug:*** Modo de Debugging, 0 OFF y 1 ON.

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.

Las funciones que implementa son:

- ***StatusManager:*** Actualiza el estado de una entidad-equipo.
- ***EquipmentStatusChange:*** Modifica el estado de una entidad-equipo.
- ***MapStatusChange:*** Modifica el estado de un mapa.

6.3.9 Módulo Drawer

Este Módulo se encuentran conformado por diez archivos, que se encuentran dentro del subdirectorio ***Wbnms_Core/Drawer***.

Directorio Wbnms_Core/ Drawer:

- ***Drawerd.pl:*** Demonio Drawer. Graficador (Ver Anexo – Código Drawerd.pl).
  - ***Drawerd.rc:*** Script de control del demonio Drawer (Ver Anexo – Código Drawerd.rc).
  - ***start:*** script de inicio del demonio (Ver Anexo – Código start).
  - ***stop:*** script de stop del demonio (Ver Anexo – Código stop).
- ***Drawer.pm:*** Clase graficadora (Ver Anexo – Código Drawer.pm).
- ***EquipmentManager.pm:*** Clase de administración de entidades-equipos del módulo (Ver Anexo – Código EquipmentManager.pm).
- ***MapManager.pm:*** Clase de administración de mapas del módulo (Ver Anexo – Código MapManager.pm).
- ***MonitorManager.pm:*** Clase de Interfaz del Monitor de estados (Ver Anexo – Código MonitorManager.pm).
- ***BmpManager.pm:*** Clase de administración de archivos gráficos (GIF) (Ver Anexo – Código BmpManager.pm).

- ***CoorManager.pm:*** Clase de administración de ubicación gráfica (Ver Anexo – Código CoorManager.pm).

6.3.9.1 Drawerd.pl

Se encarga de Graficar las entidades-equipos y mapas, de manera que representen sus estados de gestión, para ser presentados al usuario. Además genera el código html necesario para ofrecer la navegabilidad de los mismos.

Es requerido el archivo de configuración ***WbnmsConf***, del cual se extraen los siguientes datos:

$WbnmsConf:: CONFDRAWER {'pidfile' } = archivo de PID
$WbnmsConf::CONFDRAWER{'checkinterval'}=intervalo de verficación
$WbnmsConf:: CONFDRAWER {'Debug' }= 1 Debug ON,0 Debug OFF

Clases y librerías Utilizadas:

- ***ABMManager*** para almacenar los eventos recibidos..
- ***POSIX*** para manejar los procesos (PIDs).
- ***Drawer*** para realizar los gráficos.

El funcionamiento del demonio Drawer (Drawerd.pl) descripto en el diagrama de flujo de la Figura 6-7 es el siguiente:

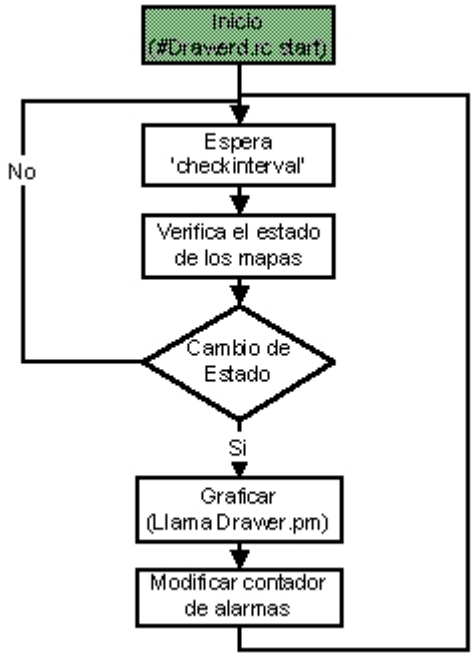


Figura 6-7 Diagrama de Flujo de demonio Drawer

El demonio es iniciado con el sistema operativo. Luego entra en un período de espera dado por el ***'checkinterval'***. Al terminar este tiempo de espera, el demonio verifica si existió un cambio de estado en los mapas configurados. Si no ha habido

ningún cambio, entonces vuelve a comenzar otro período de espera. De lo contrario, se dispara un proceso de Graficación (Drawer.pm), que realiza todos los cambios pertinentes en las representaciones de los mapas. Luego se actualiza la presentación del contador de alarmas (reportes).Después de lo cual, vuelve a comenzar el ciclo, al volver a otro período de espera.

6.3.9.2 Drawerd.rc

Este es el script de control del demonio Drawerd.pl. A través de él, se inicia, se apaga y se consulta el estado del demonio. Para ello se vale de los scripts de Start, Stop.

Drawerd.rc { start | stop | status | restart }

6.3.9.3 Drawer.pm

Esta clase se encarga de realizar el graficado de los mapas del sistema de gestión y del contador de alarmas. Para verificar el estado de las entidades-equipos, recurre a la tabla ***EquipmentMonitor*** y para el estado de los mapas a la tabla ***MapsMonitor***.

Requiere del archivo de configuración ***WbnmsConf*** del cuál extrae los siguientes datos:

$WbnmsConf::CONFPATH{'File_DB'}=Path al directorio File_DB.
$WbnmsConf::CONFDRAWER{'coloredweigh'}=Ancho del area de estado.
$WbnmsConf::CONFDRAWER{'DrawerWebpath'}=Path al directorio Web destino de los gráficos generados
$WbnmsConf::CONFDRAWER{'AlarmsCounterfile'}= nombre del archivo gráfico contador de alarmas.

Clases y Librerías Utilizadas:

- ***GD*** para graficar(librería gráfica).
- ***ABMManager*** para el acceso a la Base de Datos.

Los atributos de esta clase son:

- ***state_id:*** Identificador de estado(no implementado).
- ***equipment_id:*** Identificador de entidad-equipo(no implementado).
- ***map_id:*** Identificador de mapa.
- ***time:*** fecha y hora de procesamiento del mapa.
- ***Debug:*** Debugging option, 0 OFF y 1 ON.

Las funciones que implementa son:

- ***DrawerManager:*** Lee los estados de las entidades-equipos y mapas y llama a la función ***Draw*** para graficar.

- ***Draw:*** Utiliza la librería ***GD.pm*** para generar los archivos gráficos, GIF, de acuerdo con los parámetros que le son suministrados. Esta función es llamada desde la función ***DrawerManager***.
- ***AlarmsCounter:*** utiliza la librería ***GD.pm*** para graficar el contador de alarmas actualizado.

6.3.9.4 EquipmentManager.pm

Esta clase cumple funciones administrativas relacionadas con las entidades-equipos y sus tipos.

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***ErrorManager*** para manejar los mensajes de error.

Implementa las siguientes funciones:

- ***ReadEquipments:*** Lee los datos de las entidades-equipos y tipos.
- ***ReadMapsEquipments:*** Lee los datos de las entidades-equipos asociados a un mapa.
- ***AddEquipment:*** Agregar una nueva entidad-equipo. Son agregados en la tabla ***Equipment*** y la tabla ***EquipmentMonitor***.
- ***DelEquipment:*** Borra una entidad-equipo del sistema.
- ***ModEquipment:*** Modifica los datos de una entidad-equipo.
- ***AddType:***Agrega un nuevo tipo de entidad-equipo al sistema. Se agrega en la tabla ***EquipmentTypes***.
- ***DelType:*** Borra un tipo de entidad-equipo del sistema.
- ***ModType:*** Modifica los datos de un tipo de entidad-equipo.

6.3.9.5 MapManager.pm

Esta clase cumple funciones administrativas relacionadas con los mapas del sistema.

Requiere del archivo de configuración ***WbnmsConf*** del cuál extrae los siguientes datos:

$WbnmsConf::CONFPATH{'File_DB'}=Path al directorio File_DB.
$WbnmsConf::CONFDRAWER{'coloredweigh'}=Ancho del area de estado.
$WbnmsConf::CONFDRAWER{'DrawerWebpath'}=Path al directorio Web destino de los gráficos generados
$WbnmsConf::CONFDRAWER{'AlarmsCounterfile'}= nombre del archivo gráfico contador de alarmas.
$WbnmsConf::GENERAL{'User'}=Usuario del sistema (root).

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***File::Copy*** para mover archivos.
- ***ErrorManager*** para manejar los mensajes de error.

Implementa las siguientes funciones:

- ***ReadMaps:*** Lee los datos de los mapas. De la tabla ***Maps***.
- ***ReadMapsMaps:*** Lee los datos de los mapas contenidos en otro mapa especificado por su identificador (map_id). Para ello recurre a la tabla ***MapsMaps***.
- ***AddMap:*** Agrega un mapa al sistema. Es agregado en las tablas ***Maps***, ***MapsMonitor*** y copiado el archivo de plantilla del mapa al directorio ***'DrawerWebpath'***.
- ***DelMap:*** Borra un mapa del sistema. Realiza una verificación para que no existan ni mapas ni entidades-equipos asociados.
- ***ModMap:*** Modifica los datos de un mapa.

6.3.9.6 MonitorManager.pm

Esta clase se encarga de realizar cálculos auxiliares para la presentación gráfica de reportes-alarmas del sistema.

Requiere del archivo de configuración ***WbnmsConf*** del cuál extrae los siguientes datos:

$WbnmsConf::CONFDRAWER{'coloredweigh'}=Ancho del area de estado.
$WbnmsConf::CONFDRAWER{'DrawerWebpath'}=Path al directorio Web destino de los gráficos generados
$WbnmsConf::CONFDRAWER{'definition'}= Definición de la cuadrícula de coordenadas

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***ErrorManager*** para manejar los mensajes de error.

Implementa las siguientes funciones:

- ***Mapsize:*** Calcula el tamaño de un Mapa , identificado por el parámetro ***map_id***.
- ***MapParse:*** Lee los datos de un Mapa especificado por su identificador (map_id). Devuelve la ubicación de entidades-equipos y mapas contenidos.
- ***Counter:*** Incrementa un contador y devuelve un archivo JavaScript que presenta el resultado gráficamente.
- ***AllAlarms:*** Cuenta de la tabla ***TrapsCollector*** todos los eventos que no hallan sido ya reconocidos y devuelve el resultado.

6.3.9.7 BmpManager.pm

Esta clase es la encargada de realizar las funciones administrativas sobre los archivos de plantillas gráficos.

Requiere del archivo de configuración ***WbnmsConf*** del cuál extrae el dato:

$WbnmsConf::CONFPATH{'File_DB'}=Path al directorio File_DB.

Clases y Librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***ErrorManager*** para manejar los mensajes de error.

Implementa las siguientes funciones:

- ***UpLoad:*** Copia un archivo que ha sido levantado al servidor en forma binaria a un path determinado.
- ***DelBmp:*** Borra un archivo de plantilla gráfica. Verifica que no existan dependencias con él.
- ***Seebmp:*** Lee los directorios de plantillas, para presentarlas como vista previa a la selección.

6.3.9.8 CoorManager.pm

Esta clase administra la ubicación, en coordenadas, de mapas y entidades-equipos dentro de un mapa.

Requiere del archivo de configuración ***WbnmsConf*** del cuál extrae los siguientes dato:

$WbnmsConf::CONFDRAWER{'DrawerWebpath'}= Path web de los archivos gráficos del Drawerd.
$WbnmsConf::CONFDRAWER{'definition'}= Definición de la cuadrícula de coordenadas.

Clases y Librerías Utilizadas:

- ***GD*** para calcular las dimensiones de los gráficos(librería gráfica).
- ***ABMManager*** para el acceso a la Base de Datos.
- ***ErrorManager*** para manejar los mensajes de error.

Implementa las siguientes funciones:

- ***CoorCalculate:*** Calcula todo el rango de coordenadas para un gráfico determinado.
- ***ReadOcupatedCoors:*** Obtiene para un mapa , las coordenadas ya ocupadas.

- ***FormatView:*** Da formato a un conjunto de coordenadas. Coordenadas separadas por ***':'***.
- ***AddMap:*** Asocia un mapa a una coordenada determinada de un mapa.
- ***AddEquipment:*** Asocia una entidad-equipo a una coordenada determinada de un mapa.
- ***DelEquipment:*** Borra la asociación de unas entidad-equipo con una coordenada determinada en el mapa.
- ***DelMap:*** Borra la asociación de un mapa con una coordenada determinada en el su mapa raíz.
- ***ModMap:*** Modifica la posición de un mapa dentro de su mapa raíz.
- ***ModEquipment:*** Modifica la posición de una entidad-equipo dentro un mapa.

6.3.10 Módulo 'ActionModules'

Este Módulo se encuentran conformado por cinco archivos, que se encuentran dentro del subdirectorio ***Wbnms_Core/ActionModules***. Este módulo agrupa los sub-módulos de alerta del sistema.

Directorio Wbnms_Core/ ActionModules:

- ***ActionGuiManager.pm:*** Clase de administración de Usuarios y Asignaciones de Alertas (Ver Anexo – Código ActionGuiManager.pm).
- ***Mailer.pm:*** sub-módulo de envío de e-mails (Ver Anexo – Código Mailer.pm).
- ***Trapper.pm:***sub-módulo de envío de traps (Ver Anexo – Código Trapper.pm).
- ***SMSMessenger.pm:*** sub-módulo de envío de mensajes SMS (Ver Anexo – Código SMSMessenger.pm).
    - ***SendSMS.pm:*** librería de envío de mensajes SMS (Código CDSATG – Siemens Argentina).

6.3.10.1 ActionGuiManager.pm

Esta clase implementa funciones administrativas referentes a los Usuarios de Alertas y a la Asignación de Alertas.

Clase Utilizada:

- ***ABMManager*** para el acceso a la Base de Datos.

Las funciones que implementa son:

- ***ReadActionUsers:*** Lee los datos de los usuarios de alerta configurados. Para ello recurre a la tabla ***UsersActions***.
- ***AddActionUser:*** Agrega a la tabla ***UsersActions*** los datos de un nuevo usuario de alertas.
- ***DelActionUser:*** Borra un usuario de alertas de la tabla ***UsersActions***.
- ***ModActionUser:*** Modifica los datos de un usuario de alertas.

- ***ReadActionAssignations:*** Lee los datos de las asignaciones de alerta realizadas. Para ello consulta las tablas ***ActionsUsersEquipments***, ***AlarmsActions***, ***Equipment*** y ***Maps***.
- ***AddAsignationAction:*** Crea una nueva asociación entre un reporte y un módulo de alerta. En la tabla ***ActionsUsersEquipments.***
- ***DelAsignationAction:*** Borra una asociación de la tabla ***ActionsUsersEquipments***.
- ***ModAsignationAction:*** realiza una modificación en la asociación.

6.3.10.2 Mailer.pm

Esta clase conforma el sub-módulo de envío de alertas vía e-mails. Para ello recurre al protocolo SMTP.

Es requerido el archivo de configuración ***WbnmsConf*** del que se extrae la siguiente información:

```
%WbnmsConf::CONFMAILER =
 (
          'logpath' => "Path al directorio de logs del sub-módulo",
        'mailer.log' => "archivo de log",
          'from' => "remitente del e-mail(AlarmSurveillance)",
          'linkroot' => "link de consulta enviado (www.wbnms.com.ar)",
);
```

Clases Utilizada:

- ***Net::SMTP*** para el envío de e-mails.

Los atributos de esta clase son:

- ***UserData:*** Datos del usuario de alerta.
- ***AlarmData:*** Datos de la alerta a enviar.
- ***Debug:*** Debugging option, 0 OFF y 1 ON.

Las funciones que implementa son:

- ***SendMail:*** Da formato y envía el e-mail al usuario de alerta definido.
- ***logger_Mail:*** Escribe en el archive de log.

6.3.10.3 Trapper.pm

Esta clase representa el sub-módulo de envío de PDU Traps SNMPv1 a los usuarios de alerta configurados.

Clases y librerías Utilizadas:

- ***ABMManager*** para el acceso a la Base de Datos.
- ***SNMP_util*** para el envío de traps.

Los atributos de esta clase son:

- ***UserData:*** Datos del usuario de alerta.
- ***AlarmData:*** Datos de la alerta a enviar.
- ***Debug:*** Debugging option, 0 OFF y 1 ON.

Las funciones que implementa son:

- ***SendTrap:*** envía los Traps al usuario de alerta definido por ***UserData***.
- ***TrapManager:*** Da formato al mensaje y llama a la función ***SendTrap***.

6.3.10.4 SMSMessenger.pm

Esta clase maneja el envío de mensajes SMS dentro del sub-módulo de SMS.

Es requerido el archivo de configuración ***WbnmsConf*** del que se extrae la siguiente información:

```
%WbnmsConf::CONFSMS =
 (
          'logpath' => " Path al directorio de logs del sub-módulo ",
         'sms.log' => " archivo de log ",
          'from' => " remitente del SMS(AlarmSurveillance)",
);
```

Librería Utilizada:

- ***sendSMS*** para el envío de mensajes SMS.

Los atributos de esta clase son:

- ***UserData:*** Datos del usuario de alerta.
- ***AlarmData:*** Datos de la alerta a enviar.
- ***Debug:*** Debugging option, 0 OFF y 1 ON.

Las funciones que implementa son:

- ***SendSMS:*** Da formato y envía el mensaje SMS al usuario de alerta definido. El envió del mismo es realizado utilizando la librería ***sendSMS***.
- ***logger_SMS:*** Escribe en el archive de log.

6.3.11 Módulo MIBCompiler

Este Módulo se encuentran conformado por ocho archivos, que se encuentran dentro del subdirectorio ***Wbnms_Core/ MIBCompiler***.

Directorio ***Wbnms_Core/ MIBCompiler:***

- ***MIBManager.pm:*** Clase administrativa del modulo (Ver Anexo – Código MIBManager.pm).
- ***mibmysql.pm:*** Clase que maneja la compilación de Mibs dentro de la estructura de la base de datos (Ver Anexo – Código mibmysql.pm).
  - ***object_identifier.pm:*** Clase que compila los OBJECT-IDENTIFIER de la MIB (Ver Anexo – Código object_identifier.pm).
  - ***object_type.pm:*** Clase que compila los OBJECT-TYPE de la MIB (Ver Anexo – Código object_type.pm).
  - ***notifycation_type.pm:*** Clase que compila los NOTYFICATION-TYPE de la MIB (Ver Anexo – Código notification_type.pm).
  - ***trap_type.pm:*** Clase que compila los TRAP-TYPE de la MIB (Ver Anexo – Código trap_type.pm).
  - ***textual_conventions.pm:*** Clase que compila los TEXTUAL-CONVENTION de la MIB (Ver Anexo – Código textual_conventions.pm).
  - ***sequence.pm:*** Clase que compila los SEQUENCE de la MIB (Ver Anexo – Código sequence.pm).

6.3.11.1 MIBManager.pm

Esta clase realiza las funciones administrativas de este módulo.

Es requerido el archivo de configuración ***WbnmsConf*** del que se extrae la siguiente información:

$WbnmsConf::CONFPATH{'MIBDir'})=Path al directorio de MIBs
$WbnmsConf::CONFMIBS{'MIBIndex'})=Path al archivo indice de MIBs Compiladas.
$WbnmsConf::CONFDB{'DataBaseUserName'}= Nombre de usuario de la DB.
$WbnmsConf::CONFDB{'DataBasePassword'}= Password del usuario de la DB
$WbnmsConf::CONFDB{'DataBaseName'}= Nombre de la DB (WBNMSDB)
$WbnmsConf::CONFDB{'DataBaseHost'}=Hots de la DB.

Clases y librerías Utilizadas:

- ***mibmysql*** para la compilación de MIBs.
- ***ErrorManager*** para presentar los mensajes de error.
- ***Time::localtime*** para la presentación de fecha.

Los atributos de esta clase son:

- ***MIB:*** Nombre de MIB.
- ***Debug:*** Debugging option, 0 OFF y 1 ON.

Las funciones que implementa son:

- ***ReadMIBs:*** Lee las MIBs disponibles el directorio ***'MIBDir'*** y las ya compiladas, registradas en ***'MIBIndex'***.

- ***AddMIBs:*** Agrega un conjunto de MIBs al archivo ***'MIBIndex'***.
- ***DelMIBs:*** Borra una MIB del archivo ***'MIBIndex'***.
- ***MibLoad:*** se vale de la función ***mib2mysql*** de la librería ***mibmysql*** para compilar una MIB.
- ***UpLoad :*** Levanta (*UpLoad*) una MIB al servidor ***WBNMS***.
- ***Time:*** Devuelve formateada la fecha (***día/mes/año::hora:min:sec***).

6.3.11.2 mibmysql.pm

Esta librería provee la función de compilación de MIBs, ***mib2mysql***. El funcionamiento de la cual se encuentra detallado en la siguiente figura:

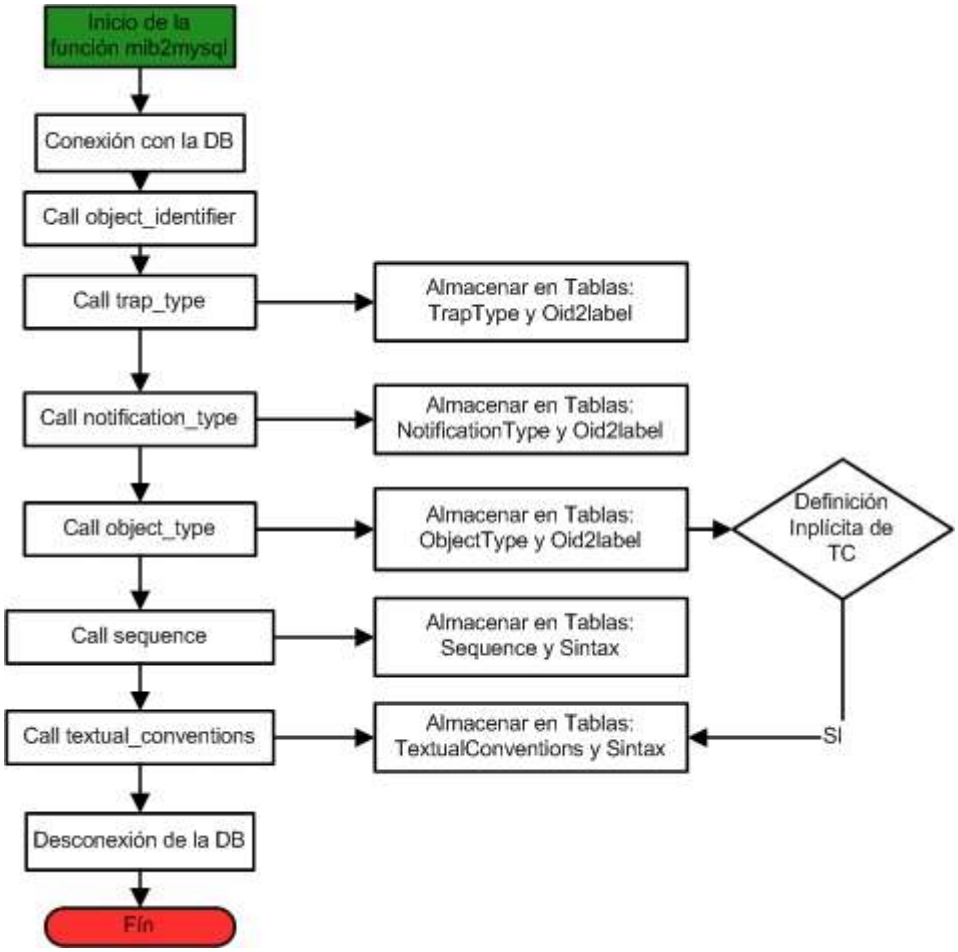


Figura 6-8 Diagrama de flujo del proceso de compilación de MIBs

Se conecta a la Base de Datos y luego llama a las distintas librerías de compilación y almacenando en la DB los datos extraídos de la MIB. Si existe en algún punto un mensaje de error, se termina el proceso.

Clases y librerías Utilizadas:

- ***DBI*** para la conexión con la DB.
- ***object_identifier*** para compilar los objetos tipo OBJECT-IDENTIFIER.

- ***trap_type*** para compilar los objetos tipo TRAP-TYPE.
- ***notification_type*** para compilar los objetos tipo NOTIFICATION-TYPE.
- ***object_type*** para compiler los objectos tipo OBJECT-TYPE.
- ***sequence*** para compiler los objectos tipo SEQUENCE.
- ***textual_conventions*** para compiler los objectos tipo TEXTUAL-CONVENTIONS.

6.3.11.3 object_identifier.pm

Esta librería implementa la función ***snmpMIB_to_OI***. La cual compila los datos relacionados con los objetos del tipo OBJECT_IDENTIFIER existentes en la MIB y los almacena en el archivo del sistema que registra la estructura jerárquica de la MIB del sistema ***WBNMS***. Este archivo, que se encuentra en el directorio ***$WbnmsConf::CONFPATH{'var'}*** (requiere el archivo ***WbnmsConf***), se llama ***ObjectIdentifiers*** y tiene el siguiente formato:

<nombre del OID>:<OID>:<nombre del OID de jerarquia superior (padre)>

La función intenta ubicar jerárquicamente los objetos leídos de la MIB, si no tiene éxito entonces se marca el error de ***falta cargar MIB previa***.

6.3.11.4 trap_type.pm

Esta librería contiene la función ***snmpMIB_to_TT***. La cual compila los datos relacionados con los objetos del tipo TRAP-TYPE (RFC1215) existentes en la MIB y los devuelve en la variable ***%trap_type::ObjectData***.
Requiere el archivo de configuración ***WbnmsConf*** para obtener el valor de ***$WbnmsConf::CONFPATH{'var'}***. Ya que necesita el Path al archivo ***ObjectIdentifiers,*** de manera de verificar que los objetos a compilar no necesiten ninguna MIB precedente.

Se extraen los siguientes datos por objeto:

- ***ENTERPRISE:*** Identificador de empresa (enterprise number).
- ***SPECIFICTRAP:*** Identificador del tipo de trap.
- ***DESCRIPTION:*** Descripción del trap.
- ***VARIABLES:*** variables que contiene el trap.

Además se genera un ***OID = ENTERPRISE+.+ SPECIFICTRAP***

Todos estos datos son agrupados por objetos y guardados en la variable ***%trap_type::ObjectData***.

6.3.11.5 notification_type.pm

Esta librería contiene la función ***snmpMIB_to_NT***. La cual compila los datos relacionados con los objetos del tipo NOTIFICATION-TYPE (SNMPv2-SMI) existentes en la MIB y los devuelve en la variable ***%notification_type::ObjectData***.
Requiere el archivo de configuración ***WbnmsConf*** para obtener el valor de ***$WbnmsConf::CONFPATH{'var'}***. Ya que necesita el Path al archivo ***ObjectIdentifiers,*** de manera de verificar que los objetos a compilar no necesiten ninguna MIB precedente.

Se extraen los siguientes datos por objeto:

- ***OBJECTS:*** Variables del trap.
- ***STATUS:*** Indica el estado de situación del objeto en la MIB (obligatorio, opcional).
- ***DESCRIPTION:*** Descripción del trap.
- ***OID:*** Identificador de objeto.

Todos estos datos son agrupados por objetos y guardados en la variable ***%notification_type::ObjectData***.

6.3.11.6 object_type.pm

Esta librería contiene la función ***snmpMIB_to_OT***. La cual compila los datos relacionados con los objetos del tipo OBJECT-TYPE (RFC1155-SMI para SNMPv1 y en la SNMPv2-SMI para SNMPv2) existentes en la MIB y los devuelve en la variables ***%object_type::ObjectData*** y ***%object_type::AlarmTypes***.
Requiere el archivo de configuración ***WbnmsConf*** para obtener el valor de ***$WbnmsConf::CONFPATH{'var'}***. Ya que necesita el Path al archivo ***ObjectIdentifiers,*** de manera de verificar que los objetos a compilar no necesiten ninguna MIB precedente.

Se extraen los siguientes datos por objeto:

- ***SYNTAX:*** sintaxis del Objeto.
- ***MAX-ACCES:*** Modo de acceso (SNMPv2).
- ***ACCESS:*** Modo de acceso (SNMPv1).
- ***STATUS:*** Indica el estado de situación del objeto en la MIB (obligatorio, opcional).
- ***DESCRIPTION:*** Descripción del trap.
- ***OID:*** Identificador de objeto.

Todos estos datos son agrupados por objetos y guardados en la variable ***%object_type::ObjectData***. Existen definiciones de convenciones textuales dentro del campo ***SYNTAX***, en este caso se almacenan en la variable ***%object_type::AlarmTypes.***

6.3.11.7 sequence.pm

Esta librería contiene la función ***snmpMIB_to_S***. La cual compila los datos relacionados con los objetos del tipo SEQUENCE (ASN.1) existentes en la MIB y los devuelve en la variable ***%sequence::sequence***.

Se extraen los siguientes datos por objeto:

- ***SEQUENCE:*** Definición de los Objetos que componen la secuencia (Tabla).

El Conjunto de Objetos son agrupados por secuencia y guardados en la variable ***%sequence::sequence***.

6.3.11.8 textual_conventions.pm

Esta librería contiene la función ***snmpMIB_to_TC***. La cual compila los datos relacionados con los objetos del tipo TEXTUAL-CONVENTION (SNMPv2-TC) existentes en la MIB y los devuelve en la variable ***%textual_conventions::AlarmTypes***.

Se extraen los siguientes datos por objeto:

- ***SYNTAX:*** Definición la convención. Traducción entre texto y número entero que lo representa.

Se agrupan las distintas convenciones con el nombre que las asocia y so almacenadas en la variable ***%textual_conventions::AlarmTypes***.

6.3.12 Módulo MIBBrowser

Este Módulo se encuentran conformado por dos archivos, que se encuentran dentro del subdirectorio ***Wbnms_Core/ MIBBrowser***.

Directorio ***Wbnms_Core/ MIBBrowser:***

- ***MIBBrowser.pm:*** Clase de servicio de la Interfaz (Ver Anexo – Código MIBBrowser.pm).
- ***OID2js.pm:*** Clase asociada a la generación de archivos soporte de la presentación gráfica de la MIB (Ver Anexo – Código OID2js.pm).

6.3.12.1 MIBBrowser.pm

Esta clase implementa funciones que brindan un servicio a la Interfaz.

Es requerido el archivo de configuración ***WbnmsConf*** del que se extrae la siguiente información:

$WbnmsConf::CONFPATH{'var'}=Path al directorio de Variables del Sistema

Clases y librerías Utilizadas:

- ***ABMManager*** para el manejo de la Base de Datos.
- ***SNMP_util*** para el manejo de los OID.

Las funciones que implementa son:

- ***ReadOID:*** Lee los datos referidos a los OID definidos en el sistema. Esto es, en la tabla ***Oid2Label*** y en el archivo ***ObjectIdentifiers***.
- ***FindChild:*** Busca los Objetos dependientes a partir de uno (padre) suministrado.
- ***RequestInfo:*** Consulta a la DB los datos de un Objeto específico suministrado.

6.3.12.2 OID2js.pm

Esta clase es la generadora de los archivos necesarios para la presentación gráfica de la MIB del sistema ***WBNMS***.

Es requerido el archivo de configuración ***WbnmsConf*** del que se extrae la siguiente información:

$WbnmsConf::CONFPATH{'var'}=Path al directorio de Variables del Sistema
$WbnmsConf::CONFMIBS{'MIBbrowserjs'}= Nombre del archivo JavaScript que contiene la información de árbol de la MIB del Sistema.
$WbnmsConf::CONFMIBS{'MIBbrowsercgi'}=nombre del archivo CGI que procesa la navegación.

Clases y librerías Utilizadas:

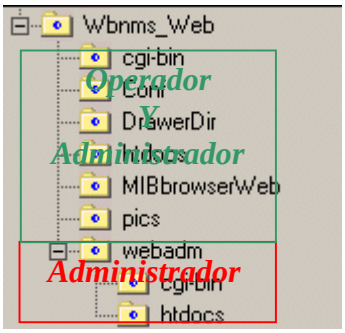
- ***ABMManager*** para el manejo de la Base de Datos.
- ***SNMP_util*** para el manejo de los OID.

Las funciones que implementa son:

- ***OID2js:*** Lee los datos de la MIB del sistema y genera un nuevo archivo ***'MIBbrowserj'*** con los datos jerárquicos de la misma.
- ***ReadOID:*** Lee los datos referidos a los OID definidos en el sistema. Esto es, en la tabla ***Oid2Label*** y en el archivo ***ObjectIdentifiers***. Estos datos son formateados según la necesidad de la función ***OID2js*** que es la que requiere de ellos.

6.4 Interfaz Web

Esta compuesta principalmente por archivos html y CGI. La estructura de archivos es la siguiente:



Existen dos Niveles, según sea el usuario Operador o Administrador, para el acceso a los determinados archivos de la interfaz. Existe un directorio ***htdocs*** de archivos HTML y uno ***cgi-bin*** de archivos CGI asignados a cada uno de los dos niveles. De esta forma, todos los archivos que se encuentren en los subdirectorios sobre el raíz de la interfaz (Wbnms_Web), serán accesibles por todo los usuarios; y los que se encuentren dentro del subdirectorio ***webadm*** seran solo accesibles por los usuarios del tipo Administrador.

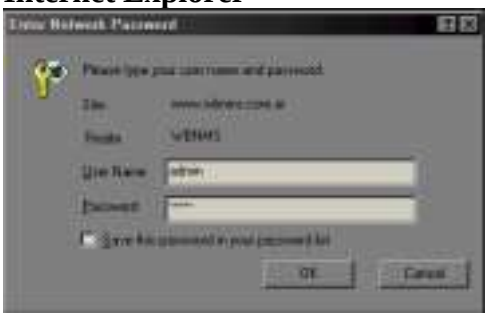
Se denomina como área de Administración a la cual solo tiene acceso el usuario Administrador. A la otra área se la denomina área de Operación.
Marginalmente de estas áreas, la primer pantalla que enfrenta un usuario es la de autentificación.

6.4.1 Autentificación

Es presentada por el navegador ante el pedido del servidor Web (Apache). En ella el usuario debe ingresar su nombre de usuario y clave correcta para ingresar al sistema ***WBNMS***.

Estos son ejemplos de ventanas de validación, propuestas por los dos navegadores base del mercado.

Internet Explorer



Mozilla

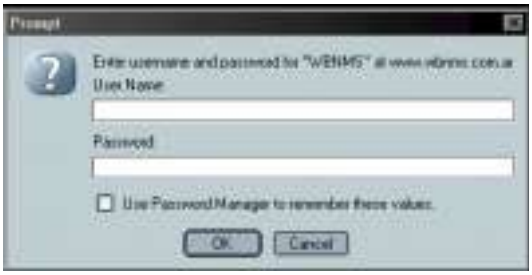


Figura 6-9 Ventanas de Validación

Si la validación es descartada por el servidor Web, entonces este envía un mensaje de error ***401*** (Requerimiento de Autentificación) que es mostrado en la ventana principal del navegador del cliente y guarda en un archivo de log el intento fallido.

Internet Explorer

Figura 6-10 Mensaje de Error de Validación

Si el usuario, en cambio es validado correctamente, entonces ingresa a la pantalla principal de la Interfaz ***WBNMS***. Se establece la sesión y permanece abierta hasta que el usuario del sistema lo abandona completamente.

6.4.2 Estructura del Sitio Web

Todas las pantallas que conforman la Interfaz ***WBNMS*** guardan un solo formato.
La estructura básica de las pantallas de la Interfaz del sistema esta compuesta por tres Marcos (Frames) detallados en la Figura 6-11:

- ***Marco Principal:*** En el se despliegan todas las pantallas de trabajo del sistema ***WBNMS*** .
- ***Marco de Menú Principal:*** Con la carga inicial del sistema , se despliega en este Marco el menú principal del sistema. El cual contiene las opciones de básicas, o de alto nivel.
- ***Marco de Menú Secundario:*** En el se despliegan los menús secundarios o de bajo nivel cuando una opción es elegida del menú principal.

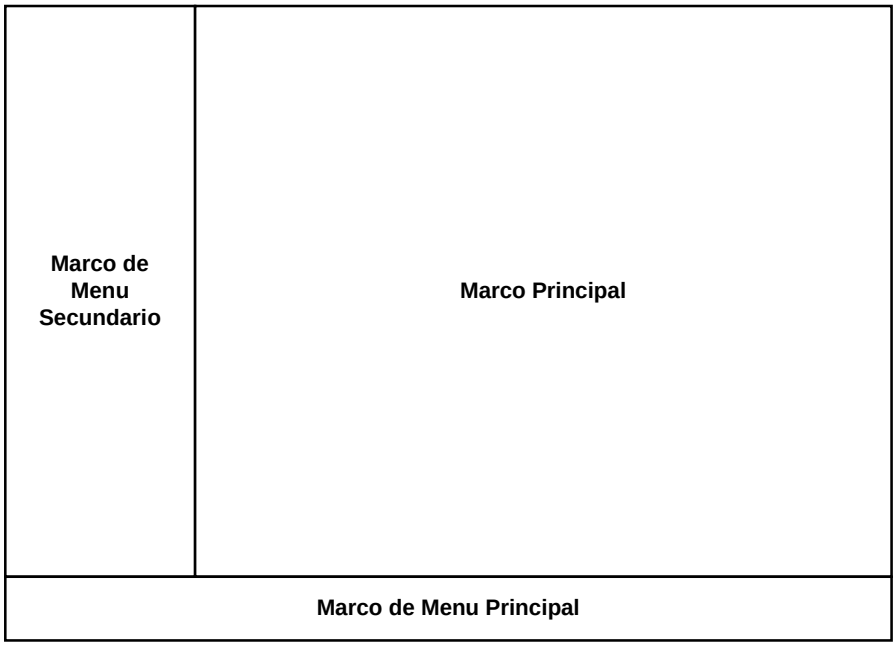


Figura 6-11 Estructura de Marcos (Frames) de la Pantalla

Esta estructura (Frame Set) se carga inicialmente desde el archivo ***Main.html*** que se encuentra dentro del directorio ***Wbnms_Web/htdocs*** (Ver Anexo Código - Main.html).

6.4.3 Archivos de Configuración

Se utiliza de archivos de configuración de estilos (CSS) para unificar el estilo de todas las pantallas del sistema ***WBNMS***. Asimismo se utiliza un archivo JavaScript (js) para el estilo dinámico de los menús. Estos archivos se encuentran dentro del directorio ***Wbnms_Web/Conf***:

- ***WBNMS.css:*** Cargado desde todos archivos de Interfaz (Ver Anexo Código - WBNMS.css).
- ***menu.js:*** Cargado desde todos los archivos de Interfaz que requieran de un menú (Ver Anexo Código - menu.js).

6.4.4 Pantalla Principal

El Usuario que accede al sistema ***WBNMS*** carga en primer lugar el archivo ***Main.html***, el cual, como ya se dijo, da la estructura a la Interfaz, pero además carga dentro de esta estructura las siguientes pantallas descriptas en la Figura 6-12:

- ***Menú principal:*** dado por el archivo ***Wbnms_Web/htdocs/Menu.html*** dentro del ***Marco de Menú Principal.***
- ***Pantalla de Logo principal:*** dado por el archivo ***Wbnms_Web/htdocs/Mainhome.html*** dentro del ***Marco Principal***.

- ***Pantalla de Logo lateral:*** dado por el archivo ***Wbnms_Web/htdocs/Lefthome.html*** dentro del ***Marco de Menú Secundario***.

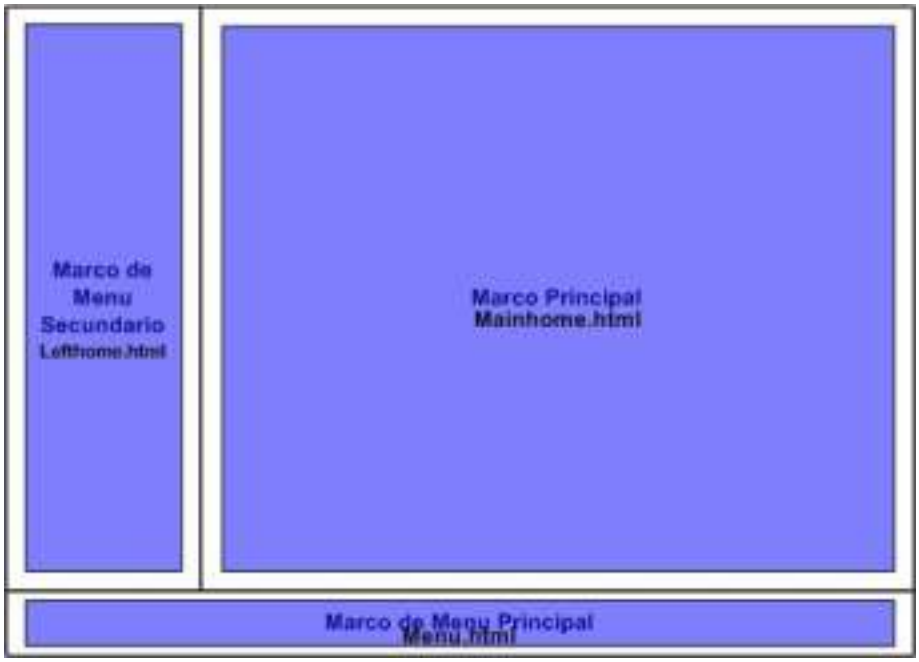


Figura 6-12 Descripción de los Marcos y contenido

La pantalla queda de la siguiente forma:

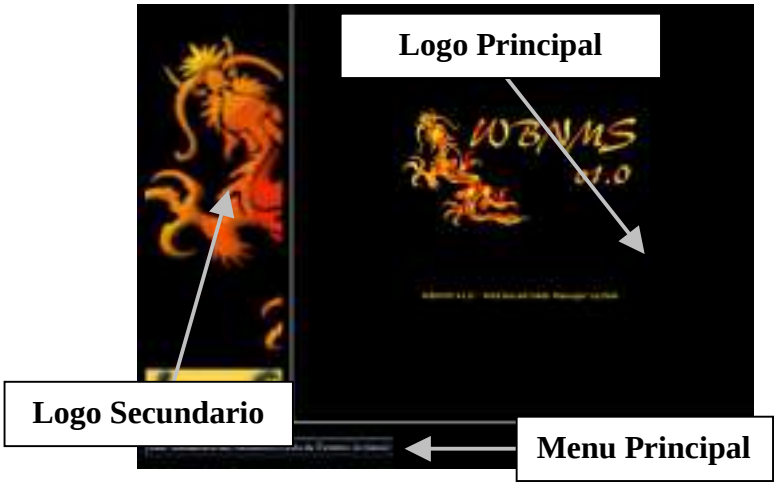


Figura 6-13 Detalle de la Pantalla Principal

6.4.4.1 Menu.html

Este es el menú principal de la Interfaz del sistema ***WBNMS*** (Ver Anexo Código-Menu.html). Desde él se accede a las pantallas de Operación y Administración.

Además se encuentra la opción de ***Salir*** del sistema así como también un acceso a la pantalla de ***About***. A continuación se detallan cada una de sus opciones siguiendo la Figura 6-14.

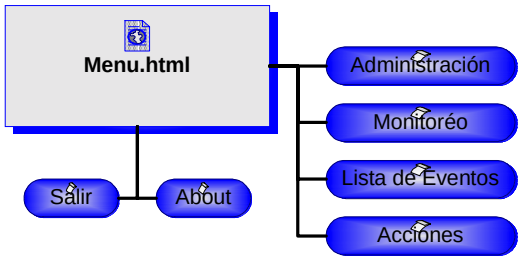


Figura 6-14 Opciones del Menú Principal

- ***Administración:*** Modo de Administración. Llama al archivo ***/webadm/htdocs/Administration.html***.
- ***Monitoreo:*** Pantallas de Monitoreo de Mapas. Llama al archivo ***/htdocs/Monitor.html***.
- ***Lista de Eventos:*** Muestra las pantallas de Lista de Eventos. Llama al archivo ***/htdocs/LogAlarms.html***.
- ***Acciones:*** Carga las pantallas de ejecución de Perfiles de Acción.Llama al archivo ***/htdocs/Actions.html***.
- ***About:*** Muestra una nueva ventana, un mensaje de About del Sistema ***WBNMS***. Llama al archivo ***/htdocs/About.html***.
- ***Salir:*** Cierra la pantalla principal.

6.4.5 Área de Operación

Pertenecen a esta área todos los archivos de Interfaz a los que el usuario Operador no tenga restringido el acceso. Entre ellos se encuentran todos los cargados en la pantalla principal. Las pantallas de Operación propiamente dichas, son aquellas cargadas desde los archivos referenciados desde el ***Menú Principal*** como ***Monitoreo***, ***Lista de Eventos*** y ***Acciones***.

6.4.5.1 Monitoreo

Esta opción llama al archivo ***/htdocs/Monitor.html*** (Ver Anexo Código-Monitor.html), el cual carga ***cgi-bin/MonitorMain.cgi*** en el ***Marco Principal***

(mainFrame) y ***cgi-bin/MonitorMenu.cgi*** en el ***Marco de Menú Secundario*** (leftFrame). En la siguiente figura se ejemplifica la pantalla desplegada.

Figura 6-15 Pantalla de Monitoreo

6.4.5.1.1 MonitorMain.cgi

Es archivo CGI muestra un mapa, si es que esta definido, de lo contrario muestra un mensaje requiriendo que el usuario seleccione uno del menú de la izquierda (Ver Código - MonitorMain.cgi) .

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***MonitorManager*** para dar formato al Mapa.
  - MapParse
  - Mapsize
  - Counter
- ***MapManager*** para obtener los datos del Mapa.
  - ReadMaps

En la siguiente figura se detallan los distintos componentes de la vista generada.

Figura 6-16 Vista de Mapa

- Al hacer clic sobre un icono de mapa, se abre una ventana secundaria, que despliega el mapa elegido (maximizado).Un Ejemplo de esta acción es mostrado en la Figura 6-17.

Figura 6-17 Ventana Pop-Up sobre un submapa

Esta segunda ventana de navegación es también creada por el archivo CGI ***MonitorMain.cgi*** y posee las mismas propiedades de navegación que la ventana raíz.

- Al hacer clic sobre un icono de equipo, se abre una ventana secundaria, que despliega la información del equipo y un menú de opciones. Un Ejemplo de esta acción es mostrado en la Figura 6-18.

Figura 6-18 Ventana Pop-Up sobre una equipo

Esta ventana de Propiedades es desplegada por el archivo CGI ***MonitorEquipmentView.cgi*** (Ver Código - MonitorEquipmentView.cgi)

6.4.5.1.1.1 MonitorEquipmentView.cgi

Muestra los datos de un equipo seleccionado y brinda acceso a las opciones de Ver Lista de Eventos, Ver Alertas y Tomar Acciones (Perfiles de Acción). Las dos primeras opciones llaman al mismo archivo CGI, aunque con distintos parámetros, ***LogAlarmViewer.cgi***. La última opción llama al archivo CGI ***GetSetViewer.cgi***.
La siguiente figura muestra un ejemplo de esta ventana:

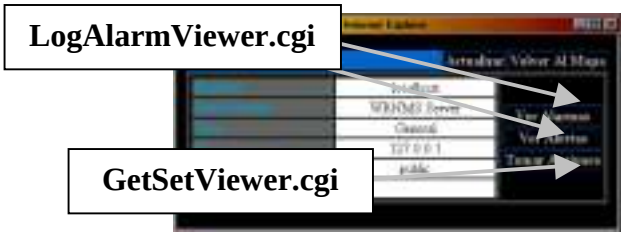


Figura 6-19 Menu de opciones de equipo

Al hacer clic en alguna de estas opciones, la ventana de propiedades se cierra y se abre una nueva ventana desplegando la opción elegida. Un ejemplo de estas acciones es ejemplificada en la siguiente figura.

Figura 6-20 Selección de Opciones de Equipo

Estos archivos CGI son descriptos mas adelante en la implementación, ya que son compartidos por otras opciones de Menú.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***MonitorManager*** para dar formato al Mapa.
  - Mapsize
- ***EquipmentManager*** para obtener los datos del Equipo.
  - ReadEquipments

6.4.5.1.2 MonitorMenu.cgi

Este archivo provee la posibilidad al usuario seleccionar un mapa, de entre los configurados en el sistema, para ser mostrado como raíz por le archivo ***MonitorMain.cgi***. Además este archivo muestra y actualiza cada ***$WbnmsConf::CONFDRAWER{'checkinterval'}*** segundos , el estado del sistema y un contador de las alarmas del sistema (Ver Código - MonitorMenu.cgi).

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***MapManager*** para obtener los datos del Mapa.
  - ReadMaps

En la siguiente figura se detallan los distintos componentes de la vista generada.

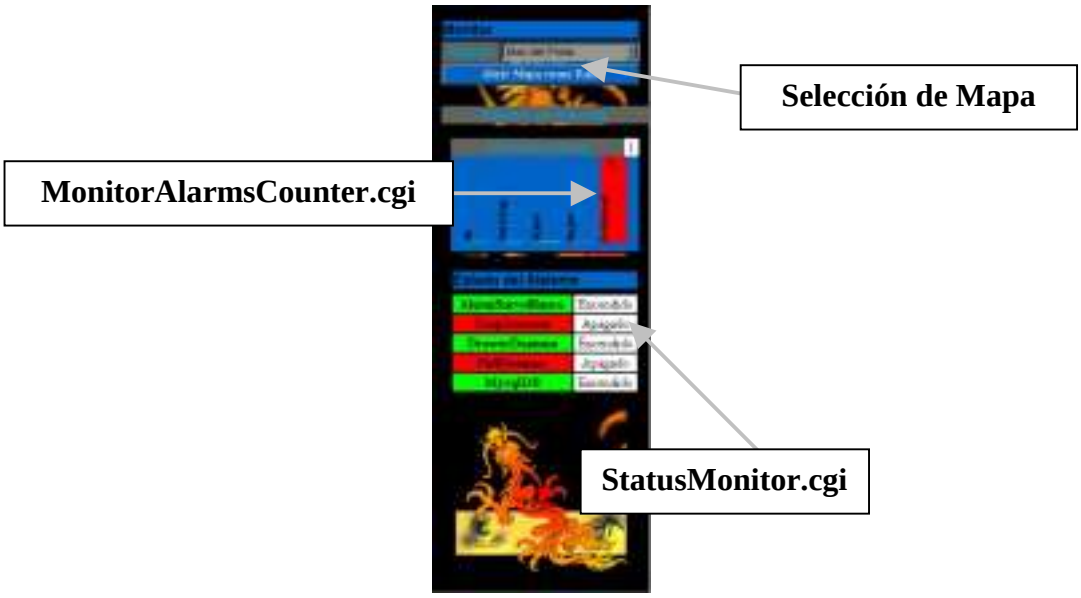


Figura 6-21 Pantalla de Menú de Monitoreo

A su vez, este archivo CGI llama a otros dos, para las funciones de contador de alarmas y verificación de estado.

6.4.5.1.2.1 MonitorAlarmsCounter.cgi

Se encuentra en el directorio /cgi-bin/ y cumple a función de contador de alarmas (Ver Código - MonitorAlarmsCounter.cgi). Un ejemplo de la pantalla generada es la mostrada por la siguiente figura.

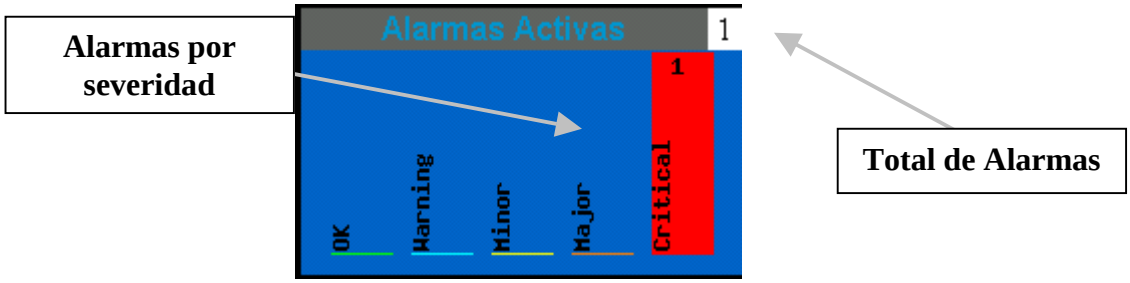


Figura 6-22 Monitor de Alarmas

6.4.5.1.2.2 StatusMonitor.cgi

Se encuentra en el directorio /cgi-bin/ y cumple a función de monitorear el estado del sistema (Ver Código - StatusMonitor.cgi). Un ejemplo de la pantalla generada es la mostrada por la siguiente figura.

Figura 6-23 Monitor de Estado del Sistema

6.4.5.2 Lista de Eventos

Esta opción llama al archivo ***/htdocs/LogAlarms.html*** (Ver Código - LogAlarms.html) que a su vez abre en el marco principal el archivo CGI ***/cgi-bin/LogAlarmViewer.cgi*** (Ver Código - LogAlarmViewer.cgi) y en el Marco de Menú Secundario el archivo CGI ***/cgi-bin/LogAlarmMenu.cgi*** (Ver Código - LogAlarmMenu.cgi) . La siguiente figura muestra la pantalla generada.

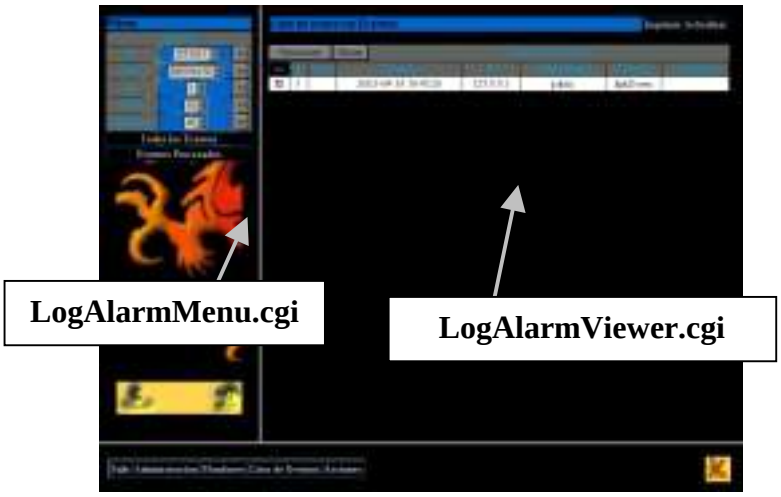


Figura 6-24 Pantalla de Lista de Eventos

6.4.5.2.1 LogAlarmViewer.cgi

Provee, según los parámetros con los que sea llamado, la posibilidad de mostrar las lista de eventos registrados por el sistema. Incluyendo eventos no procesados y procesados como reportes. De esta forma existen dos posibles pantallas mostradas en la Figura 6-25.

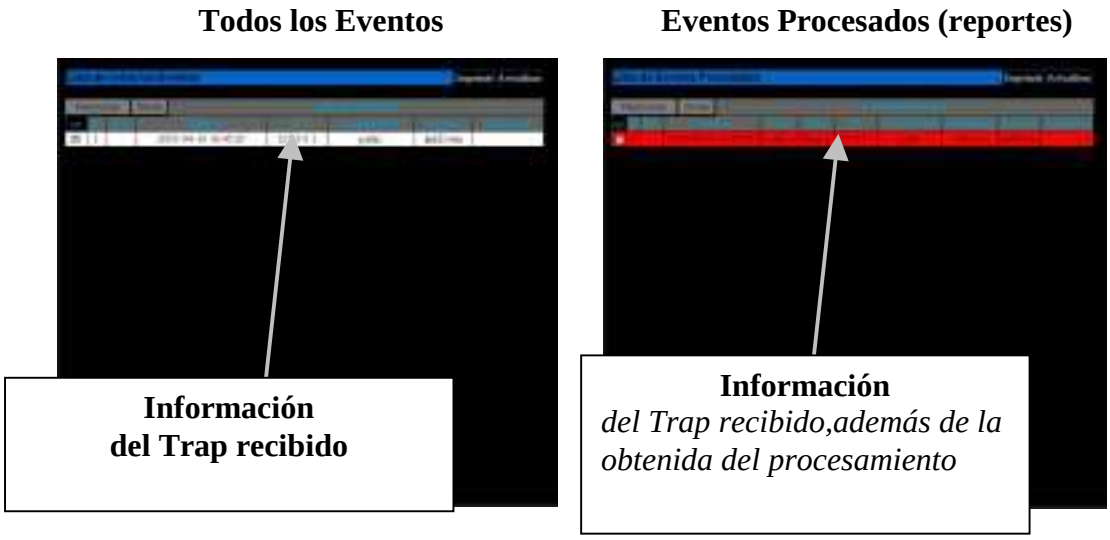


Figura 6-25 Tipos de pantalla de Log

En el primer caso, en que son mostrados los eventos sin procesar, el archivo CGI llama a la función ***ReadAllLog***, implementada por la clase ***LogManager***, para obtener dicha información. En el segunda caso, en cambio, llama a la función ***ReadProcessedLog***, implementada por la misma clase. Este segundo caso es también el utilizado para la opción de ***Ver Alarmas*** (Monitoreo). En este caso es filtrada por equipo.

Las Opciones que provee esta pantalla especificadas en la Figura 6-26 son las de ***Reconocer*** (***ACK***) y ***Borrar*** Eventos. La primera hace que el evento no este presente para el cálculo de estado del equipo al que pertenece pero sin eliminarlo del sistema, destacando quien, usuario o sistema, realizó el ***ACK***. La segunda opción, elimina el evento del sistema.

Existe la posibilidad de retroceder etapas de filtrado, pulsando el botón **"<<"** (Retroceder). Además de la opción de seleccionar todos los eventos.

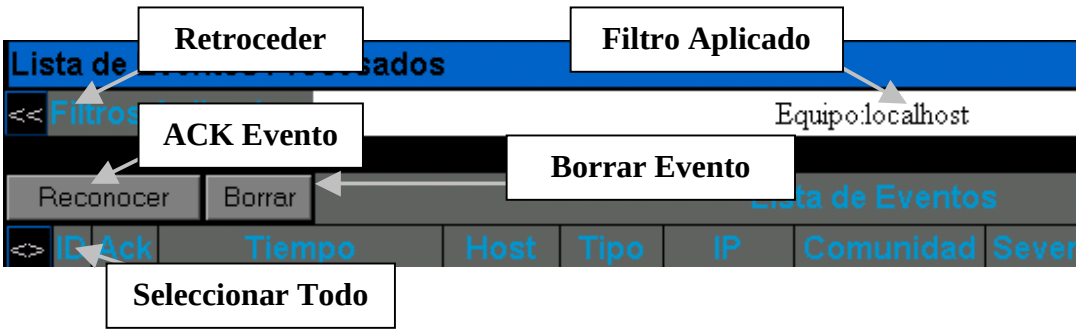


Figura 6-27 Detalle del menú de la pantalla de Log para eventos procesados

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.

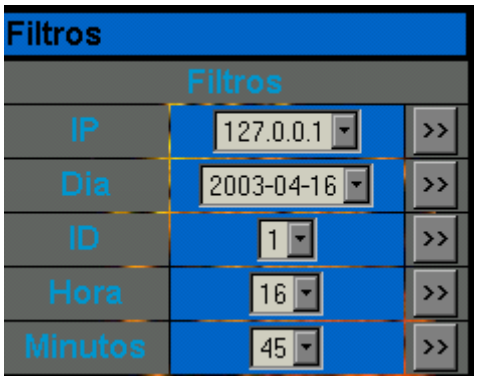
- ***LogManager*** para obtener los datos del Mapa.
  - AckEvent
  - AckAlarm
  - DelEvent
  - ReadProcessedLogWUsers
  - ReadProcessedLog
  - ReadAllLog
  - FilterLog
  - ProcessedAlarmStatus

6.4.5.2.2 LogAlarmMenu.cgi

Este archivo CGI provee las opciones de filtrado de Eventos. Incluyendo las opciones de visualización del ***Marco Principal*** de ***Todos los Eventos*** y ***Eventos Procesados***. Los resultados de las distintas solicitudes de filtrado son mostrados en el ***Marco Principal*** llamando al archivo ***LogAlarmViewer.cgi***.

Las Opciones de Filtrado son distintas, según el tipo de eventos mostrados procesados o no. En la siguiente figura se muestra un ejemplo de cada una de estas pantallas:

Todos los Eventos



Eventos Procesados

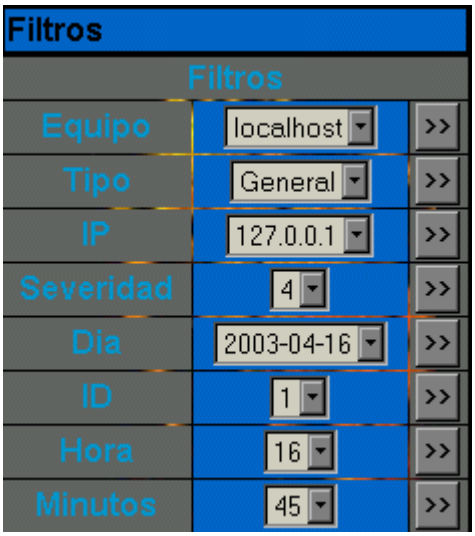


Figura 6-28 Pantallas posibles de menú de filtrado de eventos

La opción de Eventos Procesados, agrega la selección por ***Equipo***, ***Tipo*** y ***Severidad***.
Las opciones de filtrado, son acumulativas, esto es, que se superponen a medida que el usuario las selecciona.

6.4.5.3 Acciones

Esta opción llama al archivo ***/htdocs/Actions.html*** (Ver Código -) que a su vez abre en el marco principal el archivo CGI ***/cgi-bin/GetSetViewer.cgi*** (Ver Código - GetSetViewer.cgi) y en el ***Marco de Menú Secundario*** el archivo CGI ***/cgi-bin/ActionMenu.cgi*** (Ver Código - ActionMenu.cgi). De esta forma, la pantalla generada es la mostrada en la siguiente figura:

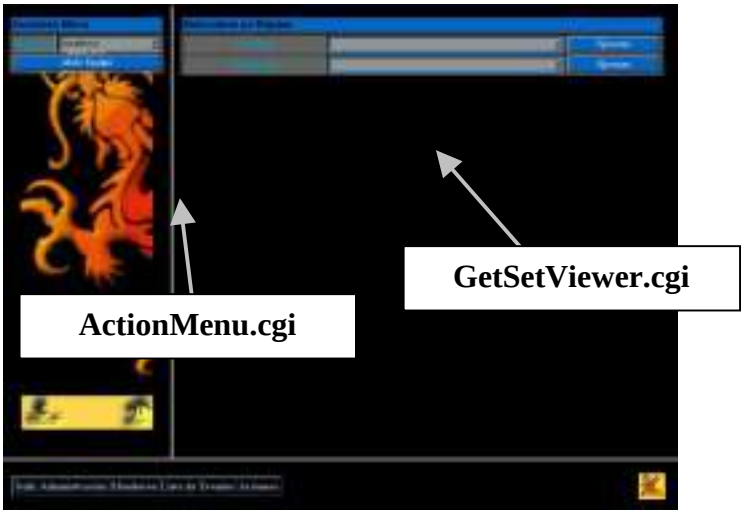


Figura 6-29 Pantalla de Acciones

6.4.5.3.1 ActionMenu.cgi

Este archivo CGI construye el menú de selección de Equipos para la ejecución de Perfiles de Acción. La selección de un Equipo es requerido desde la pantalla desplegada en el ***Marco Principal*** (***GetSetViewer.cgi***) desde la Figura 6-30. Al seleccionarlo, es abierto en el ***Marco Principal*** las opciones para los ***Perfiles de Acción*** asociados al tipo del equipo.

Figura 6-30 Menú de selección de perfiles de acción por equipo

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***GetSetManager*** tomar datos de configuración de los perfiles asociados a los tipos de equipos.
    - ReadEquipments

6.4.5.3.2 GetSetViewer.cgi

Al seleccionar un equipo en el menú (Figura 6-31) , se despliegan en el ***Marco Principal*** las opciones de Perfiles de Acción asociadas con el tipo de este.

Figura 6-31 Acción de selección de equipo

Este Archivo también es llamado desde el menú ***Tomar Acciones*** del archivo CGI ***MonitorEquipmentViewe.cgi***, asociándolo a un equipo seleccionado del Mapa.

Los Perfiles son de dos tipos diferentes, Lectura (Get) y Escritura (Get y Set). Al ejecutar un perfil seleccionado, el sistema realiza la acción a través del módulo ***GetSetManager*** y presenta en pantalla la respuesta del equipo consultado.

- En el caso de un perfil de ***escritura***, el sistema primero realiza un Get para obtener los valores actuales de los objetos del perfil. Luego, el usuario requiere o no un Set con nuevos valores. Un ejemplo de ello es mostrado en la siguiente figura.

Figura 6-32 Menú desplegado para un perfil de escritura

- En el caso de un perfil de lectura, es realizado por el sistema un Get de los objetos del perfil y presentados sus valores en pantalla. Un ejemplo de ello es mostrado en la siguiente figura.

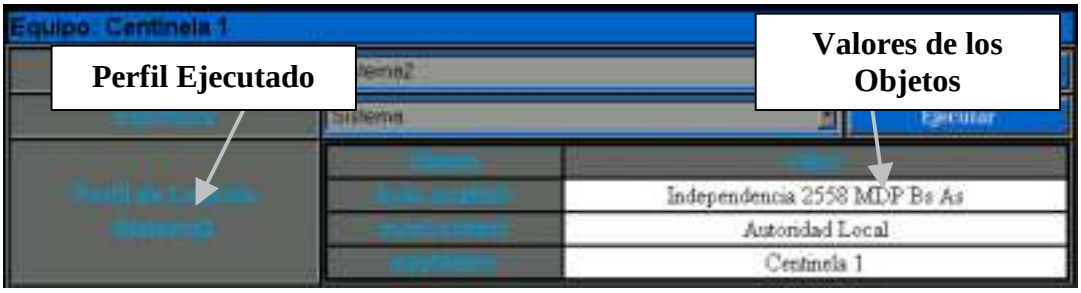


Figura 6-33 Menú desplegado para un perfil de lectura

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***GetSetManager*** para realizar las funciones de Get y Set sobre los equipos gestionados y para dar formato a los datos.
  - ReadEquipmentData
  - ReadProf
  - Get
  - Set

6.4.6 Área de Administración

Pertenecen a esta área, todos los archivos de Interfaz clasificados como de Administración, es decir dentro del directorio protegido webadm. El área de administración, archivo ***/webadm/htdocs/Administration.html*** (Ver Código - Administration.html), es desplegada en la pantalla al seleccionar la opción ***Administración*** en el ***Menú Principal.*** De esta forma la pantalla generada resulta como la mostrada en la siguiente figura.

Figura 6-34 Pantalla de Administración

Al seleccionar esta opción del Menú principal, es desplegada en el ***Marco de Menú Secundario*** el archivo ***/webadm/htdocs/menuAdmin.html*** (Ver Código - menuAdmin.html) que presenta el menú de Administración. Este Menú se divide en siete categorías:

1 ***Sistema:*** Agrupa las opciones de administración del sistema ***WBNMS***.
   - ***Usuarios del sistema:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/UsersAdmin.cgi***.
   - ***Estado del Sistema:*** despliega en el ***Marco Principal*** el archivo ***/webadm/htdocs/Status.html***.

2 ***Equipos:*** Agrupa las opciones de administración de Equipos Gestionados.
   - ***Tipos de Equipos:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/EquipmentTypesAdmin.cgi***.
   - ***Encuesta Automática:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/PollAdmin.cgi***.
   - ***Equipos:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/EquipmentsAdmin.cgi***.

3 ***Mapas:*** Contiene la opción de administración de Mapas.
   - ***Mapas:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/MapsAdmin.cgi***.

4 ***Alertas:*** Agrupa las opciones de administración de Alertas.
   - ***Asignación de Alertas:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/AssignationsActionsAdmin.cgi***.
   - ***Usuarios de Alertas:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/UsersActionsAdmin.cgi***.

5 ***Alarmas:*** Contiene la opción de administración de Alarmas (Reportes).
   - ***Alarmas:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/AlarmTriggersAdmin.cgi***.

6 ***Perfiles de Acción:*** Contiene la opción de administración de Perfiles de Acción.
   - ***Perfiles de Acción:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin/ProfilesAdmin.cgi***.

7 ***MIBs:*** Agrupa las opciones de administración de MIBs.
   - ***MIBs:*** despliega en el ***Marco Principal*** el archivo ***/webadm/cgi-bin /MIBsAdmin.cgi***.
   - ***Navegador de MIBs:*** despliega en el ***Marco Principal*** el archivo ***/cgi-bin/MIBBrowserFrameset.cgi***.

6.4.6.1 Sistema

6.4.6.1.1 UsersAdmin.cgi

Este archivo despliega la pantalla de administración de ***Usuarios del Sistema*** en el ***Marco Principal*** y permite crear y borrar los usuarios (Ver Código - UsersAdmin.cgi). En la siguiente figura se muestra la pantalla generada.

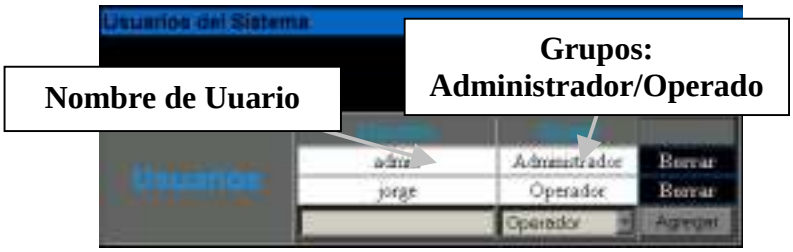


Figura 6-35 Pantalla de Administración de Usuarios

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***Auth*** para el ABM de los datos de Usuarios.
   - readGroupsandUsers
   - AddUsers
   - AddGroupUsers
   - DelUser
   - DelGroupUsers

Al Agregar un Nuevo usuario, este archivo CGI despliega una ventana emergente que pide sea ingresado el password que este va a tener. Este ventana es manejada por el archivo CGI ***/webadm/Pass.cgi*** (Ver Código - Pass.cgi) y generada resulta como la de la siguiente figura.

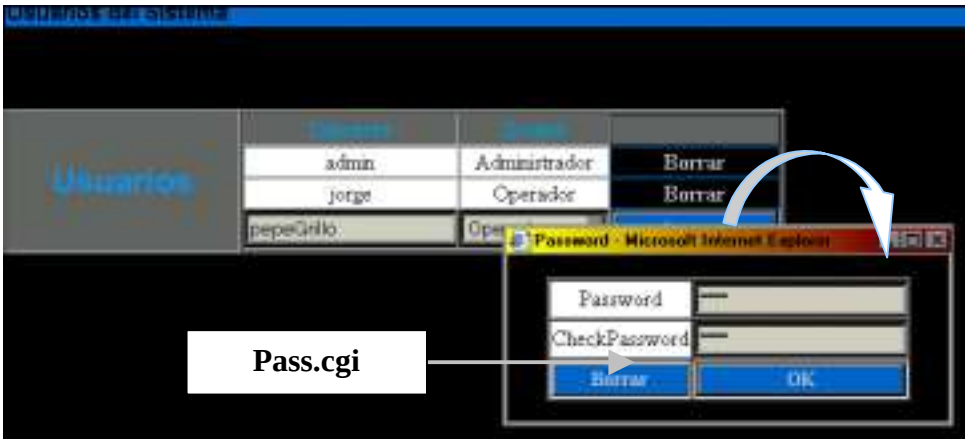


Figura 6-36 Pantalla configuración del Password de nuevo usuario

Este archivo CGI da formato a la pantalla de password para un nuevo usuario y devuelve el formulario al archivo ***UsersAdmin.cgi*** donde la información del password es procesada.

Es requerido que el administrador ingrese dos veces el password del nuevo usuario para que la información sea validada. Luego la información de usuario y grupo es almacenada en el formato requerido por el servidor ***Web Apache*** por la clase ***Auth***.

6.4.6.1.2 Status.html

Este archivo html divide el ***Marco Principal*** en dos sub Marcos, superior e inferior (Ver Código - Status.html). Llamando en el Marco superior al archivo CGI ***/webadm/cgi-bin/Status.cgi*** y en el inferior al archivo ***/webadm/htdocs/mainAdmin.html***. La pantalla generada resulta como la de la siguiente figura.

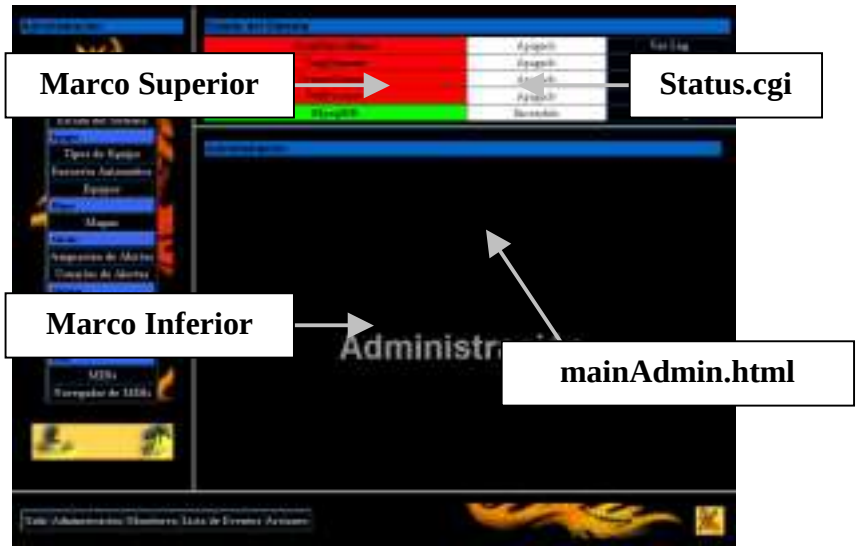


Figura 6-37 Pantalla de Administración de estado del sistema

6.4.6.1.2.1 Status.cgi

Este Archivo consulta y muestra en pantalla el estado del sistema ***WBNMS*** (Ver Código - Status.cgi). Discriminando por componentes del mismo (demonios y DB). Para la tarea de consulta de estado se vale de la clase ***StatusChecker***.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***StatusChecker*** para verificar el estado de los distintos componentes del sistema ***WBNMS***.
    - CheckStatus

A continuación, en la figura 6-38, se ejemplifica la pantalla generada con los distintos estados de los componentes del sistema

Figura 6-38 Pantalla de estados de componentes del sistema

En esta pantalla se indica con colores, rojo o verde, además de la leyenda correspondiente el estado de cada uno de los componentes del sistema. Además es posible , si es que el sistema los provee, ver los logs asociados a cada componente.
En esta primera versión solo es posible ver logs de ***Alarmsurveillance*** y ***TrapDaemon***.

Al elegir la opción ***Ver Log*** para cada componente, se despliega en el Marco Inferior una pantalla generada por el archivo ***/webadm/cgi-bin/LogDaemon.cgi***.
Este archivo también provee la posibilidad de administración de los distintos archivos de log asociados a los componentes del sistema ***WBNMS***. La siguiente figura muestra dicho proceso.

Figura 6-39 Pantalla de Administración de Logs

6.4.6.1.2.2 LogDaemon.cgi

Este archivo CGI es el encargado de mostrar y administrar los archivos de log del sistema ***WBNMS*** (Ver Código - LogDaemon.cgi). Para ello recurre a la clase ***DaemonsLogViewer***. Especificamente las funciones :

- DelLog
- ReadLogs

De esta forma en cada pantalla existe la posibilidad de elegir el log a visualizar y una vez abierto es posible borrarlo. Un ejemplo de este proceso es mostrado en la siguiente figura.

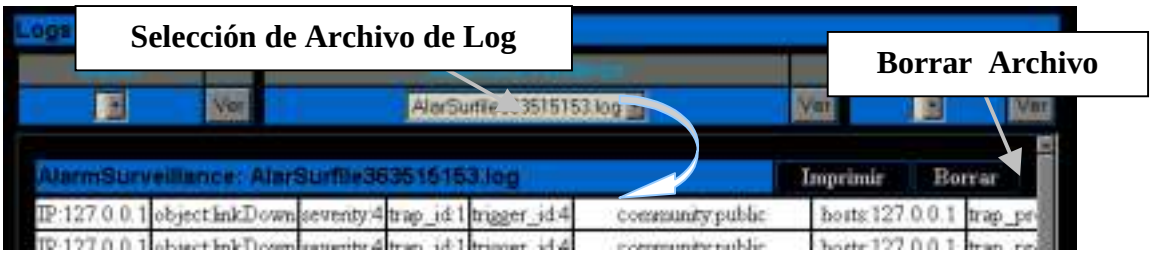


Figura 6-40 Selección y borrado de Log

La visualización del Archivo de log una vez seleccionado es realizada por el archivo CGI ***LogViewer.cgi*** (Ver Código - LogViewer.cgi). este último también utiliza la clase ***DaemonsLogViewer*** y la función ***OpenLog***.

6.4.6.2 Equipo

6.4.6.2.1 EquipmentTypesAdmin.cgi

Este archivo CGI muestra la pantalla principal de administración de los Tipos de Equipos del Sistema ***WBNMS*** (Ver Código - EquipmentTypesAdmin.cgi). Es seleccionado desde el menú de administración y desplegado en el ***Marco Principal*** (acción ejemplificada en la siguiente figura).

Figura 6-41 Selección de la pantalla de administración de los tipos de equipo

Desde esta pantalla es posible crear, modificar y borrar Tipos de Equipos.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***EquipmentManager*** para el ABM de los Tipos de Equipos y la lectura los datos de los equipos configurados.
  - ModType
  - AddType
  - DelType
  - ReadEquipments
- ***BmpManager*** para la visualización de los archivos gráficos asociados a cada tipo de equipo.
  - Seebmp

Al crear o modificar un tipo de equipo, es requerido seleccionar un archivo gráfico que lo represente (icono). Esta selección se hace desde un menú desplegable o en forma gráfica (pre visualización) con la ayuda del archivo CGI ***/webadm/cgi-bin/Seebmp.cgi*** (Ver Código - Seebmp.cgi). Estas última opción es activada en una ventana emergente al seleccionar el botón ***Tipo***.
Cuando es seleccionado desde este nueva pantalla un icono, el nombre del archivo gráfico correspondiente seleccionado automáticamente del menú desplegable de la pantalla de administración de tipos.

Es posible, subir al servidor un nuevo archivo gráfico desde la pantalla desplegada por ***Seebmp.cgi*** (Figura 6-42), con la opción ***Agregar***. Este deberá ser del tipo GIF.

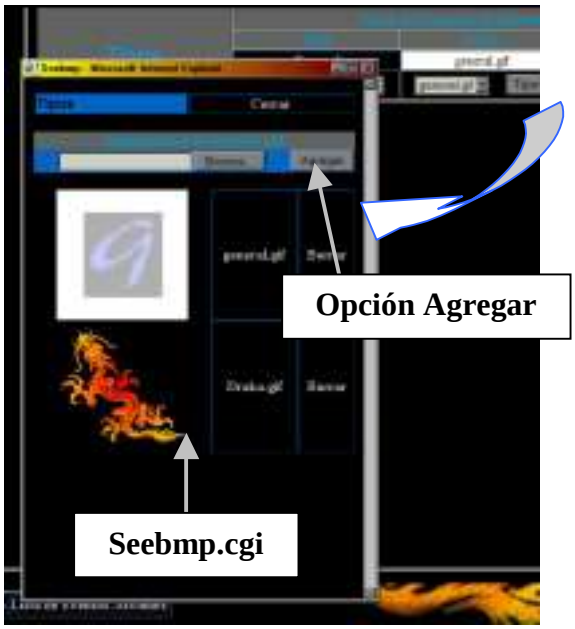


Figura 6-42 Pantalla de administración de archivo Gráfico

También es posible, borrar los archivos GIF existentes en el servidor. Estos archivos se encuentran en el directorio ***/Wbnms_Var/File_DB/BMP/Equipments***.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***BmpManager*** para la visualización y la administración los archivos gráficos.
  - Seebmp
  - UpLoad
  - DelBmp

Al hacer clic sobre el nombre de un Tipo ya creado se abre una nueva ventana para asociarlo a ***Perfiles de Acción*** configurados. Esta pantalla es manejada por el archivo CGI ***/webadm/cgi-bin/TypesProfiles.cgi*** (Ver Código - TypesProfiles.cgi).

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***GetSetManager*** para administrar y visualizar las asociaciones de ***Tipos de Equipos*** con ***Perfiles de Acción***.
  - DelProf
  - AddProf
  - ReadProf
  - ReadAllProfiles

Desde esta pantalla (Figura 6-43), el administrador asocia perfiles de acción al tipo al ***Agregar*** desde la lista de restantes (derecha) a la lista de incluidos (izquierda).

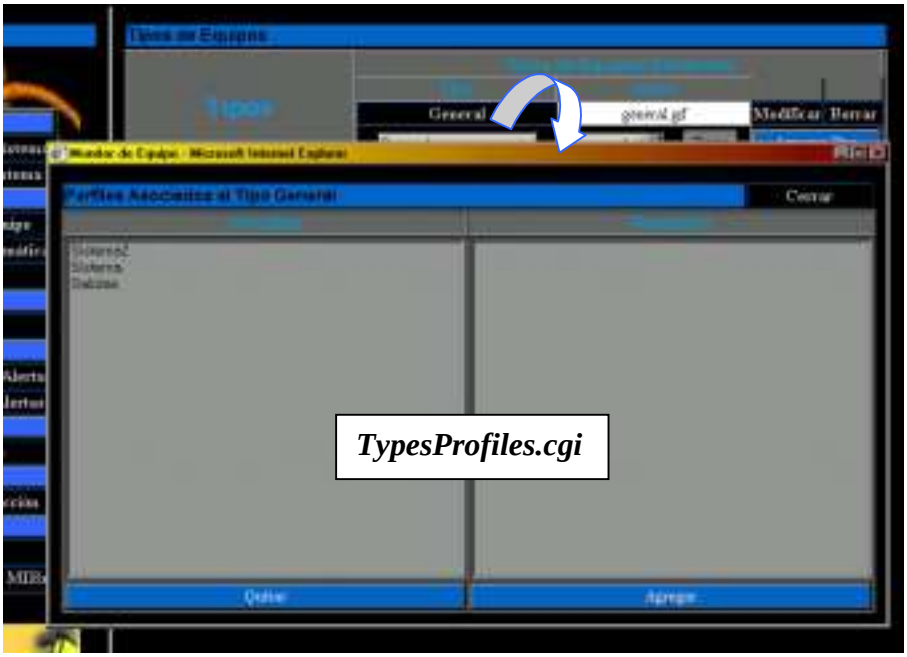


Figura 6-43 Pantalla de asociación de perfiles

6.4.6.2.2 PollAdmin.cgi

Este Archivo CGI muestra la pantalla de administración de Equipos encontrados durante un proceso de encuesta automática (demonio ***keepalive***)(Ver Código - PollAdmin.cgi).La siguiente figura ejemplifica la selección desde el menú principal.

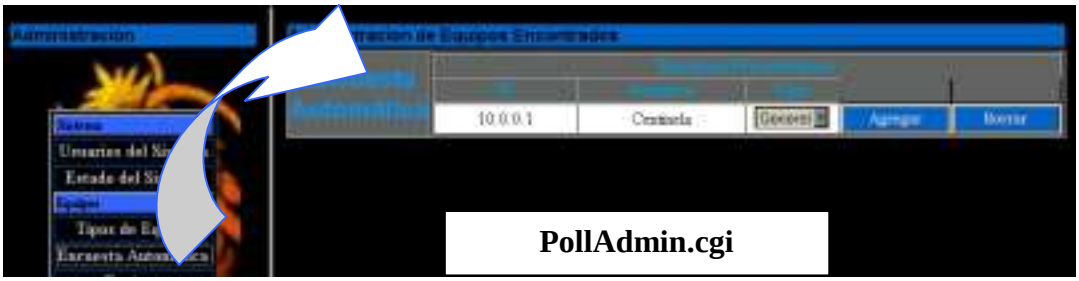


Figura 6-44 Selección de Encuesta Automatica

Desde esta pantalla, el administrador puede agregar el equipo encontrado como equipo gestionado, ***Agregar***, o descartarlo, ***Borrar***.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.

- ***PollManager*** para administrar y visualizar los equipos encontrados.
  - AddEquipment
  - DelEquipment
  - ReadEquipments

Al agregar un equipo a la gestión, este aparece ahora en la pantalla de ***Equipos***.

6.4.6.2.3 EquipmentsAdmin.cgi

Este archivo CGI muestra la pantalla (Figura 6-45) de administración de los Equipos gestionados por el Sistema (Ver Código - EquipmentsAdmin.cgi). Es posible desde ella, agregar, modificar y borrar Datos de Equipos.

Figura 6-45 Pantalla de Administración de Equipos

Los datos requeridos en dicha pantalla para la administración son:

- ***Nombre*** asociado al Equipo. Generalmente coincidente con ***sysName***.
- ***Tipo de equipo***. Este asociación definirá los perfiles de acción y el icono que lo represente.
- ***IP***. Deberá ser única, de forma de definir unívocamente al equipo en la red gestionada.
- ***Comunidad***. Definirá la comunidad SNMP a la que responderá el Equipo.
- ***Descripción***. Comentario referente al Equipo (Opcional).
- ***MIB***. Asociada al Equipo (Opcional).

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***EquipmentManager*** para administrar y visualizar los equipos gestionados.
  - DelEquipment
  - AddEquipment
  - ModEquipment
  - ReadEquipments
- ***MIBManager*** para buscar las MIBs cargadas en el sistema.
  - ReadMIBs

6.4.6.3 Mapas

6.4.6.3.1 MapsAdmin.cgi

Este Archivo CGI muestra la pantalla (Figura 6-46) de administración de Mapas del sistema ***WBNMS*** (Ver Código - MapsAdmin.cgi). En ella se pueden ver los datos de los mapas configurados así como crear nuevos o borrar y modificar los existentes.

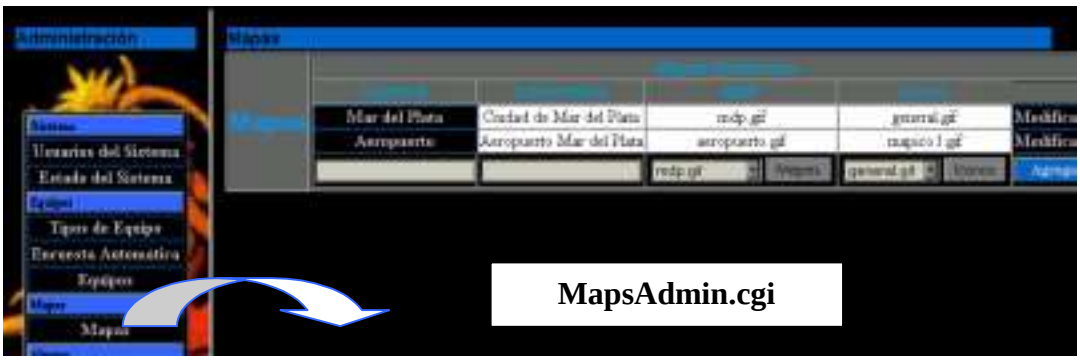


Figura 6-46 Pantalla de Administración de Mapas

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***MapManager*** para administrar y visualizar los mapas del sistema.
  - DelMap
  - AddMap
  - ModMap
  - ReadMaps
- ***BmpManager*** leer los archivos gráficos configurados como íconos y como mapas.
  - Seebmp

Para la configuración de un Mapa, son requeridos los siguientes datos:

- ***Nombre*** del Mapa.
- ***Gráfico***. Archivo gráfico(GIF) que representa el Mapa.
- ***Icono***. Archivo gráfico(GIF) que representa al icono del mapa.
- ***Comentario***. comentario asociado al Mapa (Opcional).

La elección del ***Gráfico*** y del ***Icono*** son realizados en forma de texto o gráfica (como en el caso de los Tipos de Equipos). Para el formato gráfico se utiliza el archivo CGI ***Seebmp.cgi***. La siguiente figura ejemplifica la selección gráfica.

Figura 6-47 Pantalla de Selección Gráfica utilizando Seebmp.cgi

Al seleccionar un mapa ya configurado, se despliega una nueva ventana, en la que se abre la pantalla de edición del Mapa seleccionado. Esta pantalla es generada por el archivo CGI ***/webadm/cgi-bin/AddView.cgi*** (Figura 6-48).

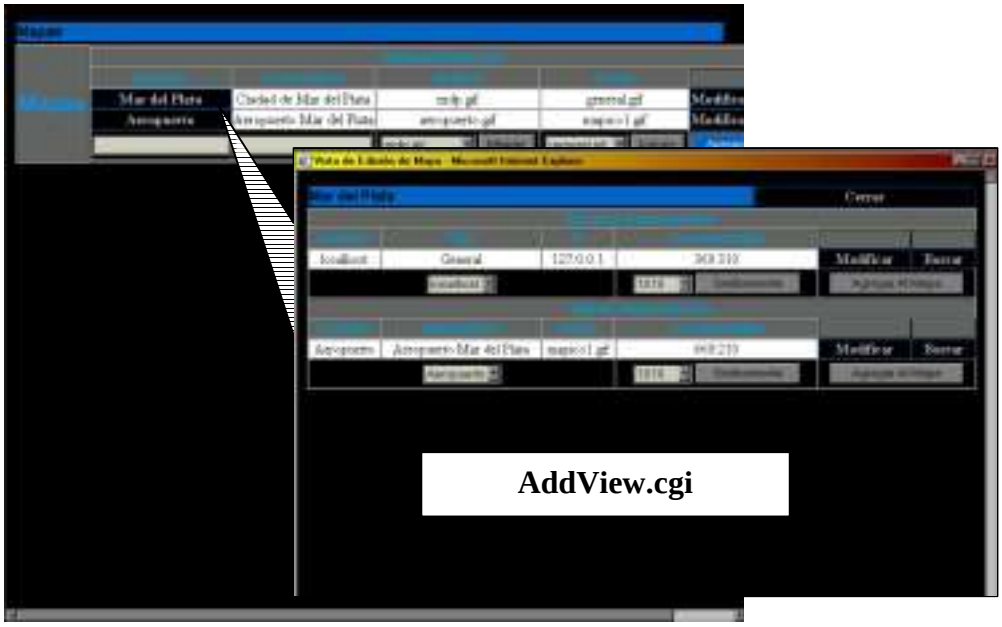


Figura 6-48 Pantalla de Edición de Mapa

6.4.6.3.1.1 AddView.cgi

Este archivo CGI muestra la pantalla de edición del Mapa seleccionado (Ver Código - AddView.cgi). Desde ella es posible administrar los equipos y sub-mapas dentro de él.
Al agregar o modificar sub-mapa o equipo, el administrador deberá asignarle una posición relativa dentro del mapa seleccionado. Esta acción podrá ser llevada a cabo mediante la selección de las coordenadas (x:y) del menú desplegable o

gráficamente con la opción ***Gráficamente***. La siguiente figura ejemplifica la acción de posicionamiento gráfico de coordenadas relativas.

Figura 6-49 Pantalla de selección gráfica de coordenadas

Al seleccionar la opción gráfica, se abre una nueva ventana en donde el archivo CGI ***/webadm/cgi-bin/GraficalCoor.cgi*** carga el mapa seleccionado con un cuadriculado de coordenadas en el (Ver Código - GraficalCoor.cgi). Este cuadriculado define las coordenadas del icono del equipo o sub-mapa, ya que al ser seleccionada una cuadrícula específica se actualiza en la pantalla ***AddView.cgi*** las coordenadas del menú desplegable.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***MapManager*** para administrar las asociaciones entre los mapas y los sub-mapas contenidos en ellos.
  - ReadMapsMaps
  - ReadMaps
- ***EquipmentManager*** para administrar las asociaciones entre los mapas y los equipos contenidos en ellos.
  - ReadMapsEquipments
  - ReadEquipments
- ***CoorManager*** para la administración y visualización de las coordenadas de los mapas.
  - AddEquipment
  - AddMap
  - DelEquipment
  - DelMap
  - CoorCalculate
  - FormatView

6.4.6.4 Alertas

6.4.6.4.1 AssignationsActionsAdmin.cgi

Este archivo CGI muestra la pantalla de asignación de ***Usuarios de Alertas*** a uno de los tres tipos definidos (e-mail,SMS,SNMP)(Ver Código - AssignationsActionsAdmin.cgi). Esta pantalla permite al administrador asignar los tipos de alerta según criterios de horario (turnos), por equipo o por mapa (zonas).La siguiente figura muestra un ejemplo de esta pantalla.

Figura 6-50 Pantalla de asignación de Alertas

Los datos necesarios para definir una Asignación de Alerta son:

- ***Usuario de Alerta***. Definido con anterioridad.
- ***Reporte***. Tipo de reporte, según los módulos de alerta definidos.
- ***Equipo***. Si es que se especifica la Alerta solo de un Equipo en particular.
- ***Mapa***. Definida una zona de incumbencia en un mapa del sistema.
- ***Inicio de Turno***. Comienzo del turno del usuario en la Asignación.
- ***Final del Turno***. Finalización del turno del usuario en la Asignación.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***ActionGuiManager*** para administrar y visualizar las asignaciones de Alertas.
  - DelAsignationAction
  - AddAsignationAction
  - ModAsignationAction
  - ReadActionAssignations

6.4.6.4.2 UsersActionsAdmin.cgi

Este archivo CGI muestra la pantalla de administración de ***Usuarios de Alertas*** (Ver Código - UsersActionsAdmin.cgi). Desde esta pantalla el administrador puede crear, modificar y borrar los datos de los usuarios de alertas. El siguiente gráfico muestra dicha pantalla.

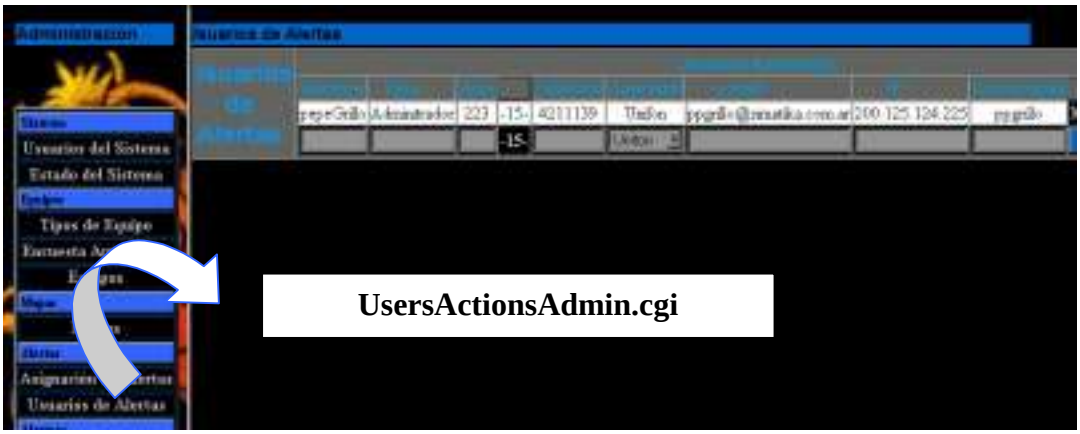


Figura 6-51 Administración de Usuarios de Alerta

Los datos necesarios para definir un ***Usuario de Alerta*** son:

- ***Nombre***. Del Usuario de Alerta.
- ***Tipo.*** Tipo de Usuario(Opcional).
- ***área***. área definida para el telefono móvil (SMS)
- ***Teléfono***. Número de teléfono móvil del Usuario(SMS).
- ***Operador***. Operador de telefonía móvil del Usuario(SMS).
- ***E-mail***. Dirección de correo electrónico del Usuario(e-mail).
- ***IP***. Dirección IP del Usuario (SNMP).
- ***Comunidad***. Comunidad SNMP del Usuario (SNMP).

Según sea los tipos de asignación que tendrán permitidos los Usuarios serán los datos necesarios para crearlos.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***ActionGuiManager*** para administrar y visualizar los datos de los Usuarios de Alertas.
  - DelActionUser
  - AddActionUser
  - ModActionUser
  - ReadActionUsers

6.4.6.5 Alarmas

6.4.6.5.1 AlarmTriggersAdmin.cgi

Este archivo CGI muestra la pantalla de administración de filtros de Alarmas (Ver Código - AlarmTriggersAdmin.cgi). Desde esta pantalla el administrador puede crear, modificar y borrar los distintos filtros según los tres tipos básicos; a saber ***General***, ***por tipo de Equipo*** y ***por Equipo***. La siguiente figura muestra dicha pantalla.

Figura 6-52 Pantalla General de administración de filtros de alarmas

Los tipos de Alarmas, clasificados de esta forma, permiten el filtrado de los eventos recibidos por el sistema ***WBNMS***.

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***AlarmTriggersManager*** para administrar y visualizar los datos de los Filtros de Alarmas.

    - AddAlarmTriggers
    - ReadAlarmTriggerfromID
    - AddActionTrigger
    - DelAlarmTriggers
    - DelActionTrigger
    - ModAlarmTriggers
    - ModActionTrigger
    - ReadEquipments
    - ReadEquipmentTypes
    - ReadAlarmTriggers
    - ReadActionTriggers
    - ReadSeverity

Se muestra a continuación el caso los filtros del tipo General, siendo análogo para los otros dos casos.

Los datos necesarios para definir un ***Filtro de Alarma***(Reporte) son:

- ***ID***. Identificador de la Alarma. Es completada automáticamente por el sistema.
- ***ID de eliminación.*** Si existe una alarma que elimine la presente, de lo contrario será 0.
- ***Severidad***. Define la gravedad de la alarma en el sistema.
- ***Alerta***. Define o no los tipos de Alertas asociados a la presente Alarma (Opcional).

- ***Objeto***. Objeto SNMP que dispara la Alarma.
- ***Variables***. Definición del filtro por las variables del Objeto SNMP(Opcional). Estas variables son configuradas luego de crear el Filtro de Alarma.

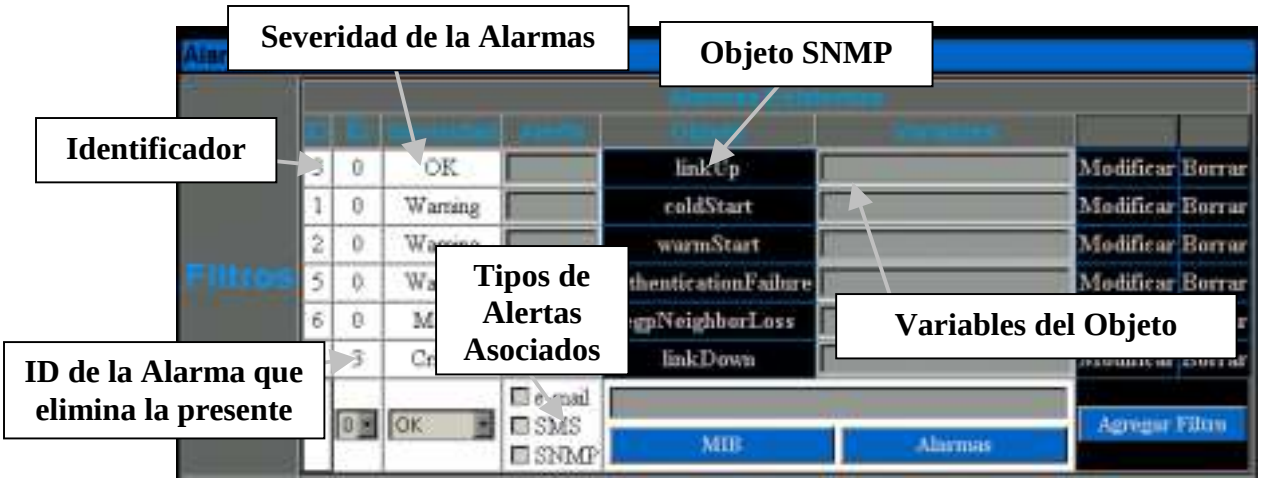


Figura 6-53 Pantalla administración de filtros de alarmas de un tipo (General) en particular

Para la elección del Objeto SNMP asociado al Filtro, la pantalla brinda la posibilidad de utilizar un navegador de MIBs (***MIB***) o un Listado de la Alarmas (***Alarmas***) cargadas en el sistema (TRAP-TYPES). Estas dos opciones son desplegadas en ventanas emergentes.

En el primer caso, es llamado el archivo ***/cgi-bin/MIBBrowserFrameset.cgi*** para desplegar en una nueva ventana la opción de selección de Objetos desde un navegador de MIBs (Figura 6-54). Este archivo CGI crea en esta nueva ventana, tres Marcos en los que llama los siguientes archivos

- ***/MIBbrowserWeb/folderTreeLeftFrameiso.html*** en el Marco Izquierdo.
- ***/MIBbrowserWeb/MIBBrowserMain.html*** en el Marco Central.
- ***/webadm/cgi-bin/MIBBrowserResultsTriggers.cgi*** en el Marco Derecho (Ver Código - MIBBrowserResultsTriggers.cgi).

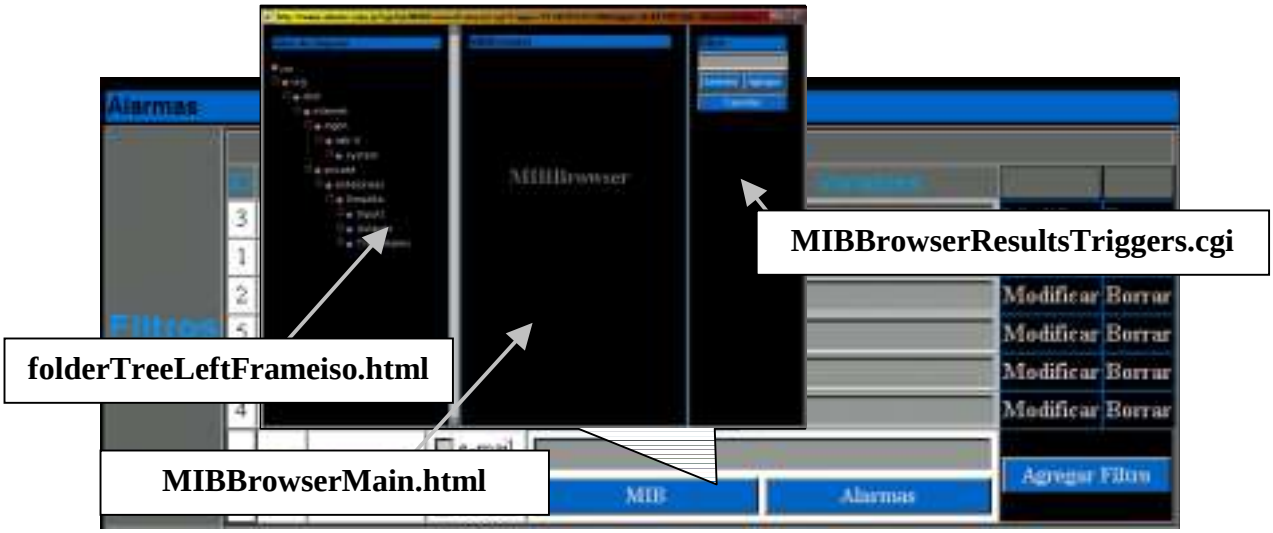


Figura 6-54 Pantalla de selección desde navegador MIB.

La opción de selección ***Alarmas*** abre una ventana en la que es cargado el archivo CGI ***/webadm/cgi-bin/AllAlarmsdisplay.cgi*** (Ver Código - AllAlarmsdisplay.cgi). Este archivo muestra una pantalla listando todas las alarmas (TRAP-TYPE y NOTIFICATION-TYPE) cargadas en la MIB del sistema (Figura 6-55).

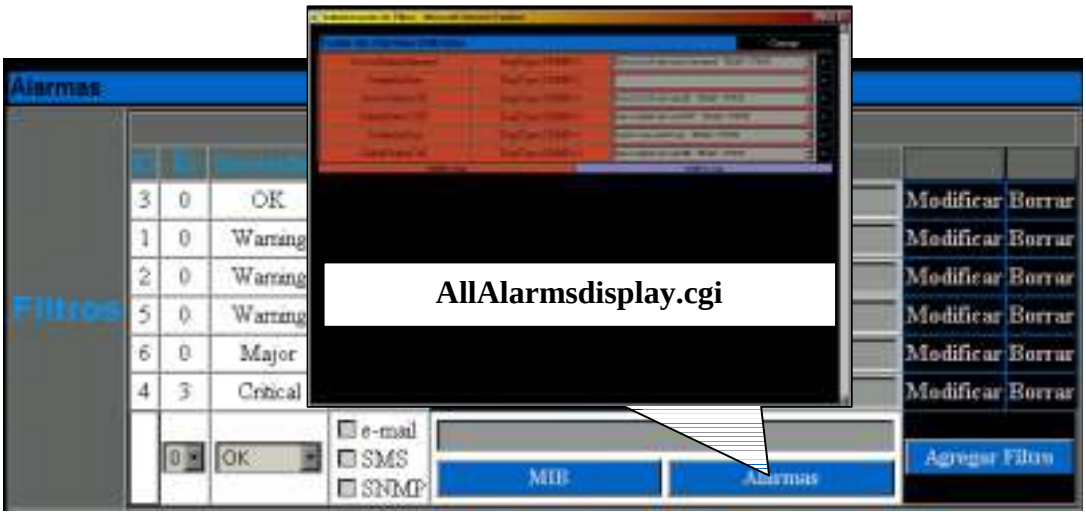


Figura 6-55 Pantalla de selección desde lista de Alarmas.

Tanto en el primer caso como en el segundo, al seleccionar un Objeto SNMP, este es actualizado en el campo correspondiente de la pantalla generada por ***AlarmTriggersAdmin.cgi***.
Una vez que es agregado el Filtro al sistema. Es posible configurar las variables del mismo, si es que es requerido. De lo contrario actuará para todo Objeto de ese tipo sin importar los valores que tomen sus variables. Para configurar las variables, haciendo clic en el nombre del objeto, se llama en una ventana emergente al archivo CGI ***/webadm/cgi-bin/FilterObjects.cgi*** (Ver Código - FilterObjects.cgi). La siguiente figura ejemplifica la acción de configuración de variables.

Figura 6-56 Acción de configuración de variables

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***AlarmTriggersManager*** para administrar y visualizar las variables de los Objetos SNMP.
  - ModAlarmTriggers
  - ReadAlarmTriggers
  - ReadObjects

6.4.6.6 Perfiles de Acción

6.4.6.6.1 ProfilesAdmin.cgi

Este archivo CGI administra y visualiza los perfiles de acción. Desde la pantalla que él despliega (Figura 6-57), el administrador puede crear y borrar dichos perfiles y asociarlos a los objetos SNMP (Ver Código - ProfilesAdmin.cgi).

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***GetSetManager*** para administrar y visualizar los datos de los Perfiles de Acción.
  - UnCreateProf
  - CreateProf
  - ReadAllProfiles

Los datos necesarios para definir un ***Perfil de Acción*** son dos:

- ***Nombre*** del perfil de Acción. deberá definir al Perfil.
- ***Tipo***. Define si es un perfil de ***Solo Lectura*** o de ***Lectura Escritura***.

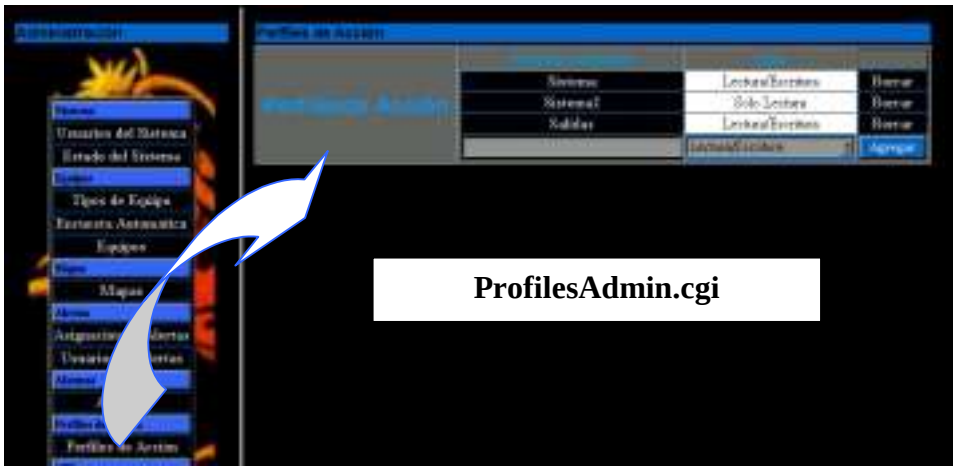


Figura 6-57 Pantalla General de Administración de Perfiles

Para asociar un Objeto SNMP a un Perfil, el administrador debe seleccionar el Perfil y abrir la ventana de asociación (Figura 6-58).

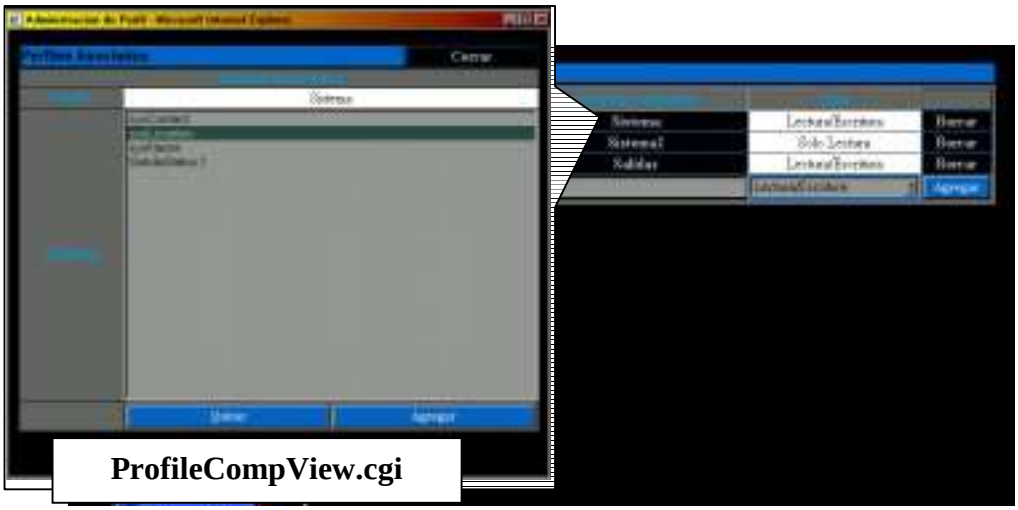


Figura 6-58 Pantalla de administración de Perfil específico (Sistema)

Desde esta pantalla es posible administrar los Objetos asociados al Perfil (Ver Código - ProfileCompView.cgi).
Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***GetSetManager*** para administrar y visualizar los Objetos SNMP del Perfiles de Acción seleccionado.
  - DelComp
  - AddComp
  - ReadAllProfiles

Para quitar los objetos solo es necesario seleccionarlos y hacer clic en ***Quitar***. Para agregar un nuevo objeto, se hace clic en la opción ***Agregar***.

Al seleccionar esta última opción, es llamado en la ventana emergente el archivo CGI ***/cgi-bin/MIBBrowserFrameset.cgi***. Este archivo CGI crea en esta nueva ventana, tres Marcos en los que llama los siguientes archivos

- ***/MIBbrowserWeb/folderTreeLeftFrameiso.html*** en el Marco Izquierdo.
- ***/MIBbrowserWeb/MIBBrowserMain.html*** en el Marco Central.
- ***/cgi-bin/MIBBrowserResults.cgi*** en el Marco Derecho.

Los archivos de los marcos Izquierdo y Central son los mismos que para el caso de la configuración de Alarmas. En cambio el archivo desplegado en el marco Derecho es exclusivo de la asignación de Objetos SNMP para Perfiles de Acción (Ver Código - MIBBrowserResults.cgi). Esta formato de pantalla es mostrado en la siguiente figura.

Figura 6-59 Pantalla de Asignación de Objetos a Perfiles de Acción

En esta ventana se seleccionan los Objetos SNMP que formaran parte del Perfil y para terminar se hace clic en la opción ***Agregar***. Esto carga los objetos seleccionados dentro del Perfil de Acción.

6.4.6.7 MIBs

6.4.6.7.1 MIBsAdmin.cgi

Este archivo CGI es el encargado de la administración de la MIB del sistema ***WBNMS*** (Ver Código - MIBsAdmin.cgi). Desde la pantalla desplegada por él, el

administrador puede cargar nuevas MIBs, subir Archivos de MIB al servidor y borrar del sistema las Mibs Cargadas (Figura 6-60).

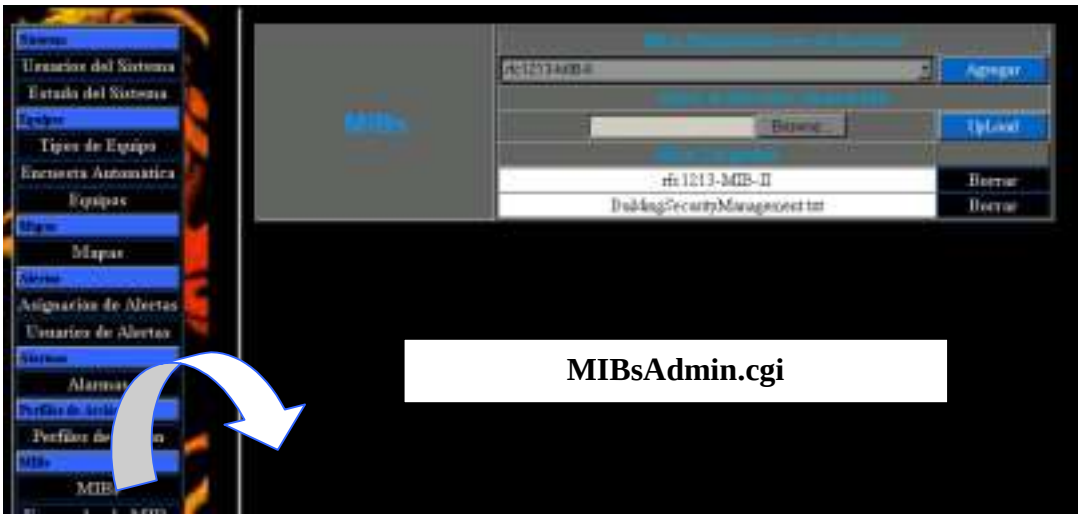


Figura 6-60 Pantalla de administración de MIBs

Utiliza las siguientes clases y Librerías:

- ***CGI*** para crear el código html.
- ***MIBManager*** para administrar y visualizar las MIBs del sistema ***WBNMS***.
  - ReadMIBs
  - MIB
  - MibLoad
  - AddMIBs
  - DelMIBs
  - UpLoad
- ***OID2js*** para generar el archive de configuración para el navegador de MIBs.
  - ***OID2js***

Para cargar nuevas MIBs en el sistema, el administrador las selecciona del menú desplegable y hace clic en la opción ***Agregar***. La MIB es compilada entonces, si el proceso termina exitosamente, el nombre de esta MIB es agregada en la lista de MIBs Cargadas, de lo contrario se muestra el mensaje de error correspondiente.
Para subir un archivo de MIB al servidor, se coloca el Path local al mismo y se selecciona la opción ***UpLoad***.

6.4.6.7.2 MIBBrowserFrameset.cgi

Este Archivo CGI ya fue descrito brevemente con anterioridad, en este punto se amplia dicha explicación (Ver Código - MIBBrowserFrameset.cgi). Tiene por objetivo crear divisiones en Marcos de la ventana a la que es dirigido. La

configuración de los Marcos por el abiertos es variable, dependiendo del objetivo final. Se pueden distinguir tres casos en los que es llamado este archivo:

- Para la configuración de Alarmas.
- Para la configuración de Perfiles de Acción.
- Para explorar la MIB del sistema.

Este último caso es el mas general y el que concuerda con opción del menú de Administración ***Navegador de MIBs***. Es por ello que, a diferencia de los dos casos anteriores, abre dos Marcos ya que solo se recorrerá la MIB sin realizar selección alguna (Figura 6-61).

Figura 6-61 Pantalla de Exploración de la MIB del sistema

Esta es la representación típica de una MIB, la forma de árbol. Los distintos objetos SNMP que componen la MIB del sistema son recorridos jerárquicamente en el ***árbol de Objetos*** (Marco Izquierdo). En el Marco Derecho (Antes Central) son mostrados los Objetos que se encuentran en los extremos de las ramas del árbol (los que definen Objetos de Gestión y no los del tipo Object Identifier).La acción de Navegación es ejemplificada en la siguiente figura.

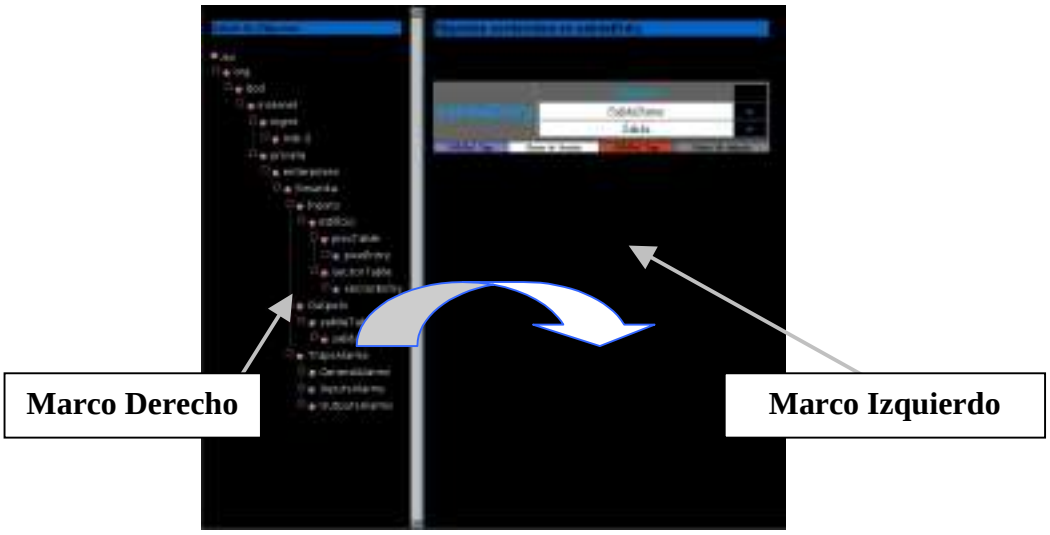


Figura 6-62 Acción de Navegación

Para dar formato al esquema de árbol de la MIB del sistema, son utilizados tres archivos Java Script:

- ***/MIBbrowserWeb/ua.js*** para la detección del tipo de Navegador.
- ***/MIBbrowserWeb/ftiens4.js*** para el formato de árbol.
- ***/MIBbrowserWeb/defineMyTreeiso.js*** archivo generado con la compilación de las MIBs.

6.5 Archivos de Instalación

Los archivos de instalación del sistema se encuentran dentro del directorio ***Install.d***. Básicamente, se pueden dividir en dos grupos, los archivos de creación y configuración de la base de datos y los archivos de instalación propiamente dichos.

6.5.1 Base de Datos

Partiendo del diseño de la Base de Datos, ***WBNMSDB***, realizada en la etapa anterior, se especificará el código SQL que construye las tablas de la Misma. Además se insertan dentro de la base de datos los valores default del sistema.
Este código es compatible con el lenguaje SQL soportado por el servidor ***MySQL***.
El archivo SQL resultante, ***WBM.sql***, estará dentro del sub directorio ***SQL.d*** dentro del directorio de instalación ***/Install.d***.
Para ver el código fuente, referirse a la sección Anexo Código - WBM.sql.

6.5.2 Archivo de Instalación

Esta dentro del directorio de instalación y cumple la función de configurar todos los archivos del sistema para que respondan a la nueva estructura de archivos y manejen los permisos de ejecución, escritura y lectura del sistema para los usuarios del sistema operativo asignados a la aplicación ***WBNMS***. Además de verificar crear usuarios para la base de datos, verificar los paquetes de software requeridos y configurar los demonios del sistema ***WBNMS*** para que sean arrancados con el sistema operativo. Este archivo de instalación, que recibe el nombre de ***install.pl***, esta escrito enteramente en Perl. Para ver el código fuente, referirse a la sección Anexo Código - install.pl.

6.6 Conclusión de la etapa de Implementación

Al termino de esta etapa, todo el código fuente de la aplicación está desarrollado y por lo tanto, queda completamente definido el sistema ***WBNMSv1.0***.
La siguiente etapa en el desarrollo consiste en la prueba de sistema, que pretende probar el funcionamiento del sistema.

7 Prueba de Sistema

7.1 Presentación del Caso de Prueba

7.1.1 Descripción General

Se implementara una red de Monitoreo de Seguridad utilizando el sistema de gestión ***WBNMS v1.0***. Para ello se cuenta con un software de monitoreo de seguridad denominado ***Centinela***, con capacidad de comunicación SNMPv1.
Se armara la red según el siguiente gráfico:

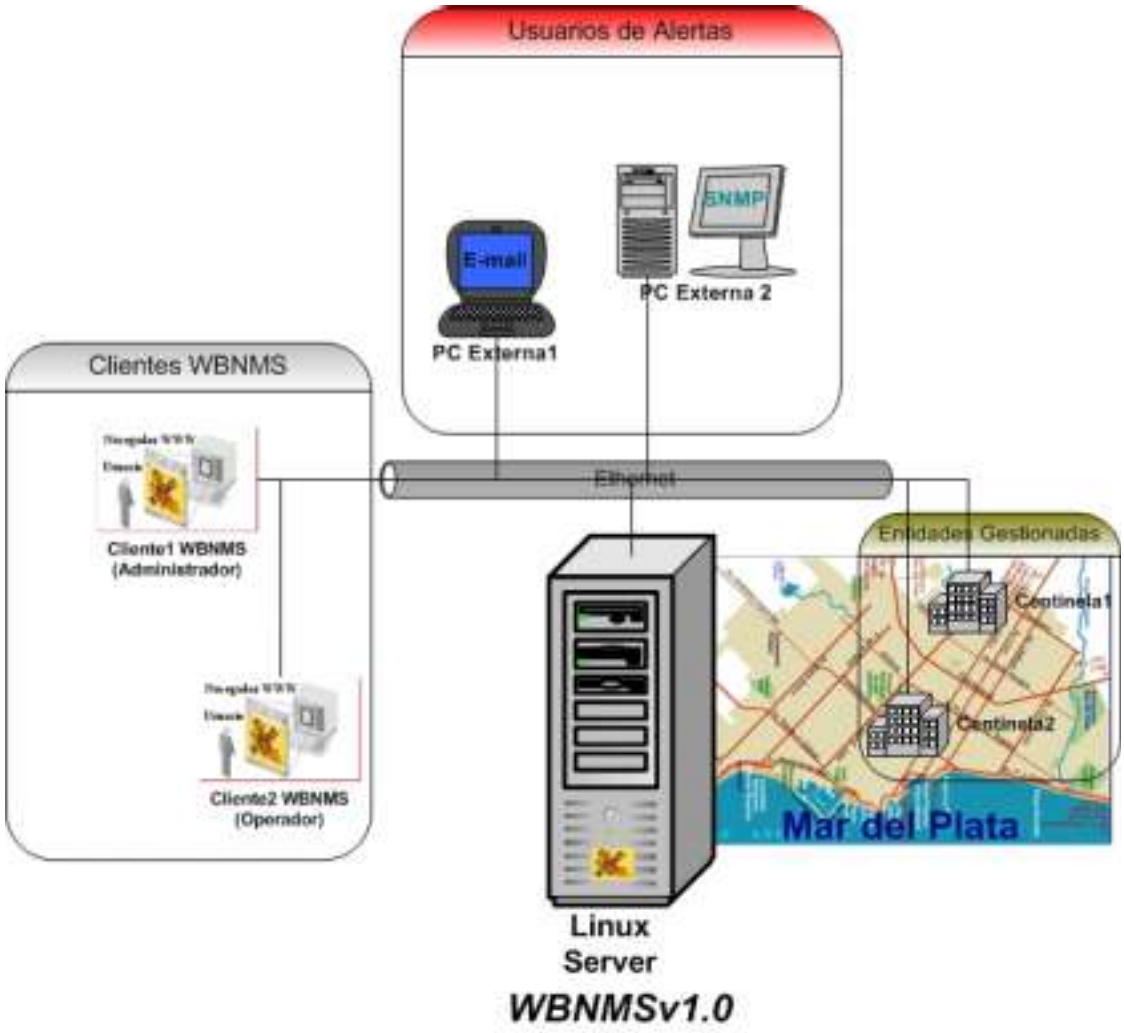


Figura 7-1 Red de Prueba General

Se conectará a una red Lan ***Ethernet*** todos los equipos, clientes, entidades gestionadas (***Centinela***), servidor ***WBNMS*** y los usuarios de alertas.

7.1.2 Sistema Centinela (versión Demo)

Es un sistema de monitoreo de seguridad (20 censores) y control remoto de interruptores (8 interruptores). En esta versión Demo, cuya pantallas se ejemplifican en la siguiente figura, el sistema distribuye los 20 censores en 4 pisos divididos en 5 sectores con un censor cada uno.

Figura 7-2 Pantallas del Sistema de Seguridad Centinela

El estado del piso-sector es representado por colores:

- verde = OK
- rojo = Alarmado

Los ocho interruptores pueden ser accionados localmente desde la interfaz ***OutManager*** provista por el software ***Centinela***. Dicha interfaz es mostrada en la siguiente figura.

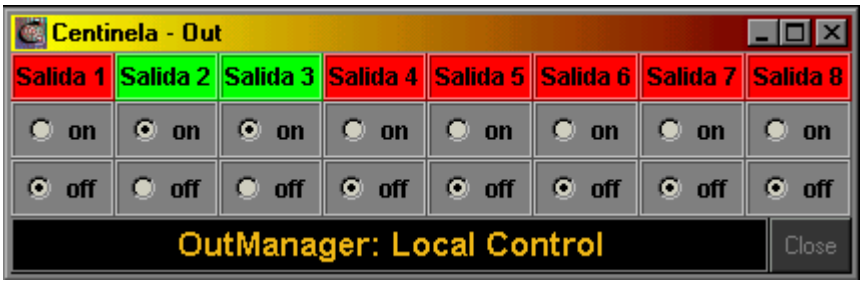


Figura 7-3 Interfaz *OutManager*

También se indican con un código de colores, el estado de cada interruptor. Siendo:

- verde = Encendido (on)
- rojo = Apagado (off)

Tanto el estado de los distintos sectores como el de los interruptores es reportado vía SNMP v1 a un gestor configurado según la MIB del sistema (***BuildingSecurityManagement.txt***). Esta última, define los distintos objetos de gestión y las alarmas que posee el ***Centinela***.

7.2 Objetivos

El objetivo de la prueba es verificar el funcionamiento del sistema de gestión ***WBNMS v1.0*** en un entorno de operación aproximado a la realidad partiendo desde la instalación del mismo.

7.3 Protocolo de Prueba

El Protocolo de prueba debe definir los pasos a seguir durante la prueba de sistema para permitir la repetición exacta de la misma. Esto asegura, si el sistema posee un comportamiento lineal (deseado), sean obtenidos los mismos resultados.

A continuación se define el Protocolo de Prueba:

1. Verificación existencia de la documentación necesaria (Manuales de Instalación, Administración y Operación).
2. Disponibilidad de Plataformas de Software y Hardware.
3. Armado de la red de Prueba.
4. Instalación del sistema ***WBNMS v1.0***.
5. Configuración de cliente.
6. Ingreso al sistema.

7. Configuración del sistema de gestión para incorporar las entidades ***Centinela*** a gestionar.
    a. Subir MIB del sistema de Monitoreo de Seguridad (***Centinela***) al servidor ***WBNMS***.
    b. Compilar MIB. Verificación de compilación de la MIB.
    c. Creación de nuevo tipo de Equipo: ***Centinela***. (Subir gráfico Centinela.gif)
    d. Configurar mapas (Argentina y dentro Mar del Plata).
    e. Crear perfiles de Acción y asociarlos al tipo creado.
    f. Agregar equipo-entidad por autodiscovery y manualmente.
    g. Configurar Alarmas.
    h. Configurar Alertas.

8. Realización de la Prueba de Gestión.

7.4 Desarrollo

Se seguirá durante desarrollo de la prueba de sistema el protocolo planteado en el punto anterior.

7.4.1 Verificación de la documentación

Documentación disponible:

- Manuales de Administración-Instalación & Operación de ***WBNMS v1.0***.
- Descriptivo del Sistema Centinela versión Demo (***www.simatika.com.ar***).
- Informe de Trabajo Final ***WBNMS*** (opcional).
- Manual de Administración ***Apache***.
- Manual de Administración ***MySQL***.

Información Implícita:

- Conocimientos de sistemas Linux/Unix.
- Conocimientos de Aplicaciones Web.
- Conocimientos de sistemas de Gestión.

7.4.2 Plataformas de Hardware y Software

Según el manual de Instalación de ***WBNMS v1.0*** se instala la plataforma de:

7.4.2.1 Hardware

- ***Servidor: Intel Pentium 200 MHz***
- ***Memoria RAM: 32 MB RAM disponibles (64 MB requeridos)***
- ***Disco Duro: 8 GB***
- ***Placa de Red: FastEthernet 10/100 Mbps – 10BASE-T_100BASE-TX***

7.4.2.2 Software

- ***Sistema Operativo: Mandrake 8.0 (kernel 2.4.3-20)***
- ***Servidor Web: Apache v1.3.19***
- ***Base de Datos: MySQL v3.23.52***
- ***Lenguaje: Perl v5.6.0***

  Módulos de Perl:
  - ***GD1.9***
  - ***SNMP_Session 0.88***
  - ***Msql-Mysql-modules 1.2219***
  - ***Data::ShowTable 3.3***
  - ***DBI 1.30***

Además se instala un servidor de correo ***Postfix v20010228*** y un servidor de nombres ***BIND DNS Server v9.1.1***.

7.4.3 Red de Prueba

Tratando de cubrir los objetivos planteados para la prueba y de acuerdo con la descripción general del sistema, se arma la siguiente red de prueba:

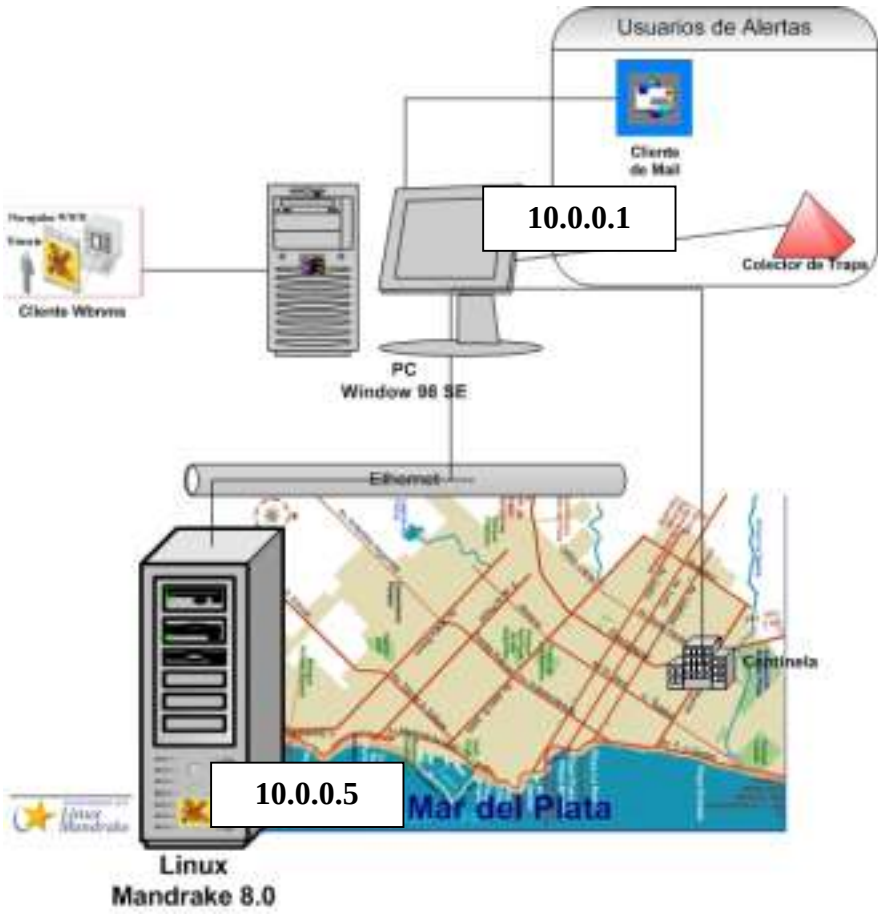


Figura 7-4 Red de Prueba Utilizada

La red consta de dos máquinas, PC y Servidor, las dos de arquitectura Intel Pentium pero con distinto sistema operativo, conectadas físicamente por una red ethernet de 10/100 Mbps (Null-MODEM). Como ya se especificó, el servidor posee un sistema operativo Linux de distribución Mandrake 8.0; mientras que la PC posee un sistema operativo Windows 98 SE. Sobre esta última, se instaló un ***cliente de correo*** estándar y un ***colector de traps SNMP***, que actuaran como Usuarios de Alertas. Además, se instaló también en dicha máquina, el sistema de monitoreo ***Centinela versión demo***, que actúa como entidad gestionada.
Como clientes se utilizarán en las pruebas los navegadores Internet Explorer 5 y el Mozilla 1.0.
Esta red, aunque reducida en recursos, es representa perfectamente las funcionalidades del sistemas, quedando para etapas posteriores, las pruebas de performance; las cuales requerirían de una prueba de campo.

7.4.4 Instalación del Sistema *WBNMS v1.0*.

Se siguieron los pasos de instalación descriptos en el manual de instalación de la aplicación ***WBNMS v1.0***:

Aclaración: **toda la instalación se realiza en forma remota desde la PC utilizando un cliente SSH (PuTTY).**

- Primer y segundo Paso: Entrar al sistema como usuario ***root*** y Copiar el Archivo ***Wbnmsv1.0.tar.gz*** al directorio donde se desea alojar la aplicación.

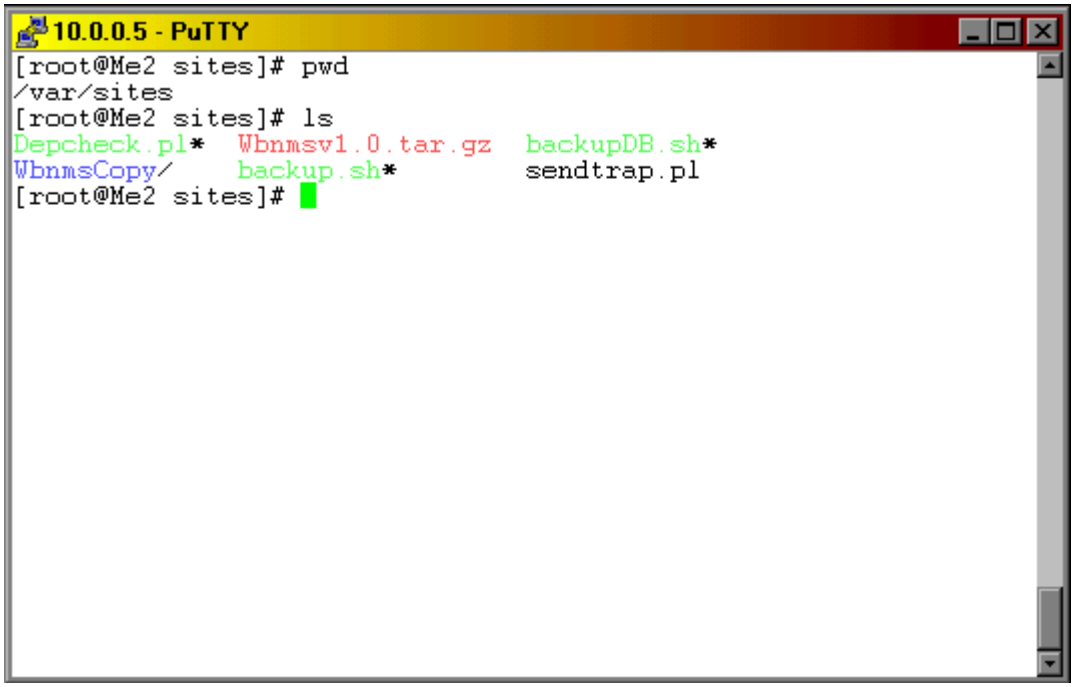


Figura 7-5 Salida del comando ls dentro de /var/sites

- Tercer Paso: Expandir los archivos de la Aplicación ejecutando siguiente comando:

tar –zxvf Wbnmsv1.0.tar.gz

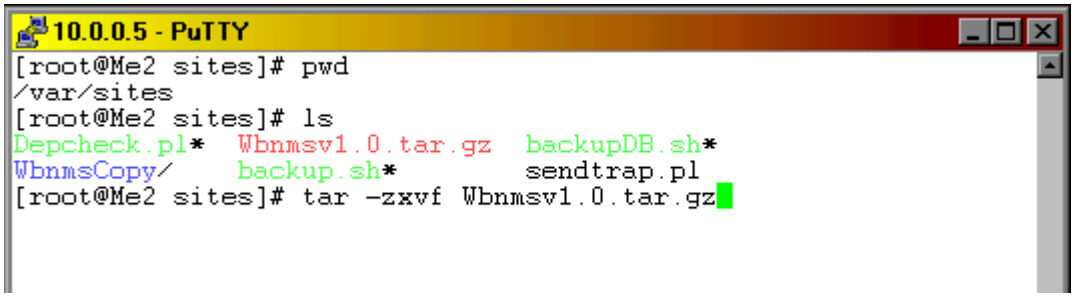


Figura 7-6 Comando de untar

Se crea el subdirectorio ***Wbnms*** sobre el directorio ***/var/sites***; este directorio creado es el raíz de la aplicación ***WBNMS***.

- Cuarto paso: Entrar al directorio de instalación de ***WBNMS v1.0*** ejecutando el siguiente comando:

cd Wbnms/Install.d

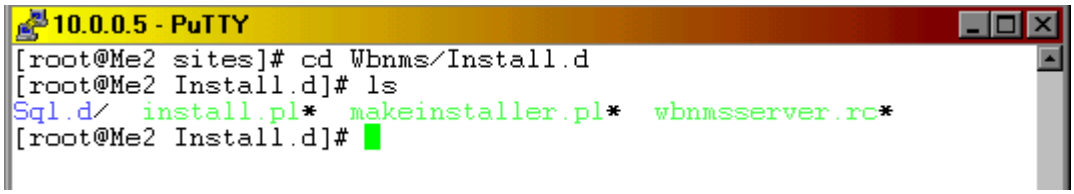


Figura 7-7 Entrando al directorio de Intalación

- Quinto Paso: Se busca la información necesaria para ejecutar el archivo de instalación.

Directorio Raíz de *WBNMS*	***/var/sites/Wbnms***
Usuario Apache	***apache***
Path al archivo de configuración del Apache	***/etc/httpd/conf/httpd.conf***
Nombre del servidor virtual	***www.wbnms.com.ar***
Dirección IP	***10.0.0.5***
Port	***80***
Path al programa mysql	***/usr/local/mysql/bin/mysql.***

- Sexto Paso: Se ejecuta el archivo de instalación install.pl con las opciones del paso anterior:

perl install.pl
/var/sites/Wbnms
apache
/etc/httpd/conf/httpd.conf
www.wbnms.com.ar
10.0.0.5
80
/usr/local/mysql/bin/mysql

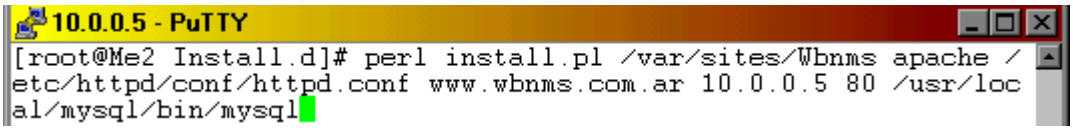


Figura 7-8 Comando de Instalación

- Séptimo Paso: Se Verifica la instalación realizada, observando el output de la ejecución del archivo ***install.pl***.

*****Verificando modulos requeridos instalados***
Modulo DBI instalado .... OK
Modulo SNMP_Session instalado .... OK
Modulo SNMP_util instalado .... OK
Modulo BER instalado .... OK
Modulo GD instalado .... OK
*****Configurando WBNMS v1.0 en el directorio /var/sites/Wbnms***
/var/sites/Wbnms/Wbnms_Core/ABM ..... OK
…
Todos los subdirectorios OK
…
/var/sites/Wbnms/Wbnms_Web/pics ..... OK
*****Configurando Vhost en el Servidor Apache***
Instalación del Vhost ..... OK
*****Configurando usuario y permiso en el servidor Mysql local.***
Instalación Mysql ..... OK

Se comprueba que la instalación ha sido realizada en forma correcta.

- Octavo Paso: Se reinicia el servidor Apache con el comando:

/etc/init.d/httpd update

Figura 7-9 Salida del comando de Intalación

- Noveno Paso: Se copia el archivo al directorio ***init.d***

cp –rp /var/sites/Wbnms/Install.d/wbnmsserver.rc /etc/init.d/wbnms

- Décimo paso: Se configura la aplicación para que sea iniciada con el sistema. Para ello se crea un link simbólico en el directorio rc3.d.

ln –s /etc/init.d/wbnms /etc/rc.d/rc3.d/S99wbnms

Figura 7-10 Creación del link de inicio automático

- Undécimo Paso: Se inicia la aplicación ***WBNMS v1.0***. Se ejecuta el siguiente comando:

/etc/init.d/wbnms start

Figura 7-11 Inicio manual de la Aplicación

Con este último paso queda completamente instalado la aplicación ***WBNMS v1.0***.

7.4.5 Configuración del Cliente

La PC que actuará de cliente ***WBNMS*** deberá ser configurada de manera que el navegador Web instalado conozca la dirección IP del servidor ***WBNMS*** a través del domino de la aplicación, ***www.wbnms.com.ar*** (virtual host configurado). Para ello existen dos caminos posibles:

1. Configurar el ***DNS*** instalado en el servidor ***Linux***.
2. Agregar el mapeo ***hostname-IP*** en el archivo ***hosts***.

Para el desarrollo de esta prueba se a elegido la primer opción, como la opción más acorde a un funcionamiento operativo normal del cliente.

7.4.6 Ingreso al sistema

Una vez configurado el cliente , el usuario es capaz, a través de él, de ingresar al sistema ***WBNMS*** al requerir la página default del dominio de la aplicación

(***www.wbnms.com.ar***) . El servidor Web responde, solicitando un usuario y password para el ingreso al sistema. Se utilizara entonces el usuario tipo administrador ***admin*** y su password ***admin***, definidos como valores default del sistema ***WBNMS***. La siguiente figura muestra el requisito de ingreso al sistema.

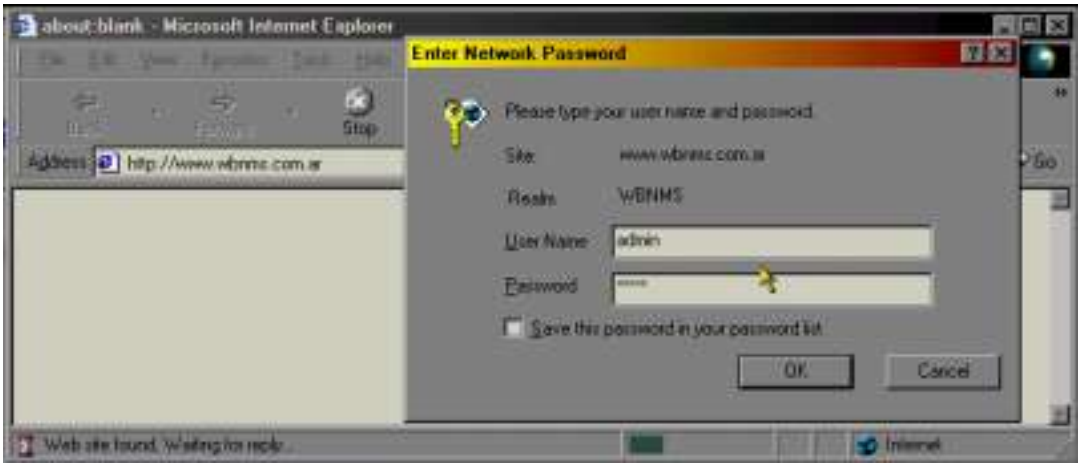


Figura 7-12 Pantalla de ingreso al sistema

Luego de estos pasos, el usuario administrador se encuentra "dentro" del sistema ***WBNMS***, con lo que es posible la configuración del mismo.

7.4.7 Configuración del Sistema de Gestión

- El primer paso de configuración consiste en cargar las MIBs básicas a la MIB del sistema. Esto deberá ser realizado contemplando las dependencias que en ellas existen. Para simplificar dicha operación, la aplicación provee una "MIB" que define los objetos básicos para su funcionamiento, denominada "***basic"***. Entonces, siguiendo el procedimiento de compilación descripto en el Manual de Administración se compila dicha MIB.

- En este punto es posible configurar el sistema para la gestión de los equipos ***Centinela***.

a) Desde el menú de administración de MIBs (Figura 7-13) se "sube" al servidor (recordar que el tipo de instalación es remota) la MIB ***BuildingSecurityManagement.txt***.

b) Incorporada dicha MIB a la lista de existencia del servidor, es seleccionada y compilada dentro de la MIB del sistema.

Figura 7-13 Pantalla de Administración

c) Para la creación del nuevo ***tipo***, que agrupara los equipos ***Centinela***, se procede a la pantalla de administración de ***Tipos de Equipo*** (Figura 7-14).

Figura 7-14 Pantallas de Administración de Tipos de Equipos

Se "sube" al servidor el icono que representará a los equipos del tipo ***Centinela*** utilizando, como muestra la figura anterior, la pantalla de administración gráfica para Tipos. Luego se secciona dicho icono y se nombra al tipo ***Centinela*** agregándolo al sistema.

d) Son creados dos mapas: ***Argentina*** y ***Mar del Plata***. Para lo cuál se recurre a la pantalla de administración de mapas (Figura 7-15).

Figura 7-15 Pantallas de Administración de Mapas

Son subidos los mapas en si, la representación gráficos, al servidor desde la pantalla de administración gráfica como muestra la figura anterior.
Luego son asociados como mapa raíz, ***Argentina***, y sub-map del anterior, ***Mar del Plata***. Esto es realizado desde la pantalla de asociación de Mapas-Equipos del mapa creado ***Argentina*** (Figura 7-16).

Figura 7-16 Pantalla de Asociación de Mapas-Equipos

Es necesario seleccionar la posición (coordenadas gráficas) que tendrá el sub-mapa (***Mar del Plata***) dentro del mapa de ***Argentina***, para ello se utiliza la pantalla de localización gráfica mostrada en la siguiente figura.

Figura 7-17 Pantalla de Localización Grafica

Desde esta pantalla se señala la posición y luego el sub-mapa es asociado al mapa raíz.

e) Para crear los Perfiles de Acción que serán utilizados durante la prueba, se accede a la pantalla de Administración de Perfiles de Acción (Figura 7-18).

Figura 7-18 Pantalla de Administración de Perfiles de Acción

Desde esta pantalla se crean los dos perfiles de acción a utilizar para los equipos ***Centinela***:

- ***Consulta de Estados de Piso***: Estado de cada uno de los 4 pisos del edificio gestionado por el equipo ***Centinela***.
- ***Estado de llaves remotas***: Estado de cada una de los interruptores del edificio gestionado por el equipo ***Centinela***.

Luego de creados los perfiles, deberán ser asociados a ellos los objetos-variables de gestión correspondientes. Entonces, para el perfil "***Consulta de Estados de Piso***" estos objetos asociados serán instancias del objeto ***PisoStatus*** definido en la MIB ***BuildingSecurityManagement.txt***. Este objeto representa el "Estado de Alarma" del piso según la instancia definida según indica la siguiente tabla.

Representación	Instancia	OID
Estado del Piso 1	PisoStatus.1	1.3.6.1.4.1.555.1.1.1.1.1.***1***
Estado del Piso 2	PisoStatus.2	1.3.6.1.4.1.555.1.1.1.1.1.***2***
Estado del Piso 3	PisoStatus.3	1.3.6.1.4.1.555.1.1.1.1.1.***3***
Estado del Piso 4	PisoStatus.4	1.3.6.1.4.1.555.1.1.1.1.1.***4***

Tabla 7-1 Definición de OIDs de estado de pisos

Desde la pantalla de administración de objetos asociados al perfil "***Consulta de Estados de Piso***" (Figura 7-19), son agregados las objetos instanciados.

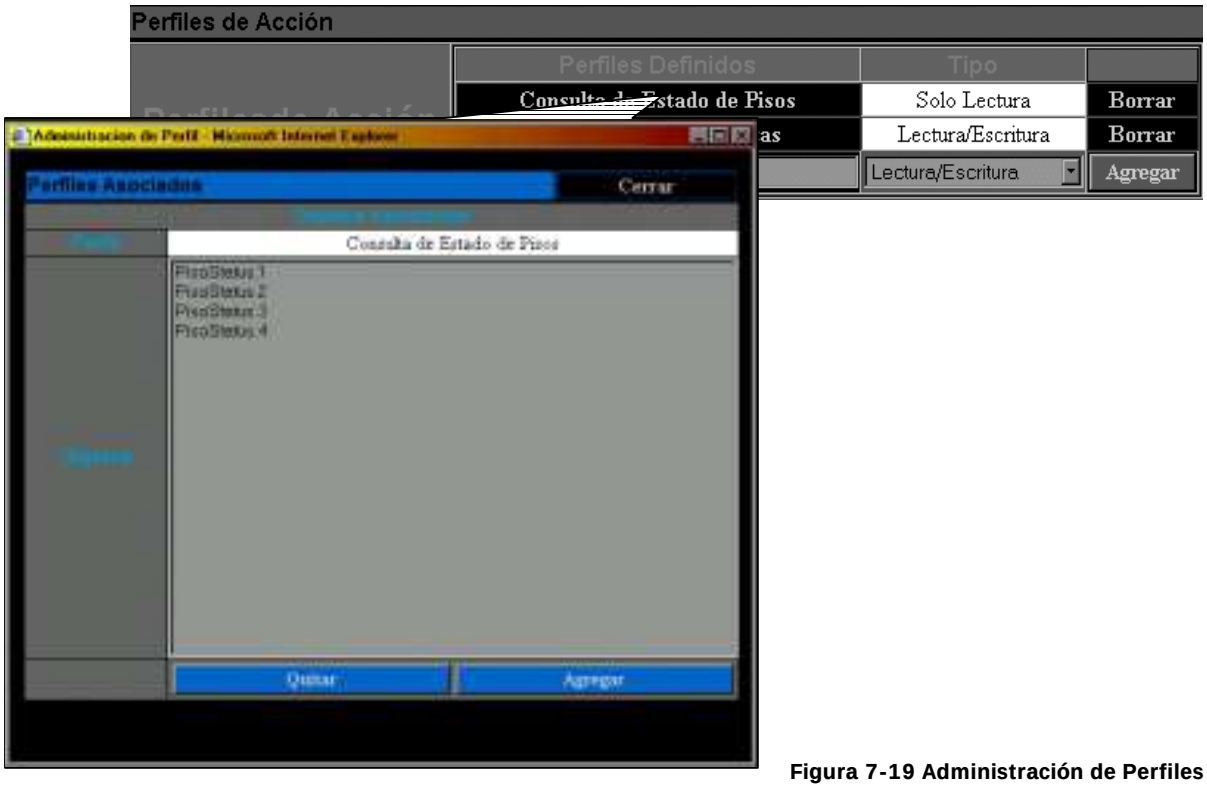


Figura 7-19 Administración de Perfiles

Para seleccionar los objetos que conformarán el perfil, se abre la pantalla de navegación de la MIB del sistema (Figura 7-20).

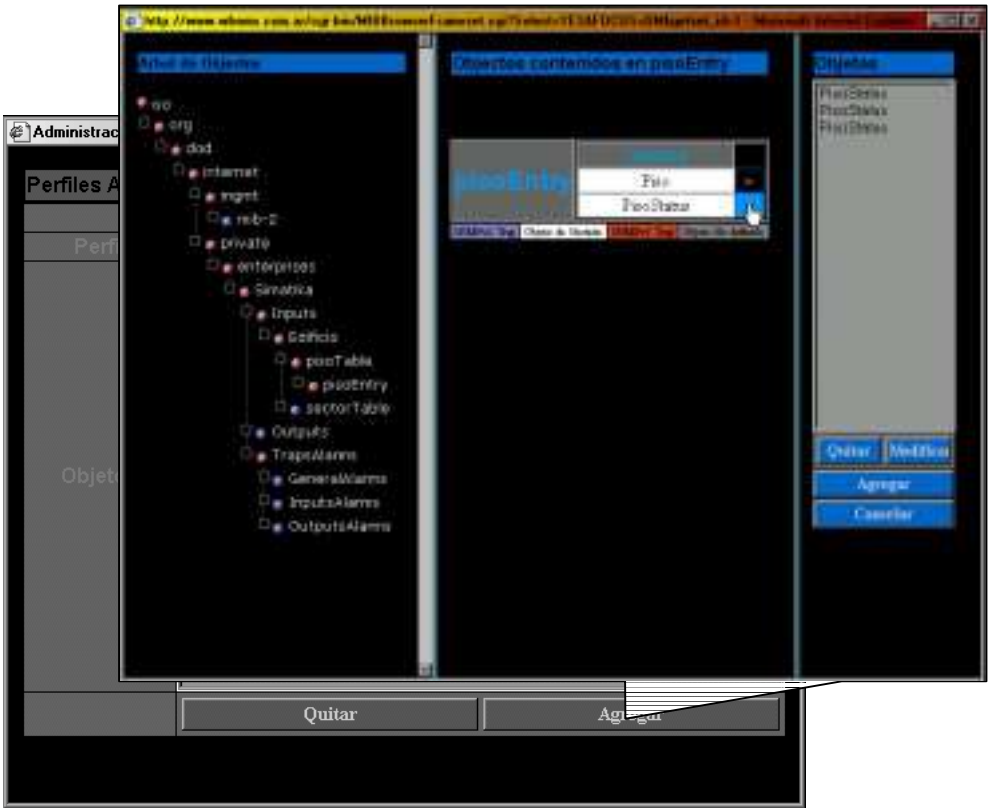


Figura 7-20 Agregado de Objetos

En esta pantalla se selecciona el objetos ***PisoStatus*** de manera de tener cuatro de ellos en la lista de objetos. Luego, estos objetos "base" se modifican, es decir, son instanciados (Figura 7-21).

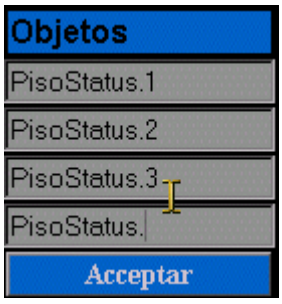


Figura 7-21 Edición de Objetos

De forma análoga, son asociados los objetos al perfil "***Estado de llaves remotas***". Para ello se considera el objeto "base" ***SalidaStatus*** y luego es instanciado por cada una de las Salidas que representan las llaves remotas según la siguiente tabla:

Representación	Instancia	OID
Estado de la Salida 1 (llave1)	SalidaStatus.1	1.3.6.1.4.1.555.2.1.1.1.***1***
Estado de la Salida 2 (llave2)	SalidaStatus.2	1.3.6.1.4.1.555.2.1.1.1.***2***
Estado de la Salida 3 (llave3)	SalidaStatus.3	1.3.6.1.4.1.555.2.1.1.1.***3***
Estado de la Salida 4 (llave4)	SalidaStatus.4	1.3.6.1.4.1.555.2.1.1.1.***4***
Estado de la Salida 5 (llave5)	SalidaStatus.5	1.3.6.1.4.1.555.2.1.1.1.***5***
Estado de la Salida 6 (llave6)	SalidaStatus.6	1.3.6.1.4.1.555.2.1.1.1.***6***
Estado de la Salida 7 (llave7)	SalidaStatus.7	1.3.6.1.4.1.555.2.1.1.1.***7***
Estado de la Salida 8 (llave8)	SalidaStatus.8	1.3.6.1.4.1.555.2.1.1.1.***8***

Tabla 7-2 Tabla de OIDs de las Salidas

Terminado este procedimiento, queda completamente configurados los perfiles requeridos para la prueba. Luego, estos perfiles deben ser asociados al "tipo de equipo" definido anteriormente como ***Centinela***. Esta asociación se hace desde la pantalla de "asociación de perfiles al Tipo ***Centinela***" desplegada desde la pantalla de administración de ***Tipos de Equipos*** como indica siguiente figura.

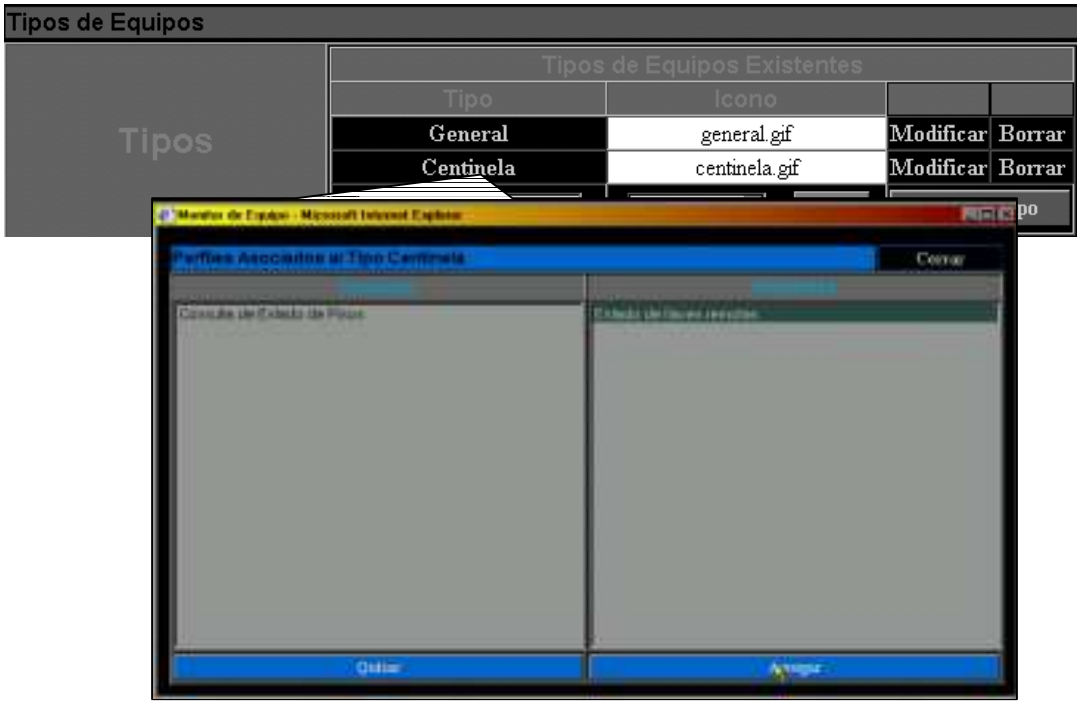


Figura 7-22 Asociación de Acciones a Perfiles

En esta pantalla son listados los perfiles incluidos en la asociación (lado izquierdo) y los perfiles restantes que aún no lo están (lado derecho).

f) Un equipo puede ser agregado al sistema de gestión "manualmente" o mediante el proceso de encuesta automática manejada por el demonio

keepalive , si es capaz de responder a la encuesta del objeto ***sysName***. Este último método es el elegido, por lo que se hace es simplemente activar el sistema ***Centinela*** en la PC y se espera a que el proceso de encuesta automática lo descubra. Cuando un nuevo equipo es descubierto por ***WBNMS***, este es incorporado a la lista de equipos nuevos en la pantalla de administración de ***encuesta automática***.

Figura 7-23 Agregar equipos desde una encuesta automática

Para que sea incorporado a la lista de equipos gestionados, el administrador deberá asociarlo a un ***Tipo de Equipo*** según corresponda, en este caso al tipo ***Centinela*** creado en pasos anteriores.
El siguiente paso de configuración, es la asociación del equipo ***Centinela*** creado con un mapa, haciéndolo "visible" para la gestión gráfica del sistema. Esto es realizado desde la pantalla de asociación de Mapas-Equipos del Mapa ***Mar del Plata*** seleccionada desde la pantalla de administración de Mapas (Figura 7-24).

Figura 7-24 Asociación de Mapa-Equipo

Siguiendo los métodos gráficos de posicionamiento ya explicados, el nuevo equipo ***Centinela*** es ubicado por sus coordenadas relativas a mapa de ***Mar del Plata***.

g) La configuración de alarmas condicionará la tarea de filtrado que realizara el sistema ***WBNMS*** (demonio ***AlarmSurveillance*** específicamente) de los eventos recibidos vía ***traps SNMP***. Por ello se definirá a continuación cuales son los traps enviados por el sistema ***Centinela***, definidos en su MIB, y bajo que condiciones, para luego especificar que "situaciones" serán verificadas durante la prueba.

El sistema ***Centinela*** define traps para las situaciones especificadas en la siguiente tabla:

Situación	trap	Variables Adjuntas	EnterpriceOID.SpecificTrap
Activación del sistema ***Centinela.***	**CentinelaStart**	Ninguna	**GeneralAlarms.1**
Desactivación del sistema ***Centinela.***	**CentinelaStop**	Ninguna	**GeneralAlarms.2**
Cambio de Estado en Sector a OK.	**SectorStatusOK**	Piso y Sector	**InputsAlarms.1**
Cambio de Estado en Sector a Alarmed.	**SectorStatusAlarmed**	Piso y Sector	**InputsAlarms.2**
Cambio de Estado en la llave a ON.	**SalidaStatusON**	Salida	**OutputsAlarms.1**
Cambio de Estado en la llave a OFF.	**SalidaStatusOFF**	Salida	**OutputsAlarms.2**

Tabla 7-3 Definición de Traps

Para la configuración total de la gestión del sistema centinela referido al monitoreo de Sectores habría que definir 40 filtros de eventos, dos por cada uno de los sectores (5 sectores x 4 pisos = 20 sectores). Esta configuración para mayor practicidad puede ser realizada directamente sobre la base de datos, sin utilizar la interfaz web y operando a través de la interfaz SQL de la MySql.
Ha efectos de la prueba, son configurados una cantidad representativa de 7 filtros de eventos (un sector por piso). De a cuerdo a este criterio, se definen los filtros de la siguiente tabla:

traps	Severidad	Variables Adjuntas
SectorStatusOK **SectorStatusAlarmed**	***OK*** ***Minor***	Piso = 1 y Sector = 1
SectorStatusOK ***SectorStatusAlarmed***	***OK*** ***Critical***	Piso = 2 y Sector = 2
SectorStatusOK **SectorStatusAlarmed**	***OK*** ***Mayor***	Piso = 3 y Sector = 4

SectorStatusAlarmed	***Warning***	Piso = ***1*** y Sector = ***Cualquiera***

Tabla 7-4 Configuración de Alarmas de Pisos

Para la configuración de la gestión de llaves remotas, la situación es similar a la de sectores, entonces , se selecciona 3 casos básicos a modo de prueba. Se definirá entonces:

traps	Severidad	Variables Adjuntas
SalidaStatusON ***SalidaStatusOFF***	***Critical*** ***OK***	Salida = 1
SalidaStatusON	***Warning***	Salida = ***Cualquiera***

Tabla 7-5 Configuración de Alarmas de Salidas

Se definen en total once filtros de eventos (alarmas) diferentes asociados al tipo ***Centinela***. Entonces desde la pantalla de administración de ***Alarmas*** son administradas estas nuevas asociaciones (figura 4-30).

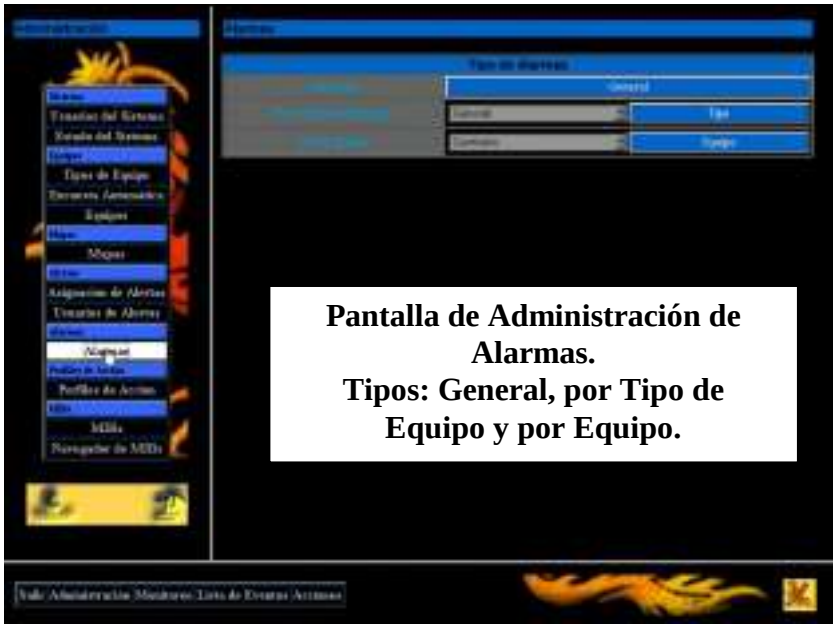


Figura 7-25 Pantalla de Administración de Alarmas

En esta pantalla es seleccionado la categoría de Alarmas asociadas al tipo ***Centinela***. Ya que es con este tipo con el que pretenderá realizar la prueba, dejando de lado la configuración por equipo y general.
Entonces desde la pantalla desplegada en la siguiente figura, que lista los filtros asociados al tipo, son configuradas la alarmas según la tabla antes mencionada. Donde en un primer paso, primero son creados los filtros con asociación a los objetos sin sus variables, y luego en un segundo paso de configuración son definidos los valores de las mismas.

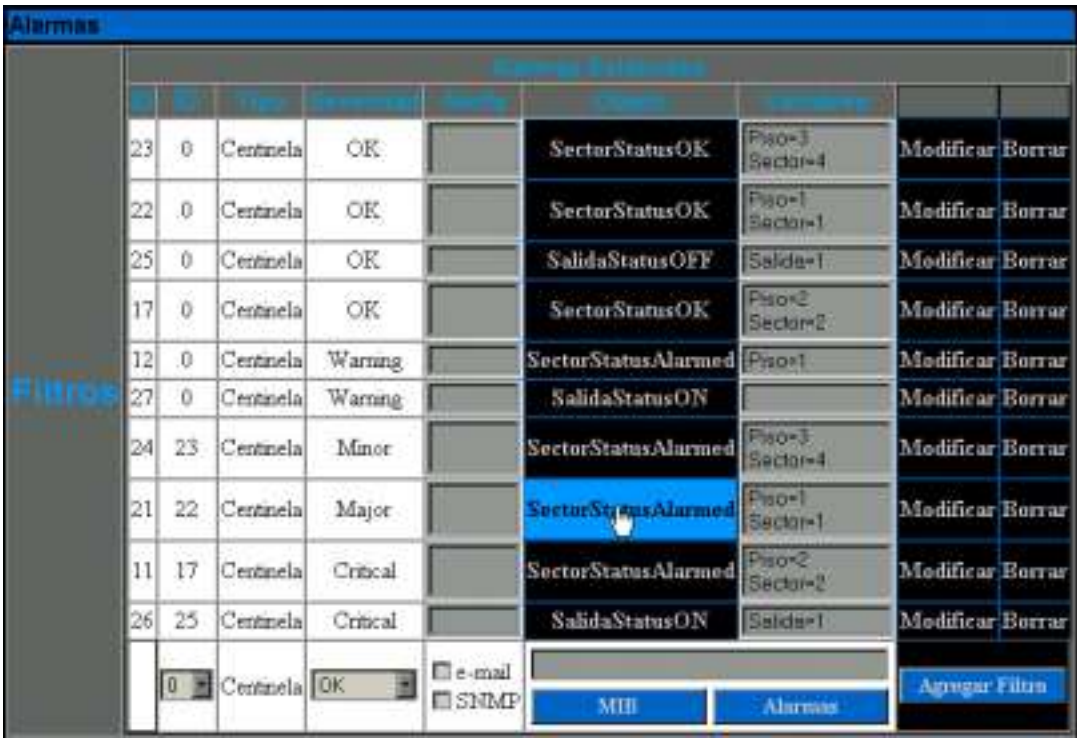


Figura 7-26 Filtros Configurados

Para la selección de los distintos objetos que compondrán las alarmas (eventos que pasen por los filtros), se utiliza la pantalla de selección de alarmas de la MIB del sistema (Figura 7-27).

Figura 7-27 Selección de Alarmas

Para la configuración de las variables asociadas a cada objeto se los selecciona, desplegando la pantalla de administración de objetos mostrada en la siguiente figura.

Figura 7-28 Administración e Objertos

h) El proceso de configuración de alertas es el conjunto de tres operaciones administrativas relacionadas: creación de Usuarios de Alertas, asignación de Alertas a usuarios definidos y asociación Alarmas-Alertas.
La ***Creación de Usuarios de Alertas*** es realizada desde la pantalla de administración de Usuarios de Alertas mostrada en la siguiente figura.

Figura 7-29 Administración de Usuarios

Desde ella se crearon dos usuarios nuevos con relación a la prueba, según los datos de la siguiente tabla.

Usuario de Alerta	e-mail	IP /comunidad SNMP
ua1	**ua1@me2.com.ar**	**10.0.0.1/public**
ua2	**ua2@me2.com.ar**	***No asignada***

Tabla 7-6 Usuarios Configurados

La ***Asignación de Alertas a usuarios definidos*** es realizada desde la pantalla de administración de Asignación de Alertas mostrada en la siguiente figura.

Figura 7-30 Configuración de Alertas

La asignación de las Alertas esta definida según criterios de horario así como también áreas de incumbencia, definido por un mapa, o equipos específicos. Es por ello que las alertas fueron asignadas a los usuarios según el criterio mostrado en la siguiente tabla.

Usuario de Alerta	Forma de reporte	Turno	Equipo	Mapa
ua1	**e-mail**	**0:00-23:59**	***Todos***	***Todos***
ua1	**SNMP**	**0:00-23:59**	***Todos***	***Todos***
ua2	**e-mail**	**0:00-23:59**	***Centinela***	***Todos***

Tabla 7-7 Asignación de Alertas

La ***asociación Alarmas-Alertas*** es realizada de acuerdo al criterio expresado en la siguiente tabla.

traps	Modo de Alerta	Severidad	Variables Adjuntas
SectorStatusOK **SectorStatusAlarmed**	***e-mail*** ***e-mail***	***OK*** ***Major***	Piso = 1 y Sector = 1
SectorStatusOK ***SectorStatusAlarmed***	***e-mail/SNMP*** ***e-mail/SNMP***	***OK*** ***Critical***	Piso = 2 y Sector = 2
SectorStatusOK **SectorStatusAlarmed**	***e-mail*** ***e-mail***	***OK*** ***Minor***	Piso = 3 y Sector = 4
SectorStatusAlarmed	***e-mail***	***Warning***	Piso = ***1*** y Sector = ***Cualquiera***
SalidaStatusON ***SalidaStatusOFF***	***e-mail/SNMP*** ***e-mail/SNMP***	***Critical*** ***OK***	Salida = 1
SalidaStatusON	***SNMP***	***Warning***	Salida = ***Cualquiera***

Tabla 7-8 Configuración de Modos de Alerta

La configuración necesaria es realizada nuevamente desde la pantalla de administración de Alarmas, donde es modificada la columna ***Alerta*** para cada filtro, indicando con ello cuál será el modo reporte de la Alerta.

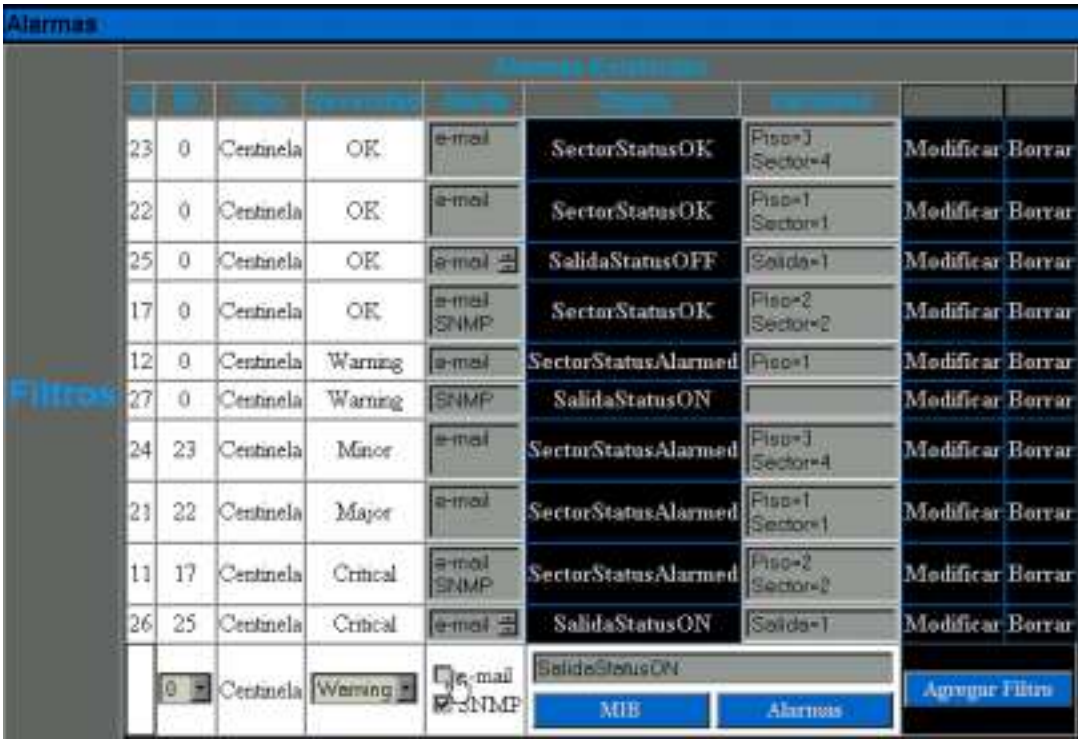


Figura 7-31 Agregado de modos de Alerta

7.4.8 Prueba de Gestión

La prueba del sistema para la gestión de los equipos ***Centinela*** consistirá en verificar la consistencia de los datos de gestión a lo largo de todo el "camino" que recorren los eventos a través del sistema, realizar encuestas y efectuar la modificación de variables en él.

Se comienza con la inicializaron del sistema ***Centinela*** en la PC y se verificando su correcto funcionamiento. Luego se prueban las situaciones representativas planteadas en la siguiente tabla.

Situación	Severidad	Objeto afectado
1.Alarma de Sector	***Warning***	***Piso1, Sector 5***
2.Alarma de Sector	***Major***	***Piso1, Sector 1***
3.Alarma de Sector	***Minor***	***Piso3, Sector 4***
4.Alarma de Sector	***Critical***	***Piso2, Sector 2***
5. Se quitan las Alarmas 4, 3 y 2	***OK***	***Piso1, Sector 1***
		Piso3, Sector 4
		Piso2, Sector 2

Tabla 7-9 Ejemplo de Situaciones de Prueba

Son generadas las alarmas descriptas una a una, en forma incremental, sumándose sus efectos en el sistema de gestión de manera de comprobar la reacción de este ante situaciones complejas (suponiendo que las alarmas simples son reconocidas correctamente).

A continuación se muestra la pantalla de monitoreo en el instante inicial de la prueba.

Figura 7-32 Pantalla de Gestión Inicial

1. Se genera el primer caso de alarma en el sistema ***Centinela*** , **Piso 1-Sector 5 Alarmado**, reportado localmente en la PC como indica la siguiente figura.

Figura 7-33 Pantalla del sistema Centinela Alarmada

El sistema ***Centinela*** reporta al gestor ***WBNMS*** el evento ocurrido como:

SectorStatusAlarmed *con variables* ***Piso = 1 y Sector = 1***

Este evento es recibido por ***WBNMS*** y procesado. Al pasar por uno de los filtros configurados anteriormente, dicho evento es reconocido como alarma y mostrado en pantalla, indicando el cambio de estado del equipo ***Centinela***. La pantalla de monitoreo resultante es la siguiente.

Figura 7-34 Pantalla de Gestión Centinela en Warning

El evento filtrado es convertido en alarma por el sistema ***WBNMS*** y reportado convenientemente. Además, si dicha alarma fue asociada a algún tipo de Alerta, esta es enviada a los usuarios configurados. En este caso, el método de alerta asociado a esta alarma es el de e-mail. Entonces son enviados las Alertas correspondientes vía e-mail a los ***Usuarios de Alertas*** configurados, ***ua1*** y ***ua2***.
Se muestra a continuación el mensaje de e-mail recibido por uno de los usuarios, mostrado desde el cliente de mail, ***Outlook Express***, instalado en la PC.

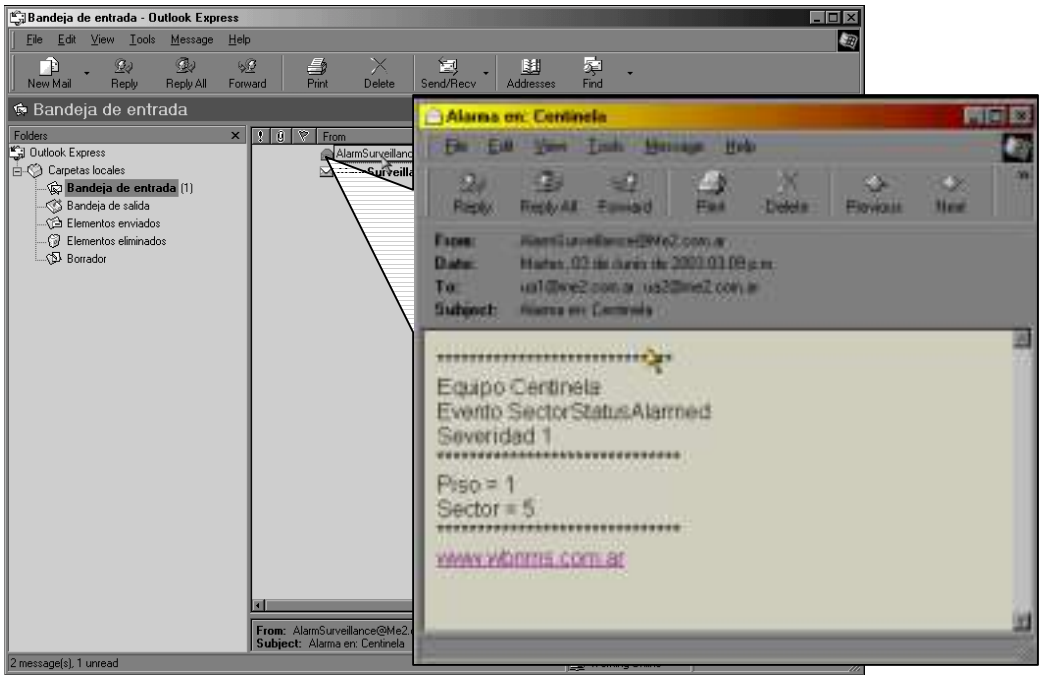


Figura 7-35 E-Mail recivido

En el siguiente diagrama de flujos se muestra el comportamiento registrado por el sistema de gestión ante el evento generado en el equipo ***Centinela***.

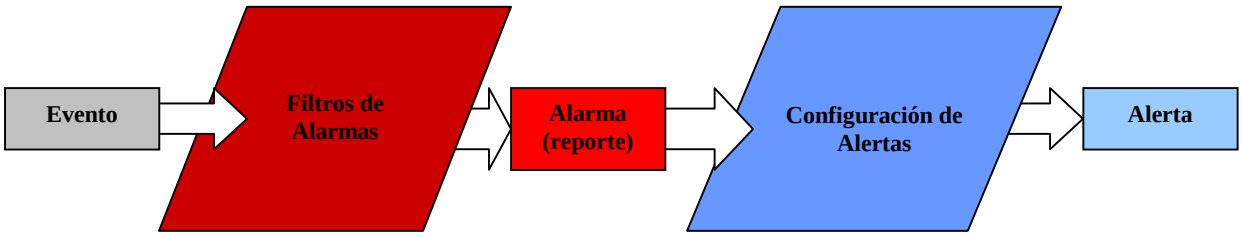


Figura 7-36 Modificación de los datos a través del sistema

Un evento pasa por un filtro de alarma y es convertido en alarma-reporte, esta última coincide con una configuración de Alerta y entonces es transformada en alerta.

2. El segundo caso de alarma, planteado en la tabla 7-9 , agrega al comportamiento planteado para el sistema de gestión con la primer alarma la complejidad de corresponderse con dos de los filtros configurados. En este caso el Filtro que determinará cuál es la alarma resultante del evento es el otorgue el mayor grado de severidad. En este caso El filtro configurado como ***Major*** se impondrá al especificado como ***Warning*** . Entonces la pantalla de monitoreo resultante será la siguiente.

Figura 7-37 Pantalla de Gestión Centinela en Major

El Monitor de Alarmas Activas, reporta la dos alarmas en estado activo, una de ellas, con severidad ***Warning*** es la generada por el evento 1. y la otra , de severidad ***Major*** es la generada por el evento asociado al evento 2. (cambio de estado en el piso 1 sector 1). Esta alarma esta asociada al método de alerta vía e-mail, por lo que se confirma que los mensajes de e-mail son correctamente recibidos por los usuarios ***ua1*** y ***ua2***.

3. Mediante este evento se pretende comprobar que al ser recibido por el sistema ***WBNMS*** y poseer una severidad asociada menor que el estado actual del equipo ***Centinela***, la alarma generada es reportada mediante alertas, pero no se efectúa ningún cambio en el estado del equipo en pantalla.

Figura 7-38 Pantalla de Gestión Centinela en Majos y Minor ignorado

Es comprobado el mensaje de alerta recibido por los usuarios ***ua1*** y ***ua2***.

4. Es generado un evento que generará una alarma de severidad máxima (***Critical***), por lo que el sistema ***WBNMS*** realizará el cambio de estado del equipo en pantalla. Entonces, la pantalla de monitoreo es la siguiente.

Figura 7-39 Pantalla de Gestión Centinela en Critical

Se confirma la recepción de los mensajes de Alertas vía e-mail de los usuarios ***ua2*** y ***ua1***. Además, para este evento, se reporta la alerta vía SNMP al usuario ***ua1***. A continuación es mostrada la pantalla del ***colector de traps*** instalado en la PC en la que se muestra la alerta recibida.

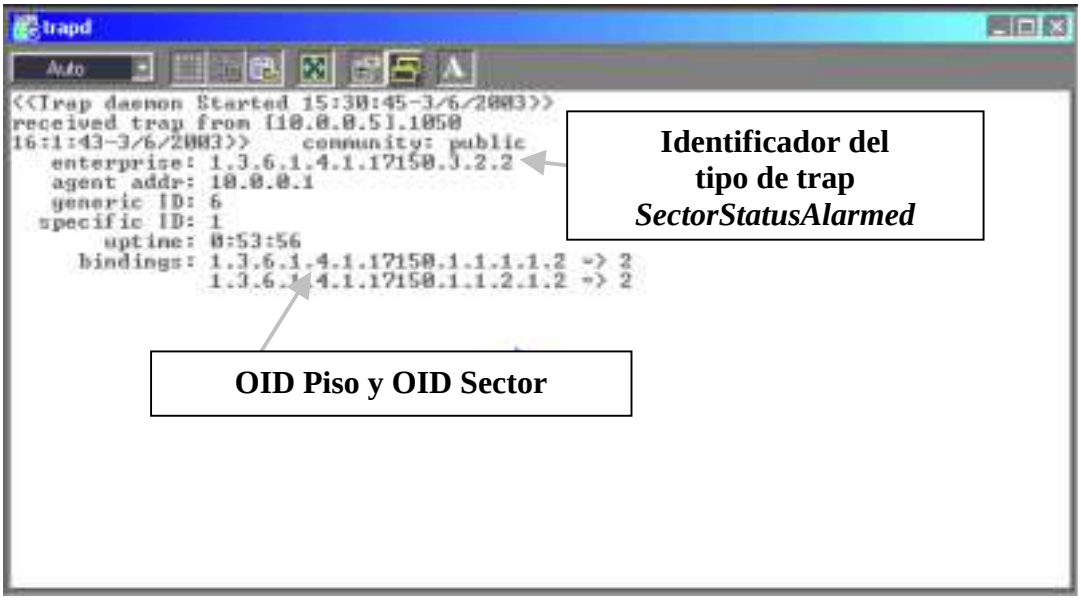


Figura 7-40 Colector de Traps

5. En este caso se procede a realizar el camino inverso, generando los eventos ***SectorStatusOK*** correspondientes , de manera de eliminar los estados de Alarma y llevar al equipo ***Centinela*** al estado inicial. Los resultados de cada paso son ilustrados en la siguiente tabla.

Evento	Descripción	Transición de Estado
SectorStatusOK Piso 2 Sector 2	Este evento reconoce (ACK) la alarma ***SectorStatusAlarmed*** con Piso = 2 y Sector = 2. El estado del equipo es recalculado según las alarmas que aún persisten. Las alertas son reportadas vía e-mail y vía SNMP.	Critical a Major
SectorStatusOK Piso 3 Sector 4	Este evento reconoce (ACK) la alarma ***SectorStatusAlarmed*** con Piso = 3 y Sector = 4. El estado del equipo no cambia ya que existe aún una alarma de severidad mayor. Las alertas son reportadas vía e-mail.	Sin Cambio
SectorStatusOK Piso 1 Sector 1	Este evento reconoce (ACK) la alarma ***SectorStatusAlarmed*** con Piso = 1 y Sector = 1. El estado del equipo es recalculado según las alarmas que aún persisten. Las alertas son reportadas vía e-mail.	Major a Warning
SectorStatusOK Piso 1 Sector 5	El evento no es reconocido como Alarmas por el sistema ya que no fue configurado para ese fin.	Sin Cambio

Tabla 7-10 Resultado de las Pruebas

Para llevar al sistema al estado inicial la alarma que persiste es reconocida (ACK) "manualmente" desde la pantalla de administración de eventos procesados mostrada en la siguiente figura.

Figura 7-41 Vista de las Alarmas Recibidas en la Prueba

Luego de realizar esta acción, el equipo ***Centinela*** se encuentra en el estado inicial (***OK***). A partir de él, se procede a probar las acciones de encuesta y modificación de variables. Para ello se plantean en la siguiente tabla, distintas situaciones representativas.

Situación	Condición	Acciones
Encuesta de estados de los Pisos.	Sin Alarmas Activas	Realizar encuesta en el estado inicial (OK).
	Con Alarma en Piso 4 Sector 3	Generar alarma en el Piso 4 Sector 3 y realizar encuesta.
Modificación del estado de las Llaves remotas (Salidas)	Salida 1 de OFF a ON	Modificar remotamente variable Salida1. Es generado un evento de Alarma con severidad ***Critical***.
	Salida 2 de OFF a ON	Modificar remotamente variable Salida2. Es generado un evento de Alarma con severidad ***Warning***.

Tabla 7-11 Acciones y sus consecuencias

Las acciones de encuesta y modificación son realizadas desde la pantalla de Toma de acciones mostrada en la siguiente figura, mediante la selección de los ***Perfiles de Acción*** adecuados.

Figura 7-42 Pantalla de Toma de Acciones de Gestión

1. Se verifica la primer situación, todos los pisos OK mediante la utilización del perfil de acción ***Consulta de Estado de Pisos***. Los resultados son mostrados en la siguiente figura.

Figura 7-43 Resultado del Perfil de Lectura 1

Luego se genera el cambio de estado en el piso 4 sector 3. Y se repite el proceso de encuesta, obteniendo en este caso la siguiente pantalla.

Figura 7-44 Resultado del Perfil de Lectura 2

En el resultado de esta nueva encuesta es posible ver el cambio de estado en el Piso 4 al producirse la alarma del sector 3.

2. El proceso de modificación es verificado mediante la utilización del perfil de Acción definido como ***Estado de las llaves remotas***. Al ejecutar dicho perfil, primero se realiza un proceso de encuesta que devuelve al usuario los valores actuales de las variables a modificar. Dicha pantalla es mostrada en la siguiente figura.

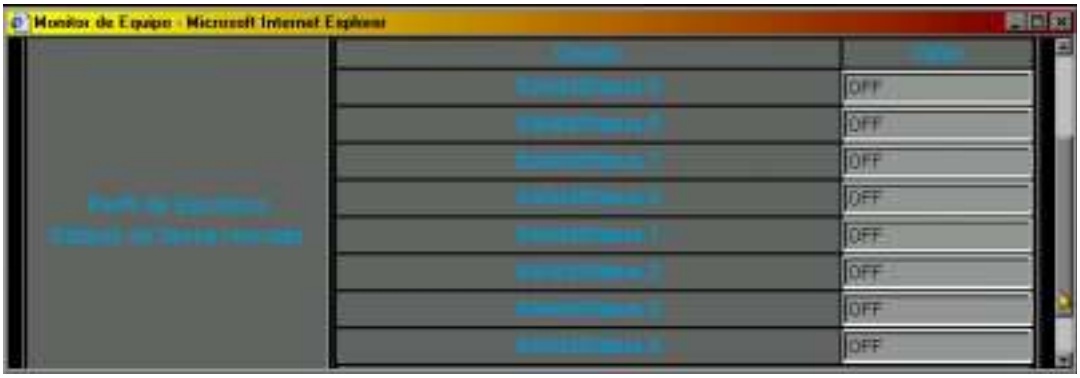


Figura 7-45 Perfil de Escritura

Estos estados se corresponden con el estado actual de las variables en el equipo ***Centinela***, como muestra la interfaz local del mismo (Figura 7-46).

Figura 7-46 Interfaz de salidas del Centinela

Luego se modifican las variables desde la pantalla de la figura 7-45 según la siguiente tabla.

Variables	Objetos.Instancia = estado
Salida 1	SalidaStatus.1 = ON
Salida 2	SalidaStatus.2 = ON

Tabla 7-12 Modificación de Salidas

Estas modificaciones producen, el cambio de estado de las salidas en el equipo ***Centinela***. En la siguiente figura se muestra dicho cambio de estado en la interfaz local de administración.

Figura 7-47 Resultado de la modificación Remota

El equipo ***Centinela***, en respuesta a estos cambios de estado, envía los eventos correspondientes reportándolos. Estos eventos son Alarmas reconocidas y se produce en consecuencia el reporte de las mismas.

7.5 Conclusiones de la Prueba de Sistema

Las pruebas realizadas indican que el sistema es completamente funcional, aunque la falta de memoria hace que tenga una respuesta lenta en presencia de carga sustancial de procesamiento simultáneo. Estos casos se dan ante el requerimiento de procesamiento de múltiples eventos y la situación sería subsanable aumentando la capacidad de memoria. Como se planteo al comienzo de la prueba, no se pretendió probar la respuesta a la carga de procesamiento, por lo que el requisito de memoria del sistema WBNMS no fue respetado, atendiendo a cuestiones de disponibilidad de hardware.
Es de destacar que este funcionamiento bajo carga implica la caída en la velocidad de procesamiento pero no se pierden condiciones de estabilidad, pudiéndose realizar el procesamiento correcto de los datos. Por lo tanto, se probaron las funcionalidades básicas del sistema y se dejó para etapas posteriores las pruebas de performance mediante la implementación de pruebas en campo.

8 Conclusión Final

Las redes de datos han adquirido una importancia crítica y creciente en diversos sectores que comprenden desde aquellos comerciales, como así también entornos gubernamentales o científicos. La constante en todos ellos es la tendencia a redes más grandes con aplicaciones cada vez más complejas y un mayor entorno de usuarios. Esto convierte a la red y sus recursos asociados en protagonistas críticos del desarrollo de determinadas actividades. La complejidad de tales sistemas impone cierto grado de automatización en la gestión, con la dificultad adicional de la diversidad de equipos que conviven bajo el mismo sistema.

La herramientas para controlar este funcionamiento deben presentar una interfaz amigable, sencilla pero potente, al servicio del administrador, como así también un soporte mínimo de software y hardware incorporados en los diversos actores del sistema de gestión. Este sistema debe permitir el monitoreo de la red como un todo pero a la vez debe ser capaz de identificar sus diversos componentes. El protagonista principal o elemento Gestor tiene la capacidad de analizar los datos suministrados por las diversas entidades gestionadas, presentarlos de manera clara y precisa al administrador y permitir el accionar de este sobre los anteriores. Para ellos deberá apoyar su función en una base de datos con toda la información pertinente. Por su lado, las entidades gestionadas poseerán la capacidad adicional de transmitir información acerca de su estado ante requerimientos del Gestor y responder a las solicitudes de acción de este.

El sistema de Gestión WBNMS posee las principales características operativas de un sistema de gestión de Nodos o Elementos de Red. El sistema ofrece un servicio de gestión remota, el cual se ideó inmerso en el contexto de una aplicación Web, a los fines del entorno usuario. Por lo tanto posee las limitaciones inherentes a este tipo de aplicaciones en su interfaz principal. Por este motivo se complementa con la interfaz de Alertas, haciendo del sistema WBNMS una combinación equilibrada entre un sistema de monitoreo en tiempo real y de interfaz Web. Ofrece además la posibilidad de efectuar encuesta y modificación de variables de gestión desde la interfaz de monitoreo, lo que brinda mayor dinamismo en cuestiones operativas.

En este trabajo no han sido desarrollada las cuestiones relacionadas con la interfaz de administración del sistema (Web) ya que es necesario destacar que, pese a la complejidad propia del sistema, es totalmente intuitiva y no requiere conocimientos profundos de Gestión para operarla.

La dimensión de la plataforma de hardware estará supeditada al volumen de los ER (Elementos de Red) gestionados. Esto permite el crecimiento de la plataforma a medida que la gestión lo requiera. El sistema fue probado en una plataforma Intel P1 con un sistema Operativo Linux Mandrake 8 pero debido a la naturaleza multiplataforma del lenguaje de programación utilizado, Perl, y de la plataforma de software, sería perfectamente factible una plataforma SPARC-Solaris (SUN) para casos de alto rendimiento y gran volumen de gestión.

El sistema se basa enteramente en tecnología Open Source, por lo tanto la distribución, utilización y modificación del código fuente de la aplicación es totalmente libre. Solo limitada según las características de una Licencia del tipo GPL(General Public Licence). Bajo estas condiciones, el software y documentación disponibles pueden ser bajados desde el sitio www.wbnms.com.ar .

Respecto de la interfaz humana de monitoreo, se eligió la implementación de una aplicación a través del protocolo estándar de Internet, HTTP (Hyper Text Transfer Protocol) ya que el mismo presenta todas las ventajas de un esquema cliente-servidor,salvando la dificultad que se plantea al mantener la sincronización entre las versiones de software en cliente y servidor, ya que se independiza completamente al cliente de todo tipo de modificaciones realizadas sobre la aplicación servidor y/o firmware del equipo a gestionar, ya que el programa cliente puede ser cualquier navegador web estándar.

Entre las ventajas de este sistema, se podrían mencionar:

- Administración Remota: todas las tareas que pueden realizarse a través de un gestor local pueden ser invocadas en forma remota, evitando la inversión de recursos y tiempo que implica su realización local, sobre todo en el caso de equipos muy distanciados geográficamente o de sistemas con gran densidad de entidades a gestionar.

- Procesamiento Paralelo I: varios clientes pueden conectarse al mismo tiempo al mismo servidor. De este modo se agrega la capacidad de realizar tareas diferentes al mismo tiempo.

- Procesamiento Paralelo II: un cliente puede estar conectado al mismo tiempo a distintos servidores y, por lo tanto, a distintos equipos, permitiéndosele comparar, si fuera necesario, sus parámetros de configuración. Esto se logra mediante diferentes ventanas del navegador.

- Control de Versiones: no es necesario llevar control sobre las versiones instaladas en la aplicación cliente ni preocuparse acerca de cuestiones de compatibilidad entre cliente y servidor, ya que el cliente será un navegador web. Todos los cambios realizados sobre el sistema quedan acotados al servidor.

- Accesibilidad: el Servidor Web está siempre conectado a Internet, cualquier cliente puede acceder a éste, en cualquier momento y desde cualquier lugar del mundo.

- Gestión realizada por el usuario: el operador tiene la posibilidad de permitirle al usuario realizar la gestión de su equipo, un ejemplo de este tipo de prestación es el caso de un sistema de acceso. Algunas de las funciones que el usuario estaría habilitado para realizar pueden ser, chequeo de alarma, estado de la performance del sistema, verificación de los datos de facturación, etc.

- Ampliación de los canales de reportes: los reportes de eventos no sólo pueden ser vistos desde la interfaz web sino que pueden ser direccionados a través de “alertas“ a usuarios predeterminados, ya sea vía e-mail o snmp. Esta funcionalidad permite, por ejemplo, mantener coordinados automáticamente equipos de mantenimiento de respuesta inmediata.

- Bajo costo de implementación : el software de gestión que compone el sistema ***WBNMS v1.0*** es de características Open Source lo que hace que el costo de implementación se reduzca al costo de asistencia técnica.

Respecto d elas limitaciones es posible mencionar:

- SNMP: si bien la mayoría de los equipos de telecomunicaciones que se encuentran hoy en el mercado soporta la gestión por SNMP, todavía existen algunos que no lo hacen. Esta condición los descarta para esta aplicación.

- Interfaz Web: presenta una marcada diferencia con una aplicación en tiempo real ya que en este caso los datos son actualizados a pedido del cliente (relación cliente-servidor) y no ante cambios en la gestión.

Otras alternativas al presente sistema, aunque están basadas también en aplicaciones cliente-servidor, son más complejas porque utilizan clientes propietarios para poder ofrecer una presentación de los datos de gestión en tiempo "real". Debe existir una compatibilidad permanente entre ambas aplicaciones, servidor y cliente, para llevar a cabo las tareas de gestión. De optar por esta alternativa, se debería desarrollar, además del lado servidor, todas las aplicaciones necesarias del lado cliente. En primer lugar, es necesario contar con la aplicación cliente instalada en la máquina desde la que se desea acceder al sistema de gestión (servidor). Sería necesario, además, mantener una actualización constante de la aplicación cliente respetando los cambios que se realizan en el firmware y/o en la aplicación servidor. Además, en caso de que se tengan distintos equipos distribuidos geográficamente, es decir con distintos servidores locales que no necesariamente tengan las mismas versiones de SW o FW, implicaría del lado del cliente la necesidad de contar con tantas versiones de la aplicación como versiones de SW o FW estén presentes.

Al utilizar un navegador web estándar como cliente, se puede acceder de manera flexible desde cualquier punto de una red al sistema de gestión. Si el servidor web posee conexión a Internet, se podría establecer una conexión segura hasta el equipo a gestionar incluso fuera de las redes de telecomunicaciones.
Por otro lado, también se salva la necesidad de actualización de software en los clientes, ya que, de ser necesario sólo se realizarían modificaciones en el servidor. Además, con el mismo cliente se puede acceder simultáneamente a distintos servidores con versiones de software distintas.

Anteriormente se destacó el bajo costo del sistema pero no hay que dejar de destacar que, como todo desarrollo de sistemas complejos, sólo se puede aspirar a una primera versión operativa y a un proceso mejora y desarrollo continuo durante el ciclo de vida del producto.

El sistema WBNMS v1.0 tal como fue planteado al comienzo de su desarrollo sigue siendo un concepto válido para la gestión de nodos y factible de ser ampliado para manejar la gestión de redes y de servicios, gracias a su soporte de base de datos y al diseño modular del software.

Respecto de la interfaz gráfica se puede mencionar que un posible mejora sería su implementación en tecnología flash con actionscripts como lenguaje de

programación, para optimizar de esta manera la interacción con el usuario, debido a características particulares de esta plataforma.

La actualización a SNMP a v3 sería importante de tener en cuenta para brindar una compatibilidad a futuro con equipos nuevos.

De acuerdo al conocimiento acumulado durante el proceso de desarrollo se puede concluir que el sistema presenta actualmente múltiples posibilidades de expansión y mejora, quedando supeditadas a las necesidades del mercado y a los avances de la tecnología. El sistema WBNMS es una más de las múltiples soluciones que existen actualmente para la gestión, pero que aspira desde su núcleo Open Source a contribuir al conocimiento respecto de este tipo de aplicaciones.

9 Código Fuente

9.1 Módulos Centrales (Core)

9.1.1 ABM

9.1.1.1 ABMManager.pm

```
package ABMManager;

use ABMDB 2.1;                  ##Order Option
use ABMFile 1.0;                ##First Version


use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '2.1';                       ## With DBRead function 2.0
                                        ## Order Option 2.1

my %fields = (                  ##Permitted fields
        ABMType => undef,
        MediaName => undef,
        Data => undef,
        DataSearch => undef,
        Keys => undef,
        Order => undef,
        ABM_DB => ABMDB->new(),
        ABM_File => ABMFile->new(),
        Debug => 0,             ## If Debug == 1 then Debug Mode
);

sub new {
        my $proto= shift;
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub CreateInto{
        my $self = shift;
        if($self->ABMType == 1){                        ## option 1_DB
                my $A_DB = $self->ABM_DB->new();
                        $A_DB->Debug($self->Debug);     ## Debug Option
                $A_DB->TableName($self->MediaName);
                $A_DB->Data(\%{$self->Data});
                $A_DB->CreateIntoDB();
        }elsif($self->ABMType == 2){                    ## option 2_File
                my $A_File = $self->ABM_File->new();
                        $A_File->Debug($self->Debug);  ## Debug Option
                $A_File->FileName($self->MediaName);
                $A_File->DataSearch($self->DataSearch);
                $A_File->Data(\@{$self->Data});
                $A_File->CreateIntoFile();
        }

}

sub ModifyInto{
        my $self = shift;
        if($self->ABMType == 1){                        ## option 1_DB
                my $M_DB = $self->ABM_DB->new();
                        $M_DB->Debug($self->Debug);  ## Debug Option
                $M_DB->TableName($self->MediaName);
                $M_DB->DataSearch(\%{$self->DataSearch});
```

```
                $M_DB->Data(\%{$self->Data});
                $M_DB->ModifyIntoDB();
        }elsif($self->ABMType == 2){                    ## option 2_File
                my $M_File = $self->ABM_File->new();
                        $M_File->Debug($self->Debug);  ## Debug Option
                $M_File->FileName($self->MediaName);
                $M_File->DataSearch($self->DataSearch);
                $M_File->Data(\@{$self->Data});
                $M_File->ModifyIntoFile();
        }


}

sub DeleteInto{
        my $self = shift;
        if($self->ABMType == 1){                        ## option 1_DB
                my $B_DB = $self->ABM_DB->new();
                        $B_DB->Debug($self->Debug);  ## Debug Option
                $B_DB->TableName($self->MediaName);
                $B_DB->DataSearch(\%{$self->DataSearch});
                $B_DB->DeleteIntoDB();
        }elsif($self->ABMType == 2){                ## option 2_File
                my $B_File = $self->ABM_File->new();
                        $B_File->Debug($self->Debug);  ## Debug Option
                $B_File->FileName($self->MediaName);
                $B_File->DataSearch($self->DataSearch);
                $B_File->DeleteIntoFile();
        }


}

sub ReadInto{
        my $self = shift;
        if($self->ABMType == 1){                        ## option 1_DB
                my $data_ref;
                my $Read_DB = $self->ABM_DB->new();
                        $Read_DB->Debug($self->Debug);  ## Debug Option
                $Read_DB->TableName($self->MediaName);
                if (defined $self->Keys){
                        $Read_DB->Keys(\@{$self->Keys});
                }
                if (defined $self->Order){
                        $Read_DB->Order($self->Order);
                }
                if (defined $self->DataSearch){
                        $Read_DB->DataSearch(\%{$self->DataSearch});
                        $data_ref =$Read_DB->ReadIntoDB();
                }else{
                        $data_ref =$Read_DB->ReadAllTableIntoDB();
                }
        return($data_ref);
        }


}

sub Read_ComparingInto{
        my $self = shift;
        if($self->ABMType == 1){                        ## option 1_DB
                my $data_ref;
                my $Read_Comp_DB = $self->ABM_DB->new();
                     $Read_Comp_DB->Debug($self->Debug);  ## Debug Option
                $Read_Comp_DB->TableName($self->MediaName);
                if (defined $self->Keys){
                        $Read_Comp_DB->Keys(\@{$self->Keys});
                }
                if (defined $self->Order){
                        $Read_Comp_DB->Order($self->Order);
                }
                if (defined $self->DataSearch){
                        $Read_Comp_DB->DataSearch($self->DataSearch);
                        $data_ref =$Read_Comp_DB->Read_ComparingIntoDB();
                }
        return($data_ref);
```

```
		}


}

sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{                                ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.1.2 ABMFile.pm

```
package ABMFile;

use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;
use strict;

use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';


my %fields = (                          ##Permited fields
		FileName => undef,
		Data => undef,
		DataSearch => undef,
		Debug => 0,                 ## If Debug == 1 then Debug Mode
);

sub new {
		my $proto= shift;                   ##it retrieves the name of the class
		my $class= ref($proto) || $proto;
		my $self = {
			_permited => \%fields,
			%fields,
		};
		bless ($self, $class);
		return $self;
}

sub CreateIntoFile{
		my $self = shift;
		if ( ! _ExistenceChecking($self) ){
			my $filename = $self->FileName;
			my $filenameItem = $self->DataSearch;
			open (INDEX,">>$WbnmsConf::CONFPATH{'File_DB'}/$filename/Index");
                                print INDEX $filenameItem."\n";
			close(INDEX);

			open(ITEM,">$WbnmsConf::CONFPATH{'File_DB'}/$filename/$filenameItem");
			foreach(@{$self->Data}){
                                print ITEM $_."\n";
			}
			close(ITEM);
		}
}

sub ModifyIntoFile{
		my $self = shift;
		if ( _ExistenceChecking($self)){
			my $filename = $self->FileName;
```

```
                my $filenameItem = $self->DataSearch;
                open(ITEM,">$WbnmsConf::CONFPATH{'File_DB'}/$filename/$filenameItem");
                foreach(@{$self->Data}){
                        print ITEM $_."\n";
                }
                close(ITEM);
        }
}

sub DeleteIntoFile{
        my $self = shift;
        if ( _ExistenceChecking($self)){
                my $filename = $self->FileName;
                my $filenameItem = $self->DataSearch;
                open (INDEX,"$WbnmsConf::CONFPATH{'File_DB'}/$filename/Index");
                my @Items;
                while(<INDEX>){
                        chomp($_);
                        if ($_ ne $filenameItem){
                                push(@Items,$_);
                        }
                }
                close(INDEX);
                open (INDEX,">$WbnmsConf::CONFPATH{'File_DB'}/$filename/Index");
                while(<INDEX>){
                        print INDEX $_."\n";
                }
                close(INDEX);

                unlink "$WbnmsConf::CONFPATH{'File_DB'}/$filename/$filenameItem"; ##Only Linux
        }
}

sub _ExistenceChecking{
        my $self = shift;
        my $filename = $self->FileName;
        open (INDEX,"$WbnmsConf::CONFPATH{'File_DB'}/$filename/Index");
        my @Items;
        while(<INDEX>){
                chomp($_);
                push(@Items,$_);
        }
        close(INDEX);
        my $test = grep($_ eq $self->DataSearch,@Items);
        return($test);
}


sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}
 sub DESTROY{                   ## AUTOLOAD search DESTROY sub

 }

1;
```

9.1.1.3 ABMDB.pm

```
package ABMDB;

use DBI;
use strict;
use vars qw($VERSION $AUTOLOAD);  ## Cheking class version
$VERSION = '2.1';               ## With DBRead function 2.0
                                        ## Order Option 2.1
```

```
## Global Variables
my $dbh;
##

my %fields = (                  ##Permited fields
        TableName => undef,
        Data => undef,
        DataSearch => undef,
        Keys => undef,
        Order => undef,
        Debug => 0,                     ## If Debug == 1 then Debug Mode

);

sub new {
        my $proto= shift;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };

        eval {_connectDB()};                                    ##Connecting with DB
        bless ($self, $class);
        return $self;
}

sub CreateIntoDB{
        my $self = shift;
                        print "Create Into: ".$self->TableName."\n" if $self->Debug == 1;
        my ($AllColumns,$AllData) = _prepareData($self,$self->Data);
        my $columns = join(',',@{$AllColumns});
        my $data = join(',',@{$AllData});
                        print "Columns and Data Formated: \n".$columns."\n".$data."\n" if $self->Debug ==
1;

        my $statement= "INSERT INTO ".$self->TableName()." ($columns) VALUES ($data) ";
        ##print $statement."\n";
                my $sth = $dbh->prepare($statement);
                  $sth->execute;
}

sub ModifyIntoDB{
        my $self = shift;
                                print "Modify Into: ".$self->TableName."\n" if $self->Debug == 1;

        my ($AllColumnsSearch,$AllDataSearch) = _prepareData($self,$self->DataSearch);
        my @where;
        foreach(@{$AllColumnsSearch}){
                my $temp= $_."=".shift(@{$AllDataSearch});
                push(@where,$temp);
        }
        my $where = join(" and ",@where);
                                print "Search Condition: ".$where."\n" if $self->Debug == 1;
        my ($AllColumnsData,$AllDataData) = _prepareData($self,$self->Data);
        my @set;
        foreach(@{$AllColumnsData}){
                my $temp= $_."=".shift(@{$AllDataData});
                push(@set,$temp);
        }
        my $set = join(",",@set);
                                print "Set: ".$set."\n" if $self->Debug == 1;
        my $statement= "UPDATE ".$self->TableName()." SET $set WHERE $where ";
                my $sth = $dbh->prepare($statement);
                  $sth->execute;
}

sub DeleteIntoDB{
        my $self = shift;
                                print "Delete from: ".$self->TableName."\n" if $self->Debug == 1;
        my ($AllColumnsSearch,$AllDataSearch) = _prepareData($self,$self->DataSearch);
        my @where;
        foreach(@{$AllColumnsSearch}){
                my $temp= $_."=".shift(@{$AllDataSearch});
                push(@where,$temp);
```

```
		}
		my $where = join(" and ",@where);
                                        print "Search Condition: ".$where."\n" if $self->Debug == 1;

		my $statement= "DELETE FROM ".$self->TableName()." WHERE $where ";
			my $sth = $dbh->prepare($statement);
			  $sth->execute;
}

sub ReadIntoDB{
		my $self = shift;
                                         print "Read from: ".$self->TableName."\n" if $self->Debug == 1;
		my ($AllColumnsSearch,$AllDataSearch) = _prepareData($self,$self->DataSearch);
		my @where;
		foreach(@{$AllColumnsSearch}){
			 my $temp= $_."=".shift(@{$AllDataSearch});
			 push(@where,$temp);
		}
		my $where = join(" and ",@where);
                                       print "Search Condition: ".$where."\n" if $self->Debug == 1;
		my $columns;
		if (defined $self->Keys){
			$columns= join(",",@{$self->Keys});

		}else{
			$columns="*";
		}
		my $orderby;
		if (defined $self->Order){
			$orderby=" ORDER BY ".$self->Order." DESC";

		}
		my $statement= "SELECT ".$columns." FROM ".$self->TableName()." WHERE $where $orderby";
			my @data;
			my $sth = $dbh->prepare($statement);
			  $sth->execute;
			while(my $hash_ref = $sth->fetchrow_hashref){
				push(@data,$hash_ref);
			}

		return(\@data);
}

sub ReadAllTableIntoDB{
		my $self = shift;
                                    print "Read from: ".$self->TableName."\n" if $self->Debug == 1;
		my $columns;
		if (defined $self->Keys){
			$columns= join(",",@{$self->Keys});

		}else{
			$columns="*";
		}
		my $orderby;
		if (defined $self->Order){
			$orderby=" ORDER BY ".$self->Order." DESC";

		}
		my $statement= "SELECT ".$columns." FROM ".$self->TableName().$orderby;
			my $sth = $dbh->prepare($statement);
			  $sth->execute;
			my @data;
			while(my $hash_ref = $sth->fetchrow_hashref){
				push(@data,$hash_ref);
			}

		return(\@data);

}

sub Read_ComparingIntoDB{
		my $self = shift;
                                print "Read from: ".$self->TableName."\n" if $self->Debug == 1;
		my $where = $self->DataSearch;
                                print "Search Condition: ".$where."\n" if $self->Debug == 1;
		my $columns;
```

```
		if (defined $self->Keys){
			$columns= join(",",@{$self->Keys});

		}else{
			$columns="*";
		}
		my $orderby;
		if (defined $self->Order){
			$orderby=" ORDER BY ".$self->Order." DESC";

		}
		my $statement= "SELECT ".$columns." FROM ".$self->TableName()." WHERE ".$where. $orderby;
			my $sth = $dbh->prepare($statement);
			  $sth->execute;
			my @data;
			while(my $hash_ref = $sth->fetchrow_hashref){
				push(@data,$hash_ref);
			}

		return(\@data);

}

sub _prepareData{
		my ($self,$data) = @_;
		my (@AllColumns,@AllData);
		while( my($column,$data)= each(%{$data})){
					print $column.":".$data."\n" if $self->Debug == 1;
			push(@AllColumns,$column);

			$data = "\"".$data."\"";

			push(@AllData,$data);
		}
		return(\@AllColumns,\@AllData);
}


sub _connectDB{
		my $self = shift;
		use lib "/var/sites/Wbnms/Wbnms_Conf";
		use WbnmsConf;
		my $username = $WbnmsConf::CONFDB{'DataBaseUserName'};
		my $password = $WbnmsConf::CONFDB{'DataBasePassword'};
		my $database = $WbnmsConf::CONFDB{'DataBaseName'};
		my $host = $WbnmsConf::CONFDB{'DataBaseHost'};
		my $data_source = "DBI:mysql:$database:$host";
		$dbh = DBI->connect( $data_source, $username, $password);
		return ($dbh);
}


sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{	## AUTOLOAD search DESTROY sub
		$dbh->disconnect;
}

1;
```

9.1.2 trapd

9.1.2.1 trapd.pl

```
#!/usr/bin/perl
#
## This is a trap daemon
## Version: 2
## History: v2- 4.11.2002
## -Linux Version
#
use SNMP_Session;
use SNMP_util;
use Socket;
use POSIX; ##for getpid and localtime
use BER;

#Trapd Configuration
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

#Prototypes
sub print_trap ($$);
sub logger_trap (@);



#Beginning  with the agent's stuff
$port = $WbnmsConf::CONFTRAP{'port'};
my $session = SNMPv1_Session->open_trap_session ($port) or die "couldn't open trap session";
my ($trap, $sender, $sender_port);

#Uptime
$uptime = localtime();

print "<<Trap daemon Started at ".$uptime.">>\n" if $WbnmsConf::CONFTRAP{'Debug'} == 1;

#Write out the PID file
open(PIDFILE, ">$WbnmsConf::CONFTRAP{'pidfile'}");
printf PIDFILE "%d\n", getpid();
close(PIDFILE);

# Main
while (($trap, $sender, $sender_port) = $session->receive_trap ()) {
   print STDERR "received trap from [".inet_ntoa ($sender)."].".$sender_port."\n"; #From Socket Library
   print_trap ($session, $trap);
}

# Subrutines

sub print_trap ($$) {
   my ($this, $trap) = @_;
   my ($encoded_pair, $oid, $value);
   my ($community, $ent, $agent, $gen, $spec, $dt, $bindings)= $this->decode_trap_request ($trap);
   my ($binding, $prefix);
   my @data ;
   my %data ;
   my @Keys;
   # Datetime Mysql Format
   my $time = strftime "%Y-%m-%d %H:%M:%S", localtime;
   push(@data,$time);
   if ( $community eq $WbnmsConf::CONFTRAP{'community'} ) {
            print "    community: $community\n" if $CONFTRAP{'Debug'} == 1;
            push (@data,$community);
            if (defined $ent) {
                     $Manager = ABMManager->new();
                     $Manager->ABMType(1);
                     my $object;
                     if ($gen == 0){         ## coldStart
```

```
                        $object="coldStart";
                }
                elsif ($gen == 1){     ## warmStart
                        $object="warmStart";
                }
                elsif ($gen == 2){     ## linkDown
                        $object="linkDown";
                }
                elsif ($gen == 3){     ## linkUp
                        $object="linkUp";
                }
                elsif ($gen == 4){     ## authenticationFailure
                        $object="authenticationFailure";
                }
                elsif ($gen == 5){     ## egpNeighborLoss
                        $object="egpNeighborLoss";
                }
                elsif ($gen == 6){     ## enterpriseSpecific
                   @Keys = (oid_label);
                   $Manager->Keys(\@Keys);
                   my $oidtested = join('.',(BER::pretty_oid ($ent)),$spec);
                   my %search = ( oid => $oidtested );
                   $Manager->DataSearch(\%search);
                   $Manager->MediaName("Oid2Label");
                   my $data_ref_array=$Manager->ReadInto();
                   my $data_ref= shift(@{$data_ref_array});
                   my @oid_label= values(%{$data_ref});
                   if (!defined $oid_label[0]){$oid_label[0] = $oidtested;}
                   $object = $oid_label[0];
                 }
        %data = (
                time => $time,
                hosts => inet_ntoa ($agent),
                community => $community,
                object => $object,
        );

      ## SNMPv1 Screen and log Trap
        print "   enterprise: ".BER::pretty_oid ($ent)."\n" if $WbnmsConf::CONFTRAP{'Debug'} == 1;
        push (@data,BER::pretty_oid ($ent));
        print "   agent addr: ".inet_ntoa ($agent)."\n" if $WbnmsConf::CONFTRAP{'Debug'} == 1;
        push (@data,inet_ntoa ($agent));
        print "   generic ID: $gen\n" if $WbnmsConf::CONFTRAP{'Debug'} == 1;
        push (@data,$gen);
        print "  specific ID: $spec\n" if $WbnmsConf::CONFTRAP{'Debug'} == 1;
        push (@data,$spec);
        print "       uptime: ".BER::pretty_uptime_value ($dt)."\n" if $WbnmsConf::CONFTRAP{'Debug'} == 1;
        push (@data,BER::pretty_uptime_value ($dt));
      }
      ## Otherwise we have an SNMPv1 Trap which basically just
      ## consists of bindings.
      $prefix = "     bindings: ";
      $countflag = 1;

      while ($bindings) {
        ($binding,$bindings) = decode_sequence ($bindings);
        ($oid,$value) = decode_by_template ($binding, "%O%@");
        my @oidtest = split(/\./,BER::pretty_oid ($oid));
        my $lastnum= pop(@oidtest);
        my $oidtestedbindings;
        if ($lastnum == 0){
                $oidtestedbindings = join('.',@oidtest);
        }else{
                $oidtestedbindings = join('.',@oidtest).".".$lastnum;
        }
        my %search = ( oid => $oidtestedbindings );
        $Manager->DataSearch(\%search);
        my $data_ref_array=$Manager->ReadInto();
        my $data_ref = shift(@{$data_ref_array});
        my @oid_label= values(%{$data_ref});
         if (!defined $oid_label[0]){$oid_label[0] = BER::pretty_oid ($oid);}
        $data{'variable'.$countflag}= $oid_label[0];
        $data{'value'.$countflag}= pretty_print ($value);

        print $prefix.BER::pretty_oid ($oid)." => ".pretty_print ($value)."\n" if
$WbnmsConf::CONFTRAP{'Debug'} == 1;
```

```
                push (@data,BER::pretty_oid ($oid));
                push (@data,pretty_print ($value));
                $prefix = "                ";
                $countflag ++;
            }

            ## Save data TrapsCollector Table
            $Manager->MediaName("TrapsCollector");
            $Manager->Data(\%data);
            $Manager->CreateInto();
    } else {
            warn "decoding trap request failed:\n".$SNMP_Session::errmsg;
    }
    #foreach (@data){print $_;} debug
    &logger_trap(@data);
}

sub logger_trap(@){
            my (@data_to_log) = @_;
            open (TRAPLOG,">>$WbnmsConf::CONFTRAP{'logpath'}/$WbnmsConf::CONFTRAP{'trapfile.log'}");
            foreach(@data_to_log){
                        print TRAPLOG "$_:";
            }
                  print TRAPLOG "\n";
            close(TRAPLOG);
}
```

9.1.2.2 trapd.rc

```
#!/bin/sh
#
# trapd.rc        Start trap daemon of WBNMS.
#

case "$1" in
  start)
            echo "Starting trapd"
            /var/sites/Wbnms/Wbnms_Core/trapd/start
            ;;
  stop)
            echo "Stopping trapd"
            /var/sites/Wbnms/Wbnms_Core/trapd/stop
            ;;
  status)
            if ! [ -f /var/sites/Wbnms/Wbnms_Core/trapd/trapd.pid ]
            then echo "trapd.pl is stopped"
                exit 3
            fi
            pid=`cat /var/sites/Wbnms/Wbnms_Core/trapd/trapd.pid`
            kill -0 $pid >/dev/null 2>&1
            if [ $? == 0 ]
            then echo "trapd.pl (pid $pid) is running..."
                exit 0
            fi
            echo "trapd.pl is stopped"
            exit 3
            ;;
  restart)
            $0 stop
            $0 start
            ;;
  reload)
            $0 stop
            $0 start
            ;;

*)
            echo "Usage: $0 { start | stop | status | restart }"
            exit 1
            ;;
esac

exit 0
```

9.1.2.3 start

```
#!/bin/sh
echo Starting trapd daemon
trap '' 1
perl /var/sites/Wbnms/Wbnms_Core/trapd/trapd.pl &
```

9.1.2.4 stop

```
#!/bin/sh
echo Stopping trapd daemon
kill -9 `cat /var/sites/Wbnms/Wbnms_Core/trapd/trapd.pid`
perl /var/sites/Wbnms/Wbnms_Core/trapd/renametraplog.pl &
```

9.1.2.5 renametraplog.pl

```
#!/usr/bin/perl

use Time::localtime;
#
##mv trapd log file
#
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

$uptime = localtime->sec().localtime->min().localtime->hour().localtime->mday().localtime->mon();
rename ($WbnmsConf::CONFTRAP{'logpath'}."/".$WbnmsConf::CONFTRAP{'trapfile.log'} ,
$WbnmsConf::CONFTRAP{'logpath'}."/trapfile$uptime.log");
```

9.1.3 keepalive

9.1.3.1 keepalive.pl

```
#!/usr/bin/perl
#
## This is a keepalive daemon
## Version: 1
## History: v1- 11.03.2003
#

use lib "/var/sites/Wbnms/Wbnms_Core/keepalive";
use snmppollerManager;

use SNMP_util;
use POSIX; ##for getpid and localtime


#Trapd Configuration
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object

use lib "/var/sites/Wbnms/Wbnms_Core/ABM";  ## Must by configured at installation time
use ABMManager 2.0;

#Prototypes
sub keepalive ($);
sub iprange ($);

#Uptime
$uptime = localtime();
#Variables
my $maxmanagersnum = 5; ## 5 recomended
```

```
$SIG{CHLD} = 'IGNORE';
####
print "<<KeepAlive daemon Started at ".$uptime.">>\n" if $WbnmsConf::KEEPALIVE{'Debug'} == 1;

#Write out the PID file
open(PIDFILE, ">$WbnmsConf::KEEPALIVE{'pidfile'}");
printf PIDFILE "%d\n", getpid();
close(PIDFILE);

# Main
while(1){
        &keepalive();
        if (defined $WbnmsConf::KEEPALIVE{'checkinterval'}){
                $interval = $WbnmsConf::KEEPALIVE{'checkinterval'};
        }else{
                $interval = 900; ## Default value
        }
        sleep($interval);
        #&keepalive();
}
# Subrutines

sub keepalive ($) {
   my ($this) = @_;
   ###Test Equipments Status
   my $type = 0;
   my $Manager = ABMManager->new();
   $Manager->ABMType(1);
   $Manager->MediaName("Equipment");
   my $EquipmentsData= $Manager->ReadInto();
   my @existips;
   print "<< ".scalar(@{$EquipmentsData})." Already Registered Equipments>>\n" if
$WbnmsConf::KEEPALIVE{'Debug'} == 1;
   my @snmps;
   my $count =1;
   my $snmppermanager = int(scalar(@ips)/$maxmanagersnum) || 2 ;
   my $laps=int(scalar(@ips)/$snmppermanager)+1;
   my $limit=scalar(@ips)-$snmppermanager*int(scalar(@ips)/$snmppermanager)-1;
   foreach my $equipment (@{$EquipmentsData}){
            push(@existips,${$equipment}{'IP'});

           if (scalar(@snmps) <= ($snmppermanager - 1) or (scalar(@snmps) <= $limit and $count == $laps)){
                   my $snmppar= ${$equipment}{'community'}."\@".${$equipment}{'IP'};
                   push(@snmps,$snmppar);
            }
            if(scalar(@snmps) > ($snmppermanager - 1) or (scalar(@snmps) > $limit and $count == $laps)){

                   my @snmpstest;
                   push(@snmpstest,@snmps);
                   undef @snmps;
                   $count ++;
                   my $pid = fork();
                   if ($pid == 0){
                           my $pollerManager = snmppollerManager->new();
                           $pollerManager->snmppollerManager(\@snmpstest,$type);
                           exit(0);
                   }else{
                           waitpid($pid, 0);

                   }
            }
   }

   ###Scan ip range
   my $type = 1;
   my @ips = &iprange;
   print "<< ".scalar(@ips)." Search New Equipments>>\n" if $WbnmsConf::KEEPALIVE{'Debug'} == 1;
   my @newips;
   my @snmps;
   my $count =1;
   my $snmppermanager = int(scalar(@ips)/$maxmanagersnum) || 2 ;
   my $laps=int(scalar(@ips)/$snmppermanager)+1;
   my $limit=scalar(@ips)-$snmppermanager*int(scalar(@ips)/$snmppermanager)-1;
   foreach my $ip (@ips){
           if (0 == grep($_ eq $ip,@existips)){
              push(@newips,$ip);
```

```
                if (scalar(@snmps) <= ($snmppermanager - 1) or (scalar(@snmps) <= $limit and $count == $laps)){
                        my $snmppar=
$WbnmsConf::KEEPALIVE{'community'}."\@".$ip.":".$WbnmsConf::KEEPALIVE{'port'}.":".$WbnmsConf::KEEPA
LIVE{'timeout'}.":".$WbnmsConf::KEEPALIVE{'retries'};
                        push(@snmps,$snmppar);
                }
                if(scalar(@snmps) > ($snmppermanager - 1) or (scalar(@snmps) > $limit and $count == $laps)){

                        my @snmpstest;
                        push(@snmpstest,@snmps);
                        undef @snmps;
                        $count ++;
                        my $pid = fork();
                        if ($pid == 0){
                                my $pollerManager = snmppollerManager->new();
                                $pollerManager->snmppollerManager(\@snmpstest,$type);
                                exit(0);
                        }else{
                                waitpid($pid, 0);

                        }
                }

        }
   }
}

sub iprange ($){
        my ($this) = @_;
        my @startscan = split(/\./,$WbnmsConf::KEEPALIVE{'startscan'});
        my @endscan = split(/\./,$WbnmsConf::KEEPALIVE{'endscan'});
        my @ips;
        for my $D ($startscan[0]..$endscan[0]){
                for  my $C ($startscan[1]..$endscan[1]){
                        for my $B ($startscan[2]..$endscan[2]){
                                for my $A ($startscan[3]..$endscan[3]){
                                        push(@ips,$D."\.".$C."\.".$B."\.".$A);
                                }
                        }
                }
        }
return(@ips);
}


sub logger_keepalive(@){
        my (@data_to_log) = @_;
        open
(KEEPLOG,">>$WbnmsConf::KEEPALIVE{'logpath'}/$WbnmsConf::KEEPALIVE{'keepalivefile.log'}");
        foreach(@data_to_log){
                print KEEPLOG "$_;";
        }
              print KEEPLOG "\n";
        close(KEEPLOG);
return(1);
}
```

9.1.3.2 keepalive.rc

```
#!/bin/sh
#
# keepalive.rc       Start keepalive daemon of WBNMS.
#

case "$1" in
  start)
        echo "Starting keepalive"
        /var/sites/Wbnms/Wbnms_Core/keepalive/start
        ;;
  stop)
        echo "Stopping keepalive"
        /var/sites/Wbnms/Wbnms_Core/keepalive/stop
        ;;
```

```
 status)
            if ! [ -f /var/sites/Wbnms/Wbnms_Core/keepalive/keepalive.pid ]
            then echo "keepalive.pl is stopped"
                exit 3
            fi
            pid=`cat /var/sites/Wbnms/Wbnms_Core/keepalive/keepalive.pid`
            kill -0 $pid >/dev/null 2>&1
            if [ $? == 0 ]
            then echo "keepalive.pl (pid $pid) is running..."
                exit 0
            fi
            echo "keepalive.pl is stopped"
            exit 3
            ;;
 restart)
            $0 stop
            $0 start
            ;;
 reload)
            $0 stop
            $0 start
            ;;

*)
            echo "Usage: $0 { start | stop | status | restart }"
            exit 1
            ;;
esac

exit 0
```

9.1.3.3 start

```
#!/bin/sh
echo Starting keepalive daemon
trap " 1
perl /var/sites/Wbnms/Wbnms_Core/keepalive/keepalive.pl &
```

9.1.3.4 stop

```
#!/bin/sh
echo Stopping keepalive daemon
kill -9 `cat /var/sites/Wbnms/Wbnms_Core/keepalive/keepalive.pid`
perl /var/sites/Wbnms/Wbnms_Core/keepalive/renamekeepalivelog.pl &
```

9.1.3.5 renamekeepalivelog.pl

```
!/usr/bin/perl

use Time::localtime;
#
##mv keepalive log file
#
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

$uptime = localtime->sec().localtime->min().localtime->hour().localtime->mday().localtime->mon();
rename ($WbnmsConf::KEEPALIVE{'logpath'}."/".$WbnmsConf::KEEPALIVE{'keepalivefile.log'} ,
$WbnmsConf::KEEPALIVE{'logpath'}."/keepalivefile$uptime.log");
```

9.1.3.6 PollManager.pm

```
package PollManager;

use strict;
use vars qw($VERSION $AUTOLOAD);##Cheking class version
$VERSION = '1.0';
```

```
#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;

my %fields = (      ##Permited fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto)= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub ReadEquipments(){
        my ($self) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->MediaName("NewEquipment");
        $Manager->Keys(undef);
        my $NewEquipmentData= $Manager->ReadInto();
        ##Read Types
      $Manager->MediaName("EquipmentTypes");
        $Manager->DataSearch(undef);
        $Manager->Keys(undef);
        my $EquipmentTypes= $Manager->ReadInto();
        my %types;
        foreach my $type(@{$EquipmentTypes}){
                $types{${$type}{'equipmenttype_id'}}= ${$type}{'equipmenttype_label'};
        }
return($NewEquipmentData,\%types);
}

sub AddEquipment($){
        my ($self,$data) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        ##Add into Equipment Table
        $Manager->MediaName("Equipment");
        $Manager->Data($data);
        $Manager->CreateInto();
        ##Read data from Table
        $Manager->DataSearch($data);
        $Manager->Keys(undef);
        my $data_ref_array = $Manager->ReadInto();
        my $equipment_id=${$data_ref_array}[0]{'equipment_id'};
        ##Create into table EquipmentMonitor
        $Manager->MediaName("EquipmentMonitor");
        my %statedata =(
                                'equipment_id'=>$equipment_id,
                                'state_id'=> 0,
                                'changed' => 1,
        );
        $Manager->Data(\%statedata);
        $Manager->CreateInto();
        ##Delete from NewEquipment Table
        my %deldata=('IP'=>${$data}{'IP'},);
        $Manager->MediaName("NewEquipment");
        $Manager->DataSearch(\%deldata);
        $Manager->DeleteInto();
}

sub DelEquipment($){
```

```
		my ($self,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Delete Item from NewEquipment table
		$Manager->MediaName("NewEquipment");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
return($output);
}

sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.3.7 snmppollerManager.pm

```
package snmppollerManager;

use snmppoller;
use strict;
use vars qw($VERSION $AUTOLOAD);##Cheking class version
$VERSION = '1.0';

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

my %fields = (  ##Permited fields
		Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
		my ($proto )= @_;  #it retrieves the name of the class
		my $class= ref($proto) || $proto;
		my $self = {
				_permited => \%fields,
				%fields,
		};
		bless ($self, $class);
		return $self;
}

sub snmppollerManager($$){
		my ($self,$snmps,$type)= @_;
		my $output = 0;
		foreach my $snmppar(@{$snmps}){
				my $pid = fork();
				if ($pid == 0){
						my $poller = snmppoller->new();
						$output = $poller->snmppoll($snmppar,$type);
						exit(0);
				}
#				else{
#						waitpid($pid, 0);
#
#				   }
}

return($output);
}
```

```
sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}
1;
```

9.1.3.8 snmppoller.pm

```
package snmppoller;

use SNMP_util;
use POSIX; ##for getpid and localtime
use strict;
use vars qw($VERSION $AUTOLOAD);##Cheking class version
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;



my %fields = (  ##Permitted fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto )= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub snmppoll($$){
        my ($self,$snmppar,$type)= @_;
        my $output = 0;
        my ($sysName) = &snmpget( $snmppar ,"1.3.6.1.2.1.1.5.0"); ##sysName
        if ($type == 0){
                if ($sysName){
                        $output = 1;
                }else{                  ## failure Report like linkDown trap
                        my ($community,$IP) = split(/\@/,$snmppar);
                        my $time = strftime "%Y-%m-%d %H:%M:%S", localtime;
                        my %data = (
                                                time => $time,
                                                hosts => $IP,
                                                community => $community,
                                                object => "linkDown",
                                                variable1 => "Encuesta",
                                                value1 => "Automatica",
                                        );
                        my $Manager = ABMManager->new();
                    $Manager->ABMType(1);
```

```
                        ## Save data TrapsCollector Table
                            $Manager->MediaName("TrapsCollector");
                            $Manager->Data(\%data);
                            $Manager->CreateInto();
                            $output = 0;
                    }
            }
            if($type == 1){
                    if ($sysName){
                            $output = 1;
                            my ($community,$SeudoIP) = split(/\@/,$snmppar);
                            my ($IP,@temp)=split(/:/,$SeudoIP);
                            my %data = (
                                            equipment => $sysName,
                                            IP => $IP,
                                            community => $community,
                                        );
                            my $Manager = ABMManager->new();
                        $Manager->ABMType(1);
                        ## Save data TrapsCollector Table
                            $Manager->MediaName("NewEquipment");
                            $Manager->Data(\%data);
                            $Manager->CreateInto();
                    }else{
                            $output = 0;
                    }

            }


return($output);
}


sub AUTOLOAD{
            my $self = shift;
            my $type = ref($self) || die;
            my $name = $AUTOLOAD;
            $name =~ s/.*://;
            unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
            if(@_){$self->{$name}= shift}
            return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}
1;
```

9.1.4 GetSet

9.1.4.1 GetSetManager.pm

```
package GetSetManager;

use SNMP_util;
use strict;
use vars qw($VERSION $AUTOLOAD);##Cheking class version
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

#Call Error Manager
use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;
```

```
my %fields = (  ##Permitted fields
        IP=>undef,
        community=>undef,
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto )= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub ReadEquipments(){
        my ($self)= @_;
        my (@Equipmets,%labels);
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->MediaName("Equipment");
        my $Data= $Manager->ReadInto();
        foreach my $Equip (@{$Data}){
                 push(@Equipmets,${$Equip}{'equipment_id'});
                 $labels{${$Equip}{'equipment_id'}}=${$Equip}{'equipment'};
                }
return(\@Equipmets,\%labels);
}

sub ReadProf($){
        my ($self,$equipmenttype_id)= @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->MediaName("EquipmentTypeGetSet,GetSet");
        my $searchfile = "EquipmentTypeGetSet.equipmenttype_id = \"".$equipmenttype_id."\" and
EquipmentTypeGetSet.getset_id = GetSet.getset_id";
        $Manager->DataSearch($searchfile);
        $Manager->Keys(undef);
        my $Profiles= $Manager->Read_ComparingInto();
        my (%GetSetProfile,@ReadP,@ReadWriteP,%labelsIds);
        foreach my $Profile (@{$Profiles}){
                if (${$Profile}{'readwrite'} == 0){push(@ReadP,${$Profile}{'getset_id'})} ## Read Only
                if (${$Profile}{'readwrite'} == 1){push(@ReadWriteP,${$Profile}{'getset_id'})}        ##
Read&Write
                $labelsIds{${$Profile}{'getset_id'}}=${$Profile}{'getset_label'};
                my @ProfArray = split(/:/,${$Profile}{'getset_comp'});
                $GetSetProfile{${$Profile}{'getset_id'}}=\@ProfArray;
        }
return(\%GetSetProfile,\@ReadP,\@ReadWriteP,\%labelsIds);
}

sub ReadAllProfiles($$$){
        my ($self,$R,$WR,$Data)=@_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->MediaName("GetSet");
        $Manager->DataSearch($Data);
        my $Data= $Manager->ReadInto();
        my (@AllProfiles,%ProfilesLabels);
        foreach my $prof (@{$Data}){
                if (0 == grep($_ == ${$prof}{'getset_id'},(@{$R},@{$WR}))){
                        push(@AllProfiles,${$prof}{'getset_id'});
                        $ProfilesLabels{${$prof}{'getset_id'}}=${$prof}{'getset_label'};
                        if (${$prof}{'readwrite'} == 0){${$prof}{'readwrite_label'}="Solo Lectura";}
                        if (${$prof}{'readwrite'} == 1){${$prof}{'readwrite_label'}="Lectura/Escritura";}
                        my @Comps=split(/:/,${$prof}{'getset_comp'});
                        ${$prof}{'getset_comp'}=\@Comps;
                }
        }

return(\@AllProfiles,\%ProfilesLabels,$Data);
}

sub AddComp($$){
```

```
		my ($self,$getset_id,$Comp)=@_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("GetSet");
		my %Data=('getset_id'=>$getset_id,);
		$Manager->DataSearch(\%Data);
		my $Data= $Manager->ReadInto();
		my @Comps=split(/:/,${$Data}[0]{'getset_comp'});
		my $newcomps=join(':',(@Comps,$Comp));
		my %NewData=('getset_comp'=>$newcomps,);
		$Manager->Data(\%NewData);
		$Manager->ModifyInto();

return(1);
}

sub DelComp($$){
		my ($self,$getset_id,$Comp)=@_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("GetSet");
		my %Data=('getset_id'=>$getset_id,);
		$Manager->DataSearch(\%Data);
		my $Data= $Manager->ReadInto();
		my @Comps=split(/:/,${$Data}[0]{'getset_comp'});
		my $newcomps=join(':',grep($_ != $Comp,@Comps));
		my %NewData=('getset_comp'=>$newcomps,);
		$Manager->Data(\%NewData);
		$Manager->ModifyInto();

return(1);
}
sub CreateProf($){
		my ($self,$data)= @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("GetSet");
		$Manager->Data($data);
		$Manager->CreateInto();
return(1);
}

sub UnCreateProf($){
		my ($self,$data)= @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("GetSet");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
return(1);
}

sub AddProf($){
		my ($self,$equipmenttype_id,$getset_id)= @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("EquipmentTypeGetSet");
		my %data=('equipmenttype_id'=>$equipmenttype_id,'getset_id'=>$getset_id,);
		$Manager->Data(\%data);
		$Manager->CreateInto();
return(1);
}

sub DelProf($){
		my ($self,$equipmenttype_id,$getset_id)= @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("EquipmentTypeGetSet");
		my %data=('equipmenttype_id'=>$equipmenttype_id,'getset_id'=>$getset_id,);
		$Manager->DataSearch(\%data);
		$Manager->DeleteInto();
return(1);
}

sub ReadEquipmentData($){
		my ($self,$equipment_id)= @_;
```

```
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("Equipment");
		my %search=('equipment_id'=>$equipment_id,);
		$Manager->DataSearch(\%search);
		$Manager->Keys(undef);
		my $Data= $Manager->ReadInto();
		$self->IP(${$Data}[0]{'IP'});
		$self->community(${$Data}[0]{'community'});
return($Data);
}

sub Get($){
		my ($self,$data)= @_;
		&_Prepare($self,$data);
		my $snmppar= $self->community()."\@".$self->IP();
		my %ObjData;
		foreach my $obj (@{$data}){
			($ObjData{$obj})= &snmpget( $snmppar ,$obj);
			if (!$ObjData{$obj}){
					my $Error = ErrorManager->new("No Hay Respuesta del Equipo");
					print $Error->Display();
			}
		}

return(\%ObjData);
}

sub Set($$){
		my ($self,$objs,$data)= @_;
		my $ObjType=&_Prepare($self,$objs);
		my $snmppar= $self->community()."\@".$self->IP();
		my %ObjData;
		foreach my $obj (@{$objs}){
			($ObjData{$obj}) = &snmpset( $snmppar , $obj , ${$ObjType}{$obj} , ${$data}{$obj}  );
		}

return(\%ObjData);
}


sub _Prepare($){
		my ($self,$data)=@_;
		my %Objtype;
		foreach my $obj (@{$data}){

				my $Manager = ABMManager->new();
				$Manager->ABMType(1);
				$Manager->MediaName("Oid2Label");
				my ($objbase,$elseobj);
				my @objcompl=split(/\./,$obj);
				if(scalar(@objcompl)>1){

					$objbase=shift(@objcompl);
					$elseobj=join('.',@objcompl);
				}else{
					$objbase= $obj;
				}

				my %search=('oid_label'=>$objbase,);
				$Manager->DataSearch(\%search);
				my $Data= $Manager->ReadInto();
				if (!defined $elseobj){
					$elseobj= 0 ;## And Instance 0
					$SNMP_util::OIDS{$objbase}= join('.',(${$Data}[0]{'oid'},$elseobj));
				}else{
					$SNMP_util::OIDS{$objbase}=${$Data}[0]{'oid'};
				}
				$Objtype{$obj}=${$Data}[0]{'syntax'};
		}
return(\%Objtype);
}


sub AUTOLOAD{
		my $self = shift;
```

```
	my $type = ref($self) || die;
	my $name = $AUTOLOAD;
	$name =~ s/.*://;
	unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
	if(@_){$self->{$name}= shift}
	return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.5 Status

9.1.5.1 StatusChecker.pm

```
package StatusChecker;


use strict;
use vars qw($VERSION $AUTOLOAD);##Cheking class version
$VERSION = '1.0';     ##

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use DBI;

my %fields = (  ##Permited fields
	Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
	my ($proto)= @_;  #it retrieves the name of the class
	my $class= ref($proto) || $proto;
	my $self = {
		_permited => \%fields,
		%fields,
	};
	bless ($self, $class);
	return $self;
}

sub CheckStatus(){
	my ($self)= @_;
	my (%testAlarmSd,%testtrapd,%testDrawerd,%testkeepalive,%testDB);
	$testAlarmSd{'AlarmSurveillance'} = 1;
	$testtrapd{'TrapDaemon'} = 1;
	$testDrawerd{'DrawerDaemon'} = 1;
	$testkeepalive{'PollDaemon'} = 1;
	$testDB{'MysqlDB'} = 1;
	my ($pid1,$pid2,$pid3,$pid4);
	opendir(PROC,"/proc"); ## Open /proc directory
	my @process = grep(!/^\./,readdir(PROC));
	closedir(PROC);

	open(PID,"$WbnmsConf::CONFALARMSUR{'pidfile'}");
	$pid1=<PID>;
	close(PID);
	unless  (grep($_ == $pid1 ,@process)) {
		$testAlarmSd{'AlarmSurveillance'} = 0; ##Stopped
	}

	open(PID,"$WbnmsConf::CONFTRAP{'pidfile'}");
	$pid2=<PID>;
	close(PID);
	unless  (grep($_ == $pid2 ,@process)) {
		$testtrapd{'TrapDaemon'} = 0; ##Stopped
	}
```

```
		open(PID,"$WbnmsConf::CONFDRAWER{'pidfile'}");
		$pid3=<PID>;
		close(PID);
		unless  (grep($_ == $pid3 ,@process)) {
			$testDrawerd{'DrawerDaemon'} = 0; ##Stopped
		}

		open(PID,"$WbnmsConf::KEEPALIVE{'pidfile'}");
		$pid4=<PID>;
		close(PID);
		unless  (grep($_ == $pid4 ,@process)) {
			$testkeepalive{'PollDaemon'} = 0; ##Stopped
		}

		&_connectDB();
		if(defined $DBI::errstr){
			$testDB{'MysqlDB'} = 0; ##Stopped
		}


return(\%testAlarmSd,\%testtrapd,\%testDrawerd,\%testkeepalive,\%testDB);
}

sub Start($){
		my ($self,$daemon)= @_;
		if ($daemon eq "AlarmSurveillance"){
			chdir("$WbnmsConf::CONFPATH{'AlarmSurveillance'}");
		}
		if ($daemon eq "TrapDaemon"){
			chdir("$WbnmsConf::CONFPATH{'trapd'}");
		}
		if ($daemon eq "DrawerDaemon"){
			chdir("$WbnmsConf::CONFDRAWER{'DrawerCorepath'}");
		}
		if ($daemon eq "PollDaemon"){
			chdir("$WbnmsConf::CONFPATH{'keepalive'}");
		}
		if (defined $daemon){
			system("./start 1>/dev/null");
		}
}

sub Stop($){
		my ($self,$daemon)= @_;
		if ($daemon eq "AlarmSurveillance"){
			chdir("$WbnmsConf::CONFPATH{'AlarmSurveillance'}");
		}
		if ($daemon eq "TrapDaemon"){
			chdir("$WbnmsConf::CONFPATH{'trapd'}");
		}
		if ($daemon eq "DrawerDaemon"){
			chdir("$WbnmsConf::CONFDRAWER{'DrawerCorepath'}");
		}
		if ($daemon eq "PollDaemon"){
			chdir("$WbnmsConf::CONFPATH{'keepalive'}");
		}
		if (defined $daemon){
			system("./stop 1>/dev/null");
		}
}

sub _connectDB{
		my $self = shift;
		my $username = $WbnmsConf::CONFDB{'DataBaseUserName'};
		my $password = $WbnmsConf::CONFDB{'DataBasePassword'};
		my $database = $WbnmsConf::CONFDB{'DataBaseName'};
		my $host = $WbnmsConf::CONFDB{'DataBaseHost'};
		my $data_source = "DBI:mysql:$database:$host";
		my $dbh = DBI->connect( $data_source, $username, $password,{PrintError => 0});


}

sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
```

```
	my $name = $AUTOLOAD;
	$name =~ s/.*://;
	unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
	if(@_){$self->{$name}= shift}
	return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.5.2 DaemonsLogViewer.pm

```
package DaemonsLogViewer;


use strict;
use vars qw($VERSION $AUTOLOAD);##Cheking class version
$VERSION = '1.0';     ##

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;


my %fields = (  ##Permited fields
	Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
	my ($proto)= @_;  #it retrieves the name of the class
	my $class= ref($proto) || $proto;
	my $self = {
		_permited => \%fields,
		%fields,
	};
	bless ($self, $class);
	return $self;
}
sub IdentifyComponents($){
	my ($self,$daemon)= @_;
	my %daemonlogdir;
	if($daemon eq "AlarmSurveillance"){
		$daemonlogdir{'AlarmSurveillance'} = $WbnmsConf::CONFALARMSUR{'logpath'};
		$daemonlogdir{'Mailer'}= $WbnmsConf::CONFMAILER{'logpath'};
	}
	if($daemon eq "TrapDaemon"){
		$daemonlogdir{'TrapDaemon'} = $WbnmsConf::CONFTRAP{'logpath'};
	}
	if($daemon eq "DrawerDaemon"){
		$daemonlogdir{'Drawerdaemon'}= "NOT";##not available
	}
	if($daemon eq "MysqlDB"){
		$daemonlogdir{'MysqlDB'}="NOT"; ##not available
	}
	if($daemon eq "PollDaemon"){
		$daemonlogdir{'PollDaemon'}="NOT"; ##not available
	}

return(\%daemonlogdir);
}

sub ReadLogs($){
	my ($self,$daemon)= @_;
	my $daemonlogdir = &IdentifyComponents($self,$daemon);
	my %daemonlogs;
	while(my ($daemoncomp,$daemonDir) = each(%{$daemonlogdir})){
		my @logs;
		if($daemonDir ne "NOT"){
		opendir(LOGS,$daemonDir);
		@logs = grep(!/^\./,readdir(LOGS));
```

```
                        closedir(LOGS);
                        }else{
                        @logs = ("No disponible");
                        }
                        $daemonlogs{$daemoncomp}=\@logs;
                }
return(\%daemonlogs,$daemonlogdir);
}

sub OpenLog($$){
                my ($self,$path,$log)= @_;
                open(LOG,"$path/$log");
                my @logtxt;
                while(<LOG>){
                        chomp();
                        my @logline = split(/;/,$_);
                        push(@logtxt,\@logline);
                }
                close(LOG);
return(\@logtxt);
}

sub DelLog($$){
                my ($self,$path,$log)= @_;
                my $deleted = unlink("$path/$log");
return($deleted);
}


sub AUTOLOAD{
                my $self = shift;
                my $type = ref($self) || die;
                my $name = $AUTOLOAD;
                $name =~ s/.*://;
                unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
                if(@_){$self->{$name}= shift}
                return $self->{$name};
}
 sub DESTROY{ ## AUTOLOAD search DESTROY sub

 }

1;
```

9.1.6 Auth

9.1.6.1 Auth.pm

```
package Auth;

use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';


use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;


my %fields = (  ##Permited fields
                Name => undef,
                Password => undef,
                CheckPass => undef,
                Group => undef,
                Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
                my $proto= shift;  #it retrieves the name of the class
                my $class= ref($proto) || $proto;
                my $self = {
                        _permited => \%fields,
                        %fields,
```

```
	};
	bless ($self, $class);
	return $self;
}

sub readGroupsandUsers(){
	my ($self)= @_;
	my (@users,%groups);
	open(USERS,$WbnmsConf::CONFAUTH{'UsersFile'});
	while(<USERS>){
		chomp();
		my ($name,$pass)= split(/:/,$_);
		push(@users,$name);
	}
	close(USERS);
	open (GROUPS,$WbnmsConf::CONFAUTH{'GroupsFile'});
	while(<GROUPS>){
		chomp();
		my ($groupname,$users)= split(/:/,$_);
		my @groupsusers = split(/ /,$users);
		$groups{$groupname} = \@groupsusers;
	}
	close(GROUPS);

return(\@users,\%groups);
}

sub AddUsers(){
	my ($self)= @_;
	my (%users,%groups);

	if (defined $self->Name() and defined $self->Password() and defined $self->CheckPass()){
		open(USERS,$WbnmsConf::CONFAUTH{'UsersFile'});
		while(<USERS>){
			chomp();
			my ($name,$pass)= split(/:/,$_);
			$users{$name}=$pass;
		}
		close(USERS);
		$users{$self->Name()}= &_crypt($self);
		open(USERS,">$WbnmsConf::CONFAUTH{'UsersFile'}");
		while(my ($name,$pass)=each(%users)){
			print USERS $name.":".$pass."\n";
		}
		close(USERS);
	}elsif($self->Password()ne$self->CheckPass()){
		warn 'Password Incorrecto';
	}else{
		warn 'Warning: Incomplete request';
	}


return(1);
}

sub AddGroupUsers(){
	my ($self)=@_;
	my (%users,%groups);
	if(defined $self->Group() and defined $self->Name()){
		open (GROUPS,$WbnmsConf::CONFAUTH{'GroupsFile'});
		while(<GROUPS>){
			chomp();
			my ($groupname,$users)= split(/:/,$_);
			if (defined $users){
				my @groupsusers = split(/ /,$users);
				$groups{$groupname} = \@groupsusers;
			}
		}
		close(GROUPS);

		if (exists($groups{$self->Group()})){
			push(@{$groups{$self->Group()}},$self->Name());
		}else{
			${$groups{$self->Group()}}[0]= $self->Name();
		}
		open (GROUPS,">$WbnmsConf::CONFAUTH{'GroupsFile'}");
```

```
                        while( my ($group,$users)=each(%groups)){
                                if (defined $users){
                                        my $usersgroup = join(' ',@{$users});
                                        print GROUPS $group.":".$usersgroup."\n";
                                }
                        }
                        close(GROUPS);
                }
return(1);
}

sub DelUser(){
                my ($self)= @_;
                my (%users,%groups);
                my $test;
                if (defined $self->Name()){
                        open(USERS,$WbnmsConf::CONFAUTH{'UsersFile'});
                        while(<USERS>){
                                chomp();
                                my ($name,$pass)= split(/:/,$_);
                                $users{$name}=$pass;
                        }
                        close(USERS);
                        $test= delete $users{$self->Name()};
                        open(USERS,">$WbnmsConf::CONFAUTH{'UsersFile'}");
                        while(my ($name,$pass)=each(%users)){
                                print USERS $name.":".$pass."\n";
                        }
                        close(USERS);
                        DelGroupUsers($self);
                }else{
                        warn 'Warning: Incomplete request';
                }


return($test);

}

sub DelGroupUsers(){
                my ($self)=@_;
                my (%groups);
                if(defined $self->Name()){
                        open (GROUPS,$WbnmsConf::CONFAUTH{'GroupsFile'});
                        while(<GROUPS>){
                                chomp();
                                my ($groupname,$users)= split(/:/,$_);
                                if (defined $users){
                                        my @groupsusers = split(/ /,$users);
                                        $groups{$groupname} = \@groupsusers;
                                }
                        }
                        close(GROUPS);
                        open (GROUPS,">$WbnmsConf::CONFAUTH{'GroupsFile'}");
                        while( my ($group,$users)=each(%groups)){
                                my @usersleft;
                                foreach(@{$users}){
                                        if($_ ne $self->Name()){
                                                push(@usersleft,$_);

                                        }
                                }
                                my $usersgroup=join(' ',@usersleft);
                                print GROUPS $group.":".$usersgroup."\n";
                        }
                        close(GROUPS);
                }
return(1);
}

sub _crypt(){
                my ($self)=@_;
                my $salt = chr(int(rand(26))+65).chr(int(rand(26))+65);
                my $pwd = crypt($self->Password(), $salt);
return($pwd);
}
```

```
sub AUTOLOAD{
	my $self = shift;
	my $type = ref($self) || die;
	my $name = $AUTOLOAD;
	$name =~ s/.*://;
	unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
	if(@_){$self->{$name}= shift}
	return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.7 Error

9.1.7.1 ErrorManager.pm

```
package ErrorManager;


use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;


my %fields = (  ##Permited fields
	message => undef,
	Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
	my ($proto,$message)= @_;  #it retrieves the name of the class
	my $class= ref($proto) || $proto;
	my $self = {
		_permited => \%fields,
		%fields,
	};
	bless ($self, $class);
	$self->message($message);
	return $self;
}

sub Display(){
	my ($self)=@_;
	my $Output="<script>window.open(\"$WbnmsConf::CONFERROR{'ErrorPath'}?Message=".$self-
>message()."\",'Error','scrollbars=no,width=320,height=120');<\/script>";
return($Output);
}


sub AUTOLOAD{
	my $self = shift;
	my $type = ref($self) || die;
	my $name = $AUTOLOAD;
	$name =~ s/.*://;
	unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
	if(@_){$self->{$name}= shift}
	return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}
```

```
1;
```

9.1.8 Alarmsurveillance

9.1.8.1 AlarmSurd.pl

```
#!/usr/bin/perl
#
##Alarm Surveillance Daemon
## Version: 1
## History: v1- 8.11.2002
## -Linux Version
#

#use POSIX; ##for getpid
use POSIX qw(:sys_wait_h getpid);

#AlarmSurd Configuration
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object

use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
#use ABMManager 2.0;
require ABMManager;
##Calling ActionManager Object
use lib "/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance";
#use ActionManager 1.0;
require ActionManager;
##Calling StatusChange Object
#use StatusChange;
require StatusChange;
#Prototypes
sub Alarm_check ($$);
sub logger_Alarm (@);



#Uptime
$uptime =localtime();

print "<<Alarm Surveillance daemon Started at ".$uptime.">>\n" if $WbnmsConf::CONFALARMSUR{'Debug'} ==
1;

#Write out the PID file
open(PIDFILE, ">$WbnmsConf::CONFALARMSUR{'pidfile'}");
printf PIDFILE "%d\n", getpid();
close(PIDFILE);

# Main
while (1) {
        if (defined $WbnmsConf::CONFALARMSUR{'checkinterval'}){
                $interval = $WbnmsConf::CONFALARMSUR{'checkinterval'};
        }else{
                $interval = 30; ## Default value
        }
        sleep($interval);
  &Alarm_check;
}

# Subrutines

sub Alarm_check ($$) {
        my ($uno,$dos)= @_;
        $SIG{CHLD} =  sub { while ( waitpid(-1,WNOHANG)>0 ) {} };
        $Manager = ABMManager->new();
        $Manager->Debug(0);
        $Manager->ABMType(1);
        $Manager->MediaName("AlarmsTrigger");  ##Read AlarmsTrigger Table
        my @Keys = ("trigger_id","trigger_object","equipment_id","equipmenttype_id","severity");
        for (1..50){
```

```
                push(@Keys,("trigger_variable$_","trigger_value$_"));
        }
        $Manager->Keys(\@Keys);
        $Manager->Order("severity");
        my $data_ref_AT=$Manager->ReadInto();
        $Manager->Order(undef);
        my (@equipments,@maps);
        foreach my $data_ref_Search (@{$data_ref_AT}){
                print ${$data_ref_Search}{'trigger_object'}.${$data_ref_Search}{'trigger_id'}."\n";
            my (%searchequipment,$Allflag);
            $Allflag=0;
            if (${$data_ref_Search}{'equipment_id'} != 0){
                %searchequipment  = (
                        equipment_id => ${$data_ref_Search}{'equipment_id'},
                        );
                        $Manager->DataSearch(\%searchequipment);
                }elsif(${$data_ref_Search}{'equipmenttype_id'} != 0){
                %searchequipment = (
                        equipmenttype_id => ${$data_ref_Search}{'equipmenttype_id'},
                        );
                        $Manager->DataSearch(\%searchequipment);
                }else{
                        $Allflag= 1; ##All Types And Equipments
                }

            ##Read IP from Equipment Table
            $Manager->MediaName("Equipment");

                $Manager->DataSearch(undef);
                my @KeysEquipment= ("IP");
                $Manager->Keys(\@KeysEquipment);
                my $data_ref=$Manager->ReadInto();

            ##
            ##Read Not Processed Incoming Alarms
                $Manager->MediaName("TrapsCollector,AlarmsTrigger,Equipment,MapsEquipments");
                my $searchfile = "AlarmsTrigger.trigger_id =\"".${$data_ref_Search}{'trigger_id'}."\" and
(Equipment.equipment_id = AlarmsTrigger.equipment_id or Equipment.equipmenttype_id =
AlarmsTrigger.equipmenttype_id or AlarmsTrigger.equipmenttype_id = AlarmsTrigger.equipment_id ) and
Equipment.equipment_id = MapsEquipments.equipment_id and TrapsCollector.object =
\"".${$data_ref_Search}{'trigger_object'}."\" and TrapsCollector.object = AlarmsTrigger.trigger_object and
(TrapsCollector.trap_trigger_id is null or TrapsCollector.trap_trigger_id != AlarmsTrigger.trigger_id) and
trap_processed = \"0\"";
                if ($Allflag == 0){
                        if (${$data_ref_Search}{'equipment_id'} != 0){
                                $searchfile = $searchfile." and TrapsCollector.hosts =
\"${$data_ref}[0]{'IP'}\" ";
                        }elsif(${$data_ref_Search}{'equipmenttype_id'} != 0){
                                        my @datasearchtype;
                                foreach my $sametype(@{$data_ref}){
                                        push(@datasearchtype,"TrapsCollector.hosts =
\"${$sametype}{'IP'}\"");
                                }
                                $searchfile = $searchfile." and (".join(' or ',@datasearchtype).")";
                        }
                }
                for (1..50){
                        if (defined ${$data_ref_Search}{'trigger_value'.$_}){
                                $searchfile = $searchfile." and  AlarmsTrigger.trigger_variable$_ =
TrapsCollector.variable$_ ";
                                $searchfile = $searchfile." and  AlarmsTrigger.trigger_value$_ =
TrapsCollector.value$_ ";
                         }else{
                                $searchfile = $searchfile." and  AlarmsTrigger.trigger_variable$_ is null";
                                $searchfile = $searchfile." and  AlarmsTrigger.trigger_value$_ is null";
                                last;
                         }
                }


                $Manager->Keys(undef);
                #print $searchfile."\n";
                $Manager->DataSearch($searchfile);
                my $data_ref_TC=$Manager->Read_ComparingInto();
                $Manager->DataSearch(undef);
                ##Process Alarms with AlarmsTriggers
```

```
print scalar(@{$data_ref_TC})." Results\n";
my $testtrap;
foreach my $data_hash_ref_TC (@{$data_ref_TC}){
	my (%data,%search,@datalog,$traptrigger,$severity);
	push(@equipments,${$data_hash_ref_TC}{'equipment_id'}); ## Store equipment
	push(@maps,${$data_hash_ref_TC}{'map_id'}); ## Store map
	while(my($column,$value) = each(%{$data_hash_ref_TC})){
		if (defined $value){
		##every line according with alarm's triggers
			if ($column eq "trap_trigger_id"){
				$traptrigger = $value;
			}
			if ($column eq "severity"){
				$severity = $value;
			}
				if ($column eq "trap_id"){
					$search{$column} = $value;
				}
				if ($column eq "trigger_id"){
					$data{'trap_trigger_id'}= $value;
				}

				##Data to be logged into log file

				my $logtext= $column.":".$value;
				push(@datalog,$logtext);
				print $column." : ".$value."\n" if
$WbnmsConf::CONFALARMSUR{'Debug'} == 1;
		}

	}
		##Read severity from AlarmsTrigger Table
	$Manager->MediaName("AlarmsTrigger");
		 my %searchalarm_id = (
						trigger_id => $traptrigger,
						);
		$Manager->DataSearch(\%searchalarm_id);
		my @KesSeverity= ("severity");
		$Manager->Keys(\@KesSeverity);
		my $data_ref=$Manager->ReadInto();
		my $data_ref_hash= shift(@{$data_ref});

	##Comparing actual Severity with previous one.
	if ($severity >=  ${$data_ref_hash}{'severity'}){
		#print $traptrigger."\n";
		print "Adding: ".$data{'trap_trigger_id'}." = ".$search{'trap_id'}."\n" if
$WbnmsConf::CONFALARMSUR{'Debug'} == 1;
		$Manager->Data(\%data);
		$Manager->DataSearch(\%search);
		$Manager->MediaName("TrapsCollector");
		$Manager->ModifyInto();

		##Equipment Status Modification
		my $StatusChanger = StatusChange->new();
		$StatusChanger->Data($data_hash_ref_TC);
		$StatusChanger->EquipmentStatusChange();
		##

		#$SIG{CHLD} = 'IGNORE';
		my $pid=fork();
		if ($pid == 0){
			###Calling action Mannager: Call each Action Modules
including Drawer
			my $ActionMannager=ActionManager-
>new($data_hash_ref_TC);  ##Passing Data to ActionManager Object
			$ActionMannager-
>Debug($WbnmsConf::CONFALARMSUR{'Debug'}); ## Passing Debug Situation
			$ActionMannager->ActionManager();
			exit(0);
		}else{
		##waitpid($pid, 0);
		}
	}
	&logger_Alarm(@datalog);  ##logging data into log file
}
```

```
		}
		##Change trap_processed status
		$Manager->MediaName("TrapsCollector");
		my %data=(trap_processed => 1);
		my %search=(trap_processed => 0);
		$Manager->Data(\%data);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();
		##Call MapsStatus
		my $StatusChanger = StatusChange->new();
		$StatusChanger->MapStatusChange(\@equipments,\@maps);



}



sub logger_Alarm(@){
		my (@data_to_log) = @_;
		open
(ALARMLOG,">>$WbnmsConf::CONFALARMSUR{'logpath'}/$WbnmsConf::CONFALARMSUR{'AlarmSurfile.log'
}");
		foreach(@data_to_log){
			print ALARMLOG "$_;";
		}
		      print ALARMLOG "\n";
		close(ALARMLOG);
		}

__END__
```

9.1.8.2 AlarmSurd.rc

```
#!/bin/sh
#
# AlarmSurd.rc        Control AlarmSur daemon of WBNMS.
#

case "$1" in
  start)
		echo "Starting AlarmSurd"
		/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/start
		;;
  stop)
		echo "Stopping AlarmSurd"
		/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/stop
		;;
  status)
		if ! [ -f /var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/AlarmSurd.pid ]
		then echo "AlarmSurd.pl is stopped"
		    exit 3
		fi
		pid=`cat /var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/AlarmSurd.pid`
		kill -0 $pid >/dev/null 2>&1
		if [ $? == 0 ]
		then echo "AlarmSurd.pl (pid $pid) is running..."
		    exit 0
		fi
		echo "AlarmSurd.pl is stopped"
		exit 3
		;;
  restart)
		$0 stop
		$0 start
		;;
  reload)
		$0 stop
		$0 start
		;;

*)
		echo "Usage: $0 { start | stop | status | restart }"
```

```
                exit 1
                ;;
esac

exit 0
```

9.1.8.3 start

```
#!/bin/sh
echo Starting AlarmSurd  daemon
trap '' 1
perl /var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/AlarmSurd.pl &
```

9.1.8.4 stop

```
#!/bin/sh
echo Stopping alarSurd  daemon
kill -9 `cat /var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/AlarmSurd.pid`
perl /var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/renameAlarmSurlog.pl &
```

9.1.8.5 renameAlarmSurlog.pl

```
#!/usr/bin/perl

use Time::localtime;
#
##mv Alarm Surveillance log file
#
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

$uptime = localtime->sec().localtime->min().localtime->hour().localtime->mday().localtime->mon();
rename ($WbnmsConf::CONFALARMSUR{'logpath'}."/".$WbnmsConf::CONFALARMSUR{'AlarmSurfile.log'} ,
$WbnmsConf::CONFALARMSUR{'logpath'}."/AlarSurfile$uptime.log");
rename ($WbnmsConf::CONFMAILER{'logpath'}."/".$WbnmsConf::CONFMAILER{'mailer.log'} ,
$WbnmsConf::CONFMAILER{'logpath'}."/Mailer$uptime.log");
rename ($WbnmsConf::CONFSMS{'logpath'}."/".$WbnmsConf::CONFSMS{'sms.log'} ,
$WbnmsConf::CONFSMS{'logpath'}."/sms$uptime.log");
```

9.1.8.6 ActionManager.pm

```
package ActionManager;


use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##


#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

#Calling Mailer Module
use lib "/var/sites/Wbnms/Wbnms_Core/ActionModules";
use Mailer;


#Calling Traps Module
use lib "/var/sites/Wbnms/Wbnms_Core/ActionModules";  ## Must by configured at installation time
use Trapper;
```

```
my %fields = (  ##Permited fields
        Data => undef,
        trigger_id => undef,
        equipment_id => undef,
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto,$data)= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        $self->Data($data);
        return $self;
}

sub ActionManager{
        my $self = shift;

        ########################
        ##Call Action Modules
        my $Manager = ABMManager->new();
                $Manager->Debug(0);
        $Manager->ABMType(1);
        $Manager->MediaName("ActionsTriggers");
        my @Keys = ("action_id");
        $Manager->Keys(\@Keys);
        my %Search = (trigger_id => ${$self->Data}{'trigger_id'});
        $Manager->DataSearch(\%Search);
        my $data_ref_array =$Manager->ReadInto();

        ##Get time from trap information
        my ($date,$time)=split(/ /,${$self->Data}{'time'});
        my ($hour,$min,$sec)=split(/:/,$time);
        $time= $hour.":".$min;

        foreach my $data_ref_hash (@{$data_ref_array}){
                while(my($column,$value) = each(%{$data_ref_hash})){
                                                                print $column.":>".$value."\n" if $self-
>Debug == 1;
                        ##Recall User Data associated with action to be performed

                        $Manager->MediaName("ActionsUsersEquipments,UsersActions,AlarmsActions");
                        my $searchfile = "AlarmsActions.action_id = \"".$value."\" and
(ActionsUsersEquipments.equipment_id = \"".${$self->Data}{'equipment_id'}."\" or
ActionsUsersEquipments.equipment_id = 0 ) and (ActionsUsersEquipments.map_id = \"".${$self-
>Data}{'map_id'}."\" or ActionsUsersEquipments.map_id = 0 ) and (ActionsUsersEquipments.starttime <
\"".$time."\" and ActionsUsersEquipments.endtime > \"".$time."\" ) and ActionsUsersEquipments.action_id =
\"".$value."\" and ActionsUsersEquipments.user_id = UsersActions.user_id";
                        $Manager->DataSearch($searchfile);
                        $Manager->Keys(undef);
                        my $UserData = $Manager->Read_ComparingInto();
                        ###
                        my $mailflag = 0;
                        my %mailusers;
                        foreach my $data_ref_array2(@{$UserData}){
                                                        print "Calling Module
".${$data_ref_array2}{'module_id'}."\n" if $self->Debug == 1;

                                if(${$data_ref_array2}{'module_id'} == 0){     ## Send Alarm by e-mail


                                                if(! exists($mailusers{${$data_ref_array2}{'name'}})){
                                                        $mailusers{${$data_ref_array2}{'name'}} =
${$data_ref_array2}{'email'};
                                                }else{

        $mailusers{${$data_ref_array2}{'name'}.${$data_ref_array2}{'user_id'}} = ${$data_ref_array2}{'email'};
                                                }
                                                $mailflag ++;
```

```
                                        }elsif(${$data_ref_array2}{'module_id'} == 1){ ## Send Alarm by snmp
trap
                                                ##Call SNMP Module
                                                $SIG{CHLD} = 'DEFAULT';
                                                my $pid=fork();
                                                if ($pid == 0){
                                                        my $Trapper= Trapper->new();
                                                        $Trapper->TrapManager($data_ref_array2,$self-
>Data());
                                                        exit(0);
                                                }else{
                                                        waitpid($pid, 0);
                                                }


                                        }
                                }

                                if($mailflag){
                                        #$SIG{CHLD} = sub { while ( waitpid(-1,WNOHANG)>0 ) { } };
                                        $SIG{CHLD} = 'DEFAULT';
                                        my $pid=fork();
                                                if ($pid == 0){
                                                        ##Call SMTP Module
                                                        my $Mailer = Mailer->new();
                                                        $Mailer->SendMail(\%mailusers,$self->Data());
                                                        exit(0);
                                                }else{
                                                        waitpid($pid, 0);
                                                }
                                }

                        }
                }

}


sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.8.7 AlarmTriggersManager.pm

```
package AlarmTriggersManager;


use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

use lib '/var/sites/Wbnms/Wbnms_Core/Error';
```

```
use ErrorManager;

my %fields = (  ##Permited fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto)= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub ReadAlarmTriggers($){
        my ($self,$data) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);

        my $searchfile;
        if (defined $data){
                if (exists(${$data}{'trigger_id'})){
                        $Manager->MediaName("AlarmsTrigger");
                        $searchfile = "AlarmsTrigger.trigger_id = \"".${$data}{'trigger_id'}."\"";
                }
                if (exists(${$data}{'equipment_id'}) and exists(${$data}{'equipmenttype_id'})){
                        if (${$data}{'equipment_id'} eq "0" and ${$data}{'equipmenttype_id'} ne "0"){
                                $Manager-
>MediaName("AlarmsTrigger,MonitorEquipmentState,EquipmentTypes");
                                $searchfile = "AlarmsTrigger.equipment_id =
\"".${$data}{'equipment_id'}."\" and AlarmsTrigger.equipmenttype_id = \"".${$data}{'equipmenttype_id'}."\" and
AlarmsTrigger.equipmenttype_id = EquipmentTypes.equipmenttype_id and AlarmsTrigger.severity =
MonitorEquipmentState.state_id";
                        }elsif(${$data}{'equipment_id'} ne "0" and ${$data}{'equipmenttype_id'} eq "0"){
                                $Manager-
>MediaName("AlarmsTrigger,Equipment,MonitorEquipmentState,EquipmentTypes");
                                $searchfile = "AlarmsTrigger.equipment_id =
\"".${$data}{'equipment_id'}."\" and AlarmsTrigger.equipmenttype_id = \"".${$data}{'equipmenttype_id'}."\" and
AlarmsTrigger.equipment_id = Equipment.equipment_id and AlarmsTrigger.severity =
MonitorEquipmentState.state_id";
                        }elsif(${$data}{'equipment_id'} eq "0" and ${$data}{'equipmenttype_id'} eq "0"){
                                $Manager->MediaName("AlarmsTrigger,MonitorEquipmentState");
                                $searchfile = "AlarmsTrigger.equipment_id =
\"".${$data}{'equipment_id'}."\" and AlarmsTrigger.equipmenttype_id = \"".${$data}{'equipmenttype_id'}."\" and
AlarmsTrigger.severity = MonitorEquipmentState.state_id";
                        }
                }
        }else{
                $searchfile = "AlarmsTrigger.equipment_id = Equipment.equipment_id and
AlarmsTrigger.severity = MonitorEquipmentState.state_id";
        }
        $Manager->DataSearch($searchfile);
        my $data_ref_array = $Manager->Read_ComparingInto();
        my @triggersID=(0);
        foreach my $alarm (@{$data_ref_array}){
                my @Variables;
                for my $ind (1..50){
                        if (exists(${$alarm}{'trigger_variable'.$ind}) and ${$alarm}{'trigger_variable'.$ind} ne
""){

        push(@Variables,${$alarm}{'trigger_variable'.$ind}."=".${$alarm}{'trigger_value'.$ind});

                        }else{last;}
                }
                ${$alarm}{'trigger_variables'}=\@Variables;
                push(@triggersID,${$alarm}{'trigger_id'});
        }

return($data_ref_array,\@triggersID);
}

sub ReadAlarmTriggerfromID($){
        my ($self,$data) = @_;
```

```
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("AlarmsTrigger");
		$Manager->DataSearch($data);
		my $ActionTriggerData= $Manager->ReadInto();


return(${$ActionTriggerData}[0]{'trigger_id'});
}

sub ReadActionTriggers($){
		my ($self,$triggers_id) = @_;
		my (%ActionTriggers,%actionlabels);
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->Keys(undef);
		$Manager->MediaName("ActionsTriggers,AlarmsActions");
		foreach my $trigger_id(@{$triggers_id}){
			my $searchfile="ActionsTriggers.trigger_id=\"".$trigger_id."\" and ActionsTriggers.action_id =
AlarmsActions.action_id";
			$Manager->DataSearch($searchfile);
			my $ActionTriggerData= $Manager->Read_ComparingInto();
			my @actions;
			foreach my $action(@{$ActionTriggerData}){
				push(@actions,${$action}{'action_id'});
			}
			$ActionTriggers{$trigger_id}=\@actions;
		}
		$Manager->DataSearch(undef);
		$Manager->MediaName("AlarmsActions");
		my $AllActionTriggerData= $Manager->ReadInto();

		foreach my $alarm (@{$AllActionTriggerData}){
			$actionlabels{${$alarm}{'action_id'}}=${$alarm}{'action_module_comment'};
		}

return(\%ActionTriggers,\%actionlabels);
}

sub ReadEquipments($){
		my ($self,$equipment_id) = @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("Equipment,EquipmentTypes");
		my $searchfile;
		if (defined $equipment_id){
			$searchfile = "Equipment.equipment_id = ".$equipment_id." and
Equipment.equipmenttype_id = EquipmentTypes.equipmenttype_id";
		}else{
			$searchfile = "Equipment.equipmenttype_id = EquipmentTypes.equipmenttype_id";
		}

		$Manager->DataSearch($searchfile);
		$Manager->Keys(undef);
		my $EquipmentData= $Manager->Read_ComparingInto();
		my (@Equipments,%EquipmentLabels);
		foreach my $Equipment (@{$EquipmentData}){
			push(@Equipments,${$Equipment}{'equipment_id'});
			$EquipmentLabels{${$Equipment}{'equipment_id'}}=${$Equipment}{'equipment'};
		}
return($EquipmentData,\@Equipments,\%EquipmentLabels);
}

sub ReadEquipmentTypes($){
		my ($self,$equipmenttype_id) = @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("EquipmentTypes");
		my $searchfile;
		if (defined $equipmenttype_id){
			my %searchfile =( 'equipmenttype_id' => $equipmenttype_id,);
			$Manager->DataSearch(\%searchfile);
		}
		$Manager->Keys(undef);
		my $EquipmentTypesData= $Manager->ReadInto();
		my (@Equipmenttypes,%EquipmentTypeLabels);
```

```
		foreach my $Equipmenttype (@{$EquipmentTypesData}){
			push(@Equipmenttypes,${$Equipmenttype}{'equipmenttype_id'});

		$EquipmentTypeLabels{${$Equipmenttype}{'equipmenttype_id'}}=${$Equipmenttype}{'equipmenttype_l
abel'};
		}
return($EquipmentTypesData,\@Equipmenttypes,\%EquipmentTypeLabels);
}

sub ReadSeverity(){
		my ($self) = @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("MonitorEquipmentState");
		my $SeveritiesData= $Manager->ReadInto();
		my (@Severities,%SeveritiesLabels);
		foreach my $Severity (@{$SeveritiesData}){
			push(@Severities,${$Severity}{'state_id'});
			$SeveritiesLabels{${$Severity}{'state_id'}}=${$Severity}{'state_label'};
		}
return($SeveritiesData,\@Severities,\%SeveritiesLabels);
}


sub AddAlarmTriggers($){
		my ($self,$data) = @_; ##data is a hash reference
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Cretate Into AlarmsTrigger Table
		$Manager->MediaName("AlarmsTrigger");
		$Manager->Data($data);
		$Manager->CreateInto();

return(1);
}

sub DelAlarmTriggers($){
		my ($self,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Delete Item from AlarmsTrigger table
		$Manager->MediaName("AlarmsTrigger");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();

return($output);
}

sub ModAlarmTriggers($$){
		my ($self,$search,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Modify Item from AlarmsTrigger table
		$Manager->MediaName("AlarmsTrigger");
		$Manager->DataSearch($search);
		$Manager->Data($data);
		$Manager->ModifyInto();

return($output);
}

sub AddActionTrigger($){
		my ($self,$data) = @_; ##data is a hash reference
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Cretate Into ActionsTriggers Table
		$Manager->MediaName("ActionsTriggers");
		$Manager->Data($data);
		$Manager->CreateInto();

return(1);
}

sub DelActionTrigger($){
```

```
		my ($self,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Delete Item from ActionsTriggers table
		$Manager->MediaName("ActionsTriggers");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();

return($output);
}

sub ModActionTrigger($$){
		my ($self,$search,$modlist) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Modify Item from ActionsTriggers table
		$Manager->MediaName("ActionsTriggers");
		$Manager->DataSearch($search);
		my $ActionsTriggersData=$Manager->ReadInto();
		my (@existlist,@nochangelist,@addlist,@deletelist);
		foreach my $actions(@{$ActionsTriggersData}){
			push(@existlist, ${$actions}{'action_id'});
			my $delflag = 0;
			if (0 != scalar(@{$modlist})){
				foreach my $modact (@{$modlist}){
					if(${$actions}{'action_id'} == $modact){
						$delflag = 0;
						last;
					}elsif(${$actions}{'action_id'} != $modact){
						$delflag = 1;
					}
				}
			}else{
				$delflag = 1;
			}
			if($delflag == 1){push(@deletelist,${$actions}{'action_id'});}
		}
		foreach my $actions(@{$modlist},@deletelist){
			my $addflag = 0;
			if (0 != scalar(@existlist)){
				foreach my $modact (@existlist){
					if($actions == $modact){
						$addflag = 0;
						last;
					}elsif($actions != $modact){
						$addflag = 1;
					}
				}
			}else{
				$addflag = 1;
			}
			if($addflag == 1){push(@addlist,$actions);}
		}

		foreach my $del (@deletelist){
			my %data=('trigger_id'=>${$search}{'trigger_id'},'action_id'=>$del,);
			&DelActionTrigger($self,\%data);
		}
		foreach my $add (@addlist){
			my %data=('trigger_id'=>${$search}{'trigger_id'},'action_id'=>$add,);
			&AddActionTrigger($self,\%data);

		}

return($output);
}

sub ReadObjects($){
		my ($self,$object) = @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Read Object Type. Object or Notification.
		$Manager->MediaName("Oid2Label");
		my %search=('oid_label'=>$object,);
```

```
	$Manager->DataSearch(\%search);
	my $TypeData= $Manager->ReadInto();
	if (${$TypeData}[0]{'object_type'} == 0 ){##Notification Type
		##Read from NotificationType Table
		$Manager->MediaName("NotificationType");
		my %search=('oid_label'=>$object,);
		$Manager->DataSearch(\%search);
		my $NotificationTypeData= $Manager->ReadInto();
		${$TypeData}[0]{'object_type'}="NotificationType";
		while(my ($key,$value)=each(%{${$NotificationTypeData}[0]})){
			if ($value ne ""){${$TypeData}[0]{$key}=$value;}
		}
		my $ind = 1;
		while( exists(${$TypeData}[0]{'object'.$ind})){
			##Read from Oid2Label Table
			$Manager->MediaName("Oid2Label,ObjectType");
			my $search="Oid2Label.oid_label = \"".${$TypeData}[0]{'object'.$ind}."\" and
Oid2Label.oid_label = ObjectType.oid_label";
			$Manager->DataSearch($search);
			my $TrapObjData= $Manager->Read_ComparingInto();
			##Read from Syntax and TextualConventions
			$Manager->MediaName("Syntax,TextualConventions");
			my $search="Syntax.type_label = \"".${$TypeData}[0]{'object'.$ind}."\" and
Syntax.type_id =\"1\" and Syntax.type_label = TextualConventions.convention_label ";
			$Manager->DataSearch($search);
			my $TrapTextData= $Manager->Read_ComparingInto();
			my (%textuallabels,@textualvalues);
			foreach my $data (@{$TrapTextData}){

	$textuallabels{${$data}{'convention_value'}}=${$data}{'convention_value_label'};
				push(@textualvalues,${$data}{'convention_value'});
			}
			${$TypeData}[0]{'textuallabels'.$ind}=\%textuallabels;
			${$TypeData}[0]{'textualvalues'.$ind}=\@textualvalues;
			${$TypeData}[0]{'objectdatatype'.$ind}=\@{$TrapTextData};
			${$TypeData}[0]{'objectdata'.$ind}=\%{${$TrapObjData}[0]};
			$ind++;
		}
	}elsif(${$TypeData}[0]{'object_type'} == 1 ){##Object Type
		##Read from ObjectType Table
		$Manager->MediaName("ObjectType");
		my %search=('oid_label'=>$object,);
		$Manager->DataSearch(\%search);
		my $ObjectTypeData= $Manager->ReadInto();
		${$TypeData}[0]{'object_type'} = "ObjectType";
		while(my ($key,$value)=each(%{${$ObjectTypeData}[0]})){
			if ($value ne ""){${$TypeData}[0]{$key}=$value;}
		}
		for my $ind (1..50){
			if(${$TypeData}[0]{'index'.$ind} ne ""){
				$Manager->MediaName("ObjectType,Oid2Label,Syntax");
				my $search="ObjectType.oid_label = \"".$object."\" and
Oid2Label.oid_label = ObjectType.oid_label and Syntax.type_label = Oid2Label.syntax";
				$Manager->DataSearch($search);
				my $ObjectIndexData= $Manager->Read_ComparingInto();

			}else{
				last;
			}
		}
	}elsif(${$TypeData}[0]{'object_type'} == 2 ){##Trap Type
		##Read from TrapType Table
		$Manager->MediaName("TrapType");
		my %search=('oid_label'=>$object,);
		$Manager->DataSearch(\%search);
		my $TrapTypeData= $Manager->ReadInto();
		${$TypeData}[0]{'object_type'}="TrapType";
		while(my ($key,$value)=each(%{${$TrapTypeData}[0]})){
			if ($value ne ""){${$TypeData}[0]{$key}=$value;}
		}
		my $ind = 1;
		while( exists(${$TypeData}[0]{'object'.$ind})){
			##Read from Oid2Label Table
			$Manager->MediaName("Oid2Label,ObjectType");
			my $search="Oid2Label.oid_label = \"".${$TypeData}[0]{'object'.$ind}."\" and
Oid2Label.oid_label = ObjectType.oid_label";
```

```
                                $Manager->DataSearch($search);
                                my $TrapObjData= $Manager->Read_ComparingInto();
                                ##Read from Syntax and TextualConventions
                                $Manager->MediaName("Syntax,TextualConventions");
                                my $search="Syntax.type_label = \"".${$TypeData}[0]{'object'.$ind}."\" and
Syntax.type_id =\"1\" and Syntax.type_label = TextualConventions.convention_label ";
                                $Manager->DataSearch($search);
                                my $TrapTextData= $Manager->Read_ComparingInto();
                                my (%textuallabels,@textualvalues);
                                foreach my $data (@{$TrapTextData}){

        $textuallabels{${$data}{'convention_value'}}=${$data}{'convention_value_label'};
                                        push(@textualvalues,${$data}{'convention_value'});
                                }
                                ${$TypeData}[0]{'textuallabels'.$ind}=\%textuallabels;
                                ${$TypeData}[0]{'textualvalues'.$ind}=\@textualvalues;
                                ${$TypeData}[0]{'objectdatatype'.$ind}=\@{$TrapTextData};
                                ${$TypeData}[0]{'objectdata'.$ind}=\%{${$TrapObjData}[0]};
                                $ind++;
                        }
                }


return(${$TypeData}[0]);
}

sub ReadAllAlarms(){
        my ($self) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        my @Alarms;
        $Manager->MediaName("TrapType");
        my $TrapType = $Manager->ReadInto();
        foreach my $line (@{$TrapType}){
                my %data;
                $data{'oid_label'}=${$line}{'oid_label'};
                $data{'description'}=${$line}{'description'};
                $data{'type'}="TrapType:SNMPv1";
                $data{'color'}='style= background-color:"#C84428"';
                push(@Alarms,\%data);
        }
        $Manager->MediaName("NotificationType");
        my $NotificationType = $Manager->ReadInto();
        foreach my $line (@{$NotificationType}){
                my %data;
                $data{'oid_label'}=${$line}{'oid_label'};
                $data{'description'}=${$line}{'description'};
                $data{'type'}="NotificationType:SNMPv2";
                $data{'color'}='style= background-color:"#8080C0"';
                push(@Alarms,\%data);
        }


return(\@Alarms);

}

sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.8.8 LogManager.pm

```
package LogManager;

use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.1;

use lib "/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance";
use StatusChange;

use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;

my %fields = (  ##Permitted fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto )= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub ReadAllLog(){
        my ($self) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->Order('trap_id');
        $Manager->MediaName("TrapsCollector");
        my $data_ref_array = $Manager->ReadInto();

return($data_ref_array);
}

sub ReadProcessedLog(){
        my ($self) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->MediaName("TrapsCollector,AlarmsTrigger,Equipment,EquipmentTypes");
        my $searchfile = "TrapsCollector.trap_trigger_id = AlarmsTrigger.trigger_id and
((AlarmsTrigger.equipment_id = Equipment.equipment_id) or (AlarmsTrigger.equipment_id = 0 and
TrapsCollector.hosts = Equipment.IP)) and EquipmentTypes.equipmenttype_id = Equipment.equipmenttype_id";
        $Manager->DataSearch($searchfile);
        $Manager->Order('trap_id');
        $Manager->Keys(undef);
        my $AlarmsData= $Manager->Read_ComparingInto();

return($AlarmsData);
}

sub ReadProcessedLogWUsers($){
        my ($self,$equipment_id) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager-
>MediaName("ActionsUsersEquipments,ActionsTriggers,UsersActions,AlarmsActions,TrapsCollector,AlarmsTrig
ger,Equipment,EquipmentTypes");
        my $searchfile = "TrapsCollector.trap_trigger_id = AlarmsTrigger.trigger_id and
((AlarmsTrigger.equipment_id = Equipment.equipment_id) or (AlarmsTrigger.equipment_id = 0 and
TrapsCollector.hosts = Equipment.IP)) and EquipmentTypes.equipmenttype_id = Equipment.equipmenttype_id
and ActionsTriggers.trigger_id = TrapsCollector.trap_trigger_id and ActionsTriggers.action_id =
```

```
ActionsUsersEquipments.action_id and AlarmsActions.action_id = ActionsUsersEquipments.action_id and
UsersActions.user_id = ActionsUsersEquipments.user_id";
		$Manager->DataSearch($searchfile);
		$Manager->Order('trap_id');
		$Manager->Keys(undef);
		my $AlarmsData= $Manager->Read_ComparingInto();
return($AlarmsData);
}

sub FilterLog(){
		my ($self,$data,$filter) = @_;
		my @filtereddata;
		foreach my $line (@{$data}){
			if (defined ${$line}{'time'}){
				my ($date,$time)=split(/ /,${$line}{'time'});
				my ($hour,$min,$sec)=split(/:/,$time);
				if (exists(${$filter}{'Day'}) and $date ne ${$filter}{'Day'}){next;}
				if (exists(${$filter}{'Hour'}) and $hour ne ${$filter}{'Hour'}){next;}
				if (exists(${$filter}{'Minutes'}) and $min ne ${$filter}{'Minutes'}){next;}

			}
			if (exists(${$filter}{'IP'}) and ${$line}{'hosts'} ne ${$filter}{'IP'}){next;}
			if (exists(${$filter}{'Severity'}) and ${$line}{'severity'} ne ${$filter}{'Severity'}){next;}
			if (exists(${$filter}{'Equipment'}) and ${$line}{'equipment'} ne ${$filter}{'Equipment'}){next;}
			if (exists(${$filter}{'Type'}) and ${$line}{'equipmenttype_label'} ne ${$filter}{'Type'}){next;}
			if (exists(${$filter}{'ID'}) and ${$line}{'trap_id'} ne ${$filter}{'ID'}){next;}
			if (exists(${$filter}{'equipment_id'}) and ${$line}{'equipment_id'} !=
${$filter}{'equipment_id'}){next;}
			push(@filtereddata,$line);
		}


return(\@filtereddata);
}

sub ProcessedAlarmStatus($){
		my ($self,$severity,$active) = @_;
		my %Colors;
		if ($active eq "" ){
		%Colors=(	'0' => "#00FF00", ##OK_Green
				'1' => "#00FFFF", ##Warning_Cyan
				'2' => "#FFFF00", ##Minor_Yellow
				'3' => "#FF8000", ##Mayor_Orange
				'4' => "#FF0000", ##Critical__Red
				);
	}elsif($active ne ""){
	%Colors=(	'0' => "#58AC58", ##OK_Green
				'1' => "#58ACA8", ##Warning_Cyan
				'2' => "#A8AC58", ##Minor_Yellow
				'3' => "#A88058", ##Mayor_Orange
				'4' => "#A85458", ##Critical__Red
				);

	}
		my %Labels=( '0' => "OK", ##OK
				'1' => "Advertencia", ##Warning
				'2' => "Menor", ##Minor
				'3' => "Mayor", ##Mayor
				'4' => "Critica", ##Critical
				);
		if ($severity > 3 ){$severity = 4;}

return($Colors{$severity},$Labels{$severity});
}

sub ParseAlarms($){
		my ($self,$type) = @_;
		my $AlarmsData;
		if ($type eq "ALL"){
			$AlarmsData=&ReadAllLog($self);
		}elsif($type eq "PRO"){
			$AlarmsData=&ReadProcessedLog($self);
		}
		my(@dates,@hours,@mins,@hosts,@Equipments,@severities,@types,@IDs);
		foreach my $line(@{$AlarmsData}){
			if (defined ${$line}{'time'}){
```

```
                                my ($date,$time)=split(/ /,${$line}{'time'});
                                my ($hour,$min,$sec)=split(/:/,$time);
                                if (!grep($_ eq $date ,@dates)){
                                        push(@dates,$date);
                                }
                                if (!grep($_ eq $hour ,@hours)){
                                        push(@hours,$hour);
                                }
                                if (!grep($_ eq $min ,@mins)){
                                        push(@mins,$min);
                                }

                        }
                        if (defined ${$line}{'hosts'}){
                                if (!grep($_ eq ${$line}{'hosts'} ,@hosts)){
                                        push(@hosts,${$line}{'hosts'});
                                }
                        }
                        if (defined ${$line}{'equipment'}){
                                if (!grep($_ eq ${$line}{'equipment'} ,@Equipments)){
                                        push(@Equipments,${$line}{'equipment'});
                                }
                        }
                        if (defined ${$line}{'severity'}){
                                if (!grep($_ eq ${$line}{'severity'} ,@severities)){
                                        push(@severities,${$line}{'severity'});
                                }
                        }
                        if (defined ${$line}{'equipmenttype_label'}){
                                if (!grep($_ eq ${$line}{'equipmenttype_label'} ,@types)){
                                        push(@types,${$line}{'equipmenttype_label'});
                                }
                        }
                        if (defined ${$line}{'trap_id'}){
                                if (!grep($_ eq ${$line}{'trap_id'} ,@IDs)){
                                        push(@IDs,${$line}{'trap_id'});
                                }
                        }
                }
                my %data=(
                                'Dia' => \@dates,
                                'Hora' => \@hours,
                                'Minutos' => \@mins,
                                'IP' => \@hosts,
                                'Equipo' => \@Equipments,
                                'Severidad' => \@severities,
                                'Tipo' => \@types,
                                'ID' => \@IDs,
                );
return(\%data);
}
sub AckEvent($$){
                my ($self,$data,$search) = @_;
                my $Manager = ABMManager->new();
                $Manager->ABMType(1);
                ##Modify Into TrapsCollector Table
                $Manager->MediaName("TrapsCollector");
                $Manager->DataSearch($search);
                $Manager->Data($data);
                $Manager->ModifyInto();
return(1);
}

sub AckAlarm($){
                my ($self,$trap_id,$user) = @_;
                ##Read equipment_id
                my $Manager = ABMManager->new();
                $Manager->ABMType(1);
                $Manager->MediaName("TrapsCollector,AlarmsTrigger,Equipment");
                my $searchfile = "TrapsCollector.trap_id =\"".$trap_id."\" and TrapsCollector.trap_trigger_id =
AlarmsTrigger.trigger_id and ((TrapsCollector.hosts = Equipment.IP and AlarmsTrigger.equipment_id = 0) or
(AlarmsTrigger.equipment_id = Equipment.equipment_id and AlarmsTrigger.equipment_id != 0))";
                $Manager->Keys(undef);
                $Manager->DataSearch($searchfile);
                my $EquipmentsAlarms = $Manager->Read_ComparingInto();
                $Manager->MediaName("AlarmsEquipmentMemory");
```

```
		my %data=(
				'equipment_id' => ${$EquipmentsAlarms}[0]{'equipment_id'},
				'trap_id' => $trap_id,
			);
		$Manager->DataSearch(\%data);
		$Manager->DeleteInto();
		##Modify Into TrapsCollector Table
		$Manager->MediaName("TrapsCollector");
		my %search=('trap_id' => $trap_id,);
		my %data=( 'ack' => $user,);
		$Manager->DataSearch(\%search);
		$Manager->Data(\%data);
		$Manager->ModifyInto();
		##Change Equipment State
		my $Status=StatusChange->new();
		my %data=('equipment_id' => ${$EquipmentsAlarms}[0]{'equipment_id'},);
		$Status->Data(\%data);
		$Status->StatusManager();
		my @equip=(${$EquipmentsAlarms}[0]{'equipment_id'});
		$Status->MapStatusChange(\@equip,undef);
return(1);
}

sub DelEvent($){
		my ($self,$data) = @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Delete Item from Maps table
		$Manager->MediaName("TrapsCollector");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
return(1);
}

sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.8.9 StatusChange.pm

```
package StatusChange;

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';

my %fields = (  ##Permitted fields
		Data => undef,
		_clearflag => 0,
		Debug => 0,  ## If Debug == 1 then Debug Mode
);
```

```
sub new {
	my $proto= shift;  #it retrieves the name of the class
	my $class= ref($proto) || $proto;
	my $self = {
		_permited => \%fields,
		%fields,
	};
	bless ($self, $class);
	return $self;
}

sub EquipmentStatusChange(){
	my ($self)= @_;
	##Read and Compare current Alarm with Alarm 's Memory
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	$Manager->MediaName("AlarmsEquipmentMemory,TrapsCollector,AlarmsTrigger");
	my $searchfile = "AlarmsEquipmentMemory.equipment_id =\"".${$self->Data}{'equipment_id'}."\" and
AlarmsEquipmentMemory.trap_id = TrapsCollector.trap_id and TrapsCollector.trap_trigger_id =
AlarmsTrigger.trigger_id";
	$Manager->DataSearch($searchfile);
	$Manager->Keys(undef);
	my $equipmentAlarmsData= $Manager->Read_ComparingInto();
	my (@clearedAlarms,@severities);
	foreach my $Alarm (@{$equipmentAlarmsData}){
		if (${$Alarm}{'triggernot_id'} == ${$self->Data}{'trigger_id'}){
			push(@clearedAlarms,${$Alarm}{'trap_id'});
			push(@severities,${$Alarm}{'severity'});
		}
	}
	if ( 0 == scalar(@clearedAlarms)){
		##print "Adding..\n";
		##Cretate Into AlarmsEquipmentMemory Table
		$Manager->MediaName("AlarmsEquipmentMemory");
		my %data=(
			'equipment_id' => ${$self->Data}{'equipment_id'},
			'trap_id' => ${$self->Data}{'trap_id'},
			);
		$Manager->Data(\%data);
		$Manager->CreateInto();
	}else{
		##print "Deleting..\n";
		##Delete Item from AlarmsEquipmentMemory table
		$Manager->MediaName("AlarmsEquipmentMemory");
		my %data=(
			'equipment_id' => ${$self->Data}{'equipment_id'},
			'trap_id' => $clearedAlarms[0],
			);
		$Manager->DataSearch(\%data);
		$Manager->DeleteInto();
		##Change to ACK trap into TrapsCollector Table
		$Manager->MediaName("TrapsCollector");
		my %data=(
				'marked' => "ACK",
				'ack' => ${$self->Data}{'trap_id'},
			);
		$Manager->Data(\%data);
		my %search=('trap_id'=> $clearedAlarms[0],);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();
		my %data=(
				'marked' => "ACK",
				'ack' => 'OK:'.$clearedAlarms[0],
			);
		$Manager->Data(\%data);
		my %search=('trap_id'=> ${$self->Data}{'trap_id'},);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();
		$self->_clearflag(1);
	}
	&StatusManager($self);

}
sub StatusManager(){
	my ($self)= @_;
	  ##Change into EquipmentMonitor Status
```

```
		my $Manager = ABMManager->new();
			$Manager->Debug(0);
		$Manager->ABMType(1);
		$Manager->MediaName("TrapsCollector,AlarmsEquipmentMemory,AlarmsTrigger");
		my $searchfile = "((AlarmsTrigger.equipment_id = \"".${$self->Data}{'equipment_id'}."\" and
AlarmsTrigger.equipment_id = AlarmsEquipmentMemory.equipment_id ) or (AlarmsTrigger.equipment_id = 0 and
AlarmsEquipmentMemory.equipment_id = \"".${$self->Data}{'equipment_id'}."\")) and
TrapsCollector.trap_trigger_id = AlarmsTrigger.trigger_id and TrapsCollector.marked is null";
		$Manager->Keys(undef);
		$Manager->DataSearch($searchfile);
		my $EquipmentsAlarms = $Manager->Read_ComparingInto();
		my ($status_id,$EquipmentSeverity);
		if (scalar(@{$EquipmentsAlarms})!=0){
			my $EquipmentSeverity=0;
			foreach my $Alarm (@{$EquipmentsAlarms}){
				if ($EquipmentSeverity <= ${$Alarm}{'severity'}){
					$EquipmentSeverity=${$Alarm}{'severity'};
				}
			}
			if ($EquipmentSeverity == 1){ $status_id = 1;} ## Warning
			if ($EquipmentSeverity == 2){ $status_id = 2;} ## Minor
			if ($EquipmentSeverity == 3){ $status_id = 3;} ## Mayor
			if ($EquipmentSeverity == 4){ $status_id = 4;} ## Critical
		}else{
			$status_id=0; ## OK
		}
	my  %data = (
				changed => "1",
				state_id => $status_id,
	);
	my %search = (
				equipment_id  => ${$self->Data}{'equipment_id'},
	 );
	$Manager->Data(\%data);
	$Manager->DataSearch(\%search);
	$Manager->MediaName("EquipmentMonitor");
	$Manager->ModifyInto();

return(1);
}
sub MapStatusChange(){
		my ($self,$equipments,$maps)= @_;
		my $Manager = ABMManager->new();
			$Manager->Debug(0);
		$Manager->ABMType(1);

		$Manager->MediaName("EquipmentMonitor,MapsEquipments");
		my %mapsstate;
		#print "Maptesting ..\n" if $WbnmsConf::CONFALARMSUR{'Debug'} == 1;
		foreach my $equip (@{$equipments}){
			my $searchfile = "EquipmentMonitor.equipment_id = MapsEquipments.equipment_id and
EquipmentMonitor.equipment_id = \"".$equip."\"";
			$Manager->Keys(undef);
			$Manager->DataSearch($searchfile);
			my $data_ref_TC=$Manager->Read_ComparingInto();
			if(!defined $mapsstate{${$data_ref_TC}[0]{'map_id'}} or
$mapsstate{${$data_ref_TC}[0]{'map_id'}} < ${$data_ref_TC}[0]{'state_id'}){
				$mapsstate{${$data_ref_TC}[0]{'map_id'}} = ${$data_ref_TC}[0]{'state_id'};
			}else{
				next;
			}
		}
		$Manager->MediaName("MapsMonitor,MapsMaps");
		my $searchfile = "MapsMaps.son_map_id = MapsMonitor.map_id";
		$Manager->Keys(undef);
		$Manager->DataSearch($searchfile);
		my $data_ref_first=$Manager->Read_ComparingInto();
		my %fatherson;
		foreach my $sonfather (@{$data_ref_first}){
				if(!defined $fatherson{${$sonfather}{'son_map_id'}}){ ##Only one father
					$fatherson{${$sonfather}{'son_map_id'}}=
${$sonfather}{'father_map_id'};
				}else{ ##Multiples fathers

				}
		}
```

```
		foreach my $map (@{$maps}){
			my $maped =$map;
			while($maped != 0){
				my $searchfile = "MapsMonitor.map_id = \"".$maped."\" and
MapsMaps.son_map_id = MapsMonitor.map_id";
				$Manager->Keys(undef);
				$Manager->DataSearch($searchfile);
				my $data_ref_TC=$Manager->Read_ComparingInto();
				if( (defined $mapsstate{${$data_ref_TC}[0]{'father_map_id'}} and
$mapsstate{${$data_ref_TC}[0]{'father_map_id'}} < $mapsstate{${$data_ref_TC}[0]{'son_map_id'}}) or (!defined
$mapsstate{${$data_ref_TC}[0]{'father_map_id'}})){
					$mapsstate{${$data_ref_TC}[0]{'father_map_id'}} =
$mapsstate{${$data_ref_TC}[0]{'son_map_id'}};
				}else{
					last;
				}
				$maped = $fatherson{$maped};
			}
		}

		$Manager->MediaName("MapsMonitor");
		while(my ($key,$value)=each(%mapsstate)){
			#print $key.":".$value."\n";
			my %data=( state_id => $value,
				changed  => 1,
				 );
			my %search=( map_id => $key);
			$Manager->Data(\%data);
			$Manager->DataSearch(\%search);
			$Manager->ModifyInto();
		}

}


sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.9 Drawer

9.1.9.1 Drawerd.pl

```
#!/usr/bin/perl
#
## Drawer daemon
## Version: 1
## History: v1- 21.1.2003
## -Linux Version
#

use POSIX qw(getpid);


#AlarmSurd Configuration
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;
```

```
#Calling ABM Object

use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
#use ABMManager 2.0;
require ABMManager;
##Calling ActionManager Object
use lib "/var/sites/Wbnms/Wbnms_Core/Drawer";
#use Drawer;
require Drawer;
#Prototypes
sub Check_Status ($);

#Uptime
$uptime = localtime();

print "<<Drawer daemon Started at ".$uptime.">>\n" if $WbnmsConf::CONFDRAWER{'Debug'} == 1;

#Write out the PID file
open(PIDFILE, ">$WbnmsConf::CONFDRAWER{'pidfile'}");
printf PIDFILE "%d\n", getpid();
close(PIDFILE);

# Main
my $Manager = ABMManager->new();
$Manager->Debug(0); ## ABM Debuging off
$Manager->ABMType(1);
$Manager->MediaName("EquipmentMonitor");

my $Drawer = Drawer->new();
$Drawer->Debug($WbnmsConf::CONFDRAWER{'Debug'});

while (1) {
        if (defined $WbnmsConf::CONFDRAWER{'checkinterval'}){
                $interval = $WbnmsConf::CONFDRAWER{'checkinterval'};
        }else{
                $interval = 30; ## Default value
        }
print "<<Interval: ".$interval." sec>>\n" if $WbnmsConf::CONFDRAWER{'Debug'} == 1;
        sleep($interval);
        $time=localtime();
  &Check_Status($time);
  &AlarmsCounter($time);
}

# Subrutines

sub Check_Status($){
        my ($time)= @_;
        my @Keysequip = ("equipment_id","state_id" );
        my @Keysmaps = ("map_id","state_id" );
        my %Search=(
                                changed => "1",
                        );
        my %Data=( changed => "0",);
        $Manager->Keys(\@Keysmaps);
        $Manager->DataSearch(\%Search);
        $Manager->MediaName("MapsMonitor");
        my $MapsStatus = $Manager->ReadInto();
        my @equipmentsprocessed;
        foreach my $Map (@{$MapsStatus}){
                $Drawer->state_id(${$Map}{'state_id'});
                $Drawer->map_id(${$Map}{'map_id'});
                $Drawer->time($time);
                my ($equipmentsprocessed) = $Drawer->DrawManager();
                push(@equipmentsprocessed,@{$equipmentsprocessed});
                ##Modify 'changed' field to 0
                $Manager->Keys(\@Keysmaps);
                $Manager->MediaName("MapsMonitor");
                my %Search;
                $Search{'map_id'}=${$Map}{'map_id'};
                $Manager->DataSearch(\%Search);
                $Manager->Data(\%Data);
                $Manager->ModifyInto();
        }
        ##Modify 'changed' field to 0
```

```
		$Manager->Keys(\@Keysequip);
		$Manager->MediaName("EquipmentMonitor");
		foreach my $equip (@equipmentsprocessed){
			my %Search;
			$Search{'equipment_id'}=$equip;
			$Manager->DataSearch(\%Search);
			$Manager->Data(\%Data);
			$Manager->ModifyInto();
		}

}

sub AlarmsCounter($){
		my ($time)=@_;
		$Drawer->AlarmsCounter();
return(1);
}

1;
```

9.1.9.2 Drawerd.rc

```
#!/bin/sh
#
# Drawer.rc	Drawer daemon of WBNMS.
#

case "$1" in
  start)
		echo "Starting Drawerd"
		/var/sites/Wbnms/Wbnms_Core/Drawer/start
		;;
  stop)
		echo "Stopping Drawerd"
		/var/sites/Wbnms/Wbnms_Core/Drawer/stop
		;;
  status)
		if ! [ -f /var/sites/Wbnms/Wbnms_Core/Drawer/Drawerd.pid ]
		then echo "AlarmSurd.pl is stopped"
		    exit 3
		fi
		pid=`cat /var/sites/Wbnms/Wbnms_Core/Drawer/Drawerd.pid`
		kill -0 $pid >/dev/null 2>&1
		if [ $? == 0 ]
		then echo "Drawerd.pl (pid $pid) is running..."
		    exit 0
		fi
		echo "Drawerd.pl is stopped"
		exit 3
		;;
  restart)
		$0 stop
		$0 start
		;;
  reload)
		$0 stop
		$0 start
		;;

*)
		echo "Usage: $0 { start | stop | status | restart }"
		exit 1
		;;
esac

exit 0
```

9.1.9.3 start

```
#!/bin/sh
echo Starting Drawerd daemon
```

```
trap '' 1
perl /var/sites/Wbnms/Wbnms_Core/Drawer/Drawerd.pl &
```

9.1.9.4 stop

```
#!/bin/sh
echo Stopping Drawerd  daemon
kill -9 `cat /var/sites/Wbnms/Wbnms_Core/Drawer/Drawerd.pid `
```

9.1.9.5 Drawer.pm

```
package Drawer;

use GD;
use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

my %fields = (  ##Permitted fields
            state_id => undef,
            equipment_id => undef,
            map_id => undef,
            time => undef,
            Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
            my ($proto )= @_;  #it retrieves the name of the class
            my $class= ref($proto) || $proto;
            my $self = {
                        _permited => \%fields,
                        %fields,
            };
            bless ($self, $class);
            return $self;
}

sub DrawManager{
            my $self = shift;
            my $Manager= ABMManager->new();
            $Manager->ABMType(1);
            my @equipments; ##All Equipments processed
            ##Read Map Information
            $Manager->MediaName("Maps");
            my %search = ( 'map_id' => $self->map_id(),);
            $Manager->DataSearch(\%search);
            my $Map = $Manager->ReadInto(); ##Map Information
            ##Read Equipment 's Map Information
            $Manager-
>MediaName("Maps,MapsEquipments,Equipment,MonitorEquipmentState,EquipmentTypes,EquipmentMonitor");
            my $searchfile = "Maps.map_id = \"".$self->map_id()."\" and MapsEquipments.map_id = Maps.map_id
and MapsEquipments.equipment_id = Equipment.equipment_id and Equipment.equipmenttype_id =
EquipmentTypes.equipmenttype_id and Equipment.equipment_id = EquipmentMonitor.equipment_id  and
EquipmentMonitor.state_id = MonitorEquipmentState.state_id";
            $Manager->DataSearch($searchfile);
            my $EquipmentsDatafromMap = $Manager->Read_ComparingInto(); ##Equipment 's Map Information
            foreach my $EquipmentfromMap (@{$EquipmentsDatafromMap}){
                                    push(@equipments,${$EquipmentfromMap }{'equipment_id'});
                        }
            ##Read Map's Map Information
            $Manager->MediaName("Maps,MapsMaps,MapsMonitor,MonitorEquipmentState");
```

```
		my $searchfile = "MapsMaps.father_map_id = \"".$self->map_id()."\" and  MapsMaps.son_map_id =
MapsMonitor.map_id and MapsMonitor.map_id = Maps.map_id and MapsMonitor.state_id =
MonitorEquipmentState.state_id";
		$Manager->DataSearch($searchfile);
		my $MapSonsData = $Manager->Read_ComparingInto(); ##Map's Map Information
		##Draw
		&Draw($self,$Map,$EquipmentsDatafromMap,$MapSonsData);

return(\@equipments);
}

sub Draw($$$$){
		my ($self,$MapData,$EquipmentsData,$MapSonsData) = @_;
	##Open Map Template
		open(MAP,"$WbnmsConf::CONFPATH{'File_DB'}/BMP/Maps/".${$MapData}[0]{'mapbmp'}) || die;
		my $immap = newFromGif GD::Image(\*MAP);
		close(MAP);
		my ($width,$height) = $immap->getBounds();
		##Create Black Background
		my $im = new GD::Image($width,$height);

		  # allocate some colors
		  my $white = $im->colorAllocate(255,255,255); ## First allocated color is Background color
		  my %datacolor=(
		   'red'           => $im->colorAllocate(255,0,0),
		   'cian'          => $im->colorAllocate(0,255,255),
		   'blue'          => $im->colorAllocate(0,0,255),
		   'green'         => $im->colorAllocate(0,255,0),
		   'yellow'        => $im->colorAllocate(255,255,0),
		   'orange'        => $im->colorAllocate(255,128,0),
		   'dimgrey'       => $im->colorAllocate(84,84,84),
		   'gray'          => $im->colorAllocate(191,191,191),
		   'lightgray'     => $im->colorAllocate(168,168,168),
		   'vlightgray'    => $im->colorAllocate(204,204,204),
		   'aquamarine'    => $im->colorAllocate(112,219,147),
		   'blueviolet'    => $im->colorAllocate(158,94,158),
		   'brown'         => $im->colorAllocate(165,42,42),
		   'cadetblue'     => $im->colorAllocate(94,158,158),
		   'coral'         => $im->colorAllocate(255,126,0),
		   'cornflowerblue'=> $im->colorAllocate(66,66,110),
		   'darkgreen'     => $im->colorAllocate(47,79,47),
		   'darkolivegreen'=> $im->colorAllocate(79,79,47),
		   'darkorchid'    => $im->colorAllocate(153,49,204),
		   'black'         => $im->colorAllocate(0,0,0),
		                      );

		  $im->interlaced('true');

		##Copy Map Template
	$im->copy($immap,0,0,0,0,$width,$height);
	##Copy each Equipment
		foreach my $EquipmentsMap (@{$EquipmentsData}){
                                                                        if ($self->Debug == 1){
                                                                                print "Data from
Equipment:".${$EquipmentsMap}{'equipment'}."\n";
                                                                                while(my($column,$value) =
each(%{$EquipmentsMap})){
                                                                                        if ( defined $value){
                                                                                                print
$column.": ".$value."\n";
                                                                                        }
                                                                                }
                                                                                print "--------------------------\n";
                                                                        }
                        ##Open Equipment Template

open(EQUIP,$WbnmsConf::CONFPATH{'File_DB'}."/BMP/Equipments/".${$EquipmentsMap}{'equipmentbmp'}) ||
die;
                        my $srcimequip = newFromGif GD::Image(\*EQUIP);
                        close(EQUIP);
                        my ($logowidth,$logoheight) = $srcimequip->getBounds();
                        my ($X,$Y)=split(/:/,${$EquipmentsMap}{'coor'});
                        my $porcent;
                        if (defined $WbnmsConf::CONFDRAWER{'coloredweigh'}){
                                        $porcent = $WbnmsConf::CONFDRAWER{'coloredweigh'};
                                }else{
```

```
                                $porcent = 0.1; ##Default
                        }
                $im->filledRectangle($X-$logowidth*$porcent,$Y-$logoheight*$porcent,$X-
1+$logowidth+$logowidth*$porcent,$Y-
1+$logoheight+$logoheight*$porcent,$datacolor{${$EquipmentsMap}{'state_color'}});
                ##Copy Equipment Template
                $im->copy($srcimequip,$X,$Y,0,0,$logowidth,$logoheight);
                ##Print Equipment name
                        $im->string(gdMediumBoldFont,$X-$logowidth*$porcent,$Y-
1+$logoheight+$logoheight*$porcent,${$EquipmentsMap}{'equipment'},$datacolor{${$EquipmentsMap}{'state_col
or'}});

        }
         ##Copy each Son Map
        foreach my $MapsMap (@{$MapSonsData}){
                                                                        if ($self->Debug == 1){
                                                                                print "Data from
Map:".${$MapsMap}{'mapname'}."\n";
                                                                                while(my($column,$value) =
each(%{$MapsMap})){
                                                                                        if ( defined $value){
                                                                                                print
$column.": ".$value."\n";
                                                                                        }
                                                                                }
                                                                                print "-------------------------\n";
                                                                        }
                        ##Open Map Icon Template
                open(EQUIP,$WbnmsConf::CONFPATH{'File_DB'}."/BMP/MapIcons/".${$MapsMap}{'mapicon'}) ||
die;
                my $srcimmap = newFromGif GD::Image(\*EQUIP);
                close(EQUIP);
                my ($logowidth,$logoheight) = $srcimmap->getBounds();
                my ($X,$Y)=split(/:/,${$MapsMap}{'coor'});
                my $porcent;
                if (defined $WbnmsConf::CONFDRAWER{'coloredweigh'}){
                                $porcent = $WbnmsConf::CONFDRAWER{'coloredweigh'};
                        }else{
                                $porcent = 0.1; ##Default
                        }
                $im->filledRectangle($X-$logowidth*$porcent,$Y-$logoheight*$porcent,$X-
1+$logowidth+$logowidth*$porcent,$Y-
1+$logoheight+$logoheight*$porcent,$datacolor{${$MapsMap}{'state_color'}});
                ##Copy Equipment Template
                $im->copy($srcimmap,$X,$Y,0,0,$logowidth,$logoheight);
                ##Print Equipment name
                        $im->string(gdMediumBoldFont,$X-$logowidth*$porcent,$Y-
1+$logoheight+$logoheight*$porcent,${$MapsMap}{'mapname'},$datacolor{${$MapsMap}{'state_color'}});

        }

        $im->filledRectangle(0,$height-15,180,$height,$datacolor{'black'});
        $im->string(gdMediumBoldFont,5,$height-15,$self->time,$datacolor{'white'});
    ####
        #Save data into gif file
        my $png_data = $im->gif;
        open (DISPLAY,">".$WbnmsConf::CONFDRAWER{'DrawerWebpath'}."/".$self->map_id."\.gif") || die;
        binmode DISPLAY;
    print DISPLAY $png_data;
        close DISPLAY;

}
sub AlarmsCounter(){
        my ($self) = @_;
        my $gap = 10;
        my $y=100;   # height
        my $x=200;  # width
        my $im = new GD::Image($x+$gap,$y+$gap);

        # allocate some colors

        my $white = $im->colorAllocate(255,255,255); ## First allocated color is Background color
        my %datacolor=(
        'red'           => $im->colorAllocate(255,0,0),
        'cian'          => $im->colorAllocate(0,255,255),
        'yellow'        => $im->colorAllocate(255,255,0),
```

```
		'orange'        => $im->colorAllocate(255,128,0),
		'green'         => $im->colorAllocate(0,255,0),
		'black'         => $im->colorAllocate(0,0,0),
                                );
		# make the background transparent and interlaced
		$im->transparent($white);
		$im->interlaced('true');
		##Read Data From DB
		my $Manager= ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("MonitorEquipmentState");
		my $States = $Manager->ReadInto(); ##Map Information
		##Set Bars
		my $positions=&_position($self,$gap,$x,scalar(@{$States}));
		###Drawing
		$Manager->MediaName("TrapsCollector");
		my $searchfile = "TrapsCollector.marked is null";
		$Manager->DataSearch($searchfile);
		my $AllAlarms = $Manager->Read_ComparingInto();
		foreach my $State (@{$States}){
			my @ind=(shift(@{$positions}),shift(@{$positions}));
			$Manager->MediaName("TrapsCollector,AlarmsTrigger,MonitorEquipmentState");
			my $searchfile = "TrapsCollector.marked is null and TrapsCollector.trap_trigger_id =
AlarmsTrigger.trigger_id and AlarmsTrigger.severity = MonitorEquipmentState.state_id and
MonitorEquipmentState.state_id =\"".${$State}{'state_id'}."\"";
			$Manager->DataSearch($searchfile);
			my $Alarms = $Manager->Read_ComparingInto();
			#print scalar(@{$Alarms}).":".$ind[0].":".$ind[1]."\n";
			if (scalar(@{$AllAlarms}) == 0){push(@{$AllAlarms},"add")}
			$im->filledRectangle($ind[0],($y-
(scalar(@{$Alarms})/scalar(@{$AllAlarms})*$y)),$ind[1],$y,$datacolor{${$State}{'state_color'}}); ##filled
rectangles
			$im->stringUp(gdMediumBoldFont,$ind[0]-2,$y-
2,"${$State}{'state_label'}",$datacolor{'black'});  ##Labels
			if (scalar(@{$Alarms}) != 0){$im->string(gdMediumBoldFont,(($ind[1]-$ind[0])/2)+$ind[0]-
5,($y-(scalar(@{$Alarms})/scalar(@{$AllAlarms})*$y)),scalar(@{$Alarms}),$datacolor{'black'}); }
		}
		#Save data into png file
	my $png_data = $im->gif;
	open
(DISPLAY,">$WbnmsConf::CONFDRAWER{'DrawerWebpath'}/$WbnmsConf::CONFDRAWER{'AlarmsCounterfil
e'}") || die;
	binmode DISPLAY;
	print DISPLAY $png_data;
	close DISPLAY;

}

sub _position($$$){
		my ($self,$gap,$x,$dataobjects)= @_;
		my @positions;
		my $barlength=($x/$dataobjects)-$gap;
		my $count = $gap;
		for(my $ind=0;$ind <=$dataobjects*2;$ind++ ){
		 ($positions[$ind],$positions[$ind+1])=($count, $count + $barlength);
		 $count=$count+$barlength+$gap;
		 $ind++;
		}
return (\@positions);
}

sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.9.6 EquipmentManager.pm

```
package EquipmentManager;

use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;

my %fields = (  ##Permited fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto )= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub ReadEquipments($){
        my ($self,$equipment_id) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->MediaName("Equipment,EquipmentTypes");
        my $searchfile;
        if (defined $equipment_id){
                $searchfile = "Equipment.equipment_id = ".$equipment_id." and
Equipment.equipmenttype_id = EquipmentTypes.equipmenttype_id";
        }else{
                $searchfile = "Equipment.equipmenttype_id = EquipmentTypes.equipmenttype_id";
        }

        $Manager->DataSearch($searchfile);
        $Manager->Keys(undef);
        my $EquipmentData= $Manager->Read_ComparingInto();
    ##Read Types
    $Manager->MediaName("EquipmentTypes");
        $Manager->DataSearch(undef);
        #my @Keys=('equipmenttype_id','equipmenttype_label');
        $Manager->Keys(undef);
        my $EquipmentTypes= $Manager->ReadInto();


return($EquipmentData,$EquipmentTypes);
}


sub ReadMapsEquipments($){
        my ($self,$map_id)= @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->MediaName("Equipment,EquipmentTypes,MapsEquipments");
        my $searchfile = "MapsEquipments.map_id = \"".$map_id."\" and Equipment.equipment_id =
MapsEquipments.equipment_id and Equipment.equipmenttype_id = EquipmentTypes.equipmenttype_id";
        $Manager->DataSearch($searchfile);
        $Manager->Keys(undef);
```

```
		my $EquipmentData= $Manager->Read_ComparingInto();

return($EquipmentData);
}



sub AddEquipment($){
		my ($self,$data) = @_; ##data is a hash reference
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("Equipment");
		$Manager->Data($data);
		$Manager->CreateInto();
		##Read data from Table
		$Manager->DataSearch($data);
		$Manager->Keys(undef);
		my $data_ref_array = $Manager->ReadInto();
		my $equipment_id=${$data_ref_array}[0]{'equipment_id'};
		##Create into table EquipmentMonitor
		$Manager->MediaName("EquipmentMonitor");
		my %statedata =(
				'equipment_id'=>$equipment_id,
				'state_id'=> 0,
				'changed' => 1,
		);
		$Manager->Data(\%statedata);
		$Manager->CreateInto();

return(1);
}

sub DelEquipment($){
		my ($self,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Delete Item from Equipments table
		$Manager->MediaName("Equipment");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
		##Delete Item from EquipmentMonitor table
		$Manager->MediaName("EquipmentMonitor");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
		##Delete Item from MapsEquipments table
		$Manager->MediaName("MapsEquipments");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();

return($output);
}

sub ModEquipment($){
		my ($self,$search,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Modify Item from Equipments table
		$Manager->MediaName("Equipment");
		$Manager->DataSearch($search);
		$Manager->Data($data);
		$Manager->ModifyInto();

return($output);
}

sub AddType($){
		my ($self,$data) = @_; ##data is a hash reference
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("EquipmentTypes");
		$Manager->Data($data);
		$Manager->CreateInto();

return(1);
```

```
}

sub DelType($){
	my ($self,$data) = @_; ##data is a hash reference
	my $output = "1";
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	##Check if are some dependent Equipment
	$Manager->MediaName("Equipment,EquipmentTypes");
	my $searchfile = "Equipment.equipmenttype_id =".${$data}{'equipmenttype_id'}." and
Equipment.equipmenttype_id = EquipmentTypes.equipmenttype_id";
	$Manager->DataSearch($searchfile);
	$Manager->Keys(undef);
	my $EquipmentTypes= $Manager->Read_ComparingInto();
	if (scalar(@{$EquipmentTypes}) != 0 ){
		my $Error = ErrorManager->new("Existen ".scalar(@{$EquipmentTypes})." Equipos
Dependientes. Debera Borrarlos antes de poder borrar este Tipo.");
		$output = $Error->Display();
	}else{
		##Delete Item from AlarmsTrigger table
		$Manager->MediaName("AlarmsTrigger");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
		##Delete Item from EquipmentTypeGetSet table
		$Manager->MediaName("EquipmentTypeGetSet");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
		##Delete Item from EquipmentTypes table
		$Manager->MediaName("EquipmentTypes");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
	}
return($output);
}

sub ModType($){
	my ($self,$search,$data) = @_; ##data is a hash reference
	my $output = "1";
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	##Modify Item from Equipments table
	$Manager->MediaName("EquipmentTypes");
	$Manager->DataSearch($search);
	$Manager->Data($data);
	$Manager->ModifyInto();

return($output);
}



sub AUTOLOAD{
	my $self = shift;
	my $type = ref($self) || die;
	my $name = $AUTOLOAD;
	$name =~ s/.*://;
	unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
	if(@_){$self->{$name}= shift}
	return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.9.7 MapManager.pm

```
package MapManager;

use strict;
use File::Copy;
```

```
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;

my %fields = (  ##Permited fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto )= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub ReadMaps($){
        my ($self,$map_id) = @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        if (defined $map_id){
                my %data=('map_id' => $map_id,);
                $Manager->DataSearch(\%data);
        }
        $Manager->MediaName("Maps");
        my $data_ref_array = $Manager->ReadInto();

return($data_ref_array);
}

sub ReadMapsMaps($){
        my ($self,$map_id)= @_;
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        $Manager->MediaName("Maps,MapsMaps");
        my $searchfile = "MapsMaps.father_map_id = \"".$map_id."\" and  MapsMaps.son_map_id =
Maps.map_id";
        $Manager->DataSearch($searchfile);
        $Manager->Keys(undef);
        my $MapsData= $Manager->Read_ComparingInto();

return($MapsData);
}
sub AddMap($){
        my ($self,$data) = @_; ##data is a hash reference
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        ##Cretate Into Maps Table
        $Manager->MediaName("Maps");
        $Manager->Data($data);
        $Manager->CreateInto();
        ##Read data from Table
        $Manager->DataSearch($data);
        $Manager->Keys(undef);
        my $data_ref_array = $Manager->ReadInto();
        my $map_id=${$data_ref_array}[0]{'map_id'};
        my $mapbmp=${$data_ref_array}[0]{'mapbmp'};
        ##Create into table MapsMonitor
        $Manager->MediaName("MapsMonitor");
        my %statedata =(
                        'map_id'=>$map_id,
                        'state_id'=> 0,
                        'changed' => 1,
```

```
		);
		$Manager->Data(\%statedata);
		$Manager->CreateInto();
		copy("$WbnmsConf::CONFPATH{'File_DB'}/BMP/Maps/$mapbmp","$WbnmsConf::CONFDRAWER{'D
rawerWebpath'}/".$map_id."\.gif");
	chmod(0777,"$WbnmsConf::CONFDRAWER{'DrawerWebpath'}/".$map_id."\.gif");
	my ($login,$pass,$uid,$gid) = getpwnam($WbnmsConf::GENERAL{'User'});
	chown($uid,$gid,"$WbnmsConf::CONFDRAWER{'DrawerWebpath'}/".$map_id."\.gif");

return(1);
}

sub DelMap($){
		my ($self,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		#####
		my @Keys=("equipment_id");
		$Manager->Keys(\@Keys);
		$Manager->MediaName("MapsEquipments");
		my %searchfile = ( 'map_id' => ${$data}{'map_id'},);
		$Manager->DataSearch(\%searchfile);
		my $Equipments = $Manager->ReadInto();
		#####
		$Manager->MediaName("MapsMaps");
		my @Keys=("son_map_id");
		$Manager->Keys(\@Keys);
		my %searchfile = ( 'father_map_id' => ${$data}{'map_id'},);
		$Manager->DataSearch(\%searchfile);
		my $Maps = $Manager->ReadInto();
		#####
		if (scalar(@{$Maps}) != 0 or scalar(@{$Equipments}) != 0 ){
			my $Error = ErrorManager->new("Existen ".scalar(@{$Equipments})." Equipos y
".scalar(@{$Maps})." Mapas Dependientes. Debera Borrarlos antes de poder borrar este Mapa.");
			$output = $Error->Display();
		}else{
			##Change Status into fathers maps
			$Manager->MediaName("MapsMaps");
			my %searchfile = ( 'son_map_id' => ${$data}{'map_id'},);
			$Manager->DataSearch(\%searchfile);
			$Manager->Keys(undef);
			my $fatherMaps=$Manager->ReadInto();
			my %statedata =(
				'changed' => 1,
				);
			$Manager->Data(\%statedata);
			$Manager->MediaName("MapsMonitor");

			foreach my $line (@{$fatherMaps}){
				my %searchfile = ( 'map_id' => ${$line}{'father_map_id'},);
				$Manager->DataSearch(\%searchfile);
				$Manager->ModifyInto();
			}
			##Delete all Children maps
			$Manager->MediaName("MapsMaps");
			my %searchfile = ( 'son_map_id' => ${$data}{'map_id'},);
			$Manager->DataSearch(\%searchfile);
			$Manager->DeleteInto();
			##Delete Item from Maps table
			$Manager->MediaName("Maps");
			$Manager->DataSearch($data);
			$Manager->DeleteInto();
			##Delete Item from MapsMonitor table
			$Manager->MediaName("MapsMonitor");
			$Manager->DataSearch($data);
			$Manager->DeleteInto();
			##Delete map File
			unlink($WbnmsConf::CONFDRAWER{'DrawerWebpath'}."/".${$data}{'map_id'}."\.gif");

		}

return($output);
}

sub ModMap($){
```

```
		my ($self,$search,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Modify Item from Equipments table
		$Manager->MediaName("Maps");
		$Manager->DataSearch($search);
		$Manager->Data($data);
		$Manager->ModifyInto();
		##Change Status into fathers maps
		$Manager->MediaName("MapsMaps");
		my %searchfile = ( 'son_map_id' => ${$search}{'map_id'},);
		$Manager->DataSearch(\%searchfile);
		$Manager->Keys(undef);
		my $fatherMaps=$Manager->ReadInto();
		my %statedata =(
			'changed' => 1,
			);
		$Manager->Data(\%statedata);
		$Manager->MediaName("MapsMonitor");

		foreach my $line (@{$fatherMaps}){
			my %searchfile = ( 'map_id' => ${$line}{'father_map_id'},);
			$Manager->DataSearch(\%searchfile);
			$Manager->ModifyInto();
		}

return($output);
}



sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.9.8 MonitorManager.pm

```
package MonitorManager;

use GD;
use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

#Call Error Manager
use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;

my %fields = (  ##Permited fields
		Debug => 0,  ## If Debug == 1 then Debug Mode
);
```

```
sub new {
	my ($proto )= @_;  #it retrieves the name of the class
	my $class= ref($proto) || $proto;
	my $self = {
		_permited => \%fields,
		%fields,
	};
	bless ($self, $class);
	return $self;
}

sub Mapsize($){
	my ($self,$map_id) = @_;
	open(MAP,$WbnmsConf::CONFDRAWER{'DrawerWebpath'}."/".$map_id."\.gif");
	my $immap = newFromGif GD::Image(\*MAP);
	close(MAP);
	my ($width,$height) = $immap->getBounds();
	my ($Mapwidth,$Mapheight) = ($width+100,$height+100);
return($Mapwidth,$Mapheight);
}

sub MapParse($){
	my ($self,$map_id) = @_;
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	$Manager->MediaName("MapsEquipments");
	my %searchfile = ( 'map_id' => $map_id,);
	$Manager->DataSearch(\%searchfile);
	my $EquipmentsMap = $Manager->ReadInto();
	foreach my $line (@{$EquipmentsMap}){
		my @Coor=split(/:/,${$line}{'coor'});

	${$line}{'coor'}=join(',',($Coor[0],$Coor[1],$Coor[0]+$WbnmsConf::CONFDRAWER{'definition'},$Coor[1]
+$WbnmsConf::CONFDRAWER{'definition'}));
	}
	$Manager->MediaName("MapsMaps");
	my %searchfile = ( 'father_map_id' => $map_id,);
	$Manager->DataSearch(\%searchfile);
	my $MapsMap = $Manager->ReadInto();
	foreach my $line (@{$MapsMap}){
		my @Coor=split(/:/,${$line}{'coor'});
		${$line}{'coor'}=join(',',($Coor[0]-
($WbnmsConf::CONFDRAWER{'definition'}*$WbnmsConf::CONFDRAWER{'coloredweigh'}),$Coor[1]-
($WbnmsConf::CONFDRAWER{'definition'}*$WbnmsConf::CONFDRAWER{'coloredweigh'}),$Coor[0]+$WbnmsC
onf::CONFDRAWER{'definition'},$Coor[1]+$WbnmsConf::CONFDRAWER{'definition'}));
	}

return($EquipmentsMap,$MapsMap);
}

sub Counter($){
	my ($self,$totalcount)=@_;
	my $output="<table><tr>";
	my $ind=0;
	while($ind < $totalcount){
		$output.="<td id=Count".$ind." > </td>";
		$ind++;
	}
	$output.="</tr></table><script>blinkit(".$totalcount.")</script>";

return($output);
}

sub AllAlarms(){
	my ($self)=@_;
	my $Manager= ABMManager->new();
	$Manager->ABMType(1);
	$Manager->MediaName("TrapsCollector");
	my $searchfile = "TrapsCollector.marked is null and TrapsCollector.trap_trigger_id is not null and
TrapsCollector.trap_processed = \"1\"";
	$Manager->DataSearch($searchfile);
	my $AllAlarms = $Manager->Read_ComparingInto();
	my $count=scalar(@{$AllAlarms});
return($count);
}
sub AUTOLOAD{
```

```
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}
 sub DESTROY{ ## AUTOLOAD search DESTROY sub

 }

1;
```

9.1.9.9 BmpManager.pm

```
package BmpManager;

use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

#Call Error Manager
use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;

my %fields = (  ##Permited fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto )= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub UpLoad($$){
        my ($self,$filename,$path) = @_;
        my @filename = split(/\\/,$filename);
        my $name = pop(@filename);
        # Copy a binary file to somewhere safe
      open (OUTFILE,">>$WbnmsConf::CONFPATH{'File_DB'}/$path$name")||die;
      while (my $bytesread=read($filename,my $buffer,1024)) {
        print OUTFILE $buffer;
      }
      chmod(0777,"$WbnmsConf::CONFPATH{'File_DB'}/$path$name");

return(1);
}

sub DelBmp($$$){
        my ($self,$bmp,$path,$type) = @_;
        my $output = "1";
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        my $errortxt;
        ##Check if are some dependent type or Map
        if ($type eq "Tipos"){
                $Manager->MediaName("EquipmentTypes");
                my %search=( 'equipmentbmp' => $bmp,);
                $Manager->DataSearch(\%search);
```

```
                $Manager->Keys(undef);
                $errortxt = "Tipos Dependientes";
        }
        if ($type eq "Mapas"){
                $Manager->MediaName("Maps");
                my %search=( 'mapbmp' => $bmp,);
                $Manager->DataSearch(\%search);
                $Manager->Keys(undef);
                $errortxt = "Mapas Dependientes";
        }
        if ($type eq "Iconos"){
                $Manager->MediaName("Maps");
                my %search=( 'mapicon' => $bmp,);
                $Manager->DataSearch(\%search);
                $Manager->Keys(undef);
                $errortxt = "Mapas Dependientes";
        }
        my $data_ref_array = $Manager->ReadInto();
        if (scalar(@{$data_ref_array}) != 0 ){
                my $Error = ErrorManager->new("Existen ".scalar(@{$data_ref_array})." $errortxt. Debera
Borrarlos antes de poder borrar este Archivo.");
                $output = $Error->Display();
        }else{
                unlink("$WbnmsConf::CONFPATH{'File_DB'}/$path".$bmp);
        }
return($output);
}

sub Seebmp($){
        my ($self,$type) = @_;
        my ($dir,$path);
        if ($type eq "Tipos"){
                $dir="$WbnmsConf::CONFPATH{'File_DB'}/BMP/Equipments";
                $path="/BMP/Equipments/";
        }
        if ($type eq "Mapas"){
                $dir="$WbnmsConf::CONFPATH{'File_DB'}/BMP/Maps";
                $path="/BMP/Maps/";
        }
        if ($type eq "Iconos"){
                $dir="$WbnmsConf::CONFPATH{'File_DB'}/BMP/MapIcons";
                $path="/BMP/MapIcons/";
        }
        opendir(BMP,$dir);
        my @bmps = grep(!/^\./,readdir(BMP));
        closedir(BMP);

return(\@bmps,$path);
}



sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.9.10 CoorManager.pm

```
package CoorManager;

use GD;
```

```
use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";  #
use ABMManager 2.0;

#Call Error Manager
use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;

my %fields = (  ##Permited fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto )= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub CoorCalculate($){
        my ($self,$map_id) = @_;
        open(MAP,$WbnmsConf::CONFDRAWER{'DrawerWebpath'}."/".$map_id."\.gif");
        my $immap = newFromGif GD::Image(\*MAP);
        close(MAP);
        my ($width,$height) = $immap->getBounds();
        my ($Vunits,$Hunits,$totalunits);
        $Vunits=int($height/$WbnmsConf::CONFDRAWER{'definition'});
        $Hunits=int($width/$WbnmsConf::CONFDRAWER{'definition'});
        $totalunits=$Vunits*$Hunits;
        my @Allcoors;
        my $row=0;
        my $column=0;
        my $ini=10;
        my ($EquipmentsCoor,$MapsCoor) = &ReadOcupatedCoors($self,$map_id);
        while($column != $Hunits){
                while($row != $Vunits){
                        my @coor;
                        if ( 0 == grep($_ eq
join(':',($ini+$column*$WbnmsConf::CONFDRAWER{'definition'},$ini+$row*$WbnmsConf::CONFDRAWER{'defini
tion'})),@{$EquipmentsCoor}) and 0 == grep($_ eq
join(':',($ini+$column*$WbnmsConf::CONFDRAWER{'definition'},$ini+$row*$WbnmsConf::CONFDRAWER{'defini
tion'})),@{$MapsCoor})){
                                        if
(($column*$WbnmsConf::CONFDRAWER{'definition'}+$WbnmsConf::CONFDRAWER{'definition'}+$ini) <=
$width and ($row*$WbnmsConf::CONFDRAWER{'definition'}+$WbnmsConf::CONFDRAWER{'definition'}+$ini <=
$height )){

            push(@coor,($ini+$column*$WbnmsConf::CONFDRAWER{'definition'},$ini+$row*$WbnmsConf::CONF
DRAWER{'definition'},$column*$WbnmsConf::CONFDRAWER{'definition'}+$WbnmsConf::CONFDRAWER{'defini
tion'},$row*$WbnmsConf::CONFDRAWER{'definition'}+$WbnmsConf::CONFDRAWER{'definition'}));
                                        push(@Allcoors,\@coor);
                                        }
                        }
                        $row++;
                }
                $row=0;
                $column++;
        }

        return(\@Allcoors);
}

sub ReadOcupatedCoors($){
        my ($self,$map_id) = @_;
```

```
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
	$Manager->MediaName("MapsEquipments");
	my %search=('map_id'=> $map_id,);
		$Manager->DataSearch(\%search);
		my @key=('coor');
		$Manager->Keys(\@key);
		my $EquipmentsCoor= $Manager->ReadInto();
		my @EquipmentsCoor;
		foreach my $coor (@{$EquipmentsCoor}){
			push(@EquipmentsCoor,${$coor}{'coor'});
		}
		$Manager->MediaName("MapsMaps");
	my %search=('father_map_id'=> $map_id,);
		$Manager->DataSearch(\%search);
		my $MapsCoor= $Manager->ReadInto();
		my @MapsCoor;
		foreach my $coor (@{$MapsCoor}){
			push(@MapsCoor,${$coor}{'coor'});
		}

return(\@EquipmentsCoor,\@MapsCoor);
}

sub FormatView($){
		my ($self,$coords) = @_;
		my @outputcoor;
		foreach my $coor (@{$coords}){
			push(@outputcoor,join(':',(${$coor}[0],${$coor}[1])));

		}
return(\@outputcoor);
}

sub AddMap($){
		my ($self,$data) = @_; ##data is a hash reference
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("MapsMaps");
		$Manager->Data($data);
		$Manager->CreateInto();
		##Read fathersMap
		$Manager->MediaName("MapsMonitor");
		my %data =(
				'changed' => 1,
		);
		$Manager->Data(\%data);
		my %search=( 'map_id' => ${$data}{'father_map_id'},);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();

return(1);
}

sub AddEquipment($){
		my ($self,$data) = @_; ##data is a hash reference
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("MapsEquipments");
		$Manager->Data($data);
		$Manager->CreateInto();
		##Read fathersMap
		$Manager->MediaName("MapsMonitor");
		my %data =(
				'changed' => 1,
		);
		$Manager->Data(\%data);
		my %search=( 'map_id' => ${$data}{'map_id'},);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();

return(1);
}

sub DelEquipment(){
		my ($self,$data) = @_; ##data is a hash reference
```

```
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Delete Item from Equipments table
		$Manager->MediaName("MapsEquipments");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
		##Read fathersMap
		$Manager->MediaName("MapsMonitor");
		my %data =(
					'changed' => 1,
		);
		$Manager->Data(\%data);
		my %search=( 'map_id' => ${$data}{'map_id'},);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();


return($output);
}

sub DelMap($){
		my ($self,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("MapsMaps");
		$Manager->DataSearch($data);
		$Manager->DeleteInto();
		##Read fathersMap
		$Manager->MediaName("MapsMonitor");
		my %data =(
					'changed' => 1,
		);
		$Manager->Data(\%data);
		my %search=( 'map_id' => ${$data}{'father_map_id'},);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();

return($output);
}

sub ModMap($$){
		my ($self,$search,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Modify Item from Equipments table
		$Manager->MediaName("MapsMaps");
		$Manager->DataSearch($search);
		$Manager->Data($data);
		$Manager->ModifyInto();
		##Read fathersMap
		$Manager->MediaName("MapsMonitor");
		my %data =(
					'changed' => 1,
		);
		$Manager->Data(\%data);
		my %search=( 'map_id' => ${$data}{'father_map_id'},);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();

return($output);
}

sub ModEquipment($$){
		my ($self,$search,$data) = @_; ##data is a hash reference
		my $output = "1";
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		##Modify Item from Equipments table
		$Manager->MediaName("MapsEquipments");
		$Manager->DataSearch($search);
		$Manager->Data($data);
		$Manager->ModifyInto();
		##Read fathersMap
```

```
		$Manager->MediaName("MapsMonitor");
		my %data =(
					'changed' => 1,
		);
		$Manager->Data(\%data);
		my %search=( 'map_id' => ${$data}{'map_id'},);
		$Manager->DataSearch(\%search);
		$Manager->ModifyInto();

return($output);
}
sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.10 ActionModules

9.1.10.1 ActionGuiManager.pm

```
package ActionGuiManager;

use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;



my %fields = (  ##Permited fields
		Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
		my ($proto )= @_;  #it retrieves the name of the class
		my $class= ref($proto) || $proto;
		my $self = {
			_permited => \%fields,
			%fields,
		};
		bless ($self, $class);
		return $self;
}

sub ReadActionUsers(){
		my ($self)= @_;
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		$Manager->MediaName("UsersActions");
		my $ActionUsersData= $Manager->ReadInto();
return($ActionUsersData);
}
```

```
sub AddActionUser($){
	my ($self,$data) = @_; ##data is a hash reference
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	##Cretate Into UsersActions Table
	$Manager->MediaName("UsersActions");
	$Manager->Data($data);
	$Manager->CreateInto();

return(1);
}

sub DelActionUser($){
	my ($self,$user_id)= @_;
	my $output = "1";
	my %data=('user_id'=>$user_id,);
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	##Delete Item from UsersActions table
	$Manager->MediaName("UsersActions");
	$Manager->DataSearch(\%data);
	$Manager->DeleteInto();
	##Delete Item from ActionsUsersEquipments table
	$Manager->MediaName("ActionsUsersEquipments");
	$Manager->DataSearch(\%data);
	$Manager->DeleteInto();

return($output);
}

sub ModActionUser($$){
	my ($self,$search,$data) = @_; ##data is a hash reference
	my $output = "1";
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	##Modify Item from UsersActions table
	$Manager->MediaName("UsersActions");
	$Manager->DataSearch($search);
	$Manager->Data($data);
	$Manager->ModifyInto();

return($output);
}


sub ReadActionAssignations(){
	my ($self)= @_;
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	$Manager->MediaName("ActionsUsersEquipments");
	my $ActionAssignationsData= $Manager->ReadInto();
	###Users
	my $UsersData=&ReadActionUsers($self);
	my %users;
	foreach my $user (@{$UsersData}){
		$users{${$user}{'user_id'}}=${$user}{'name'};
	}
	###Actions
	$Manager->MediaName("AlarmsActions");
	my $AlarmsActions= $Manager->ReadInto();
	my %actions;
	foreach my $action (@{$AlarmsActions}){
		$actions{${$action}{'action_id'}}=${$action}{'action_module_comment'};
	}
	###Equipements
	$Manager->MediaName("Equipment");
	my $Equipments= $Manager->ReadInto();
	my %equipments=('0'=>"Todos",);
	foreach my $equipment (@{$Equipments}){
		$equipments{${$equipment}{'equipment_id'}}=${$equipment}{'equipment'};
	}
	###Maps
	$Manager->MediaName("Maps");
	my $Maps= $Manager->ReadInto();
	my %maps=('0'=>"Todos",);
	foreach my $map (@{$Maps}){
```

```
                        $maps{${$map}{'map_id'}}=${$map}{'mapname'};
                }
return($ActionAssignationsData,\%users,\%actions,\%equipments,\%maps);
}

sub AddAsignationAction($){
        my ($self,$data) = @_; ##data is a hash reference
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        ##Cretate Into ActionsUsersEquipments Table
        $Manager->MediaName("ActionsUsersEquipments");
        $Manager->Data($data);
        $Manager->CreateInto();

return(1);
}

sub DelAsignationAction($){
        my ($self,$asignation_id)= @_;
        my $output = "1";
        my %data=('asignation_id'=>$asignation_id,);
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        ##Delete Item from ActionsUsersEquipments table
        $Manager->MediaName("ActionsUsersEquipments");
        $Manager->DataSearch(\%data);
        $Manager->DeleteInto();

return($output);
}

sub ModAsignationAction($$){
        my ($self,$search,$data) = @_; ##data is a hash reference
        my $output = "1";
        my $Manager = ABMManager->new();
        $Manager->ABMType(1);
        ##Modify Item from ActionsUsersEquipments table
        $Manager->MediaName("ActionsUsersEquipments");
        $Manager->DataSearch($search);
        $Manager->Data($data);
        $Manager->ModifyInto();

return($output);
}

sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.10.2 Mailer.pm

```
package Mailer;

use Net::SMTP;
use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;
```

```
my %fields = (  ##Permited fields
        UserData => undef,
        AlarmData => undef,
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my $proto= shift;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };


        bless ($self, $class);
        return $self;
}

sub SendMail(){
        my ($self,$UserData,$AlarmData)= @_;
        $self->UserData($UserData);
        $self->AlarmData($AlarmData);
        if ($self->Debug == 1){
                while(my($column,$value) = each(%{$self->UserData})){

                        print $column.": ".$value."\n";

                }
                while(my($column,$value) = each(%{$self->AlarmData})){
                        if (defined $value){
                         print $column.": ".$value."\n";
                        }
                }
        }
        ##print ${$self->UserData}{'email'};

   my $smtp = Net::SMTP->new($WbnmsConf::CONFMAILER{'server'});

   $smtp->mail($WbnmsConf::CONFMAILER{'from'});
   $smtp->to(values(%{$self->UserData}));
   $smtp->data();
   $smtp->datasend("To: ".join(',',values(%{$self->UserData})."\n");
   $smtp->datasend("Subject: Alarma en: ".${$self->AlarmData}{'equipment'}."\n");
   $smtp->datasend("*****************************\n");
   $smtp->datasend("Equipo ".${$self->AlarmData}{'equipment'}."\n");
   $smtp->datasend("Evento ".${$self->AlarmData}{'object'}."\n");
   $smtp->datasend("Severidad ".${$self->AlarmData}{'severity'}."\n");
   $smtp->datasend("*****************************\n");
   for(1..50){
        if (defined ${$self->AlarmData}{'value'.$_}){
                $smtp->datasend(""."${$self->AlarmData}{'variable'.$_}." = ".${$self-
>AlarmData}{'value'.$_}."\n");
        }else{last;}
   }
   $smtp->datasend("*****************************\n");
   $smtp->datasend($WbnmsConf::CONFMAILER{'linkroot'}."\n");
   $smtp->dataend();

   $smtp->quit;
   my @Data=(${$self->AlarmData}{'time'},join(',',values(%{$self->UserData})),${$self-
>AlarmData}{'equipment'},${$self->AlarmData}{'object'});
   &logger_Mail(@Data);
}

sub logger_Mail(@){
        my (@data_to_log) = @_;
        open
(MAILLOG,">>$WbnmsConf::CONFMAILER{'logpath'}/$WbnmsConf::CONFMAILER{'mailer.log'}");
        foreach(@data_to_log){
                print MAILLOG "$_;";
        }
              print MAILLOG "\n";
        close(MAILLOG);
}
```

```
sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.10.3 Trapper.pm

```
package Trapper;

use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##


use SNMP_util "0.86";

#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

my %fields = (  ##Permited fields
      UserData => undef,
      AlarmData => undef,
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my $proto= shift;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };


        bless ($self, $class);
        return $self;
}

sub SendTrap(){
        my ($self,$trap) = @_;
        snmptrap(${$self->UserData}{'community'}."\@".${$self->UserData}{'IP'},${$self-
>AlarmData}{'object'},${$self->AlarmData}{'hosts'},6,1,@{$trap});

}

sub TrapManager(){
        my ($self,$UserData,$AlarmData) =  @_;
        $self->UserData($UserData);
        $self->AlarmData($AlarmData);
        my $Manager = ABMManager->new();
                $Manager->Debug(0);
        $Manager->ABMType(1);
        $Manager->MediaName("Oid2Label");
        my %Search = (oid_label => ${$self->AlarmData}{'object'});
        $Manager->DataSearch(\%Search);
        my $data_ref_array =$Manager->ReadInto();
```

```
		$SNMP_util::OIDS{${$self->AlarmData}{'object'}}= ${$data_ref_array}[0]{'oid'};

	my @trap;
		for(1..50){
			if (defined ${$self->AlarmData}{'variable'.$_}){
				my %Search = (oid_label => ${$self->AlarmData}{'variable'.$_});
				$Manager->DataSearch(\%Search);
				my $data_ref_array =$Manager->ReadInto();
				$SNMP_util::OIDS{${$self->AlarmData}{'variable'.$_}}= ${$data_ref_array}[0]{'oid'};
				push(@trap,${$self->AlarmData}{'variable'.$_},"STRING",${$self-
>AlarmData}{'value'.$_});
			}else{
				last;
			}
	}
	&SendTrap($self,\@trap);

}


sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.10.4 SMSMessenger.pm

```
package SMSMessenger;

use sendSMS;
use strict;
use vars qw($VERSION $AUTOLOAD);

$VERSION = '0.3';     ##
#Configure file
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

my %fields = (  ##Permited fields
		UserData => undef,
		AlarmData => undef,
		Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
		my $proto= shift;  #it retrieves the name of the class
		my $class= ref($proto) || $proto;
		my $self = {
			_permited => \%fields,
			%fields,
		};
		bless ($self, $class);
		return $self;
}

sub SendSMS{
		my ($self,$UserData,$AlarmData)= @_;
		$self->UserData($UserData);
		$self->AlarmData($AlarmData);
	my $msg= ${$self->AlarmData}{'equipment'}."\n";
	$msg=$msg."Alarma ".${$self->AlarmData}{'object'}."\n";
	$msg=$msg."Severidad ".${$self->AlarmData}{'severity'}."\n";
```

```
	for(1..50){
			if (defined ${$self->AlarmData}{'value'.$_}){
				$msg=$msg."    ".${$self->AlarmData}{'variable'.$_}." = ".${$self-
>AlarmData}{'value'.$_}."\n";
			}else{last;}
	}
		####sendSMS($operator,$destAC,$destNr,$from,$msg);

			&logger_SMS($msg,${$self->UserData}{'phone'});
			sendSMS(${$self->UserData}{'phoneop'},${$self->UserData}{'phonearea'},${$self-
>UserData}{'phone'},$WbnmsConf::CONFSMS{'from'},$msg);


}

sub logger_SMS($$){
	my ($data_to_log,$phone) = @_;
	open (SMSLOG,">>$WbnmsConf::CONFSMS{'logpath'}/$WbnmsConf::CONFSMS{'sms.log'}");
		print SMSLOG $phone."\n";
		print SMSLOG "---------------------------------\n";
		print SMSLOG $data_to_log;
		print SMSLOG "---------------------------------\n";
	close(SMSLOG);
}

sub AUTOLOAD{
	my $self = shift;
	my $type = ref($self) || die;
	my $name = $AUTOLOAD;
	$name =~ s/.*://;
	unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
	if(@_){$self->{$name}= shift}
	return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.11 MIBCompiler

9.1.11.1 MIBManager.pm

```
package MIBManager;


use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '2.0';     ## with DBRead function

use Time::localtime;

use mibmysql;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Error';
use ErrorManager;


my %fields = (  ##Permited fields
	MIB => undef,
	Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
	my $proto= shift;  #it retrieves the name of the class
```

```
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub ReadMIBs(){
  my ($self)= @_;
    opendir(MIBDIR,$WbnmsConf::CONFPATH{'MIBDir'});
    my @mibs = grep(!/^\./,readdir(MIBDIR));
    closedir(MIBDIR);
    open(MIBS,$WbnmsConf::CONFMIBS{'MIBIndex'});
    my @mibsloaded;
    while(<MIBS>){
                chomp();
                push(@mibsloaded,$_);
    }
    close(MIBS);
return(\@mibs,\@mibsloaded);
}

sub AddMIBs($){
        my ($self,$mibloaded)= @_;
      push(@{$mibloaded},$self->MIB());
      open(MIBS,">$WbnmsConf::CONFMIBS{'MIBIndex'}");
      foreach my $mib (@{$mibloaded}){
                print MIBS $mib."\n";
      }
      close(MIBS);


return(1);
}

sub DelMIBs(){
        my ($self,$mibloaded)= @_;
        open(MIBS,">$WbnmsConf::CONFMIBS{'MIBIndex'}");
        my @undelmibs = grep($_ ne $self->MIB(),@{$mibloaded});
      foreach my $mib (@undelmibs){
                print MIBS $mib."\n";
      }
      close(MIBS);

return(\@undelmibs);
}

sub MibLoad(){
        my ($self)= @_;
        my $username = $WbnmsConf::CONFDB{'DataBaseUserName'};
        my $password = $WbnmsConf::CONFDB{'DataBasePassword'};
        my $database = $WbnmsConf::CONFDB{'DataBaseName'};
        my $host = $WbnmsConf::CONFDB{'DataBaseHost'};
        mib2mysql("$WbnmsConf::CONFPATH{'MIBDir'}/".$self-
>MIB(),$database,$host,$username,$password);
        my $error;
        if (defined $mibmysql::errormessage){
                my $Error=ErrorManager->new($mibmysql::errormessage);
                $error = $Error->Display();
        }

return($error);
}

sub UpLoad($){
        my ($self,$filename) = @_;
        my @filename = split(/\\/,$filename);
        my $name = pop(@filename);
        # Copy a binary file to somewhere safe
      open (OUTFILE,">>$WbnmsConf::CONFPATH{'MIBDir'}/$name")||die;
      while (my $bytesread=read($filename,my $buffer,1024)) {
        print OUTFILE $buffer;
      }
      chmod(0777,"$WbnmsConf::CONFPATH{'MIBDir'}/$name");
```

```
return(1);
}

sub Time(){
        my ($self)= @_;
        my $year = localtime->year() + 1900;
        my $mon = localtime->mon() + 1;
        my $time= localtime->mday()."\/".$mon."\/".$year."::".localtime->hour().":".localtime->sec().":". localtime-
>min();
return($time)
}

sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.11.2 mibmysql.pm

```
package mibmysql;
#######This module is for mib compilation.
#######Sintax: mib2mysql("Mib file",database,host,username,password)
require 5.004;

use strict;
use vars qw(@ISA @EXPORT $VERSION);
use Exporter;
use Carp;

use DBI;

use object_identifier;
use trap_type;
use notification_type;
use object_type;
use sequence;
use textual_conventions;



$VERSION = '0.86';

@ISA = qw(Exporter);

@EXPORT = qw(mib2mysql);


$mibmysql::Debug=0;
$mibmysql::errormessage;

sub mib2mysql($$$$$);

sub mib2mysql($$$$$) {
        my ($mib,$database,$host,$username,$password) = @_;

my $data_source = "DBI:mysql:$database:$host";
my $dbh = DBI->connect( $data_source, $username, $password);
print ">>>>>>>>>>>>Object-Identifier\n" if $mibmysql::Debug == 1;
snmpMIB_to_OI($mib);

if ( defined $object_identifier::errormessage){
```

```
        $mibmysql::errormessage= $object_identifier::errormessage;
        goto "EXIT";

}

######################
snmpMIB_to_TT($mib);
print ">>>>>>>>>>>>Trap-Types\n" if $mibmysql::Debug == 1;

foreach(keys(%trap_type::ObjectData)){

                my $TT=$_;
                my $enterprise;
                my $specifictrap;
                my $description;
                my @objects;

                        my $ref=$trap_type::ObjectData{$_};
                        foreach(keys(%$ref)){
                                if ($_=~/VARIABLES/){
                                        my $ref2=$$ref{$_};

                                        foreach(@$ref2){
                                                push(@objects,$_);


                                                }
                                        }else{
                                        if ($_=~/ENTERPRISE/){
                                                $enterprise = $$ref{$_};

                                        }
                                        if ($_=~/SPECIFICTRAP/){
                                                $specifictrap = $$ref{$_};

                                        }
                                        if ($_=~/DESCRIPTION/){
                                                $description = $$ref{$_};


                                        }


                                        }
                                }

                                my $sql = $dbh->quote($description);
                                $dbh->do("INSERT INTO TrapType
(oid_label,enterprise,trapspecificID,description,object1,object2,object3,object4,object5,object6,object7,object8,obj
ect9,object10) VALUES (
\"$TT\",\"$enterprise\",\"$specifictrap\",$sql,\"$objects[0]\",\"$objects[1]\",\"$objects[2]\",\"$objects[3]\",\"$objects[4]
\",\"$objects[5]\",\"$objects[6]\",\"$objects[7]\",\"$objects[8]\",\"$objects[9]\")") || last;
                                my $oid = $trap_type::OIDS{$TT};
                                my $object_type= 2; ##Trap Type
                                $dbh->do("INSERT INTO Oid2Label (oid,oid_label,object_type) VALUES (
\"$oid\",\"$TT\",\"$object_type\")") || last;
}


#################################################################
snmpMIB_to_NT($mib);
print ">>>>>>>>>>>>Notification-Types\n" if $mibmysql::Debug == 1;

foreach(keys(%notification_type::ObjectData)){

                my $NT=$_;
                my $status;
                my $description;
                my @objects;

                        my $ref=$notification_type::ObjectData{$_};
                        foreach(keys(%$ref)){
                                if ($_=~/OBJECTS/){
                                        my $ref2=$$ref{$_};

                                        foreach(@$ref2){
```

```
                                        push(@objects,$_);


                                        }
                                }else{
                                if ($_=~/STATUS/){
                                        $status = $$ref{$_};

                                }
                                if ($_=~/DESCRIPTION/){
                                        $description = $$ref{$_};


                                }


                                }
                        }

                        my $sql = $dbh->quote($description);
                $dbh->do("INSERT INTO NotificationType
(oid_label,status,description,object1,object2,object3,object4,object5,object6,object7,object8,object9,object10)
VALUES (
\"$NT\",\"$status\",$sql,\"$objects[0]\",\"$objects[1]\",\"$objects[2]\",\"$objects[3]\",\"$objects[4]\",\"$objects[5]\",\"$
objects[6]\",\"$objects[7]\",\"$objects[8]\",\"$objects[9]\")") || last;
                my $oid = $notification_type::OIDS{$NT};
                my $object_type= 0; ##Notification Type


                $dbh->do("INSERT INTO Oid2Label (oid,oid_label,object_type) VALUES (
\"$oid\",\"$NT\",\"$object_type\")") || last;
}


#####################################################################
snmpMIB_to_OT($mib);
print ">>>>>>>>>>>>>TEXTUAL-CONVENTIONS Defined at Object-Type\n" if $mibmysql::Debug == 1;

foreach (keys(%object_type::AlarmTypes)){
        my $TC=$_;
        my $type_id=1;
        my @ind = $object_type::AlarmTypes{$_};
       foreach my $ref(@ind){
                my @ind2 = keys(%$ref);
                foreach(@ind2){
                        my $value= $_;
                        my $value_label= $$ref{$_};

                         $dbh->do("INSERT INTO TextualConventions
(convention_label,convention_value,convention_value_label) VALUES ( \"$TC\",\"$value\",\"$value_label\")") ||
last ;
                        }
        }
        $dbh->do("INSERT INTO Syntax (type_id,type_label) VALUES ( \"$type_id\",\"$TC\")") || last;
}
print ">>>>>>>>>>>>Object-Type´s Data\n"  if $mibmysql::Debug == 1;



foreach(keys(%object_type::ObjectData)){

        my $OT=$_;
        my $status;
        my $description;
        my $maxaccess;
        my $syntax;
        my $type_id;
        my @indexes;

                my $ref=$object_type::ObjectData{$_};
                foreach(keys(%$ref)){
                        if ($_=~/INDEX/){
                                my $ref2=$$ref{$_};

                                foreach(@$ref2){
                                        push(@indexes,$_);
```

```
                                        }
                                }else{
                                if ($_=~/STATUS/){
                                        $status = $$ref{$_};

                                }
                                if ($_=~/DESCRIPTION/){
                                        $description = $$ref{$_};


                                }

                                if ($_=~/MAX-ACCESS/){
                                        $maxaccess = $$ref{$_};
                                }
                                if ($_=~/SYNTAX/){
                                        $syntax = $$ref{$_};
                                }
                                }
                        }
                        my $sql = $dbh->quote($description);

                $dbh->do("INSERT INTO ObjectType
(oid_label,max_access,status,description,index1,index2,index3,index4,index5,index6,index7,index8,index9,index
10) VALUES (
\"$OT\",\"$maxaccess\",\"$status\",$sql,\"$indexes[0]\",\"$indexes[1]\",\"$indexes[2]\",\"$indexes[3]\",\"$indexes[4]\
",\"$indexes[5]\",\"$indexes[6]\",\"$indexes[7]\",\"$indexes[8]\",\"$indexes[9]\")") || last;
                my $oid = $object_type::OIDS{$OT};
                my $object_type= 1; ##Object Type
                if ($syntax=~/SEQUENCE/){
                        my @ind = split(/:/,$syntax);
                        $syntax=$ind[1];
                        }


                $dbh->do("INSERT INTO Oid2Label (oid,oid_label,object_type,syntax) VALUES (
\"$oid\",\"$OT\",\"$object_type\",\"$syntax\")") || last;
}


######################################################################
snmpMIB_to_S($mib);
print ">>>>>>>>>>>>Sequences\n" if $mibmysql::Debug == 1;

my $type_id = 2;
foreach (keys(%sequence::sequence)){
        my $S=$_;
        my @ind = $sequence::sequence{$_};
        foreach my $ref(@ind){
                my @ind2 = keys(%$ref);
                foreach(@ind2){
                        my $object= $_;
                        my $type= $$ref{$_};
                         $dbh->do("INSERT INTO Sequences
(sequence_label,sequence_object,sequence_type) VALUES ( \"$S\",\"$object\",\"$type\")") || last;

                        }

        }
        $dbh->do("INSERT INTO Syntax (type_id,type_label) VALUES ( \"$type_id\",\"$S\")") || last;
}
######################################################################
snmpMIB_to_TC($mib);
print ">>>>>>>>>>>>Textual Conventions\n" if $mibmysql::Debug == 1;

my $type_id = 1;
foreach (keys(%textual_conventions::AlarmTypes)){
        my $TC=$_;
        my @ind = $textual_conventions::AlarmTypes{$_};
        foreach my $ref(@ind){
                my @ind2 = keys(%$ref);
                foreach(@ind2){
                        my $value= $_;
                        my $value_label= $$ref{$_};
```

```
                                        $dbh->do("INSERT INTO TextualConventions
(convention_label,convention_value,convention_value_label) VALUES ( \"$TC\",\"$value\",\"$value_label\")") ||
last;

                                    }
            }
            $dbh->do("INSERT INTO Syntax (type_id,type_label) VALUES ( \"$type_id\",\"$TC\")") || last;
}
 $dbh->disconnect();

EXIT: print "END\n" if $mibmysql::Debug == 1;
} ##End of Sub
1;
```

9.1.11.3 object_identifier.pm

```
package object_identifier;

require 5.004;

use strict;
use vars qw(@ISA @EXPORT $VERSION);
use Exporter;
use Carp;

use BER "0.82";
use SNMP_Session "0.83";
use Socket;
use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;

$VERSION = '0.86';

@ISA = qw(Exporter);

@EXPORT = qw(snmpMIB_to_OI);



sub snmpMIB_to_OI ($);


%object_identifier::ObjectData;
$object_identifier::Debug = 0;
%object_identifier::OIDS=('iso' => '1');
%object_identifier::parent;
%object_identifier::number=('iso' => '1');
$object_identifier::errormessage;

sub snmpMIB_to_OI ($) {
            my ($arg)= @_;
            my %logOID;
            my %logparent;
            my %lognumber;
            open (OI,"$WbnmsConf::CONFPATH{'var'}/ObjectIdentifiers");
            while(<OI>){
                        my @ind = split(/:/);
                        $logOID{$ind[0]} = $ind[1];
                        chomp($ind[2]);
                        $logparent{$ind[0]} = $ind[2];
                        my @indOID = split(/\./,$ind[1]);
                        my $len= scalar(@indOID);
                        $lognumber{$ind[0]}= $indOID[$len-1];
            }


            open (OIMIB,$arg);
            my $moduleidentityflag = 0;
            my $moduleidentityflag2 = 0;
            my $moduleidentity;
            while(<OIMIB>){
```

```
                        if ($moduleidentityflag == 1){
                                if ($_=~ /LAST-UPDATED/){
                                        $moduleidentityflag2 = 1;
                                }
                                if ($_=~ /::=/ and $moduleidentityflag2 == 1){
                                        $_=$moduleidentity.$_;
                                        $moduleidentityflag = 0;
                                        $moduleidentityflag2 = 0;
                                }
                        }
                        if ($_=~ /OBJECT IDENTIFIER/ and $_=~/::=/ and $_!~/--/){
                                my @ind = split(/::=/);
                                $ind[1]=~ tr/\{/ /;
                                $ind[1]=~ tr/\}/ /;
                                my @ind2 = split(' ',$ind[1]);
                                my @ind3 = split(' ',$ind[0]);
                                if (!defined $logOID{$ind3[0]}){
                                        $object_identifier::parent{$ind3[0]}= $ind2[0];
                                        $object_identifier::number{$ind3[0]}= $ind2[1];
                                }else{
                                        $object_identifier::parent{$ind3[0]}= $logparent{$ind3[0]};
                                        $object_identifier::number{$ind3[0]}= $lognumber{$ind3[0]};
                                }
                        }
                        if ($_=~ /MODULE-IDENTITY/ and $_!~ /\,/ and $_!~ /::=/ and $_!~/--/){
                                $moduleidentityflag = 1; ##Activate the module identity search
                                $_=~ s/MODULE-IDENTITY/OBJECT IDENTIFIER/;
                                $moduleidentity=$_;
                        }
                }
                foreach(keys(%logparent)){
                        if(!defined $object_identifier::parent{$_}){
                                $object_identifier::parent{$_}= $logparent{$_};
                        }
                }
                foreach(keys(%lognumber)){
                        if(!defined $object_identifier::number{$_}){
                                $object_identifier::number{$_}= $lognumber{$_};
                        }
                }
        close(OIMIB);
        foreach  (keys(%object_identifier::parent)){
                if(!defined $logparent{$_}){
                my $hijo = $_;
                my $OIDlabel =$_;
                my $safe = 0;
                        while($object_identifier::parent{$hijo}!~ /iso/ and $OIDlabel ne "iso" ){
                                $OIDlabel=$OIDlabel.".".$object_identifier::parent{$hijo};
                                $hijo = $object_identifier::parent{$hijo};
                                if ($safe > 80){
                                        $object_identifier::errormessage="Debera cargar la MIB previa.";
                                        goto "EXIT";
                                }
                                $safe++;
                        }
                $OIDlabel=$OIDlabel."."."iso";
                my @oid = split(/\./,$OIDlabel);

                @oid = reverse(@oid);
                my $ind = 0;
                        foreach(@oid){
                        $oid[$ind] = $object_identifier::number{$_};
                        $ind++;
                        }
                $OIDlabel= join('.',@oid);
                $object_identifier::OIDS{$_} = $OIDlabel;

                }else{
                $object_identifier::OIDS{$_}=$logOID{$_};
                }
        }
close(OI);
open(OI,">$WbnmsConf::CONFPATH{'var'}/ObjectIdentifiers");
close(OI);
open (OI,">>$WbnmsConf::CONFPATH{'var'}/ObjectIdentifiers");
      foreach(keys(%object_identifier::OIDS)){
```

```
                if ($_!~ /--/){
                        print OI $_.":".$object_identifier::OIDS{$_}.":".$object_identifier::parent{$_}."\n";
                        }
}
close(OI);
EXIT: print "END\n" if $object_identifier::Debug == 1;
                }
1;
```

9.1.11.4 object_type.pm

```
package object_type;

require 5.004;

use strict;
use vars qw(@ISA @EXPORT $VERSION);
use Exporter;
use Carp;

use BER "0.82";
use SNMP_Session "0.83";
use Socket;
use lib "/var/Wbnms/Wbnms_Conf";
use WbnmsConf;

$VERSION = '0.86';

@ISA = qw(Exporter);

@EXPORT = qw(snmpMIB_to_OT);



sub snmpMIB_to_OT ($);
sub readObjectIdentifier;

%object_type::AlarmTypes;
%object_type::ObjectData;
$object_type::TableIndex;
$object_type::Debug = 0;
%object_type::Links;
%object_type::OIDS;

sub snmpMIB_to_OT ($) {
  my($arg) = @_;
  my($quote, $buf, $var, $code, $val, $tmp, $tmpv, $strt);
  my($ret);
  readObjectIdentifier;
  my(%Link) = %object_type::Links;

  if (!open(MIB, $arg)) {
    carp "snmpMIB_to_OT: Can't open $arg: $!"
      unless ($SNMP_Session::suppress_warnings > 1);
    return -1;
  }
  print "snmpMIB_to_OT: loading $arg\n" if $object_type::Debug;
  $ret = 0;
  while(<MIB>) {
    s/--.*--//g;            # throw away comments (-- anything --)
    s/--.*//;               # throw away comments (-- anything EOL)
    if ($quote) {
      next unless /"/;
      $quote = 0;
    }
    chop;
    $buf .= ' ' . $_;
    $buf =~ s/\s+/ /g;

    if ($buf =~ / DEFINITIONS ::= BEGIN/) {
      undef %Link;
      %Link = %object_type::Links;
```

```
  }
  $buf =~ s/OBJECT-TYPE/OBJECT IDENTIFIER/;
  #$buf =~ s/NOTIFICATION-TYPE/OBJECT IDENTIFIER/;
  #$buf =~ s/TEXTUAL-CONVENTION/OBJECT IDENTIFIER/;
  $buf =~ s/OBJECT-IDENTITY/OBJECT IDENTIFIER/;
  $buf =~ s/MODULE-IDENTITY/OBJECT IDENTIFIER/;
  $buf =~ s/ IMPORTS .*\;//;
  $buf =~ s/ SEQUENCE {.*}//;
  $buf =~ s/ SYNTAX .*//;
  $buf =~ s/ [\w-]+ ::= OBJECT IDENTIFIER//;
  $buf =~ s/ OBJECT IDENTIFIER .* ::= {/ OBJECT IDENTIFIER ::= {/;
  $buf =~ s/".*"//;
  if ($buf =~ /"/) {
    $quote = 1;
  }

  if ($buf =~ / ([\w\-]+) OBJECT IDENTIFIER ::= {([^}]+)}/) {
    $var = $1;
    $buf = $2;
    undef $val;
    $buf =~ s/ +$//;
    ($code, $val) = split(' ', $buf, 2);

    if (!defined($val) || (length($val) <= 0)) {
            $object_type::OIDS{$var} = $code;
            $ret++;
            print "'$var' => '$code'\n" if $object_type::Debug;
    } else {
            $strt = $code;
            while($val =~ / /) {
              ($tmp, $val) = split(' ', $val, 2);
              if ($tmp =~ /([\w\-]+)\((\d+)\)/) {
                $tmp = $1;
                $tmpv = "$object_type::OIDS{$strt}.$2";
                $Link{$tmp} = $strt;
                if (defined($object_type::OIDS{$tmp})) {
                  if ($tmpv ne $object_type::OIDS{$tmp}) {
                          $strt = "$strt.$tmp";
                          $object_type::OIDS{$strt} = $tmpv;
                          $ret++;
                  }
                } else {
                  $object_type::OIDS{$tmp} = $tmpv;
                  $ret++;
                  $strt = $tmp;
                }
              }
            }

            if (!defined($object_type::OIDS{$strt})) {
              carp "snmpMIB_to_OT: $arg: \"$strt\" prefix unknown, load the parent MIB first.\n"
                unless ($SNMP_Session::suppress_warnings > 1);
            }
            $Link{$var} = $strt;
            $val = "$object_type::OIDS{$strt}.$val";
            if (defined($object_type::OIDS{$var})) {
              if ($val ne $object_type::OIDS{$var}) {
                $var = "$strt.$var";
              }
            }

            $object_type::OIDS{$var} = $val;
            $ret++;

            print "'$var' => '$val'\n" if $object_type::Debug;
    }
    undef $buf;
  }
 }
 close(MIB);


##################################################
#################   SYNTAX   ####################
```

```
 ####Flags
 my $SearchOBJECTTYPE = 0;
 my $SearchDESCRIPTION = 0;
 ##my @syntax;
 my $SearchSYNTAX = 0;
 my $SearchtestSYNTAX = 0;
 #########

my %alarmsTgroup;


foreach  my $TEXT(keys(%object_type::OIDS)){
my @ObjectTableINDEX;
my @alarmsT;
my $DESCRIPTION;
my %ObjectType;
my %alarmsTgroup;

open(MIB, $arg);
 while(<MIB>) {
 if ($_=~/$TEXT/ and $_=~/OBJECT-TYPE/){$SearchOBJECTTYPE = 1;} ## Is or not an Object-Type
 if ($SearchOBJECTTYPE == 1){

        if ( $_=~/SYNTAX/){
                my @syntax=split();
                if ($syntax[2]=~/\{/){$SearchSYNTAX = 1;}elsif(!defined $syntax[2]){$SearchtestSYNTAX =
1;}
                if ($syntax[1]=~/SEQUENCE/){
                        $ObjectType{'SYNTAX'} =$syntax[1].":".$syntax[3];

                }else{
                        if ($_=~/\(*\)/ and $_!~/{/){ ## Type Definition
                                shift(@syntax);
                            $ObjectType{'SYNTAX'} = join('',@syntax);
                        }else{
                            $ObjectType{'SYNTAX'} = $syntax[1];
                      }

                }
        }
        if ($_=~/\{/ and $SearchtestSYNTAX == 1 ){
                        $SearchSYNTAX = 1;
                        $SearchtestSYNTAX = 0;
        }
        if($SearchSYNTAX == 1){
                    if ($_=~ /\}/){
                       $SearchSYNTAX = 0;
                       next;
                    }
                    $_=~ tr/\(/ (/;
                if ($_=~ /\(*\)/){
                        my @ind;
                        @ind= split();

                          if ($ind[0]!~ /--/){
                                $ind[1]=~ tr/\)\,\(//d;

                                $alarmsTgroup{$ind[1]}=$ind[0];
                          }

                        }
    }

        if ($_=~/ACCESS/){
                my @ind;
                @ind= split();
                $ObjectType{'MAX-ACCESS'} = $ind[1];  ##MAX-ACCESS for SNMPv2 and ACCESS for
SNMPv1

                $SearchtestSYNTAX = 0;
        }
        if ($_=~/STATUS/){
                my @ind;
                @ind= split();
                $ObjectType{'STATUS'} = $ind[1];
```

```
		}
		if ($SearchDESCRIPTION == 1){
			if ($_=~/::=/ or $_=~/INDEX/){
				if($_=~/INDEX/){
					$_=~ tr/\{/ /;
					$_=~ tr/\,/ /;
					$_=~ tr/\}/ /;
					my @ind = split();
					shift(@ind);
					@ObjectTableINDEX=@ind;
					$ObjectType{'INDEX'} = \@ind;
				}
			$ObjectType{'DESCRIPTION'} = $DESCRIPTION;
				undef $DESCRIPTION;
				$SearchDESCRIPTION = 0;

				last;
			}else{
				$_=~s/\"/ /g;
				$DESCRIPTION = $DESCRIPTION.$_;
			}
		}
		if ($_=~/DESCRIPTION/){$SearchDESCRIPTION = 1;}
 }
 }
  close(MIB);

  if ($SearchOBJECTTYPE == 1){
		if(defined %alarmsTgroup){$object_type::AlarmTypes{$TEXT}=\%alarmsTgroup;} ##Textual
Conventions defined at Sintax label
   $object_type::ObjectData{$TEXT}=\%ObjectType;   ##Object's data
   $object_type::TableIndex{$TEXT}=\@ObjectTableINDEX;
   $SearchOBJECTTYPE = 0;
  }
}

 ###############End################
 return $ret;

}
sub readObjectIdentifier{
		my %logOID;
		my %logparent;
		open (OI,"$WbnmsConf::CONFPATH{'var'}/ObjectIdentifiers");
		while(<OI>){
			my @ind = split(/:/);
			$logOID{$ind[0]} = $ind[1];
			chomp($ind[2]);
			$logparent{$ind[0]} = $ind[2];
		}
		close(OI);
		%object_type::OIDS = %logOID;
		%object_type::Links =%logparent;
		}
1;
```

9.1.11.5 notification_type.pm

```
package notification_type;

require 5.004;

use strict;
use vars qw(@ISA @EXPORT $VERSION);
use Exporter;
use Carp;

use BER "0.82";
use SNMP_Session "0.83";
use Socket;

use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;
```

```
$VERSION = '0.86';

@ISA = qw(Exporter);

@EXPORT = qw(snmpMIB_to_NT);



sub snmpMIB_to_NT ($);
sub readObjectIdentifier;


%notification_type::ObjectData;
$notification_type::Debug = 0;
%notification_type::Links;
%notification_type::OIDS;

sub snmpMIB_to_NT ($) {
  my($arg) = @_;
  my($quote, $buf, $var, $code, $val, $tmp, $tmpv, $strt);
  my($ret);
  readObjectIdentifier;
  my(%Link) = %notification_type::Links;


  if (!open(MIB, $arg)) {
    carp "snmpMIB_to_NT: Can't open $arg: $!"
      unless ($SNMP_Session::suppress_warnings > 1);
    return -1;
  }
  print "snmpMIB_to_NT: loading $arg\n" if $notification_type::Debug;
  $ret = 0;
  while(<MIB>) {
    s/--.*--//g;            # throw away comments (-- anything --)
    s/--.*//;               # throw away comments (-- anything EOL)
    if ($quote) {
      next unless /"/;
      $quote = 0;
    }
    chop;

    $buf .= ' ' . $_;
    $buf =~ s/\s+/ /g;

    if ($buf =~ / DEFINITIONS ::= BEGIN/) {
      undef %Link;
      %Link = %notification_type::Links;

    }
    #$buf =~ s/OBJECT-TYPE/OBJECT IDENTIFIER/;
    $buf =~ s/NOTIFICATION-TYPE/OBJECT IDENTIFIER/;
    #$buf =~ s/TEXTUAL-CONVENTION/OBJECT IDENTIFIER/;
    $buf =~ s/OBJECT-IDENTITY/OBJECT IDENTIFIER/;
    $buf =~ s/MODULE-IDENTITY/OBJECT IDENTIFIER/;
    $buf =~ s/ IMPORTS .*\;//;
    $buf =~ s/ SEQUENCE {.*}//;
    $buf =~ s/ SYNTAX .*//;
    $buf =~ s/ [\w-]+ ::= OBJECT IDENTIFIER//;
    $buf =~ s/ OBJECT IDENTIFIER .* ::= {/ OBJECT IDENTIFIER ::= {/;
    $buf =~ s/".*"//;
    if ($buf =~ /"/) {
      $quote = 1;
    }

    if ($buf =~ / ([\w\-]+) OBJECT IDENTIFIER ::= {([^}]+)}/) {
      $var = $1;
      $buf = $2;
      undef $val;
      $buf =~ s/ +$//;
      ($code, $val) = split(' ', $buf, 2);

      if (!defined($val) || (length($val) <= 0)) {
              $notification_type::OIDS{$var} = $code;
              $ret++;
              print "'$var' => '$code'\n" if $notification_type::Debug;
```

```
    } else {
            $strt = $code;
            while($val =~ / /) {
              ($tmp, $val) = split(' ', $val, 2);
              if ($tmp =~ /([\w\-]+)\((\d+)\)/) {
                $tmp = $1;
                $tmpv = "$notification_type::OIDS{$strt}.$2";
                $Link{$tmp} = $strt;
                if (defined($notification_type::OIDS{$tmp})) {
                  if ($tmpv ne $notification_type::OIDS{$tmp}) {
                          $strt = "$strt.$tmp";
                          $notification_type::OIDS{$strt} = $tmpv;
                          $ret++;
                  }
                } else {
                  $notification_type::OIDS{$tmp} = $tmpv;
                  $ret++;
                  $strt = $tmp;
                }
              }
            }

            if (!defined($notification_type::OIDS{$strt})) {
              carp "snmpMIB_to_NT: $arg: \"$strt\" prefix unknown, load the parent MIB first.\n"
                unless ($SNMP_Session::suppress_warnings > 1);
            }
            $Link{$var} = $strt;
            $val = "$notification_type::OIDS{$strt}.$val";
            if (defined($notification_type::OIDS{$var})) {
              if ($val ne $notification_type::OIDS{$var}) {
                $var = "$strt.$var";
              }
            }

            $notification_type::OIDS{$var} = $val;
            $ret++;

            print "'$var' => '$val'\n" if $notification_type::Debug;
    }
    undef $buf;
  }
}
close(MIB);


#####################################################
#################   SYNTAX   ####################


 ####Flags
 my $SearchNOTIFICATIONTYPE = 0;
 my $SearchDESCRIPTION = 0;
 my $Searchobjects = 0;
 #########



foreach  my $TEXT(keys(%notification_type::OIDS)){

my @alarmsT;
my %ObjectType;
my @alarmsTgroup;
open(MIB, $arg);
  while(<MIB>) {

  if ($_=~/$TEXT/ and $_=~/NOTIFICATION-TYPE/){$SearchNOTIFICATIONTYPE = 1;} ## Is or not an Object-
Type
  if ($SearchNOTIFICATIONTYPE == 1){
            if ( $_=~/OBJECTS/){
                    $_=~ tr/\{/ /;
                    $_=~ tr/\,/ /;
                    $_=~ tr/\}/ /;

                    my @objects=split();
                    shift(@objects);
                    @alarmsTgroup=@objects;
```

```
                        $ObjectType{'OBJECTS'}=\@alarmsTgroup;
                        }
                if ($_=~/STATUS/){
                        my @ind;
                        @ind= split();
                        $ObjectType{'STATUS'} = $ind[1];

                }
                if ($SearchDESCRIPTION == 1){

                        if ($_=~/::=/ or $_=~/INDEX/){


                        $SearchDESCRIPTION = 0;


                        last;
                        }else{

                        $ObjectType{'DESCRIPTION'} = $ObjectType{'DESCRIPTION'}.$_;
                        }
                }
                if ($_=~/DESCRIPTION/){$SearchDESCRIPTION = 1;}
 }
 }
  close(MIB);
   if ($SearchNOTIFICATIONTYPE == 1){

   $notification_type::ObjectData{$TEXT}=\%ObjectType;   ##Object's data
   $SearchNOTIFICATIONTYPE = 0;
   }
}

 ###############End################
 return $ret;

}

sub readObjectIdentifier{
        my %logOID;
        my %logparent;
        open (OI,"$WbnmsConf::CONFPATH{'var'}/ObjectIdentifiers");
        while(<OI>){
                my @ind = split(/:/);
                $logOID{$ind[0]} = $ind[1];
                chomp($ind[2]);
                $logparent{$ind[0]} = $ind[2];
        }
        close(OI);
        %notification_type::OIDS = %logOID;
        %notification_type::Links =%logparent;
        }

1;
```

9.1.11.6 trap_type.pm

```
package trap_type;

require 5.004;

use strict;
use vars qw(@ISA @EXPORT $VERSION);
use Exporter;
use Carp;

use BER "0.82";
use SNMP_Session "0.83";
use Socket;

use lib "/var/sites/Wbnms/Wbnms_Conf";
use WbnmsConf;
```

```
$VERSION = '0.86';

@ISA = qw(Exporter);

@EXPORT = qw(snmpMIB_to_TT);



sub snmpMIB_to_TT ($);
sub readObjectIdentifier;


%trap_type::ObjectData;
$trap_type::Debug = 0;

%trap_type::Links;
%trap_type::OIDS;

sub snmpMIB_to_TT ($) {
  my($arg) = @_;
  my($quote, $buf, $var, $code, $val, $tmp, $tmpv, $strt);
  my(%Link) = %trap_type::Links;

  if (!open(MIB, $arg)) {
    carp "snmpMIB_to_TT: Can't open $arg: $!"
      unless ($SNMP_Session::suppress_warnings > 1);
    return -1;
  }
  print "snmpMIB_to_TT: loading $arg\n" if $trap_type::Debug;
  readObjectIdentifier;
  my $counter=0;
  my ($trap,$enterprise,$specifictrap,@trapflag);
  while(<MIB>) {
            s/--.*--//g; # throw away comments (-- anything --)
            s/--.*//;    # throw away comments (-- anything EOL)
            s/\s+/ /g;
            s/".*"//;
            if ($_ =~/TRAP-TYPE/){
                        my @trap = split();
                        $trap = $trap[0];
                        $trapflag[1] = 1;
            }
            if ($_=~/ENTERPRISE/ and $trapflag[1] == 1){
                        s/{//g;
                        s/}//g;
                        my @ind= split();
                        $enterprise = $ind[1];
                        if (!defined($trap_type::OIDS{$enterprise})) {
                                    carp "snmpMIB_to_TT: $arg: \"$enterprise\" prefix unknown, load the parent MIB
first.\n"
                                    unless ($SNMP_Session::suppress_warnings > 1);
                        }
                        $trapflag[2] = 1;
            }
            if ($_=~/::=/ and $trapflag[1] == 1 and $trapflag[2] == 1){
                        my @ind= split();
                        $specifictrap =$ind[1];
                        $val = "$trap_type::OIDS{$enterprise}.$specifictrap";
                        $trap_type::OIDS{$trap} = $val;
                        print "'$trap' => '$val'\n" if $trap_type::Debug;
                        $counter++;
                        #$trapflag[0]=0;
                        $trapflag[1]=0;
                        $trapflag[2]=0;
            }


  }
  close(MIB);


 ##################################################
 #################  SYNTAX  ######################
```

```
 ####Flags
 my $SearchTRAPTYPE = 0;
 my $SearchDESCRIPTION = 0;
 my $Searchobjects = 0;
 #########


foreach  my $TEXT(keys(%trap_type::OIDS)){
my @alarmsT;
my %ObjectType;
my @alarmsTgroup;
my $buf;
open(MIB, $arg);
 while(<MIB>) {
            s/--.*--//g; # throw away comments (-- anything --)
            s/--.*//;     # throw away comments (-- anything EOL)
            s/\s+/ /g;
 if ( $_=~/$TEXT TRAP-TYPE/){$SearchTRAPTYPE = 1;} ## Is or not an Trap-Type
 if ($SearchTRAPTYPE == 1){
            if ( $_=~/VARIABLES/){
                        $Searchobjects=1;
            }
            if ($Searchobjects == 1 and $_ !~ /DESCRIPTION/){
                        $_=~ s/VARIABLES//;
                        $_=~ s/\{//;
                        $_=~ tr/\,/ /;
                        $_=~ s/\}//;
                        $buf .=$_;
            }elsif($Searchobjects == 1 and $_ =~ /DESCRIPTION/){
                        $buf=~s/\s+/ /g;
                        $buf=~s/^\s//g;
                        my @objects=split(/ /,$buf);
                        @alarmsTgroup=@objects;
                        $ObjectType{'VARIABLES'}=\@alarmsTgroup;
                        undef $buf;
                        $Searchobjects=0;
            }
            if ($_=~/ENTERPRISE/ ){
                        s/{//g;
                        s/}//g;
                        my @ind= split();
                        $ObjectType{'ENTERPRISE'} = $ind[1];

            }
            if ($SearchDESCRIPTION == 1){
                        if ($_=~/::=/){
                                    $ObjectType{'DESCRIPTION'}= $buf;
                                    $SearchDESCRIPTION = 0;
                                    undef $buf;
                        }else{
                                    $buf .=$_;
                        }
            }
            if ($_=~/DESCRIPTION/){
                        $SearchDESCRIPTION = 1;
            }
            if ($_=~/::=/){
                        my @ind= split();
                        $ObjectType{'SPECIFICTRAP'} = $ind[1];
                        last; ##Exit
            }
 }
 }
  close(MIB);
   if ($SearchTRAPTYPE == 1){
              $trap_type::ObjectData{$TEXT}=\%ObjectType;   ##Object's data
              $SearchTRAPTYPE = 0;
   }
}

 ###############End###############
 return $counter;

}

sub readObjectIdentifier{
```

```
		my %logOID;
		my %logparent;
		open (OI,"$WbnmsConf::CONFPATH{'var'}/ObjectIdentifiers");
		while(<OI>){
			my @ind = split(/:/);
			$logOID{$ind[0]} = $ind[1];
			chomp($ind[2]);
			$logparent{$ind[0]} = $ind[2];
		}
		close(OI);
		%trap_type::OIDS = %logOID;
		%trap_type::Links = %logparent;
		}

1;
```

9.1.11.7 textual_conventions.pm

```
package textual_conventions;

require 5.004;

use strict;
use vars qw(@ISA @EXPORT $VERSION);
use Exporter;
use Carp;

use BER "0.82";
use SNMP_Session "0.83";
use Socket;

$VERSION = '0.86';

@ISA = qw(Exporter);

@EXPORT = qw(snmpMIB_to_TC);

%textual_conventions::AlarmTypes;
@textual_conventions::TC_index;


sub snmpMIB_to_TC ($);


sub snmpMIB_to_TC ($) {
  my($arg) = @_;

  open(MIB, $arg);

  ####Flags
  my $SearchTEXTUAL = 0;
  my $SearchSYNTAX = 0;
  #########
  while(<MIB>) {
  my @ind;
  @ind= split(/ ::=/);
  if ($ind[1]=~ /TEXTUAL-CONVENTION/){push(@textual_conventions::TC_index,$ind[0])}
  }
 close(MIB);

my %alarmsTgroup;
foreach  my $TEXT(@textual_conventions::TC_index){
my @alarmsT;
my %alarmsTgroup;
open(MIB, $arg);
  while(<MIB>) {
  if ($_=~ /$TEXT/){$SearchTEXTUAL = 1;}
  if ($SearchTEXTUAL == 1){if ($_=~ /SYNTAX/){$SearchSYNTAX= 1;}}
  if ($SearchSYNTAX == 1){
		if ($_=~ /\}/){
			$SearchSYNTAX = 0;
			last;
		}
```

```
            $_=~ tr/\(/ (/;
            if ($_=~ /\(*\)/){
                        my @ind;
                        @ind= split();

                        if ($ind[0]!~ /--/){
                                    $ind[1]=~ tr/\)\,\(//d;
                                    $alarmsTgroup{$ind[1]}=$ind[0];
                        }

            }
  }
  }
  close(MIB);
  $textual_conventions::AlarmTypes{$TEXT}=\%alarmsTgroup;
  $SearchTEXTUAL = 0;

}
}

1;
```

9.1.11.8 sequence.pm

```
package sequence;

require 5.004;

use strict;
use vars qw(@ISA @EXPORT $VERSION);
use Exporter;
use Carp;

use BER "0.82";
use SNMP_Session "0.83";
use Socket;

$VERSION = '0.86';

@ISA = qw(Exporter);

@EXPORT = qw(snmpMIB_to_S);

%sequence::sequence;
@sequence::S_index;


sub snmpMIB_to_S ($);


sub snmpMIB_to_S ($) {
  my($arg) = @_;

  open(MIB, $arg);

  ####Flags
  my $SearchSEQUENCE = 0;
  my $SearchSYNTAX = 0;
   my $SearchStartSYNTAX = 0;
  my $buf;
  #########
  while(<MIB>) {
              $buf .= ' ' . $_;
              $buf =~ s/\s+/ /g;
              if ($buf =~ /::= SEQUENCE/){
                          $buf =~ s/ ::= SEQUENCE//g;
                          $buf =~ s/\{//g;
                          my @ind= split(/ /,$buf);
                          my $sequence = pop(@ind);
                          push(@sequence::S_index,$sequence);
                          undef $buf;
              }
```

```
 }
 close(MIB);



 foreach  my $TEXT(@sequence::S_index){
            my %alarmsTgroup;
            my $buf;
            open(MIB, $arg);
             while(<MIB>) {
                        if ($_=~ /$TEXT/ and $_=~ /::=/ ){$SearchSEQUENCE = 1;}
                        if ($SearchSEQUENCE == 1 and $_=~ /SEQUENCE/){$SearchSYNTAX=1;}
                        if ($SearchSEQUENCE == 1 and $SearchSYNTAX == 1){
                                    $_=~ s/SEQUENCE//;
                                    $buf .= ' ' . $_;

                                    if ($_=~ /\}/){
                                                $buf =~ tr/\,/ /;
                                                $buf =~ s/\{//g;
                                                $buf =~ s/$TEXT.*::=//g;
                                                $buf =~ s/\}//g;
                                                $buf =~ s/\(.*\)//g;
                                                $buf =~ s/\s+/ /g;
                                                $buf =~ s/^\s+//g;
                                                my @ind= split(/ /,$buf);
                                                my $count=0;
                                                for my $index(0..(scalar(@ind)-1)/2){
                                                            $alarmsTgroup{$ind[$count]}=$ind[$count+1];
                                                            $count++;
                                                            $count++;
                                                }

                                                $SearchSEQUENCE= 0;
                                                $SearchSYNTAX=0;
                                                undef $buf;
                                                last;
                                    }
                        }
             }
             close(MIB);
            $sequence::sequence{$TEXT}=\%alarmsTgroup;

  }
}
1;
```

9.1.12 MIBBrowser

9.1.12.1 MIBBrowser.pm

```
package MIBBrowser;


use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

use SNMP_util;

my %fields = (  ##Permited fields
            Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
```

```
	my ($proto)= @_;  #it retrieves the name of the class
	my $class= ref($proto) || $proto;
	my $self = {
		_permited => \%fields,
		%fields,
	};
	bless ($self, $class);
	return $self;
}



sub ReadOID(){
	my ($self) = @_;
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	my @Keys = ( "oid_label","oid","object_type");
	$Manager->Keys(\@Keys);
	$Manager->MediaName("Oid2Label");
	my $data_ref_array = $Manager->ReadInto();
	##Open ObjectIdentifiers file
	open(OI,"$WbnmsConf::CONFPATH{'var'}/ObjectIdentifiers");
	while(<OI>){
		my @ind = split(/:/);
		$SNMP_util::OIDS{$ind[0]}=$ind[1];
	}
	close(OI);
	my (%OID2Label,%OBJType);
	while ( my ($OIDname, $OIDnun) = each(%SNMP_util::OIDS)) {
		$OID2Label{$OIDnun}= $OIDname;
	}
	foreach my $data_ref_hash ( @{$data_ref_array} ){
		while(my ($key,$value)=each(%{$data_ref_hash})){
			$SNMP_util::OIDS{${$data_ref_hash}{'oid_label'}} = ${$data_ref_hash}{'oid'};
			$OID2Label{${$data_ref_hash}{'oid'}} = ${$data_ref_hash}{'oid_label'};
			$OBJType{${$data_ref_hash}{'oid_label'}} = ${$data_ref_hash}{'object_type'};
		}
	}

return(\%SNMP_util::OIDS,\%OID2Label,\%OBJType);
}

sub FindChild($){
	my ($self,$father) = @_;
	my @children;
	my ( $nameOID,$OIDname,$OBJType)= &ReadOID($self);
	my %OID2Label=%{$OIDname};
	%SNMP_util::OIDS=%{$nameOID};
	my @FatherOIDs = split(/\./,$SNMP_util::OIDS{$father});
	my $fatherlevel= scalar(@FatherOIDs);
	while (my ($OIDname, $OIDnun) = each(%SNMP_util::OIDS)) {
		my @ChildOIDs = split(/\./,$OIDnun);
		my $childlevel= scalar(@ChildOIDs);
		pop(@ChildOIDs);
		my $index = $childlevel-$fatherlevel;
		if ($index == 1 and join('',@FatherOIDs) eq join('',@ChildOIDs)){
			push(@children,$OIDname);
		}
	}

return(\@children,$OBJType);
}

sub RequestInfo($){
	my ($self,$object) = @_;
	my $Manager = ABMManager->new();
	$Manager->ABMType(1);
	##Read Object Type. Object or Notification.
	$Manager->MediaName("Oid2Label");
	my %search=('oid_label'=>$object,);
	$Manager->DataSearch(\%search);
	my $TypeData= $Manager->ReadInto();
	my $OBJData;
	if (${$TypeData}[0]{'object_type'} == 0 ){##Notification Type
		##Read from NotificationType Table
		$Manager->MediaName("NotificationType");
```

```
                my %search=('oid_label'=>$object,);
                $Manager->DataSearch(\%search);
                $OBJData= $Manager->ReadInto();
        }elsif(${$TypeData}[0]{'object_type'} == 1 ){##Object Type
                ##Read from ObjectType Table
                $Manager->MediaName("ObjectType");
                my %search=('oid_label'=>$object,);
                $Manager->DataSearch(\%search);
                $OBJData= $Manager->ReadInto();
        }elsif(${$TypeData}[0]{'object_type'} == 2 ){##Trap Type
                ##Read from ObjectType Table
                $Manager->MediaName("TrapType");
                my %search=('oid_label'=>$object,);
                $Manager->DataSearch(\%search);
                $OBJData= $Manager->ReadInto();
        }

return(${$OBJData}[0]);         ##hash ref
}

sub AUTOLOAD{
        my $self = shift;
        my $type = ref($self) || die;
        my $name = $AUTOLOAD;
        $name =~ s/.*://;
        unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
        if(@_){$self->{$name}= shift}
        return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.1.12.2 OID2js.pm

```
package OID2js;


use strict;
use vars qw($VERSION $AUTOLOAD);
$VERSION = '1.0';     ##

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

#Calling ABM Object
use lib "/var/sites/Wbnms/Wbnms_Core/ABM";
use ABMManager 2.0;

use SNMP_util;

my %fields = (  ##Permited fields
        Debug => 0,  ## If Debug == 1 then Debug Mode
);

sub new {
        my ($proto)= @_;  #it retrieves the name of the class
        my $class= ref($proto) || $proto;
        my $self = {
                _permited => \%fields,
                %fields,
        };
        bless ($self, $class);
        return $self;
}

sub OID2js(){
        my ($self) =@_;
        my ($nameOID,$OIDname)= ReadOID($self);
        my %OID2Label=%{$OIDname};
```

```
%SNMP_util::OIDS=%{$nameOID};
my $ind = 0;
my %OIDLevels;
my %OIDfathers;
my $toplevel = 0;
my %SNMPOIDalias;
while (my ($OIDname, $OIDnun) = each(%SNMP_util::OIDS)) {
	my @OIDs = split(/\./,$OIDnun);
	my $level= scalar(@OIDs);
	if ($toplevel < $level ){$toplevel = $level;}
	my $Num = pop(@OIDs);
	if ($Num == 0){$Num = pop(@OIDs);}
	my $father = join('.',@OIDs);
	my $nf= $OIDname.":".$father;
	$OIDLevels{$OIDname} = $level;
	$OIDfathers{$OIDname} = $OID2Label{$father};
	$SNMPOIDalias{$OIDname}= "aux$ind";
	$ind++;
}

my %levels;

for my $ind (2..$toplevel){  # for each level
	my @level;
	while (my ($OIDname, $OIDlevel) = each(%OIDLevels)) {
		if($OIDlevel==$ind){
		push(@level,$OIDname);
		}
	}
	$levels{$ind}=\@level;
}

my %OIDchild;

foreach my $ind1 (values(%OIDfathers)){
	my @child;
	foreach my $ind2(keys(%OIDfathers)){
		if($ind1==$ind2){
		push(@child,$ind1);
		}
	}
	$OIDchild{$ind1}=\@child;

}
open(JAVAS,">$WbnmsConf::CONFMIBS{'MIBbrowserjs'}");
print JAVAS "// Decide if the names are links or just the icons\n";
print JAVAS "USETEXTLINKS = 1  //replace 0 with 1 for hyperlinks\n";
print JAVAS "// Decide if the tree is to start all open or just showing the root folders\n";
print JAVAS "STARTALLOPEN = 0 //replace 0 with 1 to show the whole tree\n";

print JAVAS "foldersTree = gFld\(\"<i>iso</i>\", \"startPage.html\"\)\n";
my $after = "foldersTree";
my %LEVEL;
for my $lev (2 .. $toplevel ){
	my $father = $levels{$lev};
	foreach my $ref1 (@$father){
		my $gran = $OIDfathers{$ref1};
		if (defined $SNMPOIDalias{$gran} and exists($OIDchild{$ref1})){
			my $granalias=$SNMPOIDalias{$gran};
			if($gran eq "iso"){
				$granalias = "foldersTree"
			}
			$LEVEL{$ref1}="$SNMPOIDalias{$ref1} = insFld\($granalias,
gFld(\"$ref1\", \"$WbnmsConf::CONFMIBS{'MIBbrowsercgi'}?OBJ=$ref1\"\)\)\n";
		}
	}
}
my @data = keys(%OID2Label);
my $last="iso";
@data = sort  @data;

foreach my $data(@data){
	my $OIDname= $OID2Label{$data};
	if(exists($LEVEL{$OIDname})){
		print JAVAS $LEVEL{$OIDname}."\n";
	}
```

```
		}
}

sub ReadOID(){
		my ($self) =@_;
		##Read Objects from Oid2Label Table
		my $Manager = ABMManager->new();
		$Manager->ABMType(1);
		my @Keys = ( "oid_label","oid");
		$Manager->Keys(\@Keys);
		$Manager->MediaName("Oid2Label");
		my $data_ref_array = $Manager->ReadInto();
		my %OID2Label;
		##Open ObjectIdentifiers file
		open(OI,"$WbnmsConf::CONFPATH{'var'}/ObjectIdentifiers");
		while(<OI>){
			my @ind = split(/:/);
			$SNMP_util::OIDS{$ind[0]}=$ind[1];
		}
		close(OI);
		while ( my ($OIDname, $OIDnun) = each(%SNMP_util::OIDS)) {
			$OID2Label{$OIDnun}= $OIDname;
		}
		foreach my $data_ref_hash ( @{$data_ref_array} ){
			while(my ($key,$value)=each(%{$data_ref_hash})){
				$SNMP_util::OIDS{${$data_ref_hash}{'oid_label'}} = ${$data_ref_hash}{'oid'};
				$OID2Label{${$data_ref_hash}{'oid'}} = ${$data_ref_hash}{'oid_label'};
			}
		}

return(\%SNMP_util::OIDS,\%OID2Label);
}

sub AUTOLOAD{
		my $self = shift;
		my $type = ref($self) || die;
		my $name = $AUTOLOAD;
		$name =~ s/.*://;
		unless(exists $self->{_permited}->{$name}){ print "Can't access $name  field in class $type"};
		if(@_){$self->{$name}= shift}
		return $self->{$name};
}

sub DESTROY{ ## AUTOLOAD search DESTROY sub

}

1;
```

9.2 Módulos de Interfaz (Web)

9.2.1 Conf

9.2.1.1 WBNMS.css

```
body {
		background-color: #000000;
		color: #FFFFFF;

}
.menu {
		cursor: hand;
		border-top: 1px solid #0066CC;
		border-right: 1px solid #003366;
		border-bottom: 1px solid #003366;
		border-left: 1px solid #0066CC;

}
.menu:hover {
```

```
		background-color: #0099FF;
		cursor: hand;

}
.menutext {
		font-weight: bold;
		color: #CCCCCC;
		text-decoration: none;
}
.menutext:hover {
		font-weight: bold;
		color: #000000;
		text-decoration: none;
}
.menuborder {
		height: 10px;
		width: 10px;
		border-top: 2px double #999999;
		border-right: 2px double #333333;
		border-bottom: 2px double #333333;
		border-left: 2px double #999999;
		background-color: #000000;

}
.ConfTable {
		font-family: "Times New Roman", Times, serif;
		color: #000000;
		text-decoration: none;
		background-color: #FFFFFF;
		cursor: hand;
}
.ConfTableTitle {
		font-family: Arial, Helvetica, sans-serif;
		font-weight: bolder;
		color: #0099FF;
		text-decoration: none;
		background-color: #666666;
		font-size: large;

}
.ConfTableItem {
		font-family: Arial, Helvetica, sans-serif;
		color: #0099CC;
		text-decoration: none;
		background-color: #666666;
		font-weight: bold;


}
.menustatic {
		font-family: Arial, Helvetica, sans-serif;
		font-weight: bold;
		color: #000000;
		text-decoration: none;
		background-color: #0066CC;
}
.ConfTableMarked {
		font-family: "Times New Roman", Times, serif;
		color: #666666;
		text-decoration: none;
		background-color: #CCCCCC;
		cursor: hand;
		font-style: oblique;


}
.Button {
		background-color: #0066CC;
		width: 100%;
		font-family: "Times New Roman", Times, serif;
		color: #CCCCCC;
		font-weight: bold;
		font-size: 14px;
}
.Popup {
		background-color: #999999;
```

```
		width: 100%;
}
.menubody {
		background-attachment: fixed;
		background-color: #000000;
		background-image: url(../pics/fire.jpg);
		background-repeat: no-repeat;
		background-position: center center;
}
.bodymenuprincipal {
		background-attachment: fixed;
		background-image: url(../pics/fire2.gif);
		background-repeat: no-repeat;
		background-position: right center;
}
.menuborderwtext {
		border-top: 2px double #999999;
		border-right: 2px double #333333;
		border-bottom: 2px double #333333;
		border-left: 2px double #999999;
		font-family: "Times New Roman", Times, serif;
		color: #000000;
		font-weight: bolder;
		text-decoration: none;
		background-color: #3366FF;
		font-size: 12px;



}
```

9.2.1.2 menu.js

```
function MenuMouseOver(Menu){
Menu.style.background='#0099FF';
}
function MenuMouseOut(Menu){
Menu.style.background='#000000';
}
function MenuSelected(Menu,Type){
Menu.style.background='#FFFFFF';

}
```

9.2.2 htdocs

9.2.2.1 Main.html

```
<html>
<head>
<title>WBNMS</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload();
}
MM_reloadPage(true);
//-->
</script>
</head>

<frameset rows="*,10%" cols="*" framespacing="1" frameborder="yes" border="1" bordercolor="#CCCCCC">
  <frameset rows="*" cols="25%,*" framespacing="1" frameborder="yes" border="1" bordercolor="#CCCCCC">
    <frame src="Lefthome.html" name="leftFrame" scrolling="NO">
    <frame src="Mainhome.html" name="mainFrame">
  </frameset>
```

```
  <frame src="Menu.html" name="bottomFrame" scrolling="NO" noresize >
</frameset>
<noframes><body>

</body></noframes>
</html>
```

9.2.2.2 Menu.html

```
<html>

<head>

<title>WBNMS</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<style type="text/css">

<!--

@import url(/Conf/WBNMS.css);

-->

</style>

</head>

<body class=bodymenuprincipal>
<script src="/Conf/menu.js"></script></script>

<table width="100%" border="0">

  <tr>

    <td width="95%" height="36">

      <table width="440" border="1" class="menuborder">

        <tr>

          <td nowrap class="menu"  onClick="parent.close()" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><a href="#" class="menutext">Salir</a></td>

          <td class="menu"  nowrap  onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><a href="/webadm/htdocs/Administration.html" target="mainFrame"
class="menutext">Administraci&oacute;n</a></td>

          <td class="menu" nowrap  onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><a href="/htdocs/Monitor.html" target="mainFrame"
class="menutext">Monitoreo</a></td>

          <td class="menu" nowrap  onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><a href="/htdocs/LogAlarms.html" target="mainFrame"
class="menutext">Lista

            de Eventos</a></td>

          <td  class="menu" nowrap  onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><a href="/htdocs/Actions.html" target="mainFrame"
class="menutext">Acciones</a></td>

        </tr>

      </table></td>

    <td width="5%"><a href="#"
onClick=window.open("About.html",'About','scrollbar=no,width=700,height=350')><img
src="../pics/wbnmslogo.gif" alt="About" width="40" height="40" border="0"></a></td>

  </tr>
```

```
</table>

</body>

</html>
```

9.2.2.3 Monitor.html

```
<html>
<head>
<title>`Monitor de Eventos</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
@import url("/Conf/WBNMS.css");
-->
</style>
</head>
<script language="JavaScript" type="text/JavaScript">
function Load(){

window.open("/cgi-bin/MonitorMain.cgi?ROOT=YES",'mainFrame');
window.open("/cgi-bin/MonitorMenu.cgi",'leftFrame');
}
</script>

<body onLoad="Load()">

<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p align="center"><strong><font color="#FF0000" size="+5">Cargando ...</font></strong></p>
</body>
</html>
```

9.2.2.4 LogAlarms.html

```
<html>
<head>
<title>Lista de Eventos</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
@import url("/Conf/WBNMS.css");
-->
</style>
</head>
<script language="JavaScript" type="text/JavaScript">
function Load(){

window.open("/cgi-bin/LogAlarmViewer.cgi",'mainFrame');
window.open("/cgi-bin/LogAlarmMenu.cgi",'leftFrame');
}
</script>

<body onLoad="Load()">

<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p align="center"><strong><font color="#FF0000" size="+5">Cargando ...</font></strong></p>
</body>
</html>
```

9.2.2.5 Actions.html

```
<html>
<head>
<title>Acciones Get&Set</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
@import url("/Conf/WBNMS.css");
-->
</style>
</head>
<script language="JavaScript" type="text/JavaScript">
function Load(){

window.open("/cgi-bin/GetSetViewer.cgi",'mainFrame');
window.open("/cgi-bin/ActionMenu.cgi",'leftFrame');
}
</script>

<body onLoad="Load()">

<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p align="center"><strong><font color="#FF0000" size="+5">Cargando ...</font></strong></p>
</body>
</html>
```

9.2.3 cgi-bin

9.2.3.1 MonitorMain.cgi

```
#!/usr/bin/perl

#
##Main Map Monitor
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib "/var/sites/Wbnms/Wbnms_Core/Drawer";
use MonitorManager;
use MapManager;

#Main
$query = new CGI;
print $query->header;

my $Focus="";
if (defined $query->param('MapThread') and $query->param('FOCUS') eq "ON"){
        $Focus='window.focus()';
}


print $query->start_html(-title=>'Monitor',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -onLoad=>$Focus,
                -onBlur=> $Focus,
                );


##Open Map
my ($image,$map_id);
```

```
if( defined $query->param('map_id')){
        my $img="/DrawerDir/".$query->param('map_id')."\.gif";
        $map_id = $query->param('map_id');
        $image = "<img SRC=\"".$img."\"  align =\"center\" border=\"0\" usemap=\"#Map\">";
}else{
        ##Load User Preferences Information from Cookie
        my %Monitorpreferences = $query->cookie('Monitor_'.$query->user_name());
        if (exists($Monitorpreferences{'rootmap'})) {
                my $img="/DrawerDir/".$Monitorpreferences{'rootmap'}."\.gif";
                $map_id = $Monitorpreferences{'rootmap'};
                $image = "<img SRC=\"".$img."\"  align =\"center\" border=\"0\" usemap=\"#Map\">";
        }else{
                print "<table width=\"100%\"><tr><td width=\"100%\" class=\"menustatic\">Mapa
Raíz<\/td><\/tr><\/table>\n";
                $image = "Seleccione un Mapa";
        }
}

print "<META HTTP-EQUIV=\"Refresh\" CONTENT=\"".$WbnmsConf::CONFDRAWER{'checkinterval'}."\">";
print "<script src=\"/Conf/menu.js\"><\/script>";
print "<script src=\"/Conf/monitor.js\"><\/script>";
my ($MapInfo,$Equipment,$Maps,$Monitor,$Map);
if (defined $map_id){
        ##
        $Map=MapManager->new();
        $MapInfo=$Map->ReadMaps($map_id);
        $Monitor = MonitorManager->new();
        ($Equipment,$Maps)=$Monitor->MapParse($map_id);
        print "<table width=\"100%\"><tr>\n";
        if (defined $query->param('MapThread')){
                my @MapThread = split(/:/,$query->param('MapThread'));
                my $Father= pop(@MapThread);
                if (scalar(@MapThread) != 0 ){
                        my $FMapThread = join(':',@MapThread);
                        my ($FMapwidth,$FMapheight)=$Monitor->Mapsize($Father);
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onclick=MonitorView(".$Father.",\"".$FMapThread."\",".$FMapwidth.",".$FMapheight.")
class=\"menutext\">Volver<\/a><\/td>\n";
                }
        }
        print "<td class=\"menustatic\">".$Monitor-
>Counter($WbnmsConf::CONFDRAWER{'checkinterval'})."</td>";
        if ($query->param('ROOT') eq "YES"){
                print "<td  width=\"100%\" class=\"menustatic\">".${$MapInfo}[0]{'mapname'}."<\/td><td
class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.location.reload(1)\"
class=\"menutext\">Actualizar<\/a><\/td><\/tr><\/table>\n";
        }else{
                print "<td  width=\"100%\" class=\"menustatic\">".${$MapInfo}[0]{'mapname'}."<\/td><td
class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.location.reload(1)\"
class=\"menutext\">Actualizar<\/a><\/td><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.close()\"
class=\"menutext\">Cerrar<\/a><\/td><\/tr><\/table>\n";
        }
}
print <<EOF;
<table   width="100%" border="1" cellspacing="0">
EOF
  ;
print "<tr><td class=\"menustatic\" align =\"center\" >".$image."<\/td><\/tr>\n";
print "<map name=\"Map\">\n";


foreach my $shape (@{$Equipment}){
        print "<area shape=\"rect\" coords=\"".${$shape}{'coor'}."\" href=\"#\"
onclick=EquipmentView(".${$shape}{'equipment_id'}.",".$map_id.",\"".$query->param('MapThread')."\')  >\n";
}
my $MapThread = join(':',($query->param('MapThread'),$map_id));
foreach my $shape (@{$Maps}){
        my ($Mapwidth,$Mapheight)=$Monitor->Mapsize(${$shape}{'son_map_id'});
        print "<area shape=\"rect\" coords=\"".${$shape}{'coor'}."\" href=\"#\"
onclick=MonitorView(".${$shape}{'son_map_id'}.",\"".$MapThread."\",".$Mapwidth.",".$Mapheight.") >\n";
```

```
}
print "<\/map>\n";
print <<EOF;
</table>
EOF
  ;

print $query->end_html;
```

9.2.3.2 MonitorEquipmentView.cgi

```
#!/usr/bin/perl

#
##Equipments Monitor
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib "/var/sites/Wbnms/Wbnms_Core/Drawer";
use MonitorManager;
use EquipmentManager;

#Main
$query = new CGI;
print $query->header;


###
my $Monitor = MonitorManager->new();
my ($Mapwidth,$Mapheight)=$Monitor->Mapsize($query->param('map_id'));

my $Equipment = EquipmentManager->new();
my ($EquipmentData,$EquipmentTypes)=$Equipment->ReadEquipments($query->param('equipment_id'));
###
my $Focus="";
if ($query->param('FOCUS') eq "ON"){
        $Focus='window.focus()';
}
my $Close="MonitorView(".$query->param('map_id').",\"".$query-
>param('MapThread')."\",".$Mapwidth.",".$Mapheight.")";
##
print $query->start_html(-title=>'Monitor de Equipo',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -onLoad=>$Focus,
                -onBlur=>$Focus,
                );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<script src=\"/Conf/monitor.js\"><\/script>";

print "<table width=\"100%\"><tr><td  width=\"100%\"
class=\"menustatic\">${${$EquipmentData}[0]}{'equipment'}<\/td><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.location.reload(1)\"
class=\"menutext\">Actualizar<\/a><\/td><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=MonitorView(".$query->param('map_id').",\"".$query-
>param('MapThread')."\",".$Mapwidth.",".$Mapheight.") class=\"menutext\">Volver Al
Mapa<\/a><\/td><\/tr><\/table>\n";
print <<EOF;
<table   width="100%" border="1" cellspacing="2">
EOF
  ;
        print "<tr><td width=\"100%\"  nowrap align=\"center\"><table width=\"100%\">\n";
                print "<tr><td class=\"ConfTableItem\">Equipo</td><td align=\"center\"
class=\"ConfTable\">${${$EquipmentData}[0]}{'equipment'}</td></tr>\n";
```

```
                        print "<tr><td class=\"ConfTableItem\">Descripción</td><td align=\"center\"
class=\"ConfTable\">${${$EquipmentData}[0]}{'description'}</td></tr>\n";
                        print "<tr><td class=\"ConfTableItem\">Tipo</td><td align=\"center\"
class=\"ConfTable\">${${$EquipmentData}[0]}{'equipmenttype_label'}</td></tr>\n";
                        print "<tr><td class=\"ConfTableItem\">IP</td><td align=\"center\"
class=\"ConfTable\">${${$EquipmentData}[0]}{'IP'}</td></tr>\n";
                        print "<tr><td class=\"ConfTableItem\">Comunidad</td><td align=\"center\"
class=\"ConfTable\">${${$EquipmentData}[0]}{'community'}</td></tr>\n";
                        print "<tr><td class=\"ConfTableItem\">MIB</td><td align=\"center\"
class=\"ConfTable\">${${$EquipmentData}[0]}{'MIB'}</td></tr>\n";

                print "</table></td>";
                print "<td width=\"100%\" nowrap align=\"center\"><table>\n";
                print "<tr><td width=\"100%\" class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onClick=EquipmentAlarmsView(".$query->param('equipment_id').",".$query->param('map_id').",\'".$query-
>param('MapThread')."\') class=\"menutext\">Ver Alarmas<\/a><\/td></tr>\n";
                print "<tr><td width=\"100%\" class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onClick=EquipmentAlarmsUsersView(".$query->param('equipment_id').",".$query->param('map_id').",\'".$query-
>param('MapThread')."\') class=\"menutext\">Ver Alertas<\/a><\/td></tr>\n";
                print "<tr><td width=\"100%\" class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onClick=GetSetView(".$query->param('equipment_id').",".$query->param('map_id').",\'".$query-
>param('MapThread')."\',".${${$EquipmentData}[0]}{'equipmenttype_id'}.",\'".${${$EquipmentData}[0]}{'IP'}."\')
class=\"menutext\">Tomar Acciones<\/a><\/td></tr>\n";
                print "</table></td></tr>\n";
print <<EOF;
</table>
EOF
  ;

print $query->end_html;
```

9.2.3.3 MonitorMenu.cgi

```
#!/usr/bin/perl

#
##Monitor  Menu
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Drawer';
use MapManager;

#Main
$query = new CGI;

##Set Cookie Option
if (defined $query->param('CookieMap')){
my %Monitorpreferences=(
                                'user'=> $query->user_name(),
                                'rootmap' => $query->param('map_id'),
);
my $cookie=$query->cookie(-name=>'Monitor_'.$query->user_name(),
                        -value=>\%Monitorpreferences,
                        );
print $query->header(-cookie=>$cookie);
}else{
print $query->header();
}


print $query->start_html(-title=>'Mapa Raiz',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -class=>'menubody',
```

```
                    );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td width=\"100%\" class=\"menustatic\">Monitor<\/td><\/tr><\/table>\n";
##Open Map Option
if (defined $query->param('OpenMap') or defined $query->param('CookieMap')){
        print "<script>window.open(\'/cgi-bin/MonitorMain.cgi?ROOT=YES&map_id=".$query-
>param('map_id')."\',\'mainFrame\');</script>";
}


my $Map = MapManager->new();
my $Maps = $Map->ReadMaps();
## Load Maps Info
my (@maps,%maps);
foreach my $map (@{$Maps}){
                push(@maps,${$map}{'map_id'});
                $maps{${$map}{'map_id'}}= ${$map}{'mapname'};
}

###############
print <<EOF;

<table width="100%" border="0" cellspacing="1">
EOF
  ;
print $query->start_form(-action=>'/cgi-bin/MonitorMenu.cgi' );
        print "<tr><td  width=\"30%\" class=\"ConfTableItem\" nowrap align=\"center\">Mapas</td><td
width=\"70%\" nowrap class=\"menustatic\" align=\"center\">".$query->popup_menu(-name=>'map_id',-
values=>\@maps,-labels=>\%maps,'class="Popup"')."</td></tr>\n";
        print "<tr><td  class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"2\">".$query-
>submit('CookieMap','Abrir Mapa como Raíz','class="Button"')."</td></tr>\n";
print $query->endform;
print <<EOF;
</table>
<table width="100%" border="0" cellspacing="1">
EOF
  ;
        print "<tr><td><p> </p></td></tr><td  class=\"ConfTableItem\" nowrap align=\"center\">Monitor
de Alarmas</td></tr>";
        print "<tr><td  align =\"center\" colspan=\"2\"><iframe width=\"240\" FRAMEBORDER=0
SCROLLING=NO  src=\"/cgi-bin/MonitorAlarmsCounter.cgi\"></iframe><\/td><\/tr>\n";
print <<EOF;
</table>
EOF
  ;
print <<EOF;
</table>
<table width="100%" border="0" cellspacing="1">
EOF
  ;
        print "<tr><td align =\"center\"><iframe width=\"240\" height=\"180\" FRAMEBORDER=0
SCROLLING=NO  src=\"/cgi-bin/StatusMonitor.cgi\"></iframe><\/td><\/tr>\n";
        print "<tr><td align =\"center\"><img src=\"/pics/Drako.gif\" ><\/td><\/tr>\n";
print <<EOF;

</table>
EOF
  ;
print $query->end_html;
```

9.2.3.4 MonitorAlarmsCounter.cgi

```
#!/usr/bin/perl

#
## Alarm's Counter Monitor
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;
```

```
use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;
use lib '/var/sites/Wbnms/Wbnms_Core/Drawer';
use MonitorManager;

#Main
$query = new CGI;

print $query->header();

print $query->start_html(-title=>'Mapa Raiz',
                         -style=>{'src'=>'/Conf/WBNMS.css'},
                         );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<META HTTP-EQUIV=\"Refresh\" CONTENT=\"".$WbnmsConf::CONFDRAWER{'checkinterval'}."\">";
###############
my $Monitor=MonitorManager->new();
my $AllAlarms=$Monitor->AllAlarms();

print <<EOF;
<table width="100%" border="0" cellspacing="0">
EOF
   ;
       print "<tr><td  class=\"ConfTableItem\" nowrap align=\"center\">Alarmas Activas</td><td nowrap
class=\"ConfTable\" align=\"center\">$AllAlarms</td></tr>";
              my $img="/DrawerDir/ACount\.gif";
              my $image = "<img SRC=\"".$img."\"  align =\"center\" border=\"0\" >";
              print "<tr><td   onClick=\"window.location.reload(1)\" class=\"menustatic\" align =\"center\"
colspan=\"2\">".$image."<\/td><\/tr>\n";
print <<EOF;
</table>
EOF
   ;

print $query->end_html;
```

9.2.3.5 StatusMonitor.cgi

```
#!/usr/bin/perl

#
##Daemons Status Monitor
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Status';
use StatusChecker;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Status',
                         -style=>{'src'=>'/Conf/WBNMS.css'},
                         );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Estado del Sistema<\/td><\/tr><\/table>\n";
print "<META HTTP-EQUIV=\"Refresh\" CONTENT=\"".$WbnmsConf::CONFDRAWER{'checkinterval'}."\">";
my $Status = StatusChecker->new();

my @test = $Status->CheckStatus();##($AlarmSd,$trapd,$Drawerd,$testDB)

print <<EOF;
<table width="100%">
```

```
  <tr>
EOF
   ;
    foreach my $test (@test){
                my ($color,$action);
                while(my ($key,$value)=each(%{$test})){
                                if ($value == 1 ){
                                                $color = "#00FF00";
                                                $value = "Encendido";
                                }elsif($value == 0 ){
                                                $color = "#FF0000";
                                                $value = "Apagado";
                                }
                                $action="/cgi-bin-admin/LogDaemon.cgi?Daemon=$key";
                                print "<tr><td nowrap align=\"center\" bgcolor=\"".$color."\" ><strong><font
color=\"#000000\">$key<\/font><\/strong><\/td><td class=\"ConfTable\" align=\"center\">$value<\/td><\/tr>\n";
                }
     }
print <<EOF;
  </tr>
</table>
EOF
   ;


print $query->end_html;
```

9.2.3.6 LogAlarmViewer.cgi

```
#!/usr/bin/perl

#
##Alarms Logs Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance';
use LogManager;


#Main
$query = new CGI;
print $query->header;

my $Focus="";
my $Close="";
if (defined $query->param('equipment_id')){
                $Focus='window.focus();';
                ##$Close='self.close();opener.focus();';
}
print $query->start_html(-title=>'Lista de Eventos',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                                -onLoad => $Focus,
                                -onBlur => $Focus,
                        );


print "<script src=\"/Conf/menu.js\"><\/script>";
print "<script src=\"/Conf/monitor.js\"><\/script>";

my $Log=LogManager->new();


##Ack Option
if (defined $query->param('Ack')){
                foreach my $trap ($query->param()){
                                if ($trap=~/^trap_id*/){
```

```
                                my %data=('marked'=> $query->param('AckState'.$trap),);
                                my %search=('trap_id' => $query->param($trap),);
                                $Log->AckEvent(\%data,\%search);
                                if ($query->param('AckState'.$trap) eq "ACK"){$Log->AckAlarm($query-
>param($trap),$query->user_name());} ## Delete Active Alarm
                        }
                }

}
##Delete Option
if (defined $query->param('Del')){
foreach my $trap ($query->param()){
                        if ($trap=~/^trap_id*/){
                                my %data=('trap_id' => $query->param($trap),);
                                $Log->AckAlarm($query->param($trap),$query->user_name());
                                $Log->DelEvent(\%data);
                        }
                }
}

##Read Data from DB

my $data;
if ($query->param('TYPE') eq "PRO"){
                ##Processed Events -> Alarms
                if ($query->param('USERS') eq "YES"){
                        print "<table width=\"100%\"><tr><td width=\"100%\" class=\"menustatic\">Lista de Eventos
Asignados<\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"  onClick=\"print()\"
class=\"menutext\">Imprimir<\/a><\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onClick=\"window.location.reload()\" class=\"menutext\">Actualizar<\/a><\/td>";
                        if (defined $query->param('equipment_id')){print "<td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=EquipmentViewBack(".$query-
>param('equipment_id').",".$query->param('map_id').",\'".$query->param('MapThread')."\')
class=\"menutext\">Volver a Propiedades<\/a><\/td>";}
                        print "<\/tr><\/table>\n";
                        $data=$Log->ReadProcessedLogWUsers($query->param('equipment_id'));
                }else{
                        print "<table width=\"100%\"><tr><td width=\"100%\" class=\"menustatic\">Lista de Eventos
Procesados<\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"  onClick=\"print()\"
class=\"menutext\">Imprimir<\/a><\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onClick=\"window.location.reload()\" class=\"menutext\">Actualizar<\/a><\/td>";
                        if (defined $query->param('equipment_id')){print "<td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=EquipmentViewBack(".$query-
>param('equipment_id').",".$query->param('map_id').",\'".$query->param('MapThread')."\')
class=\"menutext\">Volver a Propiedades<\/a><\/td>";}
                        print "<\/tr><\/table>\n";
                        $data=$Log->ReadProcessedLog();
                }

}else{      ##All Events
                print "<table width=\"100%\"><tr><td width=\"100%\" class=\"menustatic\">Lista de todos los
Eventos<\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"  onClick=\"print()\"
class=\"menutext\">Imprimir<\/a><\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onClick=\"window.location.reload()\" class=\"menutext\">Actualizar<\/a><\/td><\/tr><\/table>\n";
                $data=$Log->ReadAllLog();
}

##Filter Equipment_id Option for Monitor feactures
if (defined $query->param('equipment_id')){
                my %Filter;
                $Filter{'equipment_id'}=$query->param('equipment_id');
                $data=$Log->FilterLog($data,\%Filter);
}
##Filter Option

my $Filter;
if (defined $query->param('Filter')){
                my (%Filter,@banner,@Filter);
```

```
		push(@Filter,"Filter=".$query->param('Filter'));
		if(defined $query->param('Dia')){
			$Filter{'Day'}=$query->param('Dia');
			push(@banner,"Día:".$query->param('Dia'));
			push(@Filter,"Dia=".$query->param('Dia'));
			}
		if(defined $query->param('Hora')){
			$Filter{'Hour'}=$query->param('Hora');
			push(@banner,"Hora:".$query->param('Hora'));
			push(@Filter,"Hora=".$query->param('Hora'));
			}
		if(defined $query->param('Minutos')){
			$Filter{'Minutes'}=$query->param('Minutos');
			push(@banner,"Min:".$query->param('Minutos'));
			push(@Filter,"Minutos=".$query->param('Minutos'));
			}
		if(defined $query->param('IP')){
			$Filter{'IP'}=$query->param('IP');
			push(@banner,"IP:".$query->param('IP'));
			push(@Filter,"IP=".$query->param('IP'));
			}
		if(defined $query->param('Severidad')){
			$Filter{'Severity'}=$query->param('Severidad');
			push(@banner,"Severidad:".$query->param('Severidad'));
			push(@Filter,"Severidad=".$query->param('Severidad'));
			}
		if(defined $query->param('Equipo')){
			$Filter{'Equipment'}=$query->param('Equipo');
			push(@banner,"Equipo:".$query->param('Equipo'));
			push(@Filter,"Equipo=".$query->param('Equipo'));
			}
		if(defined $query->param('Tipo')){
			$Filter{'Type'}=$query->param('Tipo');
			push(@banner,"Tipo:".$query->param('Tipo'));
			push(@Filter,"Tipo=".$query->param('Tipo'));
			}
		if(defined $query->param('ID')){
			$Filter{'ID'}=$query->param('ID');
			push(@banner,"ID:".$query->param('ID'));
			push(@Filter,"ID=".$query->param('ID'));
			}
		$data=$Log->FilterLog($data,\%Filter);
		print "<table width=\"100%\" border=\"0\" cellspacing=\"0\"><tr><td class=\"menu\" nowrap
align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"top.frames[0].history.go(-1);\"
class=\"menutext\"><<<\/a></td><td class=\"ConfTableItem\" nowrap align=\"center\" >Filtros Aplicados:</td><td
class=\"ConfTable\" nowrap align=\"center\"  width=\"100%\">".join(' ',@banner)."</td></tr></table>\n";
		$Filter="&".join('&',@Filter);
}

#####################
		print $query->start_form(-action=>'/cgi-bin/LogAlarmViewer.cgi' );
		print "<table width=\"100%\" border=\"0\" cellspacing=\"0\"><tr><td align=\"center\">".$query-
>submit('Ack','Reconocer')."</td><td align=\"center\" >".$query->submit('Del','Borrar')."</td><td
class=\"ConfTableItem\" nowrap align=\"center\"  width=\"100%\">Lista de Eventos</td></tr></table>\n";
		print $query->hidden('TYPE',$query->param('TYPE'));
		print $query->hidden('equipment_id',$query->param('equipment_id'));
		print $query->hidden('map_id',$query->param('map_id'));
print <<EOF;
<table width="100%" border="0" cellspacing="1">
EOF
  ;
		my $marktxt;
		if ($query->param('MarkAll') eq "YES"){
				$marktxt= "NO";
		 }else{
				$marktxt= "YES";
		 }

		 if($query->param('TYPE') eq "PRO"){
			if ($query->param('USERS') eq "YES"){
				print "<tr><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"/cgi-
bin/LogAlarmViewer.cgi?MarkAll=$marktxt&TYPE=PRO&USERS=YES&equipment_id=".$query-
>param('equipment_id')."&map_id=".$query->param('map_id').$Filter."\"  class=\"menutext\"><><\/a><\/td><td
```

```
class=\"ConfTableItem\" nowrap align=\"center\" >ID</td><td class=\"ConfTableItem\" nowrap align=\"center\"
>Ack</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Tiempo</td><td class=\"ConfTableItem\" nowrap
align=\"center\" >Severidad</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Notificado</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Rango</td><td class=\"ConfTableItem\" nowrap align=\"center\"
>Turno</td><td class=\"ConfTableItem\" nowrap align=\"center\" >E-Mail</td><td class=\"ConfTableItem\"
nowrap align=\"center\" >Notificado via</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Evento</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Variables</td></tr>\n";
                        }else{
                                print "<tr><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"/cgi-
bin/LogAlarmViewer.cgi?MarkAll=$marktxt&TYPE=PRO&equipment_id=".$query-
>param('equipment_id')."&map_id=".$query->param('map_id').$Filter."\"  class=\"menutext\"><><\/a><\/td><td
class=\"ConfTableItem\" nowrap align=\"center\" >ID</td><td class=\"ConfTableItem\" nowrap align=\"center\"
>Ack</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Tiempo</td><td class=\"ConfTableItem\" nowrap
align=\"center\" >Host</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Tipo</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >IP</td><td class=\"ConfTableItem\" nowrap align=\"center\"
>Comunidad</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Severidad</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Evento</td><td class=\"ConfTableItem\" nowrap
align=\"center\" >Variables</td></tr>\n";
                        }
                }else{
                        print "<tr><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"/cgi-
bin/LogAlarmViewer.cgi?MarkAll=$marktxt".$Filter."\"  class=\"menutext\"><><\/a><\/td><td
class=\"ConfTableItem\" nowrap align=\"center\" >ID</td><td class=\"ConfTableItem\" nowrap align=\"center\"
>Ack</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Tiempo</td><td class=\"ConfTableItem\" nowrap
align=\"center\" >IP</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Comunidad</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Evento</td><td class=\"ConfTableItem\" nowrap
align=\"center\" >Variables</td></tr>\n";
                }
                foreach my $logline (@{$data}){

                                my ($Mark,$Class);
                        if ($query->param('MarkAll') eq "YES"){
                                $Mark="Checked";
                        }

                        if (exists(${$logline}{'marked'}) and ${$logline}{'marked'} ne ""){
                                $Class="ConfTableMarked";
                                print "<input type=\"hidden\" name=\"AckStatetrap_id_".${$logline}{'trap_id'}."\"
value=\"\" />";
                                #print $query->hidden('AckStatetrap_id_'.${$logline}{'trap_id'}," ");
                        }else{
                                $Class="ConfTable";
                                print "<input type=\"hidden\" name=\"AckStatetrap_id_".${$logline}{'trap_id'}."\"
value=\"ACK\" />";
                                #print $query->hidden('AckStatetrap_id_'.${$logline}{'trap_id'},"ACK");
                        }
                        my ($color,$severitylabel);
                        my $noprintflag = 0;
                        if($query->param('TYPE') eq "PRO"){
                                ($color,$severitylabel)=$Log-
>ProcessedAlarmStatus(${$logline}{'severity'},${$logline}{'marked'});
                                if ($query->param('USERS') eq "YES"){
                                        ##Get time from trap information
                                                my ($date,$time)=split(/ /,${$logline}{'time'});
                                                my ($hour,$min,$sec)=split(/:/,$time);
                                                my
($starthourtest,$startmintest)=split(/:/,${$logline}{'starttime'});
                                                my ($endhourtest,$endmintest)=split(/:/,${$logline}{'endtime'});

                                                if (($hour > $starthourtest and  $hour < $endhourtest) or (
$hour == $starthourtest and $min > $startmintest and $min < $endmintest) or ( $hour == $endhourtest and $min
> $startmintest and $min < $endmintest)){
                                                print "<tr ><td  nowrap  class=\"$Class\" style=\"background:
$color\" align=\"center\"><input type=\"checkbox\" name=\"trap_id_".${$logline}{'trap_id'}."\"
value=\"".${$logline}{'trap_id'}."\" ".$Mark."><\/td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background:
$color\" align=\"center\">".${$logline}{'trap_id'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background:
$color\" align=\"center\">".${$logline}{'ack'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background:
$color\" align=\"center\">".${$logline}{'time'}."</td>\n";
                                                print "<td class=\"$Class\" style=\"background: $color\" nowrap
align=\"center\">".$severitylabel.":".${$logline}{'severity'}."</td>\n";
```

```
                                                print "<td  nowrap class=\"$Class\" style=\"background:
$color\" align=\"center\">".${$logline}{'name'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background:
$color\" align=\"center\">".${$logline}{'range'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background:
$color\" align=\"center\">".${$logline}{'starttime'}."-".${$logline}{'endtime'}."</td>\n";
                                                        print "<td  nowrap class=\"$Class\"
style=\"background: $color\" align=\"center\">".${$logline}{'email'}."</td>\n";
                                                        print "<td  nowrap class=\"$Class\"
style=\"background: $color\" align=\"center\">".${$logline}{'action_module_comment'}."</td>\n";

                                                }else{
                                                 $noprintflag = 1;
                                                }
                                        }else{
                                                print "<tr ><td  nowrap  class=\"$Class\" style=\"background: $color\"
align=\"center\"><input type=\"checkbox\" name=\"trap_id_".${$logline}{'trap_id'}."\"
value=\"".${$logline}{'trap_id'}."\" ".$Mark."><\/td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background: $color\"
align=\"center\">".${$logline}{'trap_id'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background: $color\"
align=\"center\">".${$logline}{'ack'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background: $color\"
align=\"center\">".${$logline}{'time'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background: $color\"
align=\"center\">".${$logline}{'equipment'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background: $color\"
align=\"center\">".${$logline}{'equipmenttype_label'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background:
$color\" align=\"center\">".${$logline}{'hosts'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\" style=\"background:
$color\" align=\"center\">".${$logline}{'community'}."</td>\n";
                                                print "<td class=\"$Class\" style=\"background: $color\" nowrap
align=\"center\">".$severitylabel.":".${$logline}{'severity'}."</td>\n";
                                                }
                                }else{
                                                print "<tr ><td nowrap  class=\"$Class\"  align=\"center\"><input
type=\"checkbox\" name=\"trap_id_".${$logline}{'trap_id'}."\" value=\"".${$logline}{'trap_id'}."\" ".$Mark."><\/td>\n";
                                                print "<td  nowrap class=\"$Class\"
align=\"center\">".${$logline}{'trap_id'}."</td>\n";
                                                print "<td  nowrap class=\"$Class\"
align=\"center\">".${$logline}{'ack'}."</td>\n";
                                        print "<td nowrap class=\"$Class\"  align=\"center\">".${$logline}{'time'}."</td>\n";
                                                print "<td nowrap class=\"$Class\"
align=\"center\">".${$logline}{'hosts'}."</td>\n";
                                                print "<td nowrap class=\"$Class\"
align=\"center\">".${$logline}{'community'}."</td>\n";
                                        }
                                                if ($noprintflag == 0){
                                                my $bgcolor;
                                                if ($color ne ""){$bgcolor= "style=\"background: $color\"";}
                                                print "<td nowrap class=\"$Class\" ".$bgcolor."
align=\"center\">".${$logline}{'object'}."</td>\n";
                                                my $count=1;
                                                my $Variables;
                                                while(${$logline}{'variable'.$count} ne ""){

            $Variables.=${$logline}{'variable'.$count}."=".${$logline}{'value'.$count}.";";
                                                        $count++;

                                                }
                                                print "<td nowrap class=\"$Class\" ".$bgcolor."
align=\"center\">".$Variables."</td>\n";
                                                print "<\/tr>\n";
                                        }
        }

print <<EOF;

</table>
EOF
  ;

print $query->endform;
print $query->end_html;
```

9.2.3.7 LogAlarmMenu.cgi

```
#!/usr/bin/perl

#
##Alarms Logs Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance';
use LogManager;


#Main
$query = new CGI;
print $query->header;

print $query->start_html(-title=>'Lista de Eventos',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        -class=>'menubody',
                        );


print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Filtros<\/td><\/tr><\/table>\n";
my $Log=LogManager->new();


##Filter Option
my $filter;
if (defined $query->param('Filter')){
        my @Filter;
        if(defined $query->param('Dia')){push(@Filter,"&Dia=".$query->param('Dia'))}
        if(defined $query->param('Hora')){push(@Filter,"&Hora=".$query->param('Hora'))}
        if(defined $query->param('Minutos')){push(@Filter,"&Minutos=".$query->param('Minutos'))}
        if(defined $query->param('IP')){push(@Filter,"&IP=".$query->param('IP'))}
        if(defined $query->param('Severidad')){push(@Filter,"&Severidad=".$query->param('Severidad'))}
        if(defined $query->param('Equipo')){push(@Filter,"&Equipo=".$query->param('Equipo'))}
        if(defined $query->param('Tipo')){push(@Filter,"&Tipo=".$query->param('Tipo'))}
        if(defined $query->param('ID')){push(@Filter,"&ID=".$query->param('ID'))}
        $filter="Filter=FILTER".join(",@Filter);

}
my $type;
if (defined $query->param('TYPE')){
        $type=$query->param('TYPE');
        print "<script>window.open(\'/cgi-
bin/LogAlarmViewer.cgi?TYPE=PRO&$filter\',\'mainFrame\');</script>";
}else{
        $type="ALL";
        print "<script>window.open(\'/cgi-bin/LogAlarmViewer.cgi?$filter\',\'mainFrame\');</script>";
}
my $data=$Log->ParseAlarms($type);
####################


print <<EOF;
<table width="100%" border="0" cellspacing="1">
EOF
  ;
        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"3\">Filtros</td></tr>\n";
        while(my ($key,$value)=each(%{$data})){
                if (scalar(@{$value}) != 0){
```

```
                                print $query->start_form(-action=>'/cgi-bin/LogAlarmMenu.cgi' );
                                print "<tr><td nowrap   align=\"center\" class=\"ConfTableItem\"  >$key</td>\n";
                                print "<td  nowrap align=\"center\" class=\"menustatic\" >".$query->popup_menu(-
name=>$key,-values=>$value,-default=>$query->param($key))."</td><td align=\"center\"
class=\"ConfTableItem\">".$query->submit('Filter','>>')."</td></tr>\n";
                                print $query->hidden('Dia',$query->param('Dia'));
                                print $query->hidden('Hora',$query->param('Hora'));
                                print $query->hidden('Minutos',$query->param('Minutos'));
                                print $query->hidden('IP',$query->param('IP'));
                                print $query->hidden('Equipo',$query->param('Equipo'));
                                print $query->hidden('Severidad',$query->param('Severidad'));
                                print $query->hidden('Tipo',$query->param('Tipo'));
                                print $query->hidden('ID',$query->param('ID'));
                                print $query->hidden('TYPE',$query->param('TYPE'));
                                print $query->endform;
                        }
                }
print "<tr><td class=\"menu\" colspan=\"3\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"/cgi-
bin/LogAlarmMenu.cgi\"  class=\"menutext\">Todos los Eventos<\/a><\/td><\/tr><tr><td colspan=\"3\"
class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"/cgi-bin/LogAlarmMenu.cgi?TYPE=PRO\"
class=\"menutext\">Eventos Procesados<\/a><\/td><\/tr>\n";
print <<EOF;

</table>
EOF
  ;

print $query->end_html;
```

9.2.3.8 GetSetViewer.cgi

```
#!/usr/bin/perl

#
##Get Set
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/GetSet';
use GetSetManager;

#Main
$query = new CGI;
print $query->header;

my $Focus="";
if ($query->param('FOCUS') eq "ON"){
        $Focus='window.setTimeout("window.focus()", 6000)';
}
print $query->start_html(-title=>'Monitor de Equipo',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -onLoad=>$Focus,
                -onBlur=>$Focus,
                );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<script src=\"/Conf/monitor.js\"><\/script>";
##
my $GetSet= GetSetManager->new();
my ($EquipData)=$GetSet->ReadEquipmentData($query->param('equipment_id'));
if (!defined $query->param('equipmenttype_id')){
        $query->param('equipmenttype_id',${$EquipData}[0]{'equipmenttype_id'});
}
my ($GetSetProfile,$ReadP,$ReadWriteP,$labelsIds)= $GetSet->ReadProf($query->param('equipmenttype_id'));
```

```
if (defined $query->param('map_id')){
	print "<table width=\"100%\"><tr><td  width=\"100%\"
class=\"menustatic\">".${$EquipData}[0]{'equipment'}."<\/td><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=EquipmentViewBack(".$query-
>param('equipment_id').",".$query->param('map_id').",\"".$query->param('MapThread')."\')
class=\"menutext\">Volver a Propiedades<\/a><\/td><\/tr><\/table>\n";
}else{
	if (defined ${$EquipData}[0]{'equipment'}){
		print "<table width=\"100%\"><tr><td  width=\"100%\" class=\"menustatic\">Equipo:
".${$EquipData}[0]{'equipment'}."<\/td><\/tr><\/table>\n";
	}else{
		print "<table width=\"100%\"><tr><td  width=\"100%\" class=\"menustatic\">Seleccione un
Equipo<\/td><\/tr><\/table>\n";
	}
}
###
my ($data,@label,$dataset);
if (defined $query->param('Get') and defined $query->param('get_id')){
	$data=$GetSet->Get(${$GetSetProfile}{$query->param('get_id')});
	@label=("Perfil de Lectura ",${$labelsIds}{$query->param('get_id')});
}
if (defined $query->param('Set') and defined $query->param('set_id')){
	$data=$GetSet->Get(${$GetSetProfile}{$query->param('set_id')});
	@label=("Perfil de Escritura ",${$labelsIds}{$query->param('set_id')});
}
if (defined $query->param('Setting')){
	my %values;
	@label=("Perfil de Escritura ",${$labelsIds}{$query->param('set_id')});
	foreach my $obj (@{${$GetSetProfile}{$query->param('set_id')}}){$values{$obj}=$query->param($obj);}
	$data=$GetSet->Set(${$GetSetProfile}{$query->param('set_id')},\%values);
	$query->param('Set','Setting');

}

###
print <<EOF;
<table  width="100%" border="1" cellspacing="2">
EOF
  ;
print $query->start_form(-action=>'/cgi-bin/GetSetViewer.cgi' );
	print "<tr><td width=\"30%\" class=\"ConfTableItem\" nowrap align=\"center\">Lectura</td><td
width=\"50%\" nowrap class=\"menustatic\" align=\"center\" >".$query->popup_menu(-name=>'get_id',-
values=>$ReadP,-labels=>$labelsIds,'class="Popup"').'</td><td  width=\"40%\" class=\"ConfTableItem\" nowrap
align=\"center\" >".$query->submit('Get','Ejecutar','class="Button"')."</td></tr>\n";
	print "<tr><td width=\"30%\" class=\"ConfTableItem\" nowrap align=\"center\">Escritura</td><td
width=\"50%\" nowrap class=\"menustatic\" align=\"center\" >".$query->popup_menu(-name=>'set_id',-
values=>$ReadWriteP,-labels=>$labelsIds,'class="Popup"').'</td><td width=\"40%\" class=\"ConfTableItem\"
nowrap align=\"center\" >".$query->submit('Set','Ejecutar','class="Button"')."</td></tr>\n";
	if(defined $query->param('set_id') or defined $query->param('get_id')){
		print "<tr><td  class=\"ConfTableItem\" nowrap align=\"center\"><table><tr><td
class=\"ConfTableItem\" nowrap align=\"center\">".$label[0]."</td></tr><tr><td class=\"ConfTableItem\" nowrap
align=\"center\">".$label[1]."</td></tr></table></td><td width=\"100%\" colspan=\"2\"><table width=\"100%\">\n";

		print "<tr><td  class=\"ConfTableItem\" nowrap align=\"center\">Objeto</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Valor</td></tr>\n";

		while(my ($key,$value) = each(%{$data})){
			print "<tr><td  class=\"ConfTableItem\" nowrap align=\"center\" >".$key."</td>";
			if (defined $query->param('Set') and defined $query->param('set_id')){
				print "<td class=\"ConfTable\" nowrap align=\"center\"><input
type=\"text\" name=\"".$key."\" value=\"".$value."\" class=\"Popup\" size=\"60\" maxlength=\"80\" /></td>";
			}else{
				print "<td  class=\"ConfTable\" nowrap align=\"center\">".$value."</td>";
			}
			print "</tr>\n";
		}
		if (defined $query->param('Set') and defined $query->param('set_id')){
			print "<tr><td colspan=\"2\">".$query-
>submit('Setting','Set','class="Button"')."</td></tr>";
		}
		print "</table></td></tr>\n";
	}
	print $query->hidden('set_id',$query->param('set_id'));
	print $query->hidden('equipment_id',$query->param('equipment_id'));
	print $query->hidden('map_id',$query->param('map_id'));
```

```
            print $query->hidden('MapThread',$query->param('MapThread'));
            print $query->hidden('equipmenttype_id',$query->param('equipmenttype_id'));
            print $query->hidden('IP',$query->param('IP'));
            print $query->hidden('FOCUS',$query->param('FOCUS'));
print $query->endform;
print <<EOF;
</table>
EOF
  ;

print $query->end_html;
```

9.2.3.9 ActionMenu.cgi

```
#!/usr/bin/perl

#
##Actions Menu
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/GetSet';
use GetSetManager;

#Main
$query = new CGI;
print $query->header;

print $query->start_html(-title=>'Monitor de Equipo',
                    -style=>{'src'=>'/Conf/WBNMS.css'},
                    -class=>'menubody',
                    );

print "<script src=\"/Conf/menu.js\"><\/script>";
##
my $GetSet= GetSetManager->new();
            my ($Equipments,$EquipLabels)=$GetSet->ReadEquipments();
            print "<table width=\"100%\"><tr><td  width=\"100%\" class=\"menustatic\">Acciones
Menu<\/td><\/tr><\/table>\n";

print <<EOF;
<table  width="100%" border="1" cellspacing="2">
EOF
  ;
print $query->start_form(-action=>'/cgi-bin/GetSetViewer.cgi',-target=>'mainFrame' );
            print "<tr><td  width=\"30%\" class=\"ConfTableItem\" nowrap align=\"center\">Equipos</td><td
width=\"70%\" nowrap class=\"menustatic\" align=\"center\">".$query->popup_menu(-name=>'equipment_id',-
values=>$Equipments,-labels=>$EquipLabels,'class="Popup"')."</td></tr>\n";
            print "<tr><td  class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"2\">".$query-
>submit('Equip','Abrir Equipo','class="Button"')."</td></tr>\n";
print $query->endform;
print <<EOF;
</table>
EOF
  ;

print $query->end_html;
```

9.2.3.10 MIBBrowserResults.cgi

```
#!/usr/bin/perl

#
##MIB Browser
#
```

```
#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

#Main
$query = new CGI;

##Load Cookie
my $readcookie = $query->cookie('TRANOBJ');
###Parse
my @Objs=split(/:/,$readcookie);
####

##Del Option
my $DelFlag=0;
if(defined $query->param('Del') ){
        foreach my $objs ($query->param('OBJ')){
                @Objs=grep($_ ne $objs,@Objs);
        }
        $DelFlag=1;
}
###Ins Option
if(defined $query->param('Ins') ){
        foreach my $objs ($query->param('OBJ')){
                @Objs=grep($_ ne $objs,@Objs);
        }
        $DelFlag=1;
}
###Accept Option
if(defined $query->param('Accept') ){
        my $ind=0;
        while(defined $query->param('obj'.$ind)){
                push(@Objs,$query->param('obj'.$ind));
                $ind++;
                $DelFlag=1;
        }

}
### ReLoad Data
my $cookie;
if ($query->param('Cancel') or $query->param('Add')){
$cookie = $query->cookie(-name=>'TRANOBJ',
                -value=>"",
                 );

}
if ($DelFlag == 1){
my $data =join(':',@Objs);
$cookie = $query->cookie(-name=>'TRANOBJ',
                -value=>$data,
                 );
}
print $query->header(-cookie=>$cookie,);
##
if ($query->param('FOCUS') eq "ON"){
        $Focus='window.focus()';
}

print $query->start_html(-title=>'Administracion de MIBs',
                 -style=>{'src'=>'/Conf/WBNMS.css'},
                 -onLoad=>$Focus,
                ## -onBlur=>$Focus,
                 );

if(defined $query->param('Add') or defined $query->param('Cancel')){ ##Add and Cancel Option
my $OBJ;
if (!defined $query->param('Cancel')){$OBJ=join(':',$query->param('OBJ'));}

print "<script>window.open(\"/cgi-bin-
admin/ProfileCompView.cgi?FOCUS=ON&Adding=YES&getset_id=".$query-
>param('getset_id')."&OBJ=".$OBJ."\",\"Profiles\")<\/script>";
```

```
print "<p> </p><p> </p><p> </p><p> </p><p align=\"center\"><strong><font
color=\"#FF0000\" size=\"+1\">Cargando ...</font></strong></p>";
print "<script>top.close();<\/script>";

}elsif (defined $query->param('Del')){ ##Del Option
print "<script>window.open(\"/cgi-bin/MIBBrowserResults.cgi?FOCUS=ON&getset_id=".$query-
>param('getset_id')."\",\"resultsfrm\")<\/script>";
print "<p> </p><p> </p><p> </p><p> </p><p align=\"center\"><strong><font
color=\"#FF0000\" size=\"+1\">Borrando ...</font></strong></p>";

}elsif(defined $query->param('Accept')){ ##Mod Option
print "<script>window.open(\"/cgi-bin/MIBBrowserResults.cgi?FOCUS=ON&getset_id=".$query-
>param('getset_id')."\",\"resultsfrm\")<\/script>";
}else{

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Objetos<\/td><\/tr><\/table>\n";

print <<EOF;
<table width="100%" border="0" cellspacing="1">
EOF
  ;
        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\"
width=\"100%\">".${$Data}[0]{'getset_label'}."</td></tr>\n";
  print $query->start_form(-action=>'/cgi-bin/MIBBrowserResults.cgi' );
        if(defined $query->param('Ins')){
                my $ind=0;
                foreach my $obj ($query->param('OBJ')){
                        print $query->textfield(-name=>'obj'.$ind,-value=>$obj,'class="Popup"');
                        $ind++;
                }
                print $query->submit('Accept','Acceptar','class="Button"');
        }
        print "<tr><td class=\"ConfTable\" nowrap align=\"center\">".$query->scrolling_list(-name=>'OBJ',-
values=>\@Objs,-size=>20,-multiple=>'true','class="Popup"')."</td></tr>\n";
        print "<tr><td class=\"ConfTableItem\"><table width=\"100%\"><tr><td>".$query-
>submit('Del','Quitar','class="Button"')."</td><td>".$query-
>submit('Ins','Modificar','class="Button"')."</td></tr></table></td></tr>";
        print "<tr><td class=\"ConfTableItem\">".$query->submit('Add','Agregar','class="Button"')."</td></tr>";
        print "<tr><td class=\"ConfTableItem\">".$query-
>submit('Cancel','Cancelar','class="Button"')."</td></tr>";
        print $query->hidden('FOCUS',$query->param('FOCUS'));
        print $query->hidden('getset_id',$query->param('getset_id'));
  print $query->endform;
print <<EOF;

</table>
EOF
  ;
}
print $query->end_html;
```

9.2.3.11 MIBBrowserFrameset.cgi

```
#!/usr/bin/perl

#
##MIB Browser
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;


#Main
$query = new CGI;

print $query->header;


if (defined $query->param('Select') or defined $query->param('Trigger')){
```

```
            print "<frameset rows=\"*\" cols=\"289,*,170\" framespacing=\"2\" frameborder=\"yes\" border=\"2\"
bordercolor=\"#006699\">";
}else{
            print "<frameset rows=\"*\" cols=\"289,*\" framespacing=\"2\" frameborder=\"yes\" border=\"2\"
bordercolor=\"#006699\">";
}

print <<EOF;
  <frame src="/MIBbrowserWeb/folderTreeLeftFrameiso.html" name="treefrm" scrolling="yes" noresize>
  <frame src="/MIBbrowserWeb/MIBBrowserMain.html" name="basefrm">
EOF
   ;
if (defined $query->param('Select')){
   print "<frame src=\"/cgi-bin/MIBBrowserResults.cgi?FOCUS=ON&getset_id=".$query->param('getset_id')."\"
name=\"resultsfrm\">";
}
if (defined $query->param('Trigger')){
   my @data=$query->param('actions_id');
   print "<frame src=\"/cgi-bin-admin/MIBBrowserResultsTriggers.cgi?FOCUS=ON&trigger_id=".$query-
>param('trigger_id')."&TYPE=".$query->param('TYPE')."&ModView=".$query-
>param('ModView')."&equipment_id=".$query->param('equipment_id')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&triggernot_id=".$query->param('triggernot_id')."&severity=".$query-
>param('severity')."&actions_id=".join(':',@data)."\" name=\"resultsfrm\">";
}
print <<EOF;
</frameset>
EOF
   ;
print $query->start_html(-title=>'Navegador de  MIBs',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        );


print $query->end_html;
```

9.2.4 webadm/htdocs

9.2.4.1 Administration.html

```
<html>
<head>
<title>Administracion</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
@import url("/Conf/WBNMS.css");
-->
</style>
</head>
<script language="JavaScript" type="text/JavaScript">
function Load(){

window.open("mainAdmin.html",'mainFrame');
window.open("menuAdmin.html",'leftFrame');
}
</script>

<body onLoad="Load()">

<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p align="center"><strong><font color="#FF0000" size="+5">Cargando ...</font></strong></p>
</body>
</html>
```

9.2.4.2 menuAdmin.html

```
<html>
<head>
<title>Administracion</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
@import url("/Conf/WBNMS.css");
-->
</style>
</head>

<body class=menubody>
<table width="100%">
  <tr>
    <td class="menustatic">Administraci&oacute;n</td>
  </tr>
</table>
<p>
  <script src="/Conf/menu.js"></script>
</p>
<p>  </p>
<table border="1" align="center"  class="menuborder" width="100%">
  <tr>
    <td nowrap class="menuborderwtext" >Sistema</td>
  </tr>
   <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/UsersAdmin.cgi"
target="mainFrame" class="menutext">Usuarios
       del Sistema</a></div></td>
  </tr>
  <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/webadm/htdocs/Status.html"
target="mainFrame"  class="menutext">Estado
       del Sistema</a></div></td>
  </tr>
  <tr>
    <td nowrap class="menuborderwtext" >Equipos</td>
  </tr>
  <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/EquipmentTypesAdmin.cgi"
target="mainFrame"  class="menutext">Tipos
       de Equipo</a></div></td>
  </tr>
   <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/PollAdmin.cgi"
target="mainFrame"  class="menutext">Encuesta
       Autom&aacute;tica </a></div></td>
  </tr>
  <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/EquipmentsAdmin.cgi"
target="mainFrame"  class="menutext">Equipos</a></div></td>
  </tr>
  <tr>
    <td nowrap class="menuborderwtext" >Mapas</td>
  </tr>
  <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/MapsAdmin.cgi"
target="mainFrame"  class="menutext">Mapas</a></div></td>
  </tr>
    <tr>
    <td nowrap class="menuborderwtext" >Alertas</td>
  </tr>
  <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/AssignationsActionsAdmin.cgi"
target="mainFrame"  class="menutext">Asignación
       de Alertas</a></div></td>
```

```
  </tr>
  <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/UsersActionsAdmin.cgi"
target="mainFrame"  class="menutext">Usuarios
       de Alertas</a></div></td>
  </tr>
   <tr>
    <td nowrap class="menuborderwtext" >Alarmas</td>
  </tr>
  <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/AlarmTriggersAdmin.cgi"
target="mainFrame"  class="menutext">Alarmas</a></div></td>
  </tr>
   <tr>
    <td nowrap class="menuborderwtext" >Perfiles de Acción</td>
  </tr>
    <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/ProfilesAdmin.cgi"
target="mainFrame"  class="menutext">Perfiles
       de Acción</a></div></td>
  </tr>
   <tr>
    <td nowrap class="menuborderwtext" >MIBs</td>
  </tr>
  <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin-admin/MIBsAdmin.cgi"
target="mainFrame" class="menutext">MIBs</a></div></td>
  </tr>
 <tr>
    <td nowrap class="menu" onClick="MenuSelected(this)" onMouseOver="MenuMouseOver(this)"
onMouseOut="MenuMouseOut(this)"><div align="center"><a href="/cgi-bin/MIBBrowserFrameset.cgi"
target="mainFrame" class="menutext">Navegador
       de MIBs</a></div></td>
  </tr>

</table>
<p> </p>
</body>
</html>
```

9.2.4.3 Status.html

```
<html>
<head>
<title>WBNMS</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<frameset rows="170,*" cols="*" framespacing="2" frameborder="yes" border="2" bordercolor="#006699">
  <frame src="/cgi-bin-admin/Status.cgi" name="topStatusFrame" scrolling="NO" noresize >
  <frame src="/webadm/htdocs/mainAdmin.html" name="mainStatusFrame">
</frameset>
<noframes><body>

</body></noframes>
</html>
```

9.2.5 webadm/cgi-bin

9.2.5.1 UsersAdmin.cgi

```
#!/usr/bin/perl

#
##Users and Groups Administration
#

#Load Libraries
```

```
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;


use lib '/var/sites/Wbnms/Wbnms_Core/Auth';
use Auth;


#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Administracion de Usuarios',
                         -style=>{'src'=>'/Conf/WBNMS.css'},
                         );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Usuarios del Sistema<\/td><\/tr><\/table>\n";
print "<p> <\/p>\n";

my $Auth = Auth->new();

if (defined $query->param('User') and defined $query->param('Group') and !defined $query->param('Del')){
        if ( defined $query->param('Password') and defined $query->param('CheckPass') and $query-
>param('Password') eq $query->param('CheckPass')){
                $Auth->Name($query->param('User'));
                $Auth->Password($query->param('Password'));
                $Auth->CheckPass($query->param('CheckPass'));
                $Auth->Group($query->param('Group'));
                $Auth->AddUsers();
                $Auth->AddGroupUsers();
                $query->param('User',"");
print <<EOF;

                <script language="JavaScript" type="text/JavaScript">
                window.open("/cgi-bin-admin/Close.cgi",'Pass');
                </script>

EOF
  ;
        }elsif(!defined $query->param('Password') and !defined $query->param('CheckPass')){

                print "<script> window.open(\"\/cgi-bin-admin\/Pass.cgi?User=".$query-
>param('User')."&Group=".$query->param('Group')."\",\'Pass\',\'scrollbars=no,width=320,height=120\');<\/script>";

        }
}
if (defined $query->param('User') and defined $query->param('Del')){
                $Auth->Name($query->param('User'));
                $Auth->DelUser();
                $query->param('User',"");

}
my ($users,$groups) = $Auth->readGroupsandUsers();
my %usergroup;
my @groups=keys(%{$groups});
while (my ($group,$usersarray)=each(%{$groups})){
        foreach my $user (@{$users}){
                if(grep($_ eq $user,@{$usersarray})){
                        $usergroup{$user}=$group;
                }


        }
}
print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table  border="1" cellspacing="0">
  <tr>
    <td width="30%" class=\"ConfTableTitle\" align="center">Usuarios</td>
    <td nowrap>
```

```
            <table width="100%" border="1" cellspacing="0">

EOF
  ;
                        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\">Usuario</td><td
class=\"ConfTableItem\" nowrap align=\"center\">Grupo</td><td class=\"ConfTableTitle\" nowrap
align=\"center\">   </td></tr>\n";
        foreach my $user(@{$users}){

                        print "<tr><td nowrap class=\"ConfTable\" align=\"center\">$user</td>\n";
                         print "<td nowrap class=\"ConfTable\" align=\"center\" >".$usergroup{$user}."</td>\n";
             print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/UsersAdmin.cgi?Del=DEL&User=".$user."\" class=\"menutext\">Borrar</a></td></tr>\n";

        }
        print $query->start_form(-action=>'/cgi-bin-admin/UsersAdmin.cgi' );
              print "<tr><td nowrap>".$query->textfield('User')."</td>\n";
              print "<td nowrap>".$query->popup_menu(-name=>'Group',-values=>\@groups,-default=>")."</td>\n";
              print "<td nowrap align=\"center\" width=\"120\">".$query-
>submit('Agregar','Agregar','class="Button"')."</td></tr>\n";
              print $query->endform;

print <<EOF;

    </table></td>
  </tr>
</table>
   </td>
  </tr>
</table>
EOF
  ;

print $query->end_html;
```

9.2.5.2 Pass.cgi

```
#!/usr/bin/perl

#
##Users and Groups Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

#Main
$query = new CGI;
print $query->header;

print $query->start_html(-title=>'Password',
                    -style=>{'src'=>'/Conf/WBNMS.css'},
                    -onLoad=>'window.focus()',
                    );
print $query->start_form(-action=>'/cgi-bin-admin/UsersAdmin.cgi',
                                    -target => 'mainFrame',
);
print <<EOF;
<table align ="center" border="1" cellspacing="1">
  <tr>
EOF
  ;
print "<td class=\"ConfTable\" nowrap align=\"center\">Password<\/td>\n";
print "<td>".$query->password_field('Password')."<\/td><\/tr>\n";
print "<tr><td class=\"ConfTable\" nowrap align=\"center\">CheckPassword<\/td>\n";
print "<td>".$query->password_field('CheckPass')."<\/td><\/tr>\n";
print "<tr><td class=\"ConfTable\" align=\"center\"nowrap>".$query-
>reset('Borrar','Borrar','class="Button"')."<\/td>\n";
print "<td class=\"ConfTable\" align=\"center\" nowrap>".$query-
>submit('OK','OK','class="Button"')."<\/td><\/tr><\/table>\n";
print $query->hidden('User',$query->param('User'));
```

```
print $query->hidden('Group',$query->param('Group'));
print $query->endform;


print $query->end_html;
```

9.2.5.3 Status.cgi

```
#!/usr/bin/perl

#
##Daemons Status Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Status';
use StatusChecker;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Status',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Estado del Sistema<\/td><\/tr><\/table>\n";

my $Status = StatusChecker->new();

my @test = $Status->CheckStatus();##($AlarmSd,$trapd,$Drawerd,$keepalive,$testDB)

print <<EOF;
<table width="100%" border="0" cellspacing="1">
  <tr></tr>
EOF
   ;
    foreach my $test (@test){
              my ($color,$action);
              while(my ($key,$value)=each(%{$test})){
                        if ($value == 1 ){
                                  $color = "#00FF00";
                                  $value = "Encendido";
                        }elsif($value == 0 ){
                                  $color = "#FF0000";
                                  $value = "Apagado";
                        }
                        $action="/cgi-bin-admin/LogDaemon.cgi?Daemon=$key";
             print "<tr><td nowrap align=\"center\" bgcolor=\"".$color."\" ><strong><font
color=\"#000000\">$key<\/font><\/strong><\/td><td class=\"ConfTable\" align=\"center\">$value<\/td><td
class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\" ><a href=\"$action\" target=\"mainStatusFrame\" class=\"menutext\">Ver
Log<\/a><\/td><\/tr>\n";

              }
     }
print <<EOF;
</table>
EOF
   ;



print $query->end_html;
```

9.2.5.4 LogDaemon.cgi

```
#!/usr/bin/perl

#
##Daemons Log Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Status';
use DaemonsLogViewer;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Logs',
                         -style=>{'src'=>'/Conf/WBNMS.css'},
                         );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Logs<\/td><\/tr><\/table>\n";

my $Log = DaemonsLogViewer->new();

if (defined $query->param('Del') and defined $query->param('LOG') and defined $query->param('Path')){
                $Log->DelLog($query->param('Path'),$query->param('LOG'));
}

if(defined $query->param('Daemon')){
my ($daemonlogs,$daemonlogdir) = $Log->ReadLogs($query->param('Daemon'));
print <<EOF;
<table width="100%" height="95%" >
  <tr>
EOF
   ;
                my $count = 0;
                while(my ($daemonC,$logs )=each(%{$daemonlogs})){
                        print "<td ><table width=\"100%\"><tr><td class=\"ConfTableItem\" nowrap
align=\"center\">".$daemonC."<\/td><td class=\"ConfTableTitle\" nowrap align=\"center\">   <\/td><\/tr>\n";
                print $query->start_form(-action=>'/cgi-bin-admin/LogViewer.cgi',-target=>'logviewFrame' );
                        print "<tr><td width=\"100%\" class=\"menustatic\" align=\"center\">".$query->popup_menu(-
name=>'LOG',-values=>$logs,-default=>'')."<\/td>\n";
                        print "<td width=\"100%\" class=\"menustatic\"  align=\"center\" >".$query-
>submit('Ver')."<\/a><\/td><\/tr><\/table></td>\n";
                        print $query->hidden('DaemonComp',$daemonC);
                        print $query->hidden('Daemon',$query->param('Daemon'));
                        print $query->hidden('Path',${$daemonlogdir}{$daemonC});
                        print $query->endform;
                        $count++;
                }

                print "<\/tr><tr height=\"100%\">\n";
                print "<td  colspan=\"$count\"><iframe width=\"100%\" height=\"100%\" src=\"/cgi-bin-
admin/LogViewer.cgi\" name=\"logviewFrame\"><\/iframe><\/td>\n";


print <<EOF;
  </tr>
</table>
EOF
   ;
}

print $query->end_html;
```

9.2.5.5 LogViewer.cgi

```
#!/usr/bin/perl

#
##Daemons Log Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Status';
use DaemonsLogViewer;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'LogsViewer',
                         -style=>{'src'=>'/Conf/WBNMS.css'},
                         );

print "<script src=\"/Conf/menu.js\"><\/script>";

my $Log = DaemonsLogViewer->new();

if( defined $query->param('DaemonComp') and defined $query->param('LOG') and defined $query-
>param('Path')){

print "<table width=\"100%\"><tr><td class=\"menustatic\">".$query->param('DaemonComp').": ".$query-
>param('LOG')."<\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"  onClick=\"print()\"
class=\"menutext\">Imprimir<\/a><\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"/cgi-bin-
admin/LogDaemon.cgi?Del=YES&Path=".$query->param('Path')."&LOG=".$query-
>param('LOG')."&Daemon=".$query->param('Daemon')."\"  target=\"mainStatusFrame\"
class=\"menutext\">Borrar<\/a><\/td><\/tr><\/table>\n";
my $logtxt = $Log->OpenLog($query->param('Path'),$query->param('LOG'));

print <<EOF;
<table width="100%">

EOF
  ;

            foreach my $logline (@{$logtxt}){
                    print "<tr>\n";
                    foreach my $item (@{$logline}){
                            print "<td nowrap class=\"ConfTable\" align=\"center\">$item</td>\n";
            }
            print "<\/tr>\n";
             }
print <<EOF;

</table>
EOF
  ;
}

print $query->end_html;
```

9.2.5.6 EquipmentTypesAdmin.cgi

```
#!/usr/bin/perl

#
##Equipment Types Administration
#
```

```
#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Drawer';
use EquipmentManager;
use BmpManager;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'EquipmentsAdmin',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Tipos de Equipos<\/td><\/tr><\/table>\n";

my $Equip = EquipmentManager->new();
my $SeeBmp = BmpManager->new();

###Delete Option
if(defined $query->param('Del')){
        my %data=(
                'equipmenttype_id' => $query->param('equipmenttype_id'),
        );
        my $error = $Equip->DelType(\%data);
        if ($error ne "1"){
                print $error;
        }
}
###Add Option
if(defined $query->param('Add')){
        my %data=(
                'equipmenttype_label' => $query->param('equipmenttype_label'),
                'equipmentbmp' => $query->param('equipmentbmp'),

        );
        $Equip->AddType(\%data);
        $query->param('ModView',"");
        $query->param('equipmenttype_id',"");
        $query->param('equipmenttype_label',"");
        $query->param('equipmentbmp',"");

}
###Modify Option
if(defined $query->param('Mod')){
        my %search=(
                'equipmenttype_id' => $query->param('equipmenttype_id'),
        );
        my %data=(
                'equipmenttype_label' => $query->param('equipmenttype_label'),
                'equipmentbmp' => $query->param('equipmentbmp'),

        );
        $Equip->ModType(\%search,\%data);
        $query->param('ModView',"");
        $query->param('equipmenttype_id',"");
        $query->param('equipmenttype_label',"");
        $query->param('equipmentbmp',"");

}
###Modify Cancel Option
if(defined $query->param('Cancel')){
$query->param('ModView',"");
$query->param('equipmenttype_id',"");
$query->param('equipmenttype_label',"");
$query->param('equipmentbmp',"");
}
###open BMP Viewer
```

```
if(defined $query->param('See')){
print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
EOF
  ;
                        print "See = window.open(\"/cgi-bin-admin/Seebmp.cgi?See=".$query-
>param('See')."&ModView=".$query->param('ModView')."&equipmenttype_label=".$query-
>param('equipmenttype_label')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&equipmentbmp=".$query-
>param('equipmentbmp')."\",\'See\',\'scrollbars=yes,width=400,height=600\');";
print <<EOF;
                        See.moveTo(250,100);
                        </script>


EOF
  ;
}
## AddView Options
if(defined $query->param('AddView')){
print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
EOF
  ;
                        print "AddViewFrame = window.open(\'/cgi-bin-
admin/TypesProfiles.cgi?FOCUS=ON&equipmenttype_id=".$query-
>param('equipmenttype_id')."&equipmenttype_label=".$query-
>param('equipmenttype_label')."\',\'AddViewFrame\',\'scrollbars=no,width=800,height=440\');";
print <<EOF
                        AddViewFrame.moveTo(100,0);
                        </script>


EOF
  ;
}

###Get Information
my ($EquipmentData,$EquipmentTypes) = $Equip->ReadEquipments();
my ($bmps,$path) = $SeeBmp->Seebmp("Tipos");


### Table Begining
print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table  width="100%" border="1" cellspacing="0">
  <tr>
    <td width="30%" class=\"ConfTableTitle\" align="center">Tipos</td>
    <td nowrap >
            <table width="100%" border="1" cellspacing="0">
EOF
  ;
            print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"8\">Tipos de Equipos
Existentes</td></tr>\n";
            print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >Tipo</td><td class=\"ConfTableItem\"
nowrap align=\"center\" >Icono</td><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"10%\" ></td><td
class=\"ConfTableItem\" nowrap align=\"center\" width=\"10%\"></td></tr>\n";
            my $modflag = 0;
  foreach my $type(@{$EquipmentTypes}){
                        if (${$type}{'equipmenttype_id'} == $query->param('equipmenttype_id') and $query-
>param('ModView') eq "MOD"){
                                    print $query->start_form(-action=>'/cgi-bin-admin/EquipmentTypesAdmin.cgi' );
                                    print "<tr><td nowrap align=\"center\" width=\"100%\">".$query-
>textfield('equipmenttype_label',$query->param('equipmenttype_label'))."</td>\n";
                                    print "<td><table><tr><td nowrap align=\"center\" width=\"100%\">".$query-
>popup_menu(-name=>'equipmentbmp',-values=>$bmps,-default=>$query->param('equipmentbmp'))."<\/td><td
nowrap align=\"center\" >".$query->submit('See','Tipos')."<\/td><\/tr><\/table><\/td>\n";
                                    print "<td nowrap align=\"center\" >".$query->submit('Mod','Modificar')."</td><td
nowrap align=\"center\" >".$query->submit('Cancel','Cancelar')."</td></tr>\n";
                                    print $query->hidden('equipmenttype_id',$query->param('equipmenttype_id'));
                                    print $query->hidden('ModView',$query->param('ModView'));
                                    print $query->endform;
                                    $modflag = 1;
```

```
                        }else{
                                print "<tr><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/EquipmentTypesAdmin.cgi?AddView=YES&equipmenttype_id=".${$type}{'equipmenttype_id'}."&equipme
nttype_label=".${$type}{'equipmenttype_label'}."&equipmentbmp=".${$type}{'equipmentbmp'}."\"
class=\"menutext\" TITLE=\"Agregar Perfiles al tipo ".${$type}{'equipmenttype_label'}."\"
>".${$type}{'equipmenttype_label'}."<\/a></td>\n";
                        print "<td nowrap class=\"ConfTable\" align=\"center\">".${$type}{'equipmentbmp'}."</td>\n";
                                print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/EquipmentTypesAdmin.cgi?ModView=MOD&equipmenttype_id=".${$type}{'equipmenttype_id'}."&equipme
nttype_label=".${$type}{'equipmenttype_label'}."&equipmentbmp=".${$type}{'equipmentbmp'}."\"
class=\"menutext\">Modificar</a></td>\n";
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/EquipmentTypesAdmin.cgi?Del=DEL&equipmenttype_id=".${$type}{'equipmenttype_id'}."\"
class=\"menutext\">Borrar</a></td></tr>\n";
                }
  }
            if ($modflag == 0 ){

                        print $query->start_form(-action=>'/cgi-bin-admin/EquipmentTypesAdmin.cgi' );
                        print "<tr><td nowrap align=\"center\">".$query->textfield('equipmenttype_label')."</td>\n";
                        print "<td><table width=\"100%\"><tr><td nowrap align=\"center\" >".$query->popup_menu(-
name=>'equipmentbmp',-values=>$bmps,-default=>$query->param('equipmentbmp'))."<\/td><td nowrap
align=\"center\" >".$query->submit('See','Tipos')."<\/td><\/tr><\/table><\/td>\n";
                        print "<td nowrap align=\"center\" colspan=\"2\" width=\"100%\">".$query-
>submit('Add','Agregar Tipo','class="Button"' )."</td></tr>\n";
                        print $query->endform;
            }

print <<EOF;
     </table></td>
  </tr>
</table>
   </td>
  </tr>
</table>
EOF
  ;

print $query->end_html;
```

9.2.5.7 Seebmp.cgi

```
#!/usr/bin/perl

#
##See BMP
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Drawer';
use BmpManager;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Seebmp',
                    -style=>{'src'=>'/Conf/WBNMS.css'},
                    -onLoad=>'window.focus()',
                    -onBlur=>'window.focus()',
                    );
```

```
print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">".$query->param('See')."<\/td><td class=\"menu\"
nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.close()\"
class=\"menutext\">Cerrar<\/a><\/td><\/tr><\/table>\n";
print $query->start_multipart_form(-action=>'/cgi-bin-admin/Seebmp.cgi' );
print "<table width=\"100%\" border=\"0\" cellspacing=\"0\" ><tr><td class=\"ConfTableItem\" nowrap
align=\"center\" colspan=\"2\">Agregar Nuevo Archivo Gif</td><\/tr><tr><td width=\"100%\" nowrap
align=\"center\" class=\"menustatic\">".$query->filefield('uploaded_file')."<\/td><td width=\"100%\" nowrap
align=\"center\" class=\"menustatic\">".$query->submit('Add','Agregar')."<\/td><\/tr><\/table>\n";
print $query->hidden('See',$query->param('See'));
print $query->hidden('mapname',$query->param('mapname'));
print $query->hidden('mapicon',$query->param('mapicon'));
print $query->hidden('mapbmp',$query->param('mapbmp'));
print $query->hidden('comment',$query->param('comment'));
print $query->hidden('equipmenttype_label',$query->param('equipmenttype_label'));
print $query->hidden('equipmenttype_id',$query->param('equipmenttype_id'));
print $query->hidden('equipmentbmp',$query->param('equipmentbmp'));
print $query->endform;

##################
##Load Files
my $Seebmp = BmpManager->new();
my ($bmps,$path) = $Seebmp->Seebmp($query->param('See'));

##UpLoad Option
if (defined $query->param('uploaded_file')){
        my $filename = $query->param('uploaded_file');
        $Seebmp->UpLoad($filename,$path);
}
## Delete File Option
if (defined $query->param('Del')){
        my $error = $Seebmp->DelBmp($query->param('bmp'),$path,$query->param('See'));
        if ($error ne "1"){
                print $error;
        }
}
####################
##ReLoad Files
($bmps,$path) = $Seebmp->Seebmp($query->param('See'));

####################

print <<EOF;
<table width="100%" border="0" cellspacing="0">
EOF
  ;

if ($query->param('See') eq "Mapas" or $query->param('See') eq "Iconos"){
        my ($iconmaps,$maps);
        foreach my $bmp (@{$bmps}){

                if ($query->param('See') eq "Mapas"){
                        $iconmaps=$query->param('mapicon');
                        $maps=$bmp;
                }
                if ($query->param('See') eq "Iconos"){
                        $iconmaps=$bmp;
                        $maps=$query->param('mapbmp');
                }

                 print "<tr><td align=\"center\"><img src=\"".$path.$bmp."\" width=\"150\"
height=\"150\"><\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/MapsAdmin.cgi?ModView=".$query->param('ModView')."&map_id=".$query-
>param('map_id')."&mapname=".$query-
>param('mapname')."&mapicon=".$iconmaps."&mapbmp=".$maps."&comment=".$query->param('comment')."\"
target=\"mainFrame\" class=\"menutext\">$bmp</a></td><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/Seebmp.cgi?Del=YES&bmp=".$bmp."&See=".$query->param('See')."&mapname=".$query-
>param('mapname')."&mapicon=".$iconmaps."&mapbmp=".$maps."&comment=".$query->param('comment')."\"
class=\"menutext\">Borrar</a></td><\/tr><\/tr>\n";
        }
}elsif($query->param('See') eq "Tipos"){
        foreach my $bmp (@{$bmps}){
```

```
                        print "<tr><td align=\"center\"><img src=\"".$path.$bmp."\" width=\"150\"
height=\"150\"><\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/EquipmentTypesAdmin.cgi?ModView=".$query->param('ModView')."&equipmenttype_label=".$query-
>param('equipmenttype_label')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&equipmentbmp=".$bmp."\" target=\"mainFrame\"
class=\"menutext\">$bmp<\/a><\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/Seebmp.cgi?Del=YES&bmp=".$bmp."&See=".$query->param('See')."&equipmenttype_label=".$query-
>param('equipmenttype_label')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&equipmentbmp=".$bmp."\"  class=\"menutext\">Borrar<\/a><\/td><\/tr><\/tr>\n";
        }

}
print <<EOF;
</table>
EOF
  ;

print $query->end_html;
```

9.2.5.8 TypesProfiles.cgi

```
#!/usr/bin/perl

#
##Equipment Types Profiles
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/GetSet';
use GetSetManager;

#Main
$query = new CGI;
print $query->header;

my $Focus="";
if ($query->param('FOCUS') eq "ON"){
        $Focus='window.focus()';
}
print $query->start_html(-title=>'Monitor de Equipo',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -onLoad=>$Focus,
                -onBlur=>$Focus,
                );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Perfiles Asociados al Tipo ".$query-
>param('equipmenttype_label')."<\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onClick=\'window.close()\') class=\"menutext\">Cerrar<\/a><\/td><\/tr><\/table>\n";

my $Profiles=GetSetManager->new();
##Del Option
if(defined $query->param('Del')){
        foreach my $prof_id ($query->param('Selected')){
                $Profiles->DelProf($query->param('equipmenttype_id'),$prof_id);
        }
}
##Add Option
if(defined $query->param('Add')){
        foreach my $prof_id ($query->param('Rest')){
                $Profiles->AddProf($query->param('equipmenttype_id'),$prof_id);
        }
}
```

```
##

my ($GetSetProfile,$ReadP,$ReadWriteP,$labelsIds)=$Profiles->ReadProf($query->param('equipmenttype_id'));
my ($AllProfiles,$ProfilesLabels,$Data)=$Profiles->ReadAllProfiles($ReadP,$ReadWriteP,undef);
### Table Begining
print <<EOF;
<table width="100%" border="1" cellspacing="1">
EOF
   ;
   print $query->start_form(-action=>'/cgi-bin-admin/TypesProfiles.cgi' );
   print "<tr><td width=\"50%\" nowrap align=\"center\" class=\"ConfTableItem\">Incluidos</td><td width=\"50%\"
nowrap align=\"center\" class=\"ConfTableItem\">Restantes</td></tr>\n";
   print "<tr><td width=\"50%\">".$query->scrolling_list(-name=>'Selected',-
values=>[@{$ReadP},@{$ReadWriteP}],-size=>20,-multiple=>'true',-
labels=>$labelsIds,'class="Popup"')."</td>\n";
   print "<td width=\"50%\">".$query->scrolling_list(-name=>'Rest',-values=>$AllProfiles,-size=>20,-
multiple=>'true',-labels=>$ProfilesLabels,'class="Popup"')."</td></tr>\n";
   print "<tr><td>".$query->submit('Del','Quitar','class="Button"')."</td><td>".$query-
>submit('Add','Agregar','class="Button"')."</td></tr>";
   print $query->hidden('FOCUS',$query->param('FOCUS'));
   print $query->hidden('equipmenttype_id',$query->param('equipmenttype_id'));
   print $query->hidden('equipmenttype_label',$query->param('equipmenttype_label'));
   print $query->endform;
print <<EOF;

</table>
EOF
   ;

print $query->end_html;
```

9.2.5.9 PollAdmin.cgi

```
#!/usr/bin/perl

#
##Poll Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/keepalive';
use PollManager;


#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Logs',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Administracion de Equipos
Encontrados<\/td><\/tr><\/table>\n";

my $Polling = PollManager->new();


##Add Option
if(defined $query->param('Add')){
        my %adddata=(
                        'IP'=>$query->param('IP'),
                        'equipment'=>$query->param('equipment'),
                        'community'=>$query->param('community'),
                        'equipmenttype_id'=>$query->param('equipmenttype_id'),
```

```
		);
		$Polling->AddEquipment(\%adddata);
}

##Del Option
if(defined $query->param('Del')){
		my %deldata=(
			'IP'=>$query->param('IP'),
		);
		$Polling->DelEquipment(\%deldata);
}
######

my ($NewEquipmentData,$types)= $Polling->ReadEquipments();
my $typesvalues = keys(%{$types});

print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table  width="100%" border="1" cellspacing="0">
  <tr>
    <td width="10%" class=\"ConfTableTitle\" align="center">Encuesta Automática</td>
    <td nowrap>
		<table width="100%" border="1" cellspacing="0" >

EOF
   ;
print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"5\">Equipos Encontrados</td></tr>\n";
print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\">IP</td><td class=\"ConfTableItem\" nowrap
align=\"center\">Nombre</td><td class=\"ConfTableItem\" nowrap align=\"center\">Tipo</td><td
class=\"ConfTableItem\" nowrap align=\"center\"></td><td class=\"ConfTableItem\" nowrap
align=\"center\"></td></tr>\n";
foreach my $NewEquipment(@{$NewEquipmentData}){
		print $query->start_form(-action=>'/cgi-bin-admin/PollAdmin.cgi' );
		print "<tr><td nowrap class=\"ConfTable\" align=\"center\">".${$NewEquipment}{'IP'}."</td><td nowrap
class=\"ConfTable\" align=\"center\">".${$NewEquipment}{'equipment'}."</td><td nowrap class=\"ConfTable\"
align=\"center\" width=\"15%\">".$query->popup_menu(-name=>'equipmenttype_id',-values=>$typesvalues,-
labels=>$types)."</td><td nowrap width=\"100\" class=\"ConfTable\" align=\"center\">".$query-
>submit('Add','Agregar','class="Button"')."</td><td nowrap width=\"100\" class=\"ConfTable\"
align=\"center\">".$query->submit('Del','Borrar','class="Button"')."</td></tr>";
		print $query->hidden('equipment',${$NewEquipment}{'equipment'});
		print $query->hidden('IP',${$NewEquipment}{'IP'});
		print $query->hidden('community',${$NewEquipment}{'community'});
		print $query->endform;
}




print <<EOF;

     </table></td>
  </tr>
</table>
   </td>
  </tr>
</table>
EOF
   ;

print $query->end_html;
```

9.2.5.10 EquipmentsAdmin.cgi

```
#!/usr/bin/perl

#
##Equipments Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;
```

```
use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Drawer';
use EquipmentManager;

use lib '/var/sites/Wbnms/Wbnms_Core/MIBCompiler';
use MIBManager;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'EquipmentsAdmin',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Equipos<\/td><\/tr><\/table>\n";

my $Equip = EquipmentManager->new();
if(defined $query->param('Del')){
        my %data=(
                        'equipment_id' => $query->param('equipment_id'),
        );
        $Equip->DelEquipment(\%data);

}
if(defined $query->param('Add')){
        my %data=(
                        'equipment' => $query->param('equipment'),
                        'equipmenttype_id' => $query->param('Type'),
                        'IP' => $query->param('IP'),
                        'community' => $query->param('community'),
                        'description' => $query->param('comment'),
                        'MIB' => $query->param('MIB'),
        );
        $Equip->AddEquipment(\%data);

}
if(defined $query->param('Mod')){
        my %search=(
                        'equipment_id' => $query->param('equipment_id'),
        );
        my %data=(
                        'equipment' => $query->param('equipment'),
                        'equipmenttype_id' => $query->param('Type'),
                        'IP' => $query->param('IP'),
                        'community' => $query->param('community'),
                        'description' => $query->param('comment'),
                        'MIB' => $query->param('MIB'),
        );
        $Equip->ModEquipment(\%search,\%data);

}



my ($EquipmentData,$EquipmentTypes) = $Equip->ReadEquipments();
my (%typelabels,@tyles);
foreach my $line (@{$EquipmentTypes}){
                        $typelabels{${$line}{'equipmenttype_id'}}=${$line}{'equipmenttype_label'};

                        push(@types,${$line}{'equipmenttype_id'});
}
my $MIB = MIBManager->new();
my ($mibs,$mibsloaded)=$MIB->ReadMIBs();
my $mibsload;
push(@{$mibsload},(" ",@{$mibsloaded}));
print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table  border="1" cellspacing="0">
```

```
  <tr>
    <td width="30%" class=\"ConfTableTitle\" align="center">Equipos</td>
    <td nowrap>
            <table width="100%" border="1" cellspacing="0">
EOF
  ;
  print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"8\">Equipos Existentes</td></tr>\n";
              print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >Nombre</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Tipo</td><td class=\"ConfTableItem\" nowrap align=\"center\"
>Descripcíon</td><td class=\"ConfTableItem\" nowrap align=\"center\" >IP</td><td class=\"ConfTableItem\"
nowrap align=\"center\" >Comunidad</td><td class=\"ConfTableItem\" nowrap align=\"center\" >MIB</td><td
class=\"ConfTableItem\" nowrap align=\"center\" ></td><td class=\"ConfTableItem\" nowrap align=\"center\"
></td></tr>\n";
              my $modflag = 0;
  foreach my $equipment(@{$EquipmentData}){
                    if (${$equipment}{'equipment_id'} == $query->param('equipment_id') and defined $query-
>param('ModView')){
                                        print $query->start_form(-action=>'/cgi-bin-admin/EquipmentsAdmin.cgi' );
                                        print "<tr><td nowrap align=\"center\" width=\"100%\">".$query-
>textfield('equipment',$query->param('equipment'))."</td>\n";
                                        print "<td nowrap align=\"center\" width=\"100%\">".$query->popup_menu(-
name=>'Type',-values=>\@types,labels=>\%typelabels,-default=>$query->param('Type'))."<\/td>\n";
                                        print "<td nowrap align=\"center\" width=\"100%\">".$query-
>textfield('comment',$query->param('comment'))."</td>\n";
                                        print "<td nowrap align=\"center\" width=\"100%\">".$query->textfield('IP',$query-
>param('IP'))."</td>\n";
                                        print "<td nowrap align=\"center\" width=\"100%\">".$query-
>textfield('community',$query->param('community'))."</td>\n";
                                        print "<td nowrap align=\"center\" width=\"100%\">".$query->popup_menu(-
name=>'MIB',-values=>$mibsload,-default=>$query->param('MIB'))."</td>\n";
                                        print "<td nowrap align=\"center\">".$query->submit('Mod','Modificar')."</td><td
nowrap align=\"center\" >".$query->submit('Cancel','Cancelar')."</td></tr>\n";
                                        print $query->hidden('equipment_id',$query->param('equipment_id'));
                                        print $query->endform;
                                        $modflag = 1;
                          }else{
                                        print "<tr><td nowrap class=\"ConfTable\"
align=\"center\">".${$equipment}{'equipment'}."</td>\n";
                                        print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$equipment}{'equipmenttype_label'}."</td>\n";
                          print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$equipment}{'description'}."</td>\n";
                          print "<td nowrap class=\"ConfTable\" align=\"center\">".${$equipment}{'IP'}."</td>\n";
                          print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$equipment}{'community'}."</td>\n";
                                        print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$equipment}{'MIB'}."</td>\n";
                                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/EquipmentsAdmin.cgi?ModView=MOD&equipment_id=".${$equipment}{'equipment_id'}."&Type=".${$equi
pment}{'equipmenttype_id'}."&equipment=".${$equipment}{'equipment'}."&comment=".${$equipment}{'description'
}."&IP=".${$equipment}{'IP'}."&community=".${$equipment}{'community'}."&MIB=".${$equipment}{'MIB'}."\"
class=\"menutext\">Modificar</a></td>\n";
                          print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/EquipmentsAdmin.cgi?Del=DEL&equipment_id=".${$equipment}{'equipment_id'}."\"
class=\"menutext\">Borrar</a></td></tr>\n";
                    }
  }
              if ($modflag == 0 ){
                                        $query->param('equipment',"");
                                        $query->param('equipmenttype_id',"");
                                        $query->param('MIB',"");
                                        $query->param('comment',"");
                                        $query->param('community',"");
                                        $query->param('IP',"");
                            print $query->start_form(-action=>'/cgi-bin-admin/EquipmentsAdmin.cgi' );
                            print "<tr><td nowrap align=\"center\" width=\"100%\">".$query-
>textfield('equipment')."</td>\n";
                            print "<td nowrap align=\"center\" width=\"100%\">".$query->popup_menu(-name=>'Type',-
values=>\@types,labels=>\%typelabels)."<\/td>\n";
                            print "<td nowrap align=\"center\" width=\"100%\">".$query->textfield('comment')."</td>\n";
                            print "<td nowrap align=\"center\" width=\"100%\">".$query->textfield('IP')."</td>\n";
                            print "<td nowrap align=\"center\" width=\"100%\">".$query->textfield('community')."</td>\n";
                            print "<td nowrap align=\"center\" width=\"100%\">".$query->popup_menu(-name=>'MIB',-
values=>$mibsload)."</td>\n";
```

```
                        print "<td nowrap align=\"center\" colspan=\"2\" width=\"120\">".$query-
>submit('Add','Agregar Equipo','class="Button"')."</td></tr>\n";
                        print $query->endform;
                }

print <<EOF;
     </table></td>
  </tr>
</table>
   </td>
  </tr>
</table>
EOF
   ;

print $query->end_html;
```

9.2.5.11 MapsAdmin.cgi

```
#!/usr/bin/perl

#
##Maps Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Drawer';
use MapManager;
use BmpManager;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Logs',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Mapas<\/td><\/tr><\/table>\n";

my $Map = MapManager->new();
my $SeeBmp = BmpManager->new();
####Delete Map
if (defined $query->param('Del')){
        my %data=
        (
                'map_id' => $query->param('map_id'),
        );
        my $error = $Map->DelMap(\%data);
        if ($error ne "1"){
                print $error;
        }
}
###Add Map
if (defined $query->param('Add')){
        my %data=
        (
                'mapname' => $query->param('mapname'),
                'mapbmp' => $query->param('mapbmp'),
                'mapicon' => $query->param('mapicon'),
                'comment' => $query->param('comment'),
        );
        $Map->AddMap(\%data);
}
####Open Add View
```

```
if (defined $query->param('AddView')){
        print <<EOF;

                <script language="JavaScript" type="text/JavaScript">
EOF
  ;
                print "AddViewFrame = window.open(\'/cgi-bin-
admin/AddView.cgi?FOCUS=ON&map_id=".$query->param('map_id')."&mapname=".$query-
>param('mapname')."&mapbmp=".$query-
>param('mapbmp')."\',\'AddViewFrame\',\'scrollbars=yes,width=800,height=500\');";
print <<EOF
                AddViewFrame.moveTo(100,0);
                </script>


EOF
  ;
        $query->param('ModView',"");
        $query->param('map_id',"");
        $query->param('mapname',"");
        $query->param('mapbmp',"");
        $query->param('mapicon',"");
        $query->param('comment',"");
}
####Open View Bmp
if (defined $query->param('See')){

print <<EOF;

                <script language="JavaScript" type="text/JavaScript">
EOF
  ;
                print "See = window.open(\"/cgi-bin-admin/Seebmp.cgi?See=".$query-
>param('See')."&ModView=".$query->param('ModView')."&map_id=".$query-
>param('map_id')."&mapname=".$query->param('mapname')."&mapicon=".$query-
>param('mapicon')."&mapbmp=".$query->param('mapbmp')."&comment=".$query-
>param('comment')."\",\'See\',\'scrollbars=yes,width=400,height=600\');";
print <<EOF
                See.moveTo(250,100);
                </script>


EOF
  ;
}

####Modify Mode
if (defined $query->param('Mod')){
        my %search=(
                 'map_id' => $query->param('map_id'),
        );
        my %data=(
                'mapname' => $query->param('mapname'),
                'mapbmp' => $query->param('mapbmp'),
                'mapicon' => $query->param('mapicon'),
                'comment' => $query->param('comment'),

        );
        $Map->ModMap(\%search,\%data);
        $query->param('ModView',"");
        $query->param('map_id',"");
        $query->param('mapname',"");
        $query->param('mapbmp',"");
        $query->param('mapicon',"");
        $query->param('comment',"");
}
###Modify Cancel Option
if(defined $query->param('Cancel')){
$query->param('ModView',"");
$query->param('map_id',"");
$query->param('mapname',"");
$query->param('mapbmp',"");
$query->param('mapicon',"");
$query->param('comment',"");
}
```

```
my $mapinfo = $Map->ReadMaps();
my ($mapicons,$pathicons) =$SeeBmp->Seebmp("Iconos");
my ($mapbmps,$pathbmps) =$SeeBmp->Seebmp("Mapas");

print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table  border="1" cellspacing="0">
  <tr>
    <td width="30%" class=\"ConfTableTitle\" align="center">Mapas</td>
    <td nowrap>
            <table width="100%" border="1" cellspacing="0">

EOF
   ;
            print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"7\">Mapas
Existentes</td></tr>\n";
            print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >Nombre</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Comentario</td><td class=\"ConfTableItem\" nowrap
align=\"center\" >Gráfico</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Icono</td><td
class=\"ConfTableItem\" nowrap align=\"center\" ></td><td class=\"ConfTableItem\" nowrap align=\"center\"
></td><td class=\"ConfTableItem\" nowrap align=\"center\" ></td></tr>\n";
            my $modflag = 0;
            foreach my $map (@{$mapinfo}){
                  if (${$map}{'map_id'} == $query->param('map_id') and $query->param('ModView') eq
"MOD"){
                        print $query->start_form(-action=>'/cgi-bin-admin/MapsAdmin.cgi' );
                        print "<tr><td nowrap align=\"center\" width=\"100%\">".$query-
>textfield('mapname',$query->param('mapname'))."</td>\n";
                        print "<td nowrap align=\"center\" width=\"100%\">".$query-
>textfield('comment',$query->param('comment'))."<td nowrap><table><td >".$query->popup_menu(-
name=>'mapbmp',-values=>$mapbmps,-default=>$query->param('mapbmp'))."<\/td><td width=\"40%\" nowrap
align=\"center\" >".$query->submit('See','Mapas')."</td><\/table></td>\n";
                        print "<td nowrap><table><td>".$query->popup_menu(-name=>'mapicon',-
values=>$mapicons,-default=>$query->param('mapicon'))."<td width=\"40%\" nowrap align=\"center\" >".$query-
>submit('See','Iconos')."</td><\/table></td>\n";
                        print "<td nowrap align=\"center\" >".$query->submit('Mod','Modificar')."</td><td
nowrap align=\"center\" >".$query->submit('Cancel','Cancelar')."</td></tr>\n";
                        print $query->hidden('map_id',$query->param('map_id'));
                        print $query->hidden('ModView',$query->param('ModView'));
                        print $query->endform;
                        $modflag = 1;
                  }else{
                        print "<tr><td class=\"menu\" nowrap align=\"center\"
onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/MapsAdmin.cgi?AddView=YES&map_id=".${$map}{'map_id'}."&mapname=".${$map}{'mapname'}."&map
bmp=".${$map}{'mapbmp'}."\"  class=\"menutext\" TITLE=\"Agregar Mapas y Equipos al mapa
".${$map}{'mapname'}."\">".${$map}{'mapname'}."<\/a></td>\n";
                  print "<td nowrap class=\"ConfTable\" align=\"center\">".${$map}{'comment'}."</td>\n";
                  print "<td nowrap class=\"ConfTable\" align=\"center\">".${$map}{'mapbmp'}."</td>\n";
                        print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$map}{'mapicon'}."</td>\n";
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/MapsAdmin.cgi?ModView=MOD&map_id=".${$map}{'map_id'}."&mapname=".${$map}{'mapname'}."&ma
picon=".${$map}{'mapicon'}."&mapbmp=".${$map}{'mapbmp'}."&comment=".${$map}{'comment'}."\"
class=\"menutext\">Modificar</a></td>\n";
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/MapsAdmin.cgi?Del=DEL&map_id=".${$map}{'map_id'}."\" class=\"menutext\">Borrar</a></td></tr>\n";
                  }
      }
      if ($modflag == 0 ){
                  print $query->start_form(-action=>'/cgi-bin-admin/MapsAdmin.cgi' );
                  print "<tr><td nowrap align=\"center\" width=\"100%\">".$query-
>textfield('mapname',$query->param('mapname'))."</td>\n";
                  print "<td nowrap align=\"center\" width=\"100%\">".$query->textfield('comment',$query-
>param('comment'))."<td nowrap><table><td >".$query->popup_menu(-name=>'mapbmp',-values=>$mapbmps,-
default=>$query->param('mapbmp'))."<\/td><td width=\"40%\" nowrap align=\"center\" >".$query-
>submit('See','Mapas')."</td><\/table></td>\n";
                  print "<td nowrap><table><td >".$query->popup_menu(-name=>'mapicon',-
values=>$mapicons,-default=>$query->param('mapicon'))."<td nowrap align=\"center\" width=\"40%\">".$query-
>submit('See','Iconos')."</td><\/table></td>\n";
                  print "<td nowrap align=\"center\" colspan=\"3\" width=\"120\">".$query-
>submit('Add','Agregar Mapa','class="Button"')."</td></tr>\n";
```

```
                        print $query->endform;
                }

print <<EOF;

     </table></td>
  </tr>
</table>
    </td>
  </tr>
</table>
EOF
   ;

print $query->end_html;
```

9.2.5.12 AddView.cgi

```
#!/usr/bin/perl

#
##AddView Maps
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Drawer';
use MapManager;
use EquipmentManager;
use CoorManager;

#Main
$query = new CGI;
print $query->header;
my $Focus="";
if (defined $query->param('FOCUS')){
                $Focus= "setTimeout(\"window.focus()\",10000)";
}

print $query->start_html(-title=>'Vista de Edición de Mapa',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        -onLoad=>'window.focus()',
                        -onBlur=>$Focus,
                        );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">".$query->param('mapname')."<\/td><td
class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.close()\"
class=\"menutext\">Cerrar<\/a><\/td><\/tr><\/table>\n";
############################
my $Coor = CoorManager->new();

##Open Coor View
if (defined $query->param('CoorView')){
print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
EOF
   ;
                        print "Coor = window.open(\"/cgi-bin-admin/GraficalCoor.cgi?ModView=".$query-
>param('ModView')."&map_id=".$query->param('map_id')."&son_map_id=".$query-
>param('son_map_id')."&equipment_id=".$query->param('equipment_id')."&mapname=".$query-
>param('mapname')."&mapbmp=".$query-
>param('mapbmp')."\",\'Coor\',\'scrollbars=yes,width=700,height=600\');";
print <<EOF
                        Coor.moveTo(100,100);
```

```
                </script>


EOF
  ;

}


##
##Add Options
if(defined $query->param('AddEquip')){
        my %data=(
                'equipment_id' => $query->param('equipment_id'),
                'map_id' => $query->param('map_id'),
                'coor' => $query->param('Coor'),

        );
        $Coor->AddEquipment(\%data);
        $query->param('ModView',"");
        $query->param('equipment_id',"");
        $query->param('coor',"");



}
if(defined $query->param('AddMap')){
        my %data=(
                'son_map_id' => $query->param('son_map_id'),
                'father_map_id' => $query->param('map_id'),
                'coor' => $query->param('Coor'),

        );
        $Coor->AddMap(\%data);
        $query->param('ModView',"");
        $query->param('son_map_id',"");
        $query->param('coor',"");



}
###Delete Option
if(defined $query->param('Del')){
        if ($query->param('equipment_id') ne ""){
                my %data=(
                        'map_id' => $query->param('map_id'),
                       'equipment_id' => $query->param('equipment_id'),
                );
                my $error = $Coor->DelEquipment(\%data);
        }
        if ($query->param('son_map_id') ne ""){
                my %data=(
                        'father_map_id' => $query->param('map_id'),
                       'son_map_id' => $query->param('son_map_id'),
                );
                my $error = $Coor->DelMap(\%data);
        }

}
##Modify Options
if(defined $query->param('ModMap')){
        my %search=(
                'son_map_id' => $query->param('son_map_id'),
                'father_map_id' => $query->param('map_id'),
        );
        my %data=(
                'coor' => $query->param('Coor'),

        );
        $Coor->ModMap(\%search,\%data);
        $query->param('ModView',"");
        $query->param('son_map_id',"");
        $query->param('Coor',"");
```

```
}
if(defined $query->param('ModEquip')){
		my %search=(
				'equipment_id' => $query->param('equipment_id'),
				'map_id' => $query->param('map_id'),
		);
		my %data=(
				'coor' => $query->param('Coor'),

		);
		$Coor->ModEquipment(\%search,\%data);
		$query->param('ModView',"");
		$query->param('equipment_id',"");
		$query->param('Coor',"");

}
###Modify Cancel Option
if(defined $query->param('Cancel')){
$query->param('ModView',"");
$query->param('equipment_id',"");
$query->param('son_map_id',"");
$query->param('coor',"");
}



##Load Information About Equipments and maps into current Map
my $Equip = EquipmentManager->new();
my $Map = MapManager->new();
my $MapEquipmentData = $Equip->ReadMapsEquipments($query->param('map_id'));
my ($Equipments,$EquipmentTypes) = $Equip->ReadEquipments();
my $MapMapData = $Map->ReadMapsMaps($query->param('map_id'));
my $Maps = $Map->ReadMaps();
my $Coors = $Coor->CoorCalculate($query->param('map_id'),$query->param('mapbmp'));
my $AllCoors = $Coor->FormatView($Coors);

##Makes some changes over data to be display correctly
my (@equipments,%equipments,@maps,%maps);
foreach my $equip (@{$Equipments}){
		push(@equipments,${$equip}{'equipment_id'});
		$equipments{${$equip}{'equipment_id'}}= ${$equip}{'equipment'};
}
foreach my $map (@{$Maps}){
		if (${$map}{'map_id'} != $query->param('map_id')){
				push(@maps,${$map}{'map_id'});
				$maps{${$map}{'map_id'}}= ${$map}{'mapname'};
		}
}
############################

print <<EOF;
<table width="100%" border="1" cellspacing="0">
EOF
  ;
  ###Equipments
  print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"6\">Equipos
Dependientes</td></tr>\n";
  print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >Nombre</td><td class=\"ConfTableItem\"
nowrap align=\"center\" >Tipo</td><td class=\"ConfTableItem\" nowrap align=\"center\" >IP</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Coordenadas</td><td class=\"ConfTableItem\" nowrap
align=\"center\" width=\"75\"></td><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"75\"></td></tr>\n";
  my $modflag = 0;
  foreach my $equipment(@{$MapEquipmentData}){
		if (${$equipment}{'equipment_id'} == $query->param('equipment_id') and $query->param('ModView') eq
"MOD"){
				print $query->start_form(-action=>'/cgi-bin-admin/AddView.cgi' );
				push(@{$AllCoors},$query->param('Coor'));
				print "<tr><td nowrap align=\"center\"  colspan=\"3\">".$query->popup_menu(-
name=>'equipment_id',-values=>\@equipments,-labels=>\%equipments,-defaul=>$query-
>param('equipment_id'))."</td><td nowrap align=\"center\" ><table><td>".$query->popup_menu(-name=>'Coor',-
values=>$AllCoors,-default=>$query->param('Coor'))."<td nowrap align=\"center\" >".$query-
>submit('CoorView','Graficamente')."</td><\/table></td><td nowrap align=\"center\" >".$query-
>submit('ModEquip','Modificar')."</td><td nowrap align=\"center\" >".$query-
>submit('Cancel','Cancelar')."</td></tr>\n";
				print $query->hidden('map_id',$query->param('map_id'));
				print $query->hidden('mapname',$query->param('mapname'));
```

```
                        print $query->hidden('mapbmp',$query->param('mapbmp'));
                        print $query->hidden('ModView',$query->param('ModView'));
                        print $query->endform;
                        $modflag = 1;
                }else{
                        print "<tr><td nowrap class=\"ConfTable\"
align=\"center\">".${$equipment}{'equipment'}."</td>\n";
                        print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$equipment}{'equipmenttype_label'}."</td>\n";
                print "<td nowrap class=\"ConfTable\" align=\"center\">".${$equipment}{'IP'}."</td>\n";
                        print "<td nowrap class=\"ConfTable\" align=\"center\">".${$equipment}{'coor'}."</td>\n";
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AddView.cgi?ModView=MOD&map_id=".$query->param('map_id')."&mapname=".$query-
>param('mapname')."&equipment_id=".${$equipment}{'equipment_id'}."&mapbmp=".$query-
>param('mapbmp')."&Coor=".${$equipment}{'coor'}."\" class=\"menutext\">Modificar</a></td>\n";
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AddView.cgi?Del=DEL&map_id=".$query->param('map_id')."&mapname=".$query-
>param('mapname')."&equipment_id=".${$equipment}{'equipment_id'}."&mapbmp=".$query-
>param('mapbmp')."\" class=\"menutext\">Borrar</a></td></tr>\n";
                }
  }
  if ($modflag == 0 ){
                        print $query->start_form(-action=>'/cgi-bin-admin/AddView.cgi' );
                        print "<tr><td nowrap align=\"center\"  colspan=\"3\">".$query->popup_menu(-
name=>'equipment_id',-values=>\@equipments,-labels=>\%equipments,-defaul=>$query-
>param('equipment_id'))."</td><td nowrap align=\"center\" ><table><td>".$query->popup_menu(-name=>'Coor',-
values=>$AllCoors,-default=>$query->param('Coor'))."<td nowrap align=\"center\" >".$query-
>submit('CoorView','Graficamente')."</td><\/table></td><td nowrap align=\"center\" colspan=\"2\">".$query-
>submit('AddEquip','Agregar Al Mapa','class="Button"')."</td></tr>\n";
                        print $query->hidden('map_id',$query->param('map_id'));
                        print $query->hidden('mapname',$query->param('mapname'));
                        print $query->hidden('mapbmp',$query->param('mapbmp'));
                        print $query->endform;
  }
  ###Son Maps
  print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"7\">Mapas Dependientes</td></tr>\n";
  print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >Nombre</td><td class=\"ConfTableItem\"
nowrap align=\"center\" >Descripcíon</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Icono</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Coordenadas</td><td class=\"ConfTableItem\" nowrap
align=\"center\" ></td><td class=\"ConfTableItem\" nowrap align=\"center\" ></td></tr>\n";
  my $modflag = 0;
  foreach my $map(@{$MapMapData}){
                if (${$map}{'son_map_id'} == $query->param('son_map_id') and $query->param('ModView') eq
"MOD"){
                        print $query->start_form(-action=>'/cgi-bin-admin/AddView.cgi' );
                        push(@{$AllCoors},$query->param('Coor'));
                        print "<tr><td nowrap align=\"center\"  colspan=\"3\">".$query->popup_menu(-
name=>'son_map_id',-values=>\@maps,-labels=>\%maps,-defaul=>$query->param('son_map_id'))."</td><td
nowrap align=\"center\" ><table><td>".$query->popup_menu(-name=>'Coor',-values=>$AllCoors,-
default=>$query->param('Coor'))."<td nowrap align=\"center\" >".$query-
>submit('CoorView','Graficamente')."</td><\/table></td><td nowrap align=\"center\" >".$query-
>submit('ModMap','Modificar')."</td><td nowrap align=\"center\" >".$query-
>submit('Cancel','Cancelar')."</td></tr>\n";
                        print $query->hidden('map_id',$query->param('map_id'));
                        print $query->hidden('mapname',$query->param('mapname'));
                        print $query->hidden('mapbmp',$query->param('mapbmp'));
                        print $query->hidden('ModView',$query->param('ModView'));
                        print $query->endform;
                        $modflag = 1;
                }else{
                        print "<tr><td nowrap class=\"ConfTable\" align=\"center\">".${$map}{'mapname'}."</td>\n";
                        print "<td nowrap class=\"ConfTable\" align=\"center\">".${$map}{'comment'}."</td>\n";
                print "<td nowrap class=\"ConfTable\" align=\"center\">".${$map}{'mapicon'}."</td>\n";
                print "<td nowrap class=\"ConfTable\" align=\"center\">".${$map}{'coor'}."</td>\n";
                print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AddView.cgi?ModView=MOD&son_map_id=".${$map}{'map_id'}."&map_id=".$query-
>param('map_id')."&mapname=".$query->param('mapname')."&mapbmp=".$query-
>param('mapbmp')."&Coor=".${$map}{'coor'}."\" class=\"menutext\">Modificar</a></td>\n";
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AddView.cgi?Del=DEL&son_map_id=".${$map}{'map_id'}."&map_id=".$query-
>param('map_id')."&mapname=".$query->param('mapname')."&mapbmp=".$query->param('mapbmp')."\"
class=\"menutext\">Borrar</a></td></tr>\n";
```

```
            }
  }
  if ($modflag == 0 ){
            print $query->start_form(-action=>'/cgi-bin-admin/AddView.cgi' );
            print "<tr><td nowrap align=\"center\" colspan=\"3\">".$query->popup_menu(-name=>'son_map_id',-
values=>\@maps,-labels=>\%maps,-defaul=>$query->param('son_map_id'))."</td><td nowrap align=\"center\"
><table><td>".$query->popup_menu(-name=>'Coor',-values=>$AllCoors,-default=>$query->param('Coor'))."<td
nowrap align=\"center\" >".$query->submit('CoorView','Graficamente')."</td><\/table></td><td nowrap
align=\"center\" colspan=\"2\">".$query->submit('AddMap','Agregar Al Mapa','class="Button"')."</td></tr>\n";
            print $query->hidden('map_id',$query->param('map_id'));
            print $query->hidden('mapname',$query->param('mapname'));
            print $query->hidden('mapbmp',$query->param('mapbmp'));
            print $query->endform;
  }
print <<EOF;
</table>
EOF
  ;

print $query->end_html;
```

9.2.5.13 GraficalCoor.cgi

```
#!/usr/bin/perl

#
##Grafical Coor
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/Drawer';
use MapManager;
use EquipmentManager;
use CoorManager;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Selección Gráfica de Coordenadas',
                  -style=>{'src'=>'/Conf/WBNMS.css'},
                  -onLoad=>'window.focus()',
                  -onBlur=>'window.focus()',
                  );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">".$query->param('mapname')."<\/td><td
class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.close()\"
class=\"menutext\">Cerrar<\/a><\/td><\/tr><\/table>\n";
##Coor Calculation
my $Coor = CoorManager->new();
my $Coors = $Coor->CoorCalculate($query->param('map_id'));

print <<EOF;
<table width="100%" border="1" cellspacing="0">
EOF
  ;
my $path =  "/DrawerDir/".$query->param('map_id')."\.gif";
print "<tr><td align =\"center\"><img SRC=\"".$path."\"  align =\"center\" border=\"0\"
usemap=\"#Map\"><\/td><\/tr>\n";
print "<map name=\"Map\">\n";
foreach my $area (@{$Coors}){
          print "<area shape=\"rect\" alt=\"". join(',',@{$area})."\" coords=\"".join(',',@{$area})."\" href=\"/cgi-bin-
admin/AddView.cgi?ModView=".$query->param('ModView')."&map_id=".$query-
```

```
>param('map_id')."&son_map_id=".$query->param('son_map_id')."&equipment_id=".$query-
>param('equipment_id')."&mapname=".$query->param('mapname')."&Coor=".join(':',(${$area}[0],${$area}[1]))."\"
target=\"AddViewFrame\" )  >\n";
}
print "<\/map>\n";
print <<EOF;
</table>
EOF
   ;

print $query->end_html;
```

9.2.5.14 AssignationsActionsAdmin.cgi

```
#!/usr/bin/perl

#
##Assignation Reports Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/ActionModules';
use ActionGuiManager;


#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Reports',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Asignación de Alertas<\/td><\/tr><\/table>\n";

my $ActionManager = ActionGuiManager->new();

#####################
if(defined $query->param('Del')){
        $ActionManager->DelAsignationAction($query->param('asignation_id'));
}
if(defined $query->param('Add')){
        my %data=(
                'user_id'=>$query->param('user_id'),
                'action_id'=>$query->param('action_id'),
                'equipment_id'=>$query->param('equipment_id'),
                'map_id'=>$query->param('map_id'),
                'starttime'=>$query->param('inihour').":".$query->param('inimin'),
                'endtime'=>$query->param('endhour').":".$query->param('endmin'),
        );
         $ActionManager->AddAsignationAction(\%data);

}
if(defined $query->param('Mod')){
        my %search=(
        'asignation_id'=>$query->param('asignation_id'),
        );
        my %moddata=(
                'user_id'=>$query->param('user_id'),
                'action_id'=>$query->param('action_id'),
                'equipment_id'=>$query->param('equipment_id'),
                'map_id'=>$query->param('map_id'),
                'starttime'=>$query->param('inihour').":".$query->param('inimin'),
                'endtime'=>$query->param('endhour').":".$query->param('endmin'),
```

```
		);
		$ActionManager->ModAsignationAction(\%search,\%moddata);
		$query->delete_all();

}



####################
my %phoneop=('1'=>"Unifon",'2'=>"Personal",'3'=>"Movicom",'4'=>"CTI",);
my @hour=(0..23);
my @min=("00","01","02","03","04","05","06","07","08","09",10..59);
####
my ($ActionAssignationsData,$users,$actions,$equipments,$maps) = $ActionManager-
>ReadActionAssignations();
my @usersvalues = keys(%{$users});
my @actionsvalues = keys(%{$actions});
my @equipmentsvalues = keys(%{$equipments});
my @mapsvalues = keys(%{$maps});

print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table width="100%" border="1" cellspacing="0">
  <tr>
    <td width="20%" class=\"ConfTableTitle\" align="center">Asignaciones</td>
    <td nowrap  >
		<table width="100%" border="1" cellspacing="0">
EOF
   ;
		print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"12\" >Asignaciones
Realizadas</td></tr>\n";
		print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"80\">Usuario</td><td
class=\"ConfTableItem\" nowrap align=\"center\" width=\"80\">Reporte</td><td class=\"ConfTableItem\" nowrap
align=\"center\" width=\"80\">Equipo</td><td class=\"ConfTableItem\" nowrap align=\"center\"
width=\"80\">Mapa</td><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"3\">Inicio del
Turno</td><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"3\">Final del Turno</td><td
class=\"ConfTableItem\" nowrap align=\"center\" width=\"20%\"></td><td class=\"ConfTableItem\" nowrap
align=\"center\" width=\"20%\"></td></tr>\n";
		my $modflag = 0;
		foreach my $Assig (@{$ActionAssignationsData}){
			if (${$Assig}{'asignation_id'} == $query->param('asignation_id') and defined $query-
>param('ModView')){
				print $query->start_form(-action=>'/cgi-bin-admin/AssignationsActionsAdmin.cgi' );
				print "<tr><td nowrap align=\"center\" >".$query->popup_menu(-name=>'user_id',-
values=>\@usersvalues,-labels=>$users,-default=>"${$Assig}{'user_id'}",'class="popup"').'</td>\n";
				print "<td nowrap align=\"center\" >".$query->popup_menu(-name=>'action_id',-
values=>\@actionsvalues,-labels=>$actions,-default=>"${$Assig}{'action_id'}",'class="popup"')."<\/td>\n";
				print "<td nowrap align=\"center\" >".$query->popup_menu(-
name=>'equipment_id',-values=>\@equipmentsvalues,-labels=>$equipments,-
default=>"${$Assig}{'equipment_id'}",'class="popup"')."</td>\n";
				print "<td nowrap align=\"center\" >".$query->popup_menu(-name=>'map_id',-
values=>\@mapsvalues,-labels=>$maps,-default=>"${$Assig}{'map_id'}",'class="popup"')."</td>";
				my ($inihour,$inimin)=split(/:/,${$Assig}{'starttime'});
				print "<td nowrap align=\"center\">".$query->popup_menu(-name=>'inihour',-
values=>\@hour,-default=>"$inihour",'class="popup"')."</td>\n";
				print "<td nowrap align=\"center\" width=\"1%\"><b>:</b></td>\n";
				print "<td nowrap align=\"center\" >".$query->popup_menu(-name=>'inimin',-
values=>\@min,-default=>"$inimin",'class="popup"')."</td>\n";
				my ($endhour,$endmin)=split(/:/,${$Assig}{'endtime'});
				print "<td nowrap align=\"center\">".$query->popup_menu(-name=>'endhour',-
values=>\@hour,-default=>"$endhour",'class="popup"')."</td>\n";
				print "<td nowrap align=\"center\" width=\"1%\"><b>:</b></td>\n";
				print "<td nowrap align=\"center\" >".$query->popup_menu(-name=>'endmin',-
values=>\@min,-default=>"$endmin",'class="popup"')."</td>\n";
				print "<td nowrap align=\"center\" width=\"70\">".$query-
>submit('Mod','Modificar','class="Button"')."</td><td nowrap align=\"center\" width=\"70\">".$query-
>submit('Cancel','Cancelar','class="Button"')."</td></tr>\n";
				print $query->hidden('asignation_id',$query->param('asignation_id'));
				print $query->endform;
				$modflag = 1;
			}else{
				print "<tr>";
				print "<td class=\"ConfTable\" nowrap align=\"center\"
>".${$users}{${$Assig}{'user_id'}}."</td>";
```

```
                                        print "<td class=\"ConfTable\" nowrap align=\"center\"
>".${$actions}{${$Assig}{'action_id'}}."</td>";
                                        print "<td class=\"ConfTable\" nowrap align=\"center\"
>".${$equipments}{${$Assig}{'equipment_id'}}."</td>";
                                        print "<td class=\"ConfTable\" nowrap align=\"center\"
>".${$maps}{${$Assig}{'map_id'}}."</td>";
                                        print "<td class=\"ConfTable\" nowrap align=\"center\"
colspan=\"3\">".${$Assig}{'starttime'}."</td>";
                                        print "<td class=\"ConfTable\" nowrap align=\"center\"
colspan=\"3\">".${$Assig}{'endtime'}."</td>";
                                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AssignationsActionsAdmin.cgi?ModView=MOD&asignation_id=".${$Assig}{'asignation_id'}."\"
class=\"menutext\">Modificar</a></td>\n";
                                print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AssignationsActionsAdmin.cgi?Del=DEL&asignation_id=".${$Assig}{'asignation_id'}."\"
class=\"menutext\">Borrar</a></td></tr>\n";
                        }
                }
                        if ($modflag == 0 ){
                                $query->delete_all();
                                print $query->start_form(-action=>'/cgi-bin-admin/AssignationsActionsAdmin.cgi' );
                                print "<tr><td nowrap align=\"center\" >".$query->popup_menu(-name=>'user_id',-
values=>\@usersvalues,-labels=>$users,'class="popup"')."</td>\n";
                                print "<td nowrap align=\"center\" >".$query->popup_menu(-name=>'action_id',-
values=>\@actionsvalues,-labels=>$actions,'class="popup"')."<\/td>\n";
                                print "<td nowrap align=\"center\" >".$query->popup_menu(-
name=>'equipment_id',-values=>\@equipmentsvalues,-labels=>$equipments,'class="popup"')."</td>\n";
                                print "<td nowrap align=\"center\" >".$query->popup_menu(-name=>'map_id',-
values=>\@mapsvalues,-labels=>$maps,'class="popup"')."</td>";
                                print "<td nowrap align=\"center\">".$query->popup_menu(-name=>'inihour',-
values=>\@hour,'class="popup"')."</td>\n";
                                print "<td nowrap align=\"center\" width=\"1%\"><b>:</b></td>\n";
                                print "<td nowrap align=\"center\" >".$query->popup_menu(-name=>'inimin',-
values=>\@min,'class="popup"')."</td>\n";
                                print "<td nowrap align=\"center\">".$query->popup_menu(-name=>'endhour',-
values=>\@hour,'class="popup"')."</td>\n";
                                print "<td nowrap align=\"center\" width=\"1%\"><b>:</b></td>\n";
                                print "<td nowrap align=\"center\" >".$query->popup_menu(-name=>'endmin',-
values=>\@min,'class="popup"')."</td>\n";
                                print "<td nowrap align=\"center\" colspan=\"2\" width=\"140\">".$query->submit(-
name=>'Add',-value=>'Agregar Asociación','class="Button"')."</td></tr>\n";;
                                print $query->endform;
                        }


print <<EOF;
     </table></td>
  </tr>
</table>
   </td>
  </tr>
</table>
EOF
  ;
print $query->end_html;
```

9.2.5.15 UsersActionsAdmin.cgi

```
#!/usr/bin/perl

#
##Users Reports Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;
```

```
use lib '/var/sites/Wbnms/Wbnms_Core/ActionModules';
use ActionGuiManager;


#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Reports',
                        -style=>{'src'=>'/Conf/WBNMS.css'},
                        );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Usuarios de Alertas<\/td><\/tr><\/table>\n";

my $ActionManager = ActionGuiManager->new();

#####################
if(defined $query->param('Del')){
        $ActionManager->DelActionUser($query->param('user_id'));
}
if(defined $query->param('Add')){
        my %data=(
                'name'=>$query->param('name'),
                'range'=>$query->param('type'),
                'email'=>$query->param('email'),
                'IP'=>$query->param('IP'),
                'community'=>$query->param('community'),
        );
        $ActionManager->AddActionUser(\%data);

}
if(defined $query->param('Mod')){
        my %search=(
        'user_id'=>$query->param('user_id'),
        );
        my %moddata=(
                'name'=>$query->param('name'),
                'range'=>$query->param('type'),
                'email'=>$query->param('email'),
                'IP'=>$query->param('IP'),
                'community'=>$query->param('community'),
        );
        $ActionManager->ModActionUser(\%search,\%moddata);
        $query->delete_all();

}



#####################

my $UsersData = $ActionManager->ReadActionUsers();

print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table width="100%" border="1" cellspacing="0">
  <tr>
    <td width="20%" class=\"ConfTableTitle\" align="center">Usuarios de Alertas</td>
    <td nowrap  >
        <table width="100%" border="1" cellspacing="0">
EOF
   ;
        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"11\" >Usuarios
Existentes</td></tr>\n";
        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"60\">Nombre</td><td
class=\"ConfTableItem\" nowrap align=\"center\" width=\"60\">Tipo</td><td class=\"ConfTableItem\" nowrap
align=\"center\" width=\"120\">e-mail</td><td class=\"ConfTableItem\" nowrap align=\"center\"
width=\"80\">IP</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Comunidad</td><td
class=\"ConfTableItem\" nowrap align=\"center\" width=\"20%\"></td><td class=\"ConfTableItem\" nowrap
align=\"center\" width=\"20%\"></td></tr>\n";
        my $modflag = 0;
        foreach my $User (@{$UsersData}){
                if (${$User}{'user_id'} == $query->param('user_id') and defined $query->param('ModView')){
```

```
                                print $query->start_form(-action=>'/cgi-bin-admin/UsersActionsAdmin.cgi' );
                                print "<tr><td nowrap align=\"center\" >".$query->textfield(-name=>'name',-
default=>${$User}{'name'},'class="popup"').'</td>\n';
                                print "<td nowrap align=\"center\" >".$query->textfield(-name=>'type',-
default=>${$User}{'range'},'class="popup"').'<\/td>\n';
                                print "<td nowrap align=\"center\" >".$query->textfield(-name=>'email',-
default=>${$User}{'email'},'class="popup"').'</td>\n';
                                print "<td nowrap align=\"center\" >".$query->textfield(-name=>'IP',-
default=>${$User}{'IP'},'class="popup"').'</td>\n';
                                print "<td nowrap align=\"center\" >".$query->textfield(-
name=>'community',default=>${$User}{'community'},'class="popup"').'</td>\n';
                                print "<td nowrap align=\"center\" width=\"70\">".$query-
>submit('Mod','Modificar','class="Button"').'</td><td nowrap align=\"center\" width=\"70\">".$query-
>submit('Cancel','Cancelar','class="Button"').'</td></tr>\n';
                                print $query->hidden('user_id',$query->param('user_id'));
                                print $query->endform;
                                $modflag = 1;
                        }else{
                                print "<tr>";
                                print "<td class=\"ConfTable\" nowrap align=\"center\" >".${$User}{'name'}."</td>";
                                print "<td class=\"ConfTable\" nowrap align=\"center\" >".${$User}{'range'}."</td>";
                                print "<td class=\"ConfTable\" nowrap align=\"center\" >".${$User}{'email'}."</td>";
                                print "<td class=\"ConfTable\" nowrap align=\"center\" >".${$User}{'IP'}."</td>";
                                print "<td class=\"ConfTable\" nowrap align=\"center\"
>".${$User}{'community'}."</td>";
                                print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/UsersActionsAdmin.cgi?ModView=MOD&user_id=".${$User}{'user_id'}."\"
class=\"menutext\">Modificar</a></td>\n";
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/UsersActionsAdmin.cgi?Del=DEL&user_id=".${$User}{'user_id'}."\"
class=\"menutext\">Borrar</a></td></tr>\n";
                }
        }
                if ($modflag == 0 ){
                        $query->delete_all();
                        print $query->start_form(-action=>'/cgi-bin-admin/UsersActionsAdmin.cgi' );
                        print "<tr><td nowrap align=\"center\" >".$query->textfield(-
name=>'name','class="popup"').'</td>\n';
                        print "<td nowrap align=\"center\" >".$query->textfield(-
name=>'type','class="popup"').'<\/td>\n';
                        print "<td nowrap align=\"center\" >".$query->textfield(-
name=>'email','class="popup"').'</td>\n';
                        print "<td nowrap align=\"center\" >".$query->textfield(-
name=>'IP','class="popup"').'</td>\n';
                        print "<td nowrap align=\"center\" >".$query->textfield(-
name=>'community','class="popup"').'</td>\n';
                        print "<td nowrap align=\"center\" colspan=\"2\" width=\"140\">".$query->submit(-
name=>'Add',-value=>'Agregar Usuario','class="Button"').'</td></tr>\n';;
                        print $query->endform;
                 }


print <<EOF;
     </table></td>
  </tr>
</table>
   </td>
  </tr>
</table>
EOF
  ;

print $query->end_html;
```

9.2.5.16 AlarmTriggersAdmin.cgi

```
#!/usr/bin/perl

#
##triggers Administration
#
```

```
#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance';
use AlarmTriggersManager;


#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Logs',
                         -style=>{'src'=>'/Conf/WBNMS.css'},
                         );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Alarmas<\/td><\/tr><\/table>\n";

my $AlarmsTriggers = AlarmTriggersManager->new();
##AddObj Option
if(defined $query->param('AddObj')){
print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
EOF
  ;
  my @data = $query->param('actions_id');
                        print "MibBrowser = window.open(\"/cgi-
bin/MIBBrowserFrameset.cgi?Trigger=YES&FOCUS=ON&trigger_id=".$query-
>param('trigger_id')."&TYPE=".$query->param('TYPE')."&ModView=".$query-
>param('ModView')."&equipment_id=".$query->param('equipment_id')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&triggernot_id=".$query->param('triggernot_id')."&severity=".$query-
>param('severity')."&actions_id=".join(':',@data)."\",\'MIBBrowser\',\'scrollbars=no,width=800,height=600\');";
print <<EOF
                        MibBrowser.moveTo(250,100);
                        </script>


EOF
  ;

}
##AllAlarms Option
if(defined $query->param('AllAlarms')){
print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
EOF
  ;
  my @data = $query->param('actions_id');
                        print "AllAlarms = window.open(\"/cgi-bin-
admin/AllAlarmsdisplay.cgi?Trigger=YES&FOCUS=ON&trigger_id=".$query-
>param('trigger_id')."&TYPE=".$query->param('TYPE')."&ModView=".$query-
>param('ModView')."&equipment_id=".$query->param('equipment_id')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&triggernot_id=".$query->param('triggernot_id')."&severity=".$query-
>param('severity')."&actions_id=".join(':',@data)."\",\'AllAlarms\',\'scrollbars=yes,width=800,height=600\');";
print <<EOF
                        AllAlarms.moveTo(250,100);
                        </script>


EOF
  ;

}

##Add Option
if(defined $query->param('Add')){
            my %adddata=(
            'triggernot_id'=>$query->param('triggernot_id'),
            'trigger_object'=>$query->param('trigger_object'),
```

```
		'equipment_id'=>$query->param('equipment_id'),
		'severity'=>$query->param('severity'),
		'equipmenttype_id'=>$query->param('equipmenttype_id'),
		);
		$AlarmsTriggers->AddAlarmTriggers(\%adddata);
		my $trigger_id = $AlarmsTriggers->ReadAlarmTriggerfromID(\%adddata);
		foreach my $action($query->param('actions_id')){
			my %adddata=('action_id'=>$action,'trigger_id'=>$trigger_id);
			$AlarmsTriggers->AddActionTrigger(\%adddata);
		}
}

##Del Option
if(defined $query->param('Del')){
		my %deldata=(
		'trigger_id'=>$query->param('trigger_id'),
		);
		$AlarmsTriggers->DelAlarmTriggers(\%deldata);
		$AlarmsTriggers->DelActionTrigger(\%deldata);
}

##Mod Option
if(defined $query->param('Mod')){
		my %search=(
		'trigger_id'=>$query->param('trigger_id'),
		);
		my %moddata=(
		'triggernot_id'=>$query->param('triggernot_id'),
		'trigger_object'=>$query->param('trigger_object'),
		'equipment_id'=>$query->param('equipment_id'),
		'severity'=>$query->param('severity'),
		'equipmenttype_id'=>$query->param('equipmenttype_id'),
		);
		$AlarmsTriggers->ModAlarmTriggers(\%search,\%moddata);
		my @actions_id=$query->param('actions_id');
		$AlarmsTriggers->ModActionTrigger(\%search,\@actions_id);
		$query->param('ModView',"");
}
##Cancel Option
if (defined $query->param('Cancel')){
		$query->param('ModView',"");
}

##AddView Option
if(defined $query->param('AddView')){
print <<EOF;

			<script language="JavaScript" type="text/JavaScript">
EOF
  ;
  my @data = $query->param('actions_id');
			print "ObjectsView = window.open(\"/cgi-bin-
admin/FilterObjects.cgi?Trigger=YES&FOCUS=ON&trigger_id=".$query->param('trigger_id')."&TYPE=".$query-
>param('TYPE')."&equipment_id=".$query->param('equipment_id')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&triggernot_id=".$query->param('triggernot_id')."&severity=".$query-
>param('severity')."&trigger_object=".$query-
>param('trigger_object')."&actions_id=".join(':',@data)."\",\'ObjectsView\',\'scrollbars=yes,width=800,height=600\');
";
print <<EOF
			ObjectsView.moveTo(250,100);
			</script>


EOF
  ;
}

my ($EquipmentData,$Equipments,$EquipmentLabels)= $AlarmsTriggers->ReadEquipments();
my ($EquipmentTypesData,$Equipmenttypes,$EquipmentTypeLabels)= $AlarmsTriggers-
>ReadEquipmentTypes();

if ( !defined $query->param('TYPE')){
		print $query->start_form(-action=>'/cgi-bin-admin/AlarmTriggersAdmin.cgi' );
print "<table  border=\"1\" cellspacing=\"1\" width=\"100%\"><tr><td class=\"menustatic\" nowrap align=\"center\"
colspan=\"2\" width=\"100%\">Tipo de Alarmas</td></tr>\n";
```

```
print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"30%\">General</td><td nowrap
class=\"ConfTable\" align=\"center\" width=\"50%\">".$query-
>submit('TYPE','General','class="Button"')."</td></tr>\n";
print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"30%\">Por Tipo de Equipo</td><td><table
width=\"100%\"><tr><td nowrap align=\"center\" width=\"50%\">".$query->popup_menu(-
name=>'equipmenttype_id',-values=>$Equipmenttypes,-
labels=>$EquipmentTypeLabels,'class="popup"')."</td><td nowrap class=\"ConfTable\" align=\"center\"
width=\"50%\">".$query->submit('TYPE','Tipo','class="Button"')."</td></tr></table></td></tr>\n";
print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"30%\">Por Equipo</td><td><table
width=\"100%\"><tr><td nowrap align=\"center\" width=\"50%\">".$query->popup_menu(-name=>'equipment_id',-
values=>$Equipments,-labels=>$EquipmentLabels,'class="popup"')."</td><td nowrap class=\"ConfTable\"
align=\"center\" width=\"50%\">".$query->submit('TYPE','Equipo','class="Button"')."</td></tr></table></td></tr>\n";
print "</table>\n";
print $query->endform;
}else{
my %data;
if ($query->param('TYPE') eq "Tipo"){
        $data{'equipmenttype_id'}= $query->param('equipmenttype_id');
        $data{'equipment_id'}= "0";
}elsif($query->param('TYPE') eq "Equipo"){
        $data{'equipment_id'}= $query->param('equipment_id');
        $data{'equipmenttype_id'}= "0";
}elsif($query->param('TYPE') eq "General"){
        $data{'equipmenttype_id'}= "0";
        $data{'equipment_id'}= "0";
}
my ($Triggers,$triggersID) = $AlarmsTriggers->ReadAlarmTriggers(\%data);
my ($ActionTriggers,$actionlabels)=$AlarmsTriggers->ReadActionTriggers($triggersID);
my @actionsvalues = keys(%{$actionlabels});
my ($SeveritiesData,$Severities,$SeveritiesLabels)= $AlarmsTriggers->ReadSeverity();
print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table  width="100%" border="1" cellspacing="0">
  <tr>
    <td width="10%" class=\"ConfTableTitle\" align="center">Filtros</td>
    <td nowrap>
        <table width="100%" border="1" cellspacing="0" >

EOF
   ;
        if ($query->param('TYPE') eq "Equipo"){
            print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"10\">Alarmas
Existentes</td></tr>\n";
                print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >ID</td><td
class=\"ConfTableItem\" nowrap align=\"center\" style=\'text-decoration: overline;\'>ID</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Equipo</td><td class=\"ConfTableItem\" nowrap
align=\"center\" >Tipo</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Severidad</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Alerta</td><td class=\"ConfTableItem\" nowrap align=\"center\"
>Objeto</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Variables</td><td class=\"ConfTableItem\"
nowrap align=\"center\" ></td><td class=\"ConfTableItem\" nowrap align=\"center\" ></td></tr>\n";
        }elsif ($query->param('TYPE') eq "Tipo"){
            print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"9\">Alarmas
Existentes</td></tr>\n";
                print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >ID</td><td
class=\"ConfTableItem\" nowrap align=\"center\" style=\'text-decoration: overline;\'>ID</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Tipo</td><td class=\"ConfTableItem\" nowrap align=\"center\"
>Severidad</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Alerta</td><td class=\"ConfTableItem\"
nowrap align=\"center\" >Objeto</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Variables</td><td
class=\"ConfTableItem\" nowrap align=\"center\" ></td><td class=\"ConfTableItem\" nowrap align=\"center\"
></td></tr>\n";
        }else{
                print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"8\">Alarmas
Existentes</td></tr>\n";
                print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >ID</td><td
class=\"ConfTableItem\" nowrap align=\"center\" style=\'text-decoration: overline;\'>ID</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Severidad</td><td class=\"ConfTableItem\" nowrap
align=\"center\" >Alerta</td><td class=\"ConfTableItem\" nowrap align=\"center\" >Objeto</td><td
class=\"ConfTableItem\" nowrap align=\"center\" >Variables</td><td class=\"ConfTableItem\" nowrap
align=\"center\" ></td><td class=\"ConfTableItem\" nowrap align=\"center\" ></td></tr>\n";
        }
         my $modflag = 0;
         foreach my $trigger (@{$Triggers}){
                if (${$trigger}{'trigger_id'} == $query->param('trigger_id') and $query->param('ModView') eq
"MOD"){
                        print $query->start_form(-action=>'/cgi-bin-admin/AlarmTriggersAdmin.cgi' );
```

```
                                print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'trigger_id'}."</td>\n";
                                print "<td nowrap class=\"ConfTable\" align=\"center\" width=\"8%\">".$query-
>popup_menu(-name=>'triggernot_id',-values=>$triggersID,-default=>${$trigger}{'triggernot_id'})."</td>\n";
                                if ($query->param('TYPE') eq "Equipo"){
                                print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'equipment'}."</td>\n";
                                print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'equipmenttype_label'}."</td>\n";
                                $query->param('equipmenttype_id',0);
                                }elsif ($query->param('TYPE') eq "Tipo"){
                                print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'equipmenttype_label'}."</td>\n";
                                $query->param('equipment_id',0);
                                }else{
                                        $query->param('equipmenttype_id',0);
                                        $query->param('equipment_id',0);
                                }
                                print "<td nowrap class=\"ConfTable\" align=\"center\" width=\"15%\">".$query-
>popup_menu(-name=>'severity',-values=>$Severities,-labels=>$SeveritiesLabels,-
default=>${$trigger}{'severity'})."</td>\n";
                                print "<td nowrap class=\"ConfTable\" align=\"left\">".$query->checkbox_group(-
name=>'actions_id',-values=>\@actionsvalues,-labels=>$actionlabels,-
default=>${$ActionTriggers}{${$trigger}{'trigger_id'}},-linebreak=>'true')."</td>\n";
                                print "<td nowrap class=\"ConfTable\" align=\"center\" colspan=\"2\"
width=\"40%\"><table width=\"100%\"><tr><td nowrap align=\"center\" width=\"100%\" colspan=\"2\">".$query-
>textfield(-name=>'trigger_object',-values=>"${$trigger}{'object'}",'class="popup"')."</td></tr><tr><td nowrap
class=\"ConfTable\" align=\"center\" width=\"40%\">".$query->submit('AddObj','MIB','class="Button"')."</td><td
nowrap class=\"ConfTable\" align=\"center\" width=\"40%\">".$query-
>submit('AllAlarms','Alarmas','class="Button"')."</td></tr></table></td>\n";
                                print "<td nowrap align=\"center\" width=\"15%\">".$query-
>submit('Mod','Modificar','class="Button"')."</td><td nowrap align=\"center\" width=\"15%\">".$query-
>submit('Cancel','Cancelar','class="Button"')."</td></tr>\n";
                                print $query->hidden('trigger_id',$query->param('trigger_id'));
                                print $query->hidden('TYPE',$query->param('TYPE'));
                                print $query->hidden('equipment_id',$query->param('equipment_id'));
                                print $query->hidden('equipmenttype_id',$query->param('equipmenttype_id'));
                                print $query->hidden('ModView',$query->param('ModView'));
                                print $query->endform;
                                $modflag = 1;
                        }else{
                                print "<tr><td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'trigger_id'}."</td>\n";
                                print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'triggernot_id'}."</td>\n";
                                if ($query->param('TYPE') eq "Equipo"){
                                print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'equipment'}."</td>\n";
                                print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'equipmenttype_label'}."</td>\n";
                        }elsif ($query->param('TYPE') eq "Tipo"){
                                print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$trigger}{'equipmenttype_label'}."</td>\n";
                        }
                        my $size = scalar(@{${$trigger}{'trigger_variables'}});
                                if ($size > 5){$size = 5;}elsif($size == 0){$size = 1;}
                        print "<td nowrap class=\"ConfTable\" align=\"center\">".${$trigger}{'state_label'}."</td>\n";
                        print "<td nowrap class=\"ConfTable\" align=\"center\">".$query->scrolling_list(-
name=>'actions',-values=>\@{${$ActionTriggers}{${$trigger}{'trigger_id'}}},-labels=>$actionlabels,-size=>$size,-
multiple=>'true','class="Popup"')."</td>\n";
                                print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AlarmTriggersAdmin.cgi?AddView=YES&TYPE=".$query-
>param('TYPE')."&trigger_id=".${$trigger}{'trigger_id'}."&triggernot_id=".${$trigger}{'triggernot_id'}."&equipment_id
=".${$trigger}{'equipment_id'}."&trigger_object=".${$trigger}{'trigger_object'}."&equipmenttype_id=".${$trigger}{'eq
uipmenttype_id'}."\"  class=\"menutext\" TITLE=\"Ver Objecto\">".${$trigger}{'trigger_object'}."<\/a></td>\n";

                                print "<td nowrap class=\"ConfTable\" align=\"center\" width=\"60%\">".$query-
>scrolling_list(-name=>'Variables',-values=>\@{${$trigger}{'trigger_variables'}},-size=>$size,-
multiple=>'true','class="Popup"')."</td>\n";
                                print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AlarmTriggersAdmin.cgi?ModView=MOD&TYPE=".$query-
>param('TYPE')."&trigger_id=".${$trigger}{'trigger_id'}."&triggernot_id=".${$trigger}{'triggernot_id'}."&equipment_id
=".${$trigger}{'equipment_id'}."&trigger_object=".${$trigger}{'trigger_object'}."&equipmenttype_id=".${$trigger}{'eq
uipmenttype_id'}."\" class=\"menutext\">Modificar</a></td>\n";
```

```
                                print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/AlarmTriggersAdmin.cgi?Del=DEL&TYPE=".$query-
>param('TYPE')."&equipment_id=".${$trigger}{'equipment_id'}."&equipmenttype_id=".${$trigger}{'equipmenttype_i
d'}."&trigger_id=".${$trigger}{'trigger_id'}."\" class=\"menutext\">Borrar</a></td></tr>\n";
                        }
        }
        if ($modflag == 0 ){
                          print $query->start_form(-action=>'/cgi-bin-admin/AlarmTriggersAdmin.cgi' );
                          print "<tr><td nowrap class=\"ConfTable\" align=\"center\"></td>\n";
                          print "<td nowrap class=\"ConfTable\" align=\"center\" width=\"8%\">".$query-
>popup_menu(-name=>'triggernot_id',-values=>$triggersID,-default=>$query->param('triggernot_id'))."</td>\n";
                          if ($query->param('TYPE') eq "Equipo"){
                        print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$Triggers}[0]{'equipment'}."</td>\n";
                          print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$Triggers}[0]{'equipmenttype_label'}."</td>\n";
                          $query->param('equipmenttype_id',0);
                           }elsif ($query->param('TYPE') eq "Tipo"){
                          print "<td nowrap class=\"ConfTable\"
align=\"center\">".${$Triggers}[0]{'equipmenttype_label'}."</td>\n";
                          $query->param('equipment_id',0);
                           }else{
                                    $query->param('equipmenttype_id',0);
                                    $query->param('equipment_id',0);
                          }
                          print "<td nowrap class=\"ConfTable\" align=\"center\" width=\"15%\">".$query-
>popup_menu(-name=>'severity',-values=>$Severities,-labels=>$SeveritiesLabels,-default=>$query-
>param('severity'))."</td>\n";
                          my @data=$query->param('actions_id');
                      print "<td nowrap class=\"ConfTable\" align=\"left\">".$query->checkbox_group(-
name=>'actions_id',-values=>\@actionsvalues,-labels=>$actionlabels,-default=>\@data,-
linebreak=>'true')."</td>\n";
                          print "<td nowrap class=\"ConfTable\" align=\"center\" colspan=\"2\" width=\"40%\"><table
width=\"100%\"><tr><td nowrap align=\"center\" width=\"100%\" colspan=\"2\">".$query->textfield(-
name=>'trigger_object',-value=>$query->param('trigger_object')||"",'class="popup"')."</td></tr><tr><td nowrap
class=\"ConfTable\" align=\"center\" width=\"40%\">".$query->submit('AddObj','MIB','class="Button"')."</td><td
nowrap class=\"ConfTable\" align=\"center\" width=\"40%\">".$query-
>submit('AllAlarms','Alarmas','class="Button"')."</td></tr></table></td>\n";
                          print "<td nowrap align=\"center\" colspan=\"2\" width=\"30%\">".$query-
>submit('Add','Agregar Filtro','class="Button"')."</td></tr>\n";
                          print $query->hidden('TYPE',$query->param('TYPE'));
                          print $query->hidden('equipment_id',$query->param('equipment_id'));
                          print $query->hidden('equipmenttype_id',$query->param('equipmenttype_id'));
                          print $query->endform;
                }

print <<EOF;

     </table></td>
  </tr>
</table>
    </td>
  </tr>
</table>
EOF
   ;
}
print $query->end_html;
```

9.2.5.17 AllAlarmsdisplay.cgi

```
#!/usr/bin/perl

#
##Select Alarms
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
```

```
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance';
use AlarmTriggersManager;

#Main
$query = new CGI;
print $query->header;
my $Focus="";
if (defined $query->param('FOCUS')){
        $Focus="setTimeout('window.focus()',10000);";
}

print $query->start_html(-title=>'Administración de Filtros',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -onLoad=>'window.focus()',
                -onBlur=>$Focus,
                );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Todas las Alarmas Definidas<\/td><td class=\"menu\"
nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.close()\"
class=\"menutext\">Cerrar<\/a><\/td><\/tr><\/table>\n";


############################
my $AlarmsTriggers = AlarmTriggersManager->new();

my ($Alarms) = $AlarmsTriggers->ReadAllAlarms();
##
##Select Option
if (defined $query->param('Select')){
print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
EOF
  ;
                        print "window.open(\"/cgi-bin-admin/AlarmTriggersAdmin.cgi?trigger_object=".$query-
>param('trigger_object')."&trigger_id=".$query->param('trigger_id')."&TYPE=".$query-
>param('TYPE')."&ModView=".$query->param('ModView')."&equipment_id=".$query-
>param('equipment_id')."&equipmenttype_id=".$query->param('equipmenttype_id')."&triggernot_id=".$query-
>param('triggernot_id')."&severity=".$query->param('severity')."\",\'mainFrame\');";
print <<EOF
                        self.close();
                        </script>


EOF
  ;
}
############################

print <<EOF;
<table width="100%" border="1" cellspacing="1">
EOF
  ;
my @data = $query->param('actions_id');
foreach my $alarm (@{$Alarms}){
        print "<tr>\n";
        print "<td class=\"ConfTable\" ".${$alarm}{'color'}." nowrap
align=\"center\">".${$alarm}{'oid_label'}."</td>";
        print "<td class=\"ConfTable\" ".${$alarm}{'color'}." nowrap align=\"center\">".${$alarm}{'type'}."</td>";
        print "<td class=\"ConfTable\" ".${$alarm}{'color'}." nowrap align=\"center\" width=\"40%\">".$query-
>textarea(-name=>'DescriptionAlarm',-default=>${$alarm}{'description'},-rows=>1,-
columns=>10,'class="Popup"')."</td>";
        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"/cgi-bin-
admin/AllAlarmsdisplay.cgi?Select=YES&trigger_object=".${$alarm}{'oid_label'}."&trigger_id=".$query-
>param('trigger_id')."&TYPE=".$query->param('TYPE')."&ModView=".$query-
>param('ModView')."&equipment_id=".$query->param('equipment_id')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&triggernot_id=".$query->param('triggernot_id')."&severity=".$query-
>param('severity')."&actions_id=".join(':',@data)."\" class=\"menutext\" TITLE=\"Seleccionar\"><></a></td></tr>\n";
        print "</tr>\n";
}
```

```
print <<EOF;
</table>
EOF
   ;
##References
my @bgcolor=('style= background-color:"#C84428"','style= background-color:"#8080C0"');
my @helptext=("SNMPv1 Trap","SNMPv2 Trap");
##
print "<table width=\"100%\" border=\"0\" cellspacing=\"1\"><tr>";
my $ind=0;
foreach my $helpitem (@helptext){
            print "<td  class=\"ConfTable\" ".$bgcolor[$ind]." align=\"center\"><font
size=\"1\">".$helpitem."</font><\/td>\n";
            $ind++;
}
print "</tr></table>";
print $query->end_html;
```

9.2.5.18 FilterObjects.cgi

```
#!/usr/bin/perl

#
##Admin Filters
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/AlarmSurveillance';
use AlarmTriggersManager;

#Main
$query = new CGI;
print $query->header;
my $Focus="";
if (defined $query->param('FOCUS')){
        $Focus="setTimeout('window.focus()',10000);";
}

print $query->start_html(-title=>'Administración de Filtros',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -onLoad=>'window.focus()',
                -onBlur=>$Focus,
                );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">".$query->param('trigger_object')."<\/td><td
class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\" onMouseOver=\"MenuMouseOver(this)\"
onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\" onClick=\"window.close()\"
class=\"menutext\">Cerrar<\/a><\/td><\/tr><\/table>\n";
############################
my $AlarmsTriggers = AlarmTriggersManager->new();
##Mod Option
if(defined $query->param('Mod')){
        my %search=(
        'trigger_id'=>$query->param('trigger_id'),
        );
        my $ind = 1;
        my %moddata;
        while( defined $query->param('object'.$ind)){
                if ($query->param('objectvalue'.$ind) ne ""){
                        $moddata{'trigger_variable'.$ind}= $query->param('object'.$ind);
                        $moddata{'trigger_value'.$ind}= $query->param('objectvalue'.$ind);
                }
                $ind++;
        }

        $AlarmsTriggers->ModAlarmTriggers(\%search,\%moddata);
```

```
print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
EOF
  ;
                        print "window.open(\"/cgi-bin-admin/AlarmTriggersAdmin.cgi?&TYPE=".$query-
>param('TYPE')."&equipment_id=".$query->param('equipment_id')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."\",\'mainFrame\');";
print <<EOF
                        </script>


EOF
  ;
}
###

my %data=('trigger_id'=>$query->param('trigger_id'),);
my ($data_ref_array,$ObjValues)=$AlarmsTriggers->ReadAlarmTriggers(\%data);
my %ObjValues;
foreach $obj(@{${$data_ref_array}[0]{'trigger_variables'}}){
        my ($key,$value) = split(/=/,$obj);
        $ObjValues{$key}=$value;
}

my $ObjData=$AlarmsTriggers->ReadObjects($query->param('trigger_object'));
############################

print <<EOF;
<table width="100%" border="1" cellspacing="0">
EOF
  ;
print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"30%\">Alarma</td><td nowrap
class=\"ConfTable\" align=\"center\" width=\"50%\">".${$ObjData}{'oid_label'}."</td></tr>\n";
${$ObjData}{'description'}=~ s/\s+/ /g;
print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"30%\">Descripcion</td><td nowrap
class=\"ConfTable\" align=\"center\" width=\"30%\">".$query->textarea(-name=>'DescriptionAlarm',-
default=>${$ObjData}{'description'},-rows=>5,-columns=>10,'class="Popup"')."</td></tr>\n";
if (defined ${$ObjData}{'enterprise'}){print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\"
width=\"30%\">Trap Tipo</td><td nowrap class=\"ConfTable\" align=\"center\"
width=\"50%\">".${$ObjData}{'enterprise'}."</td></tr>\n";}
if (defined ${$ObjData}{'trapspecificID'}){print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\"
width=\"30%\">Numero de Trap</td><td nowrap class=\"ConfTable\" align=\"center\"
width=\"50%\">".${$ObjData}{'trapspecificID'}."</td></tr>\n";}

my $ind = 1;
print "<tr><td colspan=\"2\"><table width=\"100%\" border=\"1\" cellspacing=\"2\">";
print $query->start_form(-action=>'/cgi-bin-admin/FilterObjects.cgi' );
while( exists(${$ObjData}{'object'.$ind})){
        print $query->hidden('object'.$ind,${$ObjData}{'object'.$ind});
        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"20%\">Objecto $ind</td><td
nowrap class=\"ConfTable\" align=\"center\" width=\"25%\" >".${$ObjData}{'object'.$ind}."</td>\n";
        print "<td nowrap class=\"ConfTable\" align=\"center\" width=\"25%\"
>".${$ObjData}{'objectdata'.$ind}{'syntax'}."</td>\n";
        ${$ObjData}{'objectdata'.$ind}{'description'} =~ s/\s+/ /g;
        print "<td nowrap class=\"ConfTable\" align=\"center\" rowspan=\"2\" width=\"30%\">".$query-
>textarea(-name=>'DescriptionObject',-default=>${$ObjData}{'objectdata'.$ind}{'description'},-rows=>2,-
columns=>10,'class="Popup"')."</td></tr>\n";
        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" width=\"30%\"
colspan=\"2\">".${$ObjData}{'object'.$ind}."</td>\n";
        if (scalar(@{${$ObjData}{'textualvalues'.$ind}}) != 0){
                print "<td nowrap class=\"ConfTable\" align=\"center\" width=\"25%\">".$query-
>popup_menu(-name=>'objectvalue'.$ind,-values=>${$ObjData}{'textualvalues'.$ind},-
labels=>${$ObjData}{'textuallabels'.$ind},'class="popup"',-
default=>$ObjValues{${$ObjData}{'object'.$ind}})."</td>\n";
        }else{
                print "<td nowrap class=\"ConfTable\" align=\"center\" width=\"25%\" >".$query->textfield(-
name=>'objectvalue'.$ind,-value=>$ObjValues{${$ObjData}{'object'.$ind}},'class="popup"')."</td>\n";
        }

        print "</tr>\n";
        $ind++;
}
if($ind > 1){
        print "<tr><td nowrap align=\"center\" colspan=\"4\" width=\"100%\">".$query-
>submit('Mod','Modificar','class="Button"')."</td></tr>\n";
```

```
}
        print $query->hidden('trigger_id',$query->param('trigger_id'));
        print $query->hidden('TYPE',$query->param('TYPE'));
        print $query->hidden('equipment_id',$query->param('equipment_id'));
        print $query->hidden('equipmenttype_id',$query->param('equipmenttype_id'));
        print $query->hidden('trigger_object',$query->param('trigger_object'));

print $query->endform;

print "</table></td></tr>";
print <<EOF;
</table>
EOF
  ;

print $query->end_html;
```

9.2.5.19 ProfilesAdmin.cgi

```
#!/usr/bin/perl

#
##Actions Profiles Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/GetSet';
use GetSetManager;

#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'ProfilesAdmin',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Perfiles de Acción<\/td><\/tr><\/table>\n";

my $Profiles= GetSetManager->new();

###Delete Option
if(defined $query->param('Del')){
        my %data=(
                  'getset_id' => $query->param('getset_id'),
        );
        my $error = $Profiles->UnCreateProf(\%data);
        if ($error ne "1"){
                print $error;
        }
}
###AdmProf Option
if(defined $query->param('AdmProf')){
print <<EOF;

                <script language="JavaScript" type="text/JavaScript">
EOF
  ;
                print "Profiles = window.open(\"/cgi-bin-
admin/ProfileCompView.cgi?FOCUS=ON&getset_id=".$query-
>param('getset_id')."\",\'Profiles\',\'scrollbars=no,width=600,height=500\');";
print <<EOF
                Profiles.moveTo(250,100);
                </script>
```

```
EOF
  ;
}

###Add Option
if(defined $query->param('Add')){
        my %data=(
                'getset_label' => $query->param('getset_label'),
                'readwrite' => $query->param('readwrite'),
        );
        $Profiles->CreateProf(\%data);

}
###
my ($AllProfiles,$ProfilesLabels,$Data)=$Profiles->ReadAllProfiles();

### Table Begining
print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table width="100%" border="1" cellspacing="0">
  <tr>
    <td width="30%" class=\"ConfTableTitle\" align="center">Perfilesde Acción</td>
    <td width="70%">
        <table width="100%" border="1" cellspacing="0">
EOF
  ;
     print $query->start_form(-action=>'/cgi-bin-admin/ProfilesAdmin.cgi' );
     print "<tr><td nowrap align=\"center\" class=\"ConfTableItem\" >Perfiles Definidos</td><td nowrap
align=\"center\" class=\"ConfTableItem\" >Tipo</td><td nowrap align=\"center\" class=\"ConfTableItem\"
width=\"15%\"></td></tr>\n";
     foreach my $prof(@{$Data}){
        print "<tr><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/ProfilesAdmin.cgi?AdmProf=YES&getset_id=".${$prof}{'getset_id'}."\" class=\"menutext\" TITLE=\"Agregar
Objectos al Perfil\">".${$prof}{'getset_label'}."</a></td>\n";
        print "<td nowrap class=\"ConfTable\" align=\"center\" >".${$prof}{'readwrite_label'}."</td>\n";
        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/ProfilesAdmin.cgi?Del=YES&getset_id=".${$prof}{'getset_id'}."\" class=\"menutext\" TITLE=\"Borrar
Perfil\">Borrar</a></td></tr>\n";
     }
     my %labels=(1=>"Lectura/Escritura",0=>"Solo Lectura",);
     print "<tr><td nowrap align=\"center\">".$query->textfield(-name=>'getset_label','class="Popup"')."</td><td
nowrap class=\"menustatic\" align=\"center\" >".$query->popup_menu(-name=>'readwrite',-values=>['1','0'],-
labels=>\%labels,'class="Popup"')."</td><td nowrap align=\"center\">".$query-
>submit('Add','Agregar','class="Button"')."</td></tr>\n";
     print $query->endform;
print <<EOF;
     </table></td>
  </tr>
</table>
    </td>
  </tr>
</table>
EOF
  ;

print $query->end_html;
```

9.2.5.20 ProfileCompView.cgi

```
#!/usr/bin/perl

#
##Profile Comp Admin
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;
```

```
use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/GetSet';
use GetSetManager;

#Main
$query = new CGI;
print $query->header;

my $Focus="";
if ($query->param('FOCUS') eq "ON" and !defined $query->param('Add')){
        $Focus='window.focus()';
}
print $query->start_html(-title=>'Administracion de Perfil',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -onLoad=>$Focus,
                -onBlur=>$Focus,
                );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Perfiles Asociados".$query-
>param('equipmenttype_label')."<\/td><td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"#\"
onClick=\'window.close()\') class=\"menutext\">Cerrar<\/a><\/td><\/tr><\/table>\n";

my $Profiles=GetSetManager->new();
##Del Option
if(defined $query->param('Del')){
        foreach my $prof_id ($query->param('OBJ')){
                $Profiles->DelComp($query->param('getset_id'),$prof_id);
        }
}

##Add Option
if(defined $query->param('Add')){
print <<EOF;

                <script language="JavaScript" type="text/JavaScript">
EOF
  ;
                print "MibBrowser = window.open(\"/cgi-
bin/MIBBrowserFrameset.cgi?Select=YES&FOCUS=ON&getset_id=".$query-
>param('getset_id')."\",\'MibBrowser\',\'scrollbars=no,width=800,height=600\');";
print <<EOF
                MibBrowser.moveTo(250,100);
                </script>


EOF
  ;

}
##Adding Option
if(defined $query->param('Adding')){
        foreach my $prof_id (split(/:/,$query->param('OBJ'))){
                if ($prof_id ne ""){$Profiles->AddComp($query->param('getset_id'),$prof_id);}
        }
}
##
my %search=('getset_id'=>$query->param('getset_id'),);
my ($AllProfiles,$ProfilesLabels,$Data)=$Profiles->ReadAllProfiles(undef,undef,\%search);

### Table Begining
print <<EOF;
<table width="100%" border="0" cellspacing="1">
EOF
  ;
   print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"2\">Objetos Asociados</td></tr>\n";
        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >Perfil</td><td class=\"ConfTable\"
nowrap align=\"center\" width=\"80%\" >".${$Data}[0]{'getset_label'}."</td></tr>\n";
  print $query->start_form(-action=>'/cgi-bin-admin/ProfileCompView.cgi' );
        print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" >Objetos</td><td class=\"ConfTable\"
nowrap align=\"center\" >".$query->scrolling_list(-name=>'OBJ',-values=>${$Data}[0]{'getset_comp'},-size=>20,-
multiple=>'true','class="Popup"')."</td></tr>\n";
```

```
                print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" ></td><td
class=\"ConfTableItem\"><table width=\"100%\"><tr><td>".$query-
>submit('Del','Quitar','class="Button"')."</td><td>".$query-
>submit('Add','Agregar','class="Button"')."</td></tr></table></td></tr>";
                print $query->hidden('FOCUS',$query->param('FOCUS'));
                print $query->hidden('getset_id',$query->param('getset_id'));
   print $query->endform;
print <<EOF;

</table>
EOF
   ;

print $query->end_html;
```

9.2.5.21 MIBBrowserResultsTriggers.cgi

```
#!/usr/bin/perl

#
##MIB Browser
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

#Main
$query = new CGI;

##Load Cookie
my $readcookie = $query->cookie('TRANOBJ');
###Parse
my @Objs=split(/:/,$readcookie);
####

##Del Option
my $DelFlag=0;
if(defined $query->param('Del') ){
        foreach my $objs ($query->param('trigger_object')){
                @Objs=grep($_ ne $objs,@Objs);
        }
        $DelFlag=1;
}
### ReLoad Data
my $cookie;
if ($query->param('Cancel') or $query->param('Add')){
$cookie = $query->cookie(-name=>'TRANOBJ',
                -value=>"",
                 );

}
if ($DelFlag == 1){
my $data =join(':',@Objs);
$cookie = $query->cookie(-name=>'TRANOBJ',
                -value=>$data,
                 );
}
print $query->header(-cookie=>$cookie,);
##
if ($query->param('FOCUS') eq "ON"){
        $Focus='window.focus()';
}

print $query->start_html(-title=>'Administracion de MIBs',
                -style=>{'src'=>'/Conf/WBNMS.css'},
                -onLoad=>$Focus,
                );
my @data=$query->param('actions_id');
if(defined $query->param('Add') or defined $query->param('Cancel')){ ##Add and Cancel Option
```

```
print "<script>window.open(\"/cgi-bin-admin/AlarmTriggersAdmin.cgi?trigger_id=".$query-
>param('trigger_id')."&trigger_object=".$query->param('trigger_object')."&TYPE=".$query-
>param('TYPE')."&ModView=".$query->param('ModView')."&equipment_id=".$query-
>param('equipment_id')."&equipmenttype_id=".$query->param('equipmenttype_id')."&triggernot_id=".$query-
>param('triggernot_id')."&severity=".$query-
>param('severity')."&actions_id=".join(':',@data)."\",\"mainFrame\")<\/script>";
print "<p> </p><p> </p><p> </p><p> </p><p align=\"center\"><strong><font
color=\"#FF0000\" size=\"+1\">Cargando ...</font></strong></p>";
print "<script>top.close();<\/script>";

}elsif (defined $query->param('Del')){ ##Del Option
print "<script>window.open(\"/cgi-bin-admin/MIBBrowserResultsTriggers.cgi?FOCUS=ON&trigger_id=".$query-
>param('trigger_id')."&TYPE=".$query->param('TYPE')."&ModView=".$query-
>param('ModView')."&equipment_id=".$query->param('equipment_id')."&equipmenttype_id=".$query-
>param('equipmenttype_id')."&triggernot_id=".$query->param('triggernot_id')."&severity=".$query-
>param('severity')."&actions_id=".join(':',@data)."\",\"resultsfrm\")<\/script>";
print "<p> </p><p> </p><p> </p><p> </p><p> </p><p align=\"center\"><strong><font
color=\"#FF0000\" size=\"+1\">Volviendo ...</font></strong></p>";

}else{

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Filtro<\/td><\/tr><\/table>\n";

print <<EOF;
<table width="100%" border="0" cellspacing="1">
EOF
   ;
   print $query->start_form(-action=>'/cgi-bin-admin/MIBBrowserResultsTriggers.cgi');
                print "<tr><td class=\"ConfTable\" nowrap align=\"center\" >".$query->textfield(-
name=>'trigger_object',value=>$Objs[0],'class="Popup"')."</td></tr>\n";
                print "<tr><td class=\"ConfTableItem\"><table width=\"100%\"><tr><td>".$query-
>submit('Del','Anterior','class="Button"')."</td><td>".$query-
>submit('Add','Agregar','class="Button"')."</td></tr></table></td></tr><tr><td class=\"ConfTableItem\">".$query-
>submit('Cancel','Cancelar','class="Button"')."</td></tr>";
                print $query->hidden('FOCUS',$query->param('FOCUS'));
                print $query->hidden('trigger_id',$query->param('trigger_id'));
                print $query->hidden('equipment_id',$query->param('equipment_id'));
                print $query->hidden('equipmenttype_id',$query->param('equipmenttype_id'));
                print $query->hidden('TYPE',$query->param('TYPE'));
                print $query->hidden('ModView',$query->param('ModView'));
                print $query->hidden('triggernot_id',$query->param('triggernot_id'));
                print $query->hidden('severity',$query->param('severity'));
                print $query->hidden('actions_id',join(':',@data));
   print $query->endform;
print <<EOF;

</table>
EOF
   ;
}
print $query->end_html;
```

9.2.5.22 MIBsAdmin.cgi

```
#!/usr/bin/perl

#
##MIBs Administration
#

#Load Libraries
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

use lib '/var/sites/Wbnms/Wbnms_Conf';
use WbnmsConf;

use lib '/var/sites/Wbnms/Wbnms_Core/MIBCompiler';
use MIBManager;

use lib '/var/sites/Wbnms/Wbnms_Core/MIBbrowser';
use OID2js;
```

```
#Main
$query = new CGI;
print $query->header;


print $query->start_html(-title=>'Administracion de MIBs',
                         -style=>{'src'=>'/Conf/WBNMS.css'},
                         );

print "<script src=\"/Conf/menu.js\"><\/script>";
print "<table width=\"100%\"><tr><td class=\"menustatic\">Administración de MIBS<\/td><\/tr><\/table>\n";
print "<p> <\/p>\n";
my $MIBManager = MIBManager->new();
my ($mibs,$mibsloaded) = $MIBManager->ReadMIBs();
if (defined $query->param('MIB') and defined $query->param('Add')){
        $MIBManager->MIB($query->param('MIB'));
print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
                        Wait = window.open("/cgi-bin-admin/Wait.cgi",'Wait','scrollbars=no,width=320,height=120');
                        Wait.moveTo(250,250);
                        </script>


EOF
  ;


        my $pid = fork();
        if ($pid == 0){
                my $error = $MIBManager->MibLoad();
                if (defined $error ){
                        print $error."\n";
                }else{
                        $MIBManager->AddMIBs($mibsloaded);
                        my $OID2js = OID2js->new;
                        $OID2js->OID2js();
                }
                exit(0);
        }else{
                waitpid($pid,0);
        }


print <<EOF;

                        <script language="JavaScript" type="text/JavaScript">
                        window.open("/cgi-bin-admin/MIBsAdmin.cgi",'mainFrame');
                        setTimeout("Wait.close()",1000);
                        </script>

EOF
  ;
}

if (defined $query->param('MIB') and defined $query->param('Del')){
                        $MIBManager->MIB($query->param('MIB'));
                        $mibsloaded = $MIBManager->DelMIBs($mibsloaded);

}

##UpLoad Option
if (defined $query->param('uploaded_file')){
        my $filename = $query->param('uploaded_file');
        $MIBManager->UpLoad($filename);
        ($mibs,$mibsloaded) = $MIBManager->ReadMIBs();
}

print <<EOF;
<table width="100%" border="0" cellspacing="0">
<tr><td align="left" valign="center">
<table width="100%"  border="1" cellspacing="0">
  <tr>
    <td width="30%" class=\"ConfTableTitle\" align="center">MIBs</td>
```

```
   <td nowrap>
            <table width="100%" border="1" cellspacing="0">

EOF
  ;
               print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"2\">Mibs Disponibles en el
Servidor</td></tr>\n";
         print $query->start_form(-action=>'/cgi-bin-admin/MIBsAdmin.cgi' );
               print "<tr><td nowrap>".$query->popup_menu(-name=>'MIB',-values=>$mibs,-
default=>'','class="popup"')."</td>\n";
               print "<td nowrap align=\"center\" >".$query-
>submit('Add','Agregar','class="Button"')."</a></td></tr>\n";
               print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\" colspan=\"2\" >Subir al Servidor Nueva
Mib</td></tr>\n";
               print $query->endform;
               print $query->start_multipart_form(-action=>'/cgi-bin-admin/MIBsAdmin.cgi' );
               print "<tr><td class=\"ConfTableTitle\" nowrap align=\"center\">".$query->filefield('uploaded_file')."<td
class=\"ConfTableTitle\" nowrap align=\"center\" width=\"100\">".$query-
>submit('Up','UpLoad','class="Button"')."</td></tr>\n";
               print $query->endform;
               print "<tr><td class=\"ConfTableItem\" nowrap align=\"center\">Mibs Cargadas</td><td
class=\"ConfTableTitle\" nowrap align=\"center\">   </td></tr>\n";
               foreach my $mibload (@{$mibsloaded}){

                        print "<tr><td nowrap class=\"ConfTable\" align=\"center\">$mibload</td>\n";
                        print "<td class=\"menu\" nowrap align=\"center\" onClick=\"MenuSelected(this)\"
onMouseOver=\"MenuMouseOver(this)\" onMouseOut=\"MenuMouseOut(this)\"><a href=\"\/cgi-bin-
admin\/MIBsAdmin.cgi?Del=DEL&MIB=".$mibload."\" class=\"menutext\">Borrar</a></td></tr>\n";

       }


print <<EOF;

    </table></td>
  </tr>
</table>
   </td>
  </tr>
</table>
EOF
  ;



print $query->end_html;
```

9.3 Archivos de Instalación

9.3.1 install.pl

```
#!/usr/bin/perl
#
## install.pl
## Version: 2
##
#


#Prototypes
sub iterate($$);
sub read_dir ($);
sub change_file ($$);
sub apache_vhost($$$$);
sub mysql_conf($);

##Configuration
```

```
my
($ROOTPATH,$APACHEUSER,$HTTPDCONF_PATH,$SERVER_NAME,$SERVER_IP,$SERVER_PORT,$MY
SQL_PATH);
if (defined $ARGV[0] and defined $ARGV[1] and ((defined $ARGV[2] and defined $ARGV[3] and defined
$ARGV[4] and defined $ARGV[5]) or (!defined $ARGV[2] and !defined $ARGV[3] and !defined $ARGV[4] and
!defined $ARGV[5]))){
        ($ROOTPATH,$APACHEUSER,$HTTPDCONF_PATH,$SERVER_NAME,$SERVER_IP,$SERVER_P
ORT,$MYSQL_PATH) = @ARGV;
        ###Test Installed modules
        print "**Verificando modulos requeridos instalados\n";
        my @packages=("DBI","SNMP_Session","SNMP_util","BER","GD"); ## Required Modules
        my $test=0;
        foreach my $mod(@packages){
           unless (eval "require $mod") {
              warn "Modulo $mod NO instalado .... Fail\n";
              $test = 1;
           }else{
                 print "Modulo $mod instalado .... OK\n";
           }
        }
        if ($test  == 1){ goto "END";}
        print "**Configurando WBNMS v1.0 en el directorio ".$ARGV[0]."\n";
}else{
        print "uso: perl install.pl <pathrootWBNMS> <apacheuser> [<httpdconf_path>|--] <server_name>
<server_ip> <server_port> [<mysql_path>]\n";
        print "     <pathrootWBNMS> es el directorio raiz de la aplicacion. \n";
        print "     <apacheuser> es el usuario de servidor apache. \n";
        print "     [<httpdconf_path> | --] path al arhcivo de configuración del servidor Apache
(/etc/httpd/conf/httpd.conf). \n";
        print "     <server_name> nombre del dominio destinado a la aplicación (www.wbnms.com.ar). \n";
        print "     <server_ip> IP a la que responderá el servidor virtual de la aplicación. \n";
        print "     <server_port> Port de escucha de la aplicaión. \n";
        print "     <mysql_path> path a mysql (/usr/local/mysql/bin/mysql).Si es omitido, se asume una Mysql
externa. \n";
}


# Main
my @levelpath;
if (defined $ROOTPATH and defined $APACHEUSER and (( defined $HTTPDCONF_PATH and defined
$SERVER_NAME and defined $SERVER_IP and defined $SERVER_PORT) or ( !defined $HTTPDCONF_PATH
and !defined $SERVER_NAME and !defined $SERVER_IP and !defined $SERVER_PORT) )){
        $levelpath[0] = $ROOTPATH; ##Wbnms root directory
        my ($Allfiles,$files,$dirs)=&read_dir($levelpath[0]);
        &iterate($dirs,\@levelpath);
        ##Install Vhost
        if (defined $HTTPDCONF_PATH and defined $SERVER_NAME and defined $SERVER_IP and
defined $SERVER_PORT and $HTTPDCONF_PATH ne "--"){
                print "**Configurando Vhost en el Servidor Apache\n";
                my $text="Instalación del Vhost ";
                print $text;
                my $add;
                for(1..(60 - length($text))){$add = ".".$add;};
                my $testapache =
&apache_vhost($HTTPDCONF_PATH,$SERVER_NAME,$SERVER_IP,$SERVER_PORT);
                if ($testapache){print "$add OK\n";}else{print "$add Fail\n"};
        }else{
                print "El Vhost debera ser creado manualmente en el servidor Apache.\n";
        }
        ##Mysql Configuration
        if (defined $MYSQL_PATH){
                print "**Configurando usuario y permiso en el servidor Mysql local.\n";
                my $text="Instalación Mysql ";
                print $text;
                my $add;
                for(1..(60 - length($text))){$add = ".".$add;};
                my $testmysql = mysql_conf($MYSQL_PATH);
                if ($testmysql){print "$add OK\n";}else{print "$add Fail\n"};
        }else{
                print "El Usuario Mysql y la Db WBNMS deberan ser Configurados en el server
correspondiente.\n";
        }
}
# Subrutines
sub iterate($$){
        my ($dirs,$levelpath)= @_;
```

```
		foreach my $dir (@{$dirs}){
			my ($Allfiles,$files,$dirs)=&read_dir(join('/',(@{$levelpath},$dir)));
			foreach my $file (@{$files}){
				if ($file !~ /install/){
					print join('/',(@{$levelpath},$dir))." ....";
					my $test = &change_file($file,join('/',(@{$levelpath},$dir)));
					my $add;
					for(1..(60 - length(join('/',(@{$levelpath},$dir))) ) ){$add = "."$add;};
					if ($test){print "$add OK\n";}else{print "$add Fail\n"};
				}

			}
			if (defined $dirs){
				my $pid=fork();
				if ($pid == 0 ){
					push(@{$levelpath},$dir);
					&iterate($dirs,$levelpath);
					exit(0);
				}else{waitpid($pid,0);}
			}
		}
}


sub read_dir ($) {
    my ($dirname)= @_;
    opendir(DIR,$dirname);
    my @Allfiles = grep(!/^\./,readdir(DIR));
    closedir(DIR);
    my (@files,@dirs);
    foreach my $test(@Allfiles){
			my $dirtest = opendir(DIR,$dirname."/".$test);
			if (!defined $dirtest){
				push(@files,$test);
			}else{
				closedir(DIR);
				push(@dirs,$test);
				&chown_by_name($APACHEUSER,"$dirname/$test");
				chmod(0774,"$dirname/$test");
				##chmod(0777,"$path/$filename");
			}
    }
return(\@Allfiles,\@files,\@dirs);
}

sub change_file($$){
		my ($filename,$path)=@_;
	 my $test1 = open(CHANGE,$path."/".$filename);
	 my @swap;
		while (<CHANGE>){
			push(@swap,$_);
		}
		close(CHANGE);
		my $test2 = open(CHANGE,">$path/$filename");
		foreach my $temp (@swap) {
			$temp =~ s/<<INSTALL_CONF>>/$ROOTPATH/g;
			##$temp =~ s/\/var\/sites\/Wbnms/<<INSTALL_CONF>>/g;
			print CHANGE $temp;
		}
		close(CHANGE);
		&chown_by_name($APACHEUSER,"$path/$filename");
		chmod(0774,"$path/$filename");
		##chmod(0777,"$path/$filename");
		my $test = $test1 && $test2;
return($test);
}

sub chown_by_name {
   local($user, $pattern) = @_;
   chown((getpwnam($user))[2,3], glob($pattern));
}

sub apache_vhost($$$$){
		my ($httpdconf_path,$server_name,$server_ip,$Server_port)= @_;
		my $test = open(HTTPDCONF,"$httpdconf_path");
		close(HTTPDCONF);
```

```
                        if ($test){
                                open(HTTPDCONF,">>$httpdconf_path");
                                if ($server_ip eq "Any"){$server_ip="_default_";}
                                if ($Server_port eq "80"){$Server_port="";}elsif($Server_port eq
"Any"){$Server_port=":*";}else{$Server_port=":".$Server_port;}
                                print HTTPDCONF "\n<VirtualHost $server_ip$Server_port>\n";
                                print HTTPDCONF "DocumentRoot $ROOTPATH/Wbnms_Web\n";
                                print HTTPDCONF "ServerName $server_name\n";
                                print HTTPDCONF "ScriptAlias /cgi-bin/ $ROOTPATH/Wbnms_Web/cgi-bin/\n";
                                print HTTPDCONF "ScriptAlias /cgi-bin-admin/
$ROOTPATH/Wbnms_Web/webadm/cgi-bin/\n";
                                print HTTPDCONF "Alias /BMP/ $ROOTPATH/Wbnms_Var/File_DB/BMP/\n";
                                print HTTPDCONF "</VirtualHost>\n";
                                close(HTTPDCONF);
                        }
return($test);
}

sub mysql_conf($){
        my ($mysql_path)= @_;
        my $test1 = open(MYSQL,$mysql_path);
        my $test2 = system($mysql_path." < ./Sql.d/user.sql");
        my $test3 = system($mysql_path." < ./Sql.d/WBM.sql");
        my $test = $test1 && (not $test2) && (not $test3);
return($test);
}

END: print "\n";
1;
```

9.3.2 wbnmsserver.rc

```
#!/bin/sh
#
# wbnmsserver.rc
#

case "$1" in
  start)
        echo "--------Starting WBNMS System---------"
        /var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/start
        /var/sites/Wbnms/Wbnms_Core/Drawer/start
        /var/sites/Wbnms/Wbnms_Core/trapd/start
        /var/sites/Wbnms/Wbnms_Core/keepalive/start
        ;;
  stop)
        echo "--------Stopping WBNMS System---------"
        /var/sites/Wbnms/Wbnms_Core/AlarmSurveillance/stop
        /var/sites/Wbnms/Wbnms_Core/Drawer/stop
        /var/sites/Wbnms/Wbnms_Core/trapd/stop
        /var/sites/Wbnms/Wbnms_Core/keepalive/stop
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  reload)
        $0 stop
        $0 start
        ;;

*)
        echo "Usage: $0 { start | stop | restart }"
        exit 1
        ;;
esac

exit 0
```

9.3.3 SQL

9.3.3.1 WBM.sql

```
##Create Data Base WBNMSDB

create database WBNMSDB;
use WBNMSDB;

## Create Tables

#Tables of equipments

create table Equipment ( equipment_id int not null auto_increment,
                         equipment char(30) not null ,
                         IP char (30) not null,
                         community char(30),
                         description char(200),
                         MIB char(30),
                equipmenttype_id int not null,
                primary key(equipment_id));

create table NewEquipment ( equipment char(30) not null ,
                         IP char (30) not null,
                         community char(30),
                primary key(IP));

create table EquipmentMonitor ( equipment_id int not null primary key ,
                                state_id int not null,
                                changed int not null default 1
                                );

create table MonitorEquipmentState ( state_id int not null ,
                                state_label char (30) not null,
                                state_color char (30) not null,
                                primary key(state_id,state_label));

create table EquipmentTypes ( equipmenttype_id int not null auto_increment,
                              equipmenttype_label char (50) not null,
                              equipmentbmp char (50),
                  primary key(equipmenttype_id,equipmenttype_label));

# Tables of Mibs

create table Oid2Label (oid char(50) not null,
                        oid_label char(50) not null,
                        object_type int not null,
                        syntax char(50),
                        primary key(oid, oid_label));

create table Syntax (type_id int not null,
                type_label char(30) not null,
                primary key( type_label));

create table TextualConventions (convention_label char(50) not null,
                                 convention_value int not null,
                                 convention_value_label char(50) not null);

create table Sequences (sequence_label char(50) not null,
                        sequence_object char(50) not null,
                        sequence_type char(50) not null);

create table NotificationType (oid_label char(50) not null primary key,
                               status char(30),
                               description BLOB,
                                    object1 char(50) default null,
                                    object2 char(50) default null,
                                    object3 char(50) default null,
                                    object4 char(50) default null,
                                    object5 char(50) default null,
                                    object6 char(50) default null,
                                    object7 char(50) default null,
                                    object8 char(50) default null,
```

```
                object9 char(50) default null,
                object10 char(50) default null,
                object11 char(50) default null,
                object12 char(50) default null,
                object13 char(50) default null,
                object14 char(50) default null,
                object15 char(50) default null,
                object16 char(50) default null,
                object17 char(50) default null,
                object18 char(50) default null,
                object19 char(50) default null,
                object20 char(50) default null,
                object21 char(50) default null,
                object22 char(50) default null,
                object23 char(50) default null,
                object24 char(50) default null,
                object25 char(50) default null,
                object26 char(50) default null,
                object27 char(50) default null,
                object28 char(50) default null,
                object29 char(50) default null,
                object30 char(50) default null,
                object31 char(50) default null,
                object32 char(50) default null,
                object33 char(50) default null,
                object34 char(50) default null,
                object35 char(50) default null,
                object36 char(50) default null,
                object37 char(50) default null,
                object38 char(50) default null,
                object39 char(50) default null,
                object40 char(50) default null,
                object41 char(50) default null,
                object42 char(50) default null,
                object43 char(50) default null,
                object44 char(50) default null,
                object45 char(50) default null,
                object46 char(50) default null,
                object47 char(50) default null,
                object48 char(50) default null,
                object49 char(50) default null,
                object50 char(50) default null
        );
create table TrapType (oid_label char(50) not null primary key,
        enterprise char(50) not null,
        trapspecificID int not null,
        description BLOB,
                object1 char(50) default null,
                object2 char(50) default null,
                object3 char(50) default null,
                object4 char(50) default null,
                object5 char(50) default null,
                object6 char(50) default null,
                object7 char(50) default null,
                object8 char(50) default null,
                object9 char(50) default null,
                object10 char(50) default null,
                object11 char(50) default null,
                object12 char(50) default null,
                object13 char(50) default null,
                object14 char(50) default null,
                object15 char(50) default null,
                object16 char(50) default null,
                object17 char(50) default null,
                object18 char(50) default null,
                object19 char(50) default null,
                object20 char(50) default null,
                object21 char(50) default null,
                object22 char(50) default null,
                object23 char(50) default null,
                object24 char(50) default null,
                object25 char(50) default null,
                object26 char(50) default null,
                object27 char(50) default null,
                object28 char(50) default null,
                object29 char(50) default null,
```

```
                                        object30 char(50) default null,
                                        object31 char(50) default null,
                                        object32 char(50) default null,
                                        object33 char(50) default null,
                                        object34 char(50) default null,
                                        object35 char(50) default null,
                                        object36 char(50) default null,
                                        object37 char(50) default null,
                                        object38 char(50) default null,
                                        object39 char(50) default null,
                                        object40 char(50) default null,
                                        object41 char(50) default null,
                                        object42 char(50) default null,
                                        object43 char(50) default null,
                                        object44 char(50) default null,
                                        object45 char(50) default null,
                                        object46 char(50) default null,
                                        object47 char(50) default null,
                                        object48 char(50) default null,
                                        object49 char(50) default null,
                                        object50 char(50) default null
                                    );

create table ObjectType (oid_label char(50) not null primary key,
                                    max_access char(30),
                                    status char(30),
                                    description BLOB, ###Mirar !!!
                                        index1 char(50) default null,
                                        index2 char(50) default null,
                                        index3 char(50) default null,
                                        index4 char(50) default null,
                                        index5 char(50) default null,
                                        index6 char(50) default null,
                                        index7 char(50) default null,
                                        index8 char(50) default null,
                                        index9 char(50) default null,
                                        index10 char(50) default null,
                                        index11 char(50) default null,
                                        index12 char(50) default null,
                                        index13 char(50) default null,
                                        index14 char(50) default null,
                                        index15 char(50) default null,
                                        index16 char(50) default null,
                                        index17 char(50) default null,
                                        index18 char(50) default null,
                                        index19 char(50) default null,
                                        index20 char(50) default null,
                                        index21 char(50) default null,
                                        index22 char(50) default null,
                                        index23 char(50) default null,
                                        index24 char(50) default null,
                                        index25 char(50) default null,
                                        index26 char(50) default null,
                                        index27 char(50) default null,
                                        index28 char(50) default null,
                                        index29 char(50) default null,
                                        index30 char(50) default null,
                                        index31 char(50) default null,
                                        index32 char(50) default null,
                                        index33 char(50) default null,
                                        index34 char(50) default null,
                                        index35 char(50) default null,
                                        index36 char(50) default null,
                                        index37 char(50) default null,
                                        index38 char(50) default null,
                                        index39 char(50) default null,
                                        index40 char(50) default null,
                                        index41 char(50) default null,
                                        index42 char(50) default null,
                                        index43 char(50) default null,
                                        index44 char(50) default null,
                                        index45 char(50) default null,
                                        index46 char(50) default null,
                                        index47 char(50) default null,
                                        index48 char(50) default null,
                                        index49 char(50) default null,
```

```
                                        index50 char(50) default null
                                );

#Tables of Traps

create table AlarmsTrigger (trigger_id int not null primary key auto_increment,
                                        trigger_object char(50),
                                        equipmenttype_id int,
                                        equipment_id int,
                                        triggernot_id int,
                                        severity int not null,
                                                trigger_variable1 char(50) default null,
                                                trigger_value1 char(50) default null,
                                                trigger_variable2 char(50) default null,
                                                trigger_value2 char(50) default null,
                                                trigger_variable3 char(50) default null,
                                                trigger_value3 char(50) default null,
                                                trigger_variable4 char(50) default null,
                                                trigger_value4 char(50) default null,
                                                trigger_variable5 char(50) default null,
                                                trigger_value5 char(50) default null,
                                                trigger_variable6 char(50) default null,
                                                trigger_value6 char(50) default null,
                                                trigger_variable7 char(50) default null,
                                                trigger_value7 char(50) default null,
                                                trigger_variable8 char(50) default null,
                                                trigger_value8 char(50) default null,
                                                trigger_variable9 char(50) default null,
                                                trigger_value9 char(50) default null,
                                                trigger_variable10 char(50) default null,
                                                trigger_value10 char(50) default null,
                                                trigger_variable11 char(50) default null,
                                                trigger_value11 char(50) default null,
                                                trigger_variable12 char(50) default null,
                                                trigger_value12 char(50) default null,
                                                trigger_variable13 char(50) default null,
                                                trigger_value13 char(50) default null,
                                                trigger_variable14 char(50) default null,
                                                trigger_value14 char(50) default null,
                                                trigger_variable15 char(50) default null,
                                                trigger_value15 char(50) default null,
                                                trigger_variable16 char(50) default null,
                                                trigger_value16 char(50) default null,
                                                trigger_variable17 char(50) default null,
                                                trigger_value17 char(50) default null,
                                                trigger_variable18 char(50) default null,
                                                trigger_value18 char(50) default null,
                                                trigger_variable19 char(50) default null,
                                                trigger_value19 char(50) default null,
                                                trigger_variable20 char(50) default null,
                                                trigger_value20 char(50) default null,
                                                trigger_variable21 char(50) default null,
                                                trigger_value21 char(50) default null,
                                                trigger_variable22 char(50) default null,
                                                trigger_value22 char(50) default null,
                                                trigger_variable23 char(50) default null,
                                                trigger_value23 char(50) default null,
                                                trigger_variable24 char(50) default null,
                                                trigger_value24 char(50) default null,
                                                trigger_variable25 char(50) default null,
                                                trigger_value25 char(50) default null,
                                                trigger_variable26 char(50) default null,
                                                trigger_value26 char(50) default null,
                                                trigger_variable27 char(50) default null,
                                                trigger_value27 char(50) default null,
                                                trigger_variable28 char(50) default null,
                                                trigger_value28 char(50) default null,
                                                trigger_variable29 char(50) default null,
                                                trigger_value29 char(50) default null,
                                                trigger_variable30 char(50) default null,
                                                trigger_value30 char(50) default null,
                                                trigger_variable31 char(50) default null,
                                                trigger_value31 char(50) default null,
                                                trigger_variable32 char(50) default null,
                                                trigger_value32 char(50) default null,
                                                trigger_variable33 char(50) default null,
```

```
                                        trigger_value33 char(50) default null,
                                        trigger_variable34 char(50) default null,
                                        trigger_value34 char(50) default null,
                                        trigger_variable35 char(50) default null,
                                        trigger_value35 char(50) default null,
                                        trigger_variable36 char(50) default null,
                                        trigger_value36 char(50) default null,
                                        trigger_variable37 char(50) default null,
                                        trigger_value37 char(50) default null,
                                        trigger_variable38 char(50) default null,
                                        trigger_value38 char(50) default null,
                                        trigger_variable39 char(50) default null,
                                        trigger_value39 char(50) default null,
                                        trigger_variable40 char(50) default null,
                                        trigger_value40 char(50) default null,
                                        trigger_variable41 char(50) default null,
                                        trigger_value41 char(50) default null,
                                        trigger_variable42 char(50) default null,
                                        trigger_value42 char(50) default null,
                                        trigger_variable43 char(50) default null,
                                        trigger_value43 char(50) default null,
                                        trigger_variable44 char(50) default null,
                                        trigger_value44 char(50) default null,
                                        trigger_variable45 char(50) default null,
                                        trigger_value45 char(50) default null,
                                        trigger_variable46 char(50) default null,
                                        trigger_value46 char(50) default null,
                                        trigger_variable47 char(50) default null,
                                        trigger_value47 char(50) default null,
                                        trigger_variable48 char(50) default null,
                                        trigger_value48 char(50) default null,
                                        trigger_variable49 char(50) default null,
                                        trigger_value49 char(50) default null,
                                        trigger_variable50 char(50) default null,
                                        trigger_value50 char(50) default null
                                );
create table AlarmsEquipmentMemory (equipment_id int not null,
                                trap_id int not null,
                                primary key( equipment_id,trap_id)
                                );

create table TrapsCollector (trap_id int not null primary key auto_increment,
                                marked char(30),
                                ack char(30),
                                time datetime,
                                hosts char(50),
                                trap_trigger_id int,
                                community char(50),
                                object char(50),
                                trap_processed int not null default 0,
                                        variable1 char(50) default null,
                                        value1 char(50) default null,
                                        variable2 char(50) default null,
                                        value2 char(50) default null,
                                        variable3 char(50) default null,
                                        value3 char(50) default null,
                                        variable4 char(50) default null,
                                        value4 char(50) default null,
                                        variable5 char(50) default null,
                                        value5 char(50) default null,
                                        variable6 char(50) default null,
                                        value6 char(50) default null,
                                        variable7 char(50) default null,
                                        value7 char(50) default null,
                                        variable8 char(50) default null,
                                        value8 char(50) default null,
                                        variable9 char(50) default null,
                                        value9 char(50) default null,
                                        variable10 char(50) default null,
                                        value10 char(50) default null,
                                        variable11 char(50) default null,
                                        value11 char(50) default null,
                                        variable12 char(50) default null,
                                        value12 char(50) default null,
                                        variable13 char(50) default null,
                                        value13 char(50) default null,
```

```
    variable14 char(50) default null,
    value14 char(50) default null,
    variable15 char(50) default null,
    value15 char(50) default null,
    variable16 char(50) default null,
    value16 char(50) default null,
    variable17 char(50) default null,
    value17 char(50) default null,
    variable18 char(50) default null,
    value18 char(50) default null,
    variable19 char(50) default null,
    value19 char(50) default null,
    variable20 char(50) default null,
    value20 char(50) default null,
    variable21 char(50) default null,
    value21 char(50) default null,
    variable22 char(50) default null,
    value22 char(50) default null,
    variable23 char(50) default null,
    value23 char(50) default null,
    variable24 char(50) default null,
    value24 char(50) default null,
    variable25 char(50) default null,
    value25 char(50) default null,
    variable26 char(50) default null,
    value26 char(50) default null,
    variable27 char(50) default null,
    value27 char(50) default null,
    variable28 char(50) default null,
    value28 char(50) default null,
    variable29 char(50) default null,
    value29 char(50) default null,
    variable30 char(50) default null,
    value30 char(50) default null,
    variable31 char(50) default null,
    value31 char(50) default null,
    variable32 char(50) default null,
    value32 char(50) default null,
    variable33 char(50) default null,
    value33 char(50) default null,
    variable34 char(50) default null,
    value34 char(50) default null,
    variable35 char(50) default null,
    value35 char(50) default null,
    variable36 char(50) default null,
    value36 char(50) default null,
    variable37 char(50) default null,
    value37 char(50) default null,
    variable38 char(50) default null,
    value38 char(50) default null,
    variable39 char(50) default null,
    value39 char(50) default null,
    variable40 char(50) default null,
    value40 char(50) default null,
    variable41 char(50) default null,
    value41 char(50) default null,
    variable42 char(50) default null,
    value42 char(50) default null,
    variable43 char(50) default null,
    value43 char(50) default null,
    variable44 char(50) default null,
    value44 char(50) default null,
    variable45 char(50) default null,
    value45 char(50) default null,
    variable46 char(50) default null,
    value46 char(50) default null,
    variable47 char(50) default null,
    value47 char(50) default null,
    variable48 char(50) default null,
    value48 char(50) default null,
    variable49 char(50) default null,
    value49 char(50) default null,
    variable50 char(50) default null,
    value50 char(50) default null
);
```

```
create table AlarmsActions (action_id int not null primary key auto_increment,
                            module_id int not null,
                            action_module_comment BLOB not null
                            );

create table ActionsTriggers (action_id int not null,
                              trigger_id int not null,
                              primary key( action_id,trigger_id)
                              );

create table ActionsUsersEquipments (asignation_id int not null primary key auto_increment,
                                     user_id int not null,
                                     action_id int not null,
                                     equipment_id int not null,
                                     map_id int not null,
                                     starttime char(30),
                                     endtime char(30)
                                      );

create table UsersActions (user_id int primary key auto_increment,
                           name char(50) not null,
                           range char(30),
                           email char(50),
                           IP char(50),
                           community char(30)
                           );
#Tables of Maps

create table Maps (map_id int not null primary key auto_increment,
                   mapbmp char(50),
                   mapicon char(50),
                   mapname char(50),
                   comment BLOB
                   );

create table MapsEquipments (map_id int not null ,
                             equipment_id int not null,
                             coor char(50),
                            primary key(map_id,equipment_id)
                           );

create table MapsMaps      (son_map_id int not null ,
                            father_map_id int not null,
                            coor char(50),
                           primary key(son_map_id,father_map_id)
                           );

create table MapsMonitor ( map_id int not null primary key ,
                                   state_id int not null,
                                   changed int not null default 1
                         );


#Get/Set Tables

create table EquipmentTypeGetSet ( equipmenttype_id int not null,
                                    getset_id int not null,
                                   primary key(equipmenttype_id,getset_id)
                         );

create table GetSet ( getset_id int not null primary key auto_increment,
                                   getset_label char(30) not null,
                                   getset_comp char(200),
                                   readwrite int not null default 0
                         );


###Insert Default Variables

INSERT INTO AlarmsActions VALUES (1,0,'e-mail');
INSERT INTO AlarmsActions VALUES (2,1,'SNMP');

INSERT INTO AlarmsTrigger VALUES
(1,'coldStart',0,0,0,1,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NU
LL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
```

```
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
INSERT INTO AlarmsTrigger VALUES
(2,'warmStart',0,0,0,1,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
INSERT INTO AlarmsTrigger VALUES
(3,'linkUp',0,0,0,0,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NU
LL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
INSERT INTO AlarmsTrigger VALUES
(4,'linkDown',0,0,3,4,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
INSERT INTO AlarmsTrigger VALUES
(5,'authenticationFailure',0,0,0,1,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NU
LL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);
INSERT INTO AlarmsTrigger VALUES
(6,'egpNeighborLoss',0,0,0,3,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);

INSERT INTO EquipmentTypes VALUES (1,'General','general.gif');

INSERT INTO MonitorEquipmentState VALUES (0,'OK','green');
INSERT INTO MonitorEquipmentState VALUES (1,'Warning','cian');
INSERT INTO MonitorEquipmentState VALUES (2,'Minor','yellow');
INSERT INTO MonitorEquipmentState VALUES (3,'Major','orange');
INSERT INTO MonitorEquipmentState VALUES (4,'Critical','red');
```

9.3.3.2 user.sql

```
use mysql;
INSERT INTO user (Host,User,Password) VALUES('%','WBNMS',PASSWORD('WBNMS'));
INSERT INTO user (Host,User,Password) VALUES('localhost','WBNMS',PASSWORD('WBNMS'));
INSERT INTO db
(Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,Create_priv,Drop_priv,Grant_priv,References_priv
,Index_priv,Alter_priv) VALUES('%','WBNMSDB','WBNMS','Y','Y','Y','Y','Y','Y','N','Y','Y','Y');
```

10 Anexos

10.1 Gestión de redes de telecomunicaciones TMN

Se define TMN (Telecommunication Management Network) como un conjunto de capacidades que permiten el intercambio y procesamiento de información de gestión con el propósito de ayudar a los administradores de redes a realizar sus actividades con eficacia, para brindar mejores servicios en menos tiempo y con menor costo.
Desde un punto de vista conceptual un TMN es una red independiente que asegura la interfaz con una red de telecomunicaciones en diversos puntos para el envío/recepción de información hacia/desde la red de gestión y para el control de la red gestionada. Un TMN puede utilizar partes de la red de telecomunicaciones para realizar/facilitar sus comunicaciones. La Figura 10-1 presenta la relación general existente entre una red de gestión y una red de telecomunicaciones.

Relación general entre un TMN y una red de telecomunicaciones

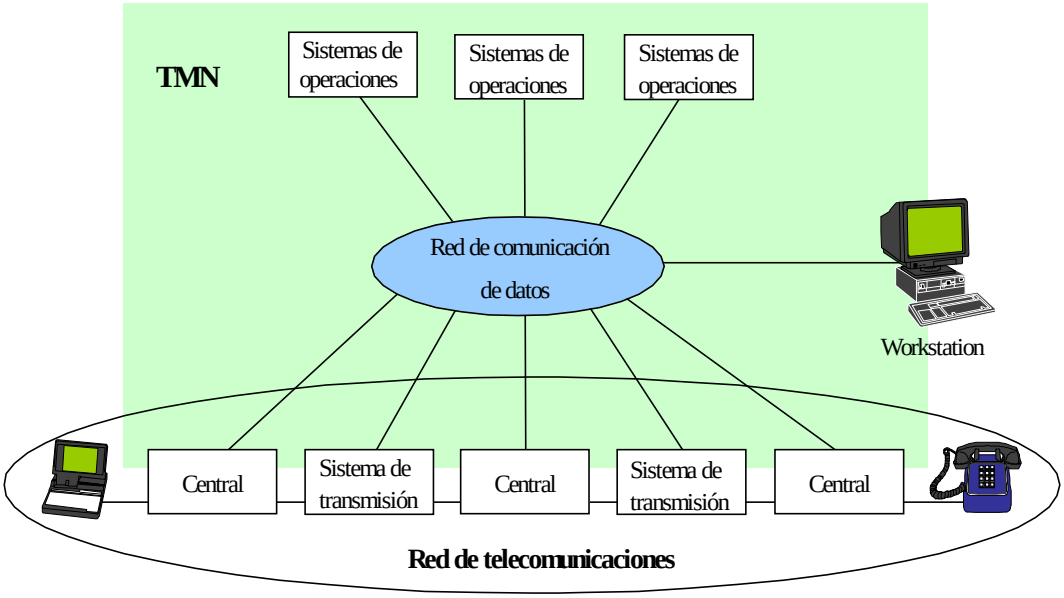


Figura 10-1

Para poder tener la capacidad de gestionar una amplia gama de equipos, redes y servicios distribuidos (integración), uno de los objetivos debe ser mantener la red de gestión lógicamente diferenciada de las redes y servicios que son gestionados.

10.1.1 Estructura en capas y áreas funcionales de un TMN.

La estructura del TMN posee forma de matriz, cuyas filas representan las Capas de Gestión y sus columnas las distintas áreas funcionales que abarcan dichas capas.

Capas de Gestión y Áreas Funcionales

Áreas Funcionales / Capas	Fallas	Configuración	Desempeño	Seguridad	Contabilidad	Otras
Gestión de Comercial (Negocios)						
Gestión de Servicios						
Gestión de Red						
Gestión de Elementos de Red						

Figura 10-2

La estructura esquematizada en la Figura 10-2 representa la interrelación entre todos los componentes del TMN.

El modelo que representa el funcionamiento de TMN se encuentra dividido en cuatro niveles formando una arquitectura lógica en capas. Cada una de estas capas tiene un objetivo determinado y, por lo tanto un conjunto de funciones asociadas.

La ***Capa de Gestión Comercial*** se asocia con el aumento de la rentabilidad a través del uso de herramientas informáticas. Sus funciones incluyen el presupuesto y planificación del negocio y la gestión de inventarios y recursos humanos.

La ***Capa de Gestión de Servicios*** se encarga de mejorar la provisión del servicio al mismo tiempo que reducir los costos operativos. Sus funciones son el registro de órdenes de servicio, la administración de abonados y la gestión de clientes.

La ***Capa de Gestión de Red*** debe asegurar la conectividad y la calidad de los vínculos entre los diversos elementos de red (E.R.). Por tanto la configuración de las conexiones en la red, el control de topología y el control global de la conmutación a líneas de reserva son sus funciones más relevantes.

La ***Capa de Gestión de Elementos de Red*** se relaciona con el monitoreo y mantenimiento de los componentes individuales. Incluyendo el control, configuración y localización de fallas en los E.R. entre las funciones más importantes que realiza.

10.1.2 Arquitectura de TMN

Al enfrentarse a un diseño de TMN se encuentran tres grandes tipos de arquitecturas:

- Arquitectura funcional
- Arquitectura de información
- Arquitectura física

10.1.2.1 Arquitectura funcional

Describe la distribución apropiada de funcionalidades dentro de TMN. Permite la creación de bloques de función y puntos de referencia para favorecer el intercambio de información entre los diversos bloques de gestión. Estos a su vez implican la estandarización de las funcionalidades de TMN, ya que un bloque físico puede abarcar varios bloques funcionales. De esta manera se logra independencia respecto de la arquitectura física. Algunos bloques de función se encuentran parcialmente dentro del TMN (ver Figura 10-3).

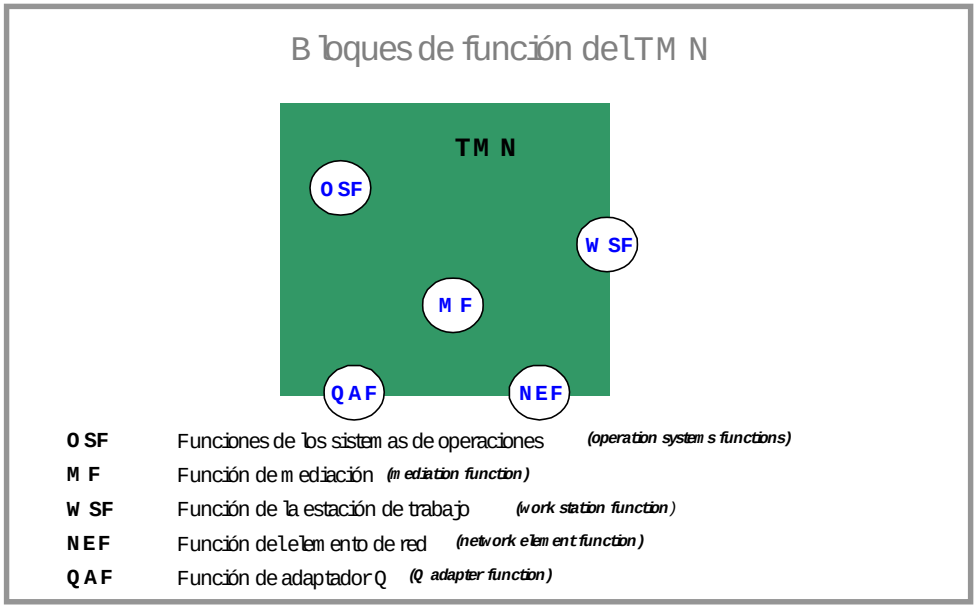


Figura 10-3

En el siguiente gráfico (Figura 10-4) se ejemplifica la relación de entre el nivel de abstracción de los datos, las capas de gestión y los bloques de función definidos en el TMN.

Ejemplo de jerarquía funcional de OSs

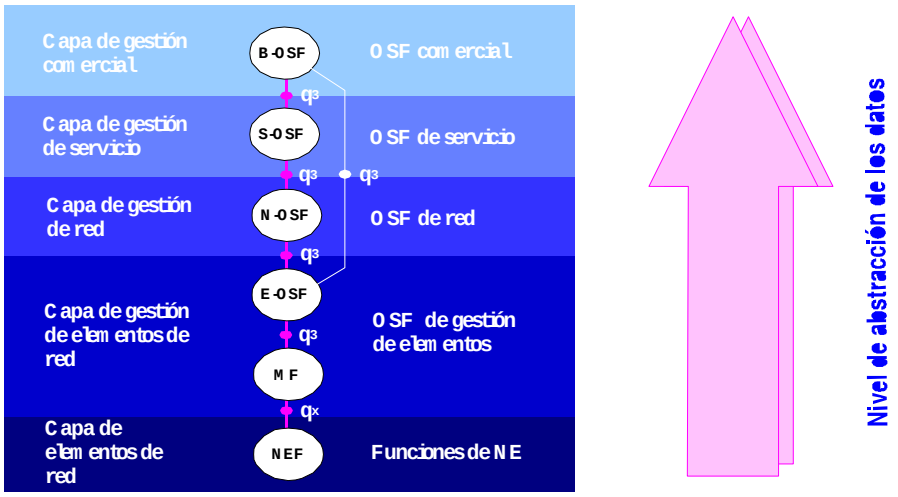


Figura 10-4

10.1.2.2 Arquitectura de información de TMN

En un sistema TMN la información es considerada desde dos puntos de vista.
El *Modelo de Información de Gestión* representa una abstracción de las funciones de gestión de telecomunicación provistas por los elementos de red y de las funciones de gestión del sistema de operaciones propiamente dicho (para la gestión de los ER). Este modelo determina qué información es posible intercambiar de manera normalizada (estandarizada). Debe tenerse en cuenta que es posible que cierta información del ER no sea intercambiada por no estar incluida en el modelo de información.
Por su parte el *Intercambio de Información de Gestión (pila de protocolos)* entre componentes físicos distantes (ER, sistema de operación, etc.) requiere el uso de una pila de protocolos, para asegurar que esta información llegue a destino.

El flujo de mensajes de gestión implica una interacción entre gestor, agente y objeto gestionados (entre ER, sistemas de operaciones, etc.), tal como lo indica la siguiente figura.

Interacción entre gestor, agente y objeto

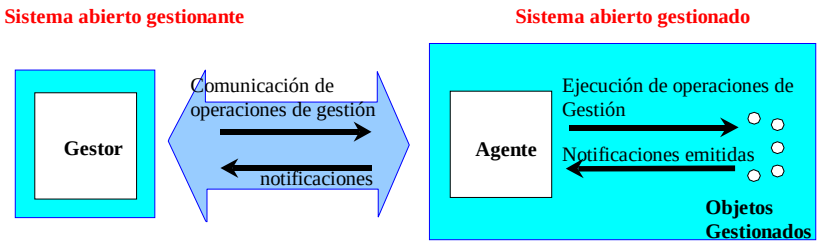


Figura 10-5

Cabe mencionar que cada uno de estos componentes puede contener distintos objetos gestionados. Estos últimos son visiones conceptuales o modelos sencillos de una realidad compleja. Se refieren, por un lado, a los recursos sometidos a gestión o funcionalidades de telecomunicaciones propias de equipos a ser gestionados y, por otro lado, a aquellos recursos que soportan funciones de gestión propiamente dichas o funcionalidades de gestión que no están relacionadas con los servicios de telecomunicaciones de los ER. Un ejemplo del primer caso sería el sincronismo o la configuración de un ER. Un ejemplo del segundo caso lo constituye el filtrado de alarmas y el registro de eventos en el ER.
Dado que el sistema de gestión es distribuido, la gestión de red es distribuida. Esto implica el intercambio de información de gestión entre procesos de gestión (gestor y agente) a fin de supervisar y controlar los diversos recursos de la red física y lógica. Dentro de este intercambio de información gestor y agente se definen por sus funciones. Un gestor emite directivas de operación de gestión y recibe notificaciones. Un agente gestiona a los "objetos gestionados" asociados a él, respondiendo a los pedidos del gestor. Asimismo el agente presenta al gestor el estado de estos "objetos gestionados" y emite notificaciones (reportes espontáneos) que reflejan cambios en el estado de los mismos (alarmas, eventos, etc.).
Para permitir la comunicación entre gestor y agente deben cumplirse algunas premisas. Ambos deben utilizar la misma pila de protocolos y manejar y conocer los distintos "objetos gestionados", es decir, reconocer las funciones de gestión soportadas. Por su parte, el gestor debe tener almacenada en su base de datos los nombres de todos los agentes y objetos gestionados que debe gestionar (red a gestionar).

10.1.2.3 Arquitectura física de TMN

La realización de funciones de TMN puede tener lugar en muy diversas configuraciones físicas. Por ejemplo se pueden atravesar diferentes topologías y plataformas de hardware. De esta manera, cada uno de los bloques funcionales se pueden agrupar o repartir en diversas plataformas físicas; cada bloque físico contiene/soporta "bloques de función" obligatorios y opcionales. A continuación, en la Figura 10-6, se detallan los bloques físicos del TMN y sus interfaces.

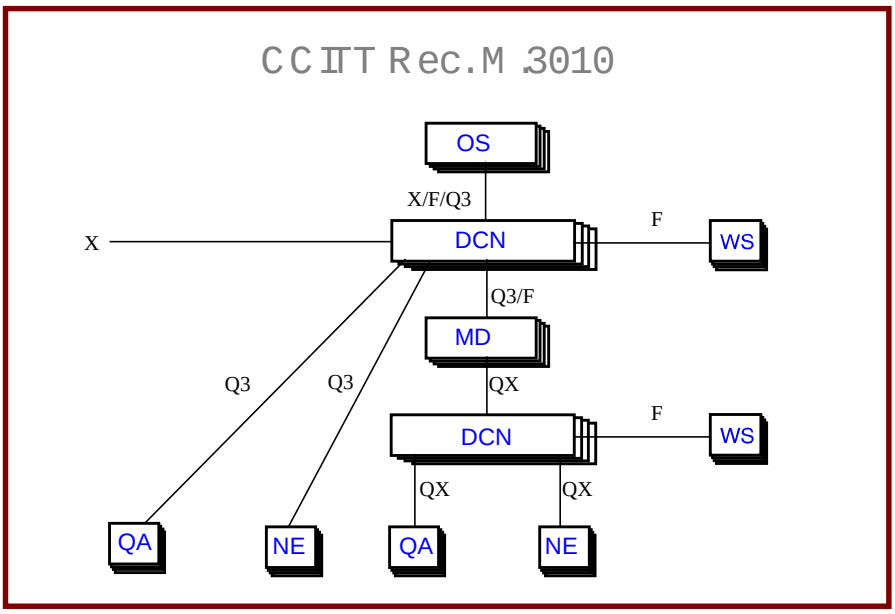


Figura 10-6

La arquitectura de TMN está compuesta por las siguientes plataformas físicas:

Sistemas de operaciones (OS): Sistema que ejecuta las funciones de sistema de operaciones(OSF) en forma obligatoria. La plataforma física del sistema de operaciones podrá proporcionar opcionalmente las funciones de mediación (MF), de adaptador Q (QAF) y de estación de trabajo (WSF).

Dispositivo de mediación (MD): Dispositivo que ejecuta las funciones de mediación (MF) en forma obligatoria. La plataforma física del dispositivo de mediación podrá proporcionar también opcionalmente las funciones de sistemas de operaciones (OSF), de adaptador Q (QAF) y de estaciones de trabajo (WSF).

Adaptador Q (QA): Dispositivo que permite conectar a TMN elementos de red o sistemas de operación con interfaces no compatibles con las del mismo.

Red de comunicaciones de datos (DCN): Red que provee la función de comunicaciones de datos para permitir las comunicaciones entre los distintos componentes del TMN. Esta red de comunicación de datos podrá constar de un cierto número de subredes interconectadas entre sí independientemente de distintos tipos de protocolos o topologías. Así por ejemplo, la red de comunicación de datos podrá contar con una subred troncal que proporcione conectividad entre las distintas subredes, y estas últimas, por su parte, proporcionar el acceso a la red de comunicaciones de datos.

Elemento de red (ER): Los elementos de red son los equipos de telecomunicación. Cabe mencionar que un elemento de red podrá contener opcionalmente cualquiera de los restantes bloques de función de TMN.

Estación de trabajo (WS): Sistema que ejecuta funciones del bloque de estación de trabajo (WSF) en forma obligatoria. Las funciones de la estación de trabajo traducen información situada en el punto de referencia *f* a un formato visual en el punto de referencia *g* y viceversa. No es ni más ni menos que la interfaz de usuario del TMN.

Interfaces: Para el intercambio de información de gestión entre bloques físicos de TMN, estos deberán estar interconectados a través de un vínculo de comunicación. Para asegurar esta comunicación ambos elementos deberán soportar la misma interfaz (tanto pila de protocolos como modelo de información). Para simplificar los problemas de comunicaciones que puede plantear una red multivendedores y multicapacidades (distintos tipos de ER) es útil valerse del concepto de interfaz inter-operable (interfaz estándar). Las interfaces inter-operables definen la pila de protocolos y los mensajes a ser transportados. Se basan en una visión orientada a objetos, por lo que todos los mensajes transportados se refieren a manipulaciones de objetos. Este tipo de interfaces están constituidas por un conjunto formalmente definido de protocolos, procedimientos (por ejemplo handshake), formatos de mensaje y semántica (contenido o significado del mensaje) utilizados para las comunicaciones de gestión. Los tipos de interfaces se pueden clasificar en:
F: Interfaz entre sistema de operaciones o dispositivos de mediación con estaciones de trabajo.
X: Interfaz entre sistemas de gestión.
Qx: Interfaz que contiene la porción del modelo de información compartida entre los dispositivos de mediación y los elementos de red.
Q3: Interfaz que contiene la porción del modelo de información compartida entre el sistema de operaciones y los componentes del TMN con los que interactúa la interfaz directamente.

10.2 Protocolo de Gestión SNMP.

10.2.1 Evolución de SNMP

El protocolo SNMP (Simple Network Management Protocol), creado en 1988, fue diseñado con el objeto de tener un sistema de gestión de routers, servers, workstations, y otros elementos presentes en una red de datos, que sea fácilmente implementable y con poco overhead. La especificación de SNMP:

- Define un protocolo para intercambio de información entre uno o más sistemas de gestión, y varios agentes.
- Define el marco de operación para el manejo de información de gestión
- Define variables de propósito general, llamadas objetos.

La versión original de SNMP, que ahora es conocida como SNMPv1, se ha convertido rápidamente en la herramienta de gestión más utilizada. Sin embargo, debido a su gran utilización el protocolo ha evidenciado sus deficiencias. Entre estas podemos incluir la falta de comunicación entre managers, la imposibilidad de

transmitir gran cantidad de datos, y la falta de seguridad. Todas estas desventajas fueron salvadas con la creación de SNMPv2 en 1993. Este último protocolo se encuentra definido como un conjunto de standards de Internet.

La solución propuesta en SNMPv2 para el problema de la seguridad resultó ser demasiado compleja. Esto condujo a los diseñadores a una nueva versión SNMPv3, la cual incluye pequeños cambios funcionales y una nueva forma de encarar el problema de la seguridad

10.2.2 Conceptos básicos

El modelo para el sistema de gestión utilizado por SNMP incluye los siguientes elementos:

- Estación de gestión
- Agente de gestión
- Base de información de gestión
- Protocolo de gestión de red

10.2.2.1 Estación de gestión

Típicamente una estación de gestión es un dispositivo que funciona en forma aislada, aunque puede ser implementada sobre un sistema de tipo compartido. En cualquier caso, la estación de gestión cumple la función de interfaz entre la persona que se encarga de realizar la gestión y el sistema de gestión de red. La estación de gestión debe estar compuesta por los siguientes ítems, como mínimo:

- Un conjunto de aplicación de gestión para el análisis de los datos, recuperación de fallas, etc.

- Una interfaz a través de la cual la persona encargada de la gestión de la red pueda monitorear y controlar la misma. Esto es, esta interfaz debe permitir al usuario realizar determinadas acciones (monitorear y controlar) las cuales son llevadas a cabo por la estación de gestión comunicándose con los elementos a gestionar dentro de la red.

- Un protocolo por medio del cual la estación de gestión y las entidades a gestionar puedan intercambiar información de gestión y control de la red.

- Una base de datos extraídos de las diferentes bases de datos de todas las entidades a gestionar dentro de la red. Esto significa que la estación de gestión debe contener al menos un resumen de la información de gestión incluida en cada elemento a gestionar dentro de la red.

Solo los últimos dos elementos son el objeto de el Standard que define al protocolo SNMP.

10.2.2.2 Agente de gestión

Tanto hosts, bridges, routers, o hubs pueden ser equipados con el software que implementa SNMP para ser gestionados a través de una estación de gestión. El agente de gestión responde a pedidos de información realizados desde la estación de gestión, y puede de forma asincrónica proveer a la estación de gestión de información importante que puede no haber sido solicitada.

10.2.2.3 Base de información de gestión

Los recursos dentro de una red son representados como objetos para poder ser gestionados. Cada objeto es, esencialmente, una variable que representa un aspecto de un sistema gestionado. El conjunto de objetos es denominado como base de información de gestión (MIB: Management Information Base). La MIB conforma los puntos de acceso a los que puede acceder la estación de gestión dentro del agente. El software que implementa el agente de gestión es el que contiene la MIB. Estos objetos están estandarizados de acuerdo a sistemas de una determinada clase. Esto significa por ejemplo que los todos bridges soportan el mismo conjunto de objetos, la misma MIB. Además extensiones propietarias pueden desarrollarse.

Estructura de Árbol de la MIB

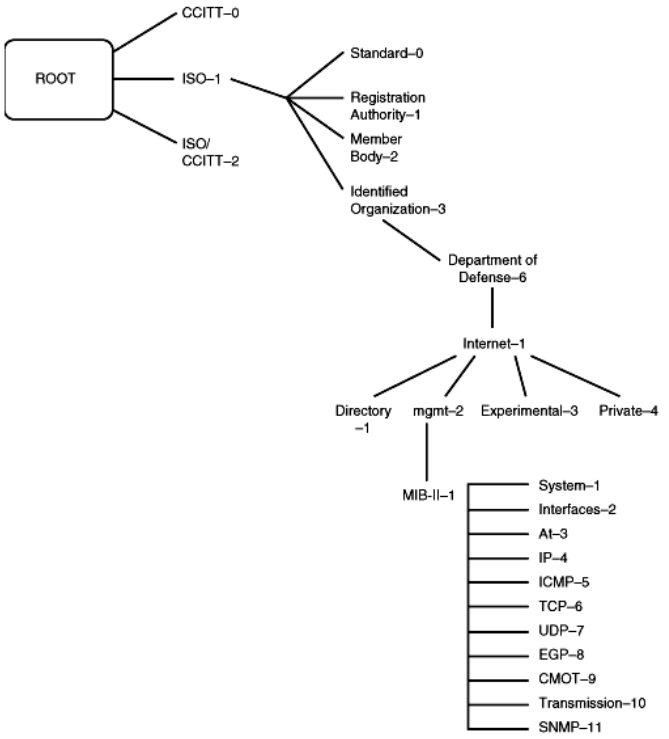


Figura 10-7

La estación de gestión realiza el monitoreo o control obteniendo o modificando el valor de un determinado objeto de la MIB contenido en el agente de gestión.

10.2.2.4 Protocolo de gestión

La estación de gestión está comunicada con el agente de gestión mediante un protocolo de gestión de red. El mismo incluye los siguientes comandos:

- **Get:** habilita a la estación de gestión a obtener el valor de un objeto del agente.
- **Set:** habilita a la estación de gestión a determinar el valor de un objeto del agente.
- **Trap:** habilita al agente a notificar a la estación de gestión de determinados eventos importantes.

No se encuentra escrito en los estándares el máximo número de estaciones de gestión o la relación de estaciones de gestión a agentes que debe cumplirse. En general, es prudente tener como mínimo dos sistemas capaces de realizar la función de la estación de gestión, de forma de obtener redundancia en caso de falla. Otro aspecto no especificado es que no se indica el número máximo de agentes que una estación de gestión puede gestionar.

10.2.2.4.1 Arquitectura del protocolo de gestión de redes.

SNMP fue diseñado para ser un protocolo a nivel de aplicación que forme parte del la pila de protocolos de TCP/IP. Para una estación de gestión que funcione en forma independiente, el proceso de gestión controla el acceso a la MIB central en la estación de gestión y provee la interfaz hacia el gestor de la red. El proceso de gestión logra la gestión de la red utilizando SNMP, el cual es implementado por encima de la siguiente pila de protocolos: UDP, IP, y el protocolo que se encarga de comunicar los datos a la interfaz física (Ethernet, FDDI, X25).

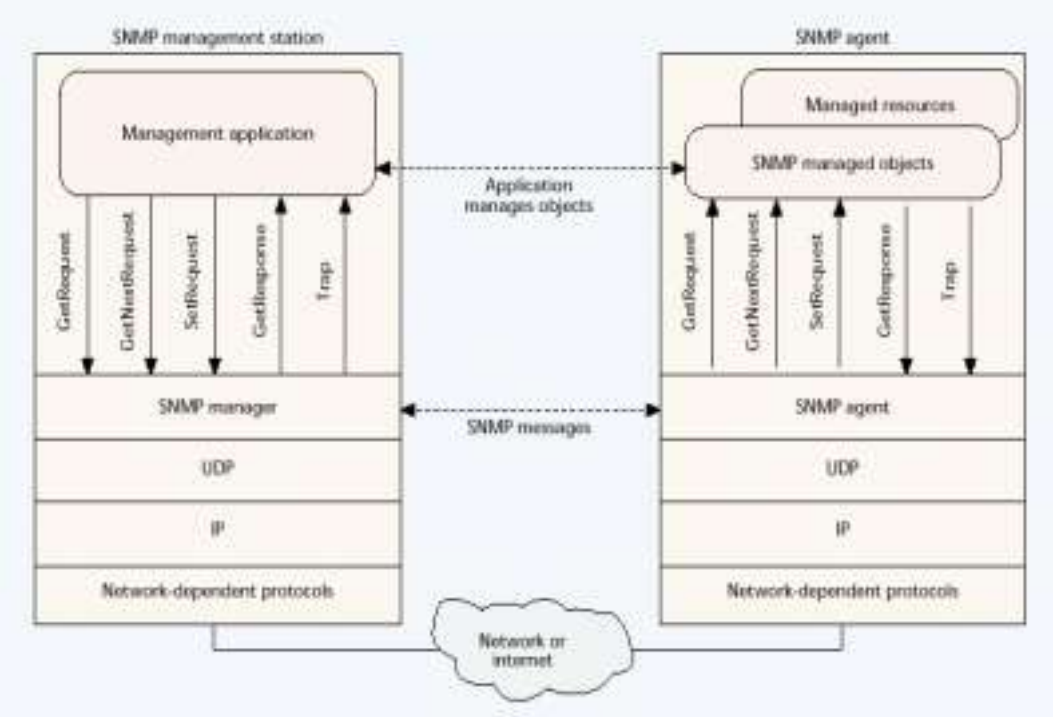


Figura 10-8

Esta misma pila de protocolos debe ser implementada del lado del agente de gestión, el cual, a su vez, necesita de un proceso que interprete los mensajes SNMP y controle remotamente el acceso a la MIB. Para los dispositivos que soporten otros protocolos como FTP, se requiere que trabajen tanto sobre TCP como sobre UDP.

Desde una estación de gestión, existen tres tipos de mensajes de tipos SNMP implementados para ser utilizados por una aplicación. Estos son: GetRequest, GetNextRequest, y SetRequest. Los primeros dos corresponden a una variación de la función Get. Los tres mensajes son respondidos por el agente por medio de un mensaje GetResponse, el cual es transportado hasta la aplicación utilizada. Además el agente debe implementar un mensaje de información denominado Trap, como respuesta a algún evento que afecte la MIB y los elementos gestionados.

SNMP asume que el protocolo UDP no sufrirá problemas de transporte de los datos. UDP es un protocolo no orientado a conexión, de forma que SNMP no está orientado a conexión. Esto es, no existen conexiones establecidas entre la estación de gestión y el agente de gestión. Sin embargo, cada mensaje transmitido es una transferencia independiente entre la estación de gestión y el agente de gestión.

10.2.3 Envío de información forzado

Si la estación de gestión es responsable por una gran cantidad de agentes de gestión, y si cada agente tiene asignado una gran cantidad de objetos, se convierte irrealizable que la estación de gestión realice un pedido de información a todos los agentes, de todos los objetos. Entonces, se recomienda que se utilizar el siguiente método. Cuando se produce la inicialización de la estación de gestión, y con períodos de un día, por ejemplo, la estación de gestión puede realizar el pedido de

los objetos que poseen información valiosa para gestión y control de una red. Entre ellos podemos nombrar a el promedio del número de paquetes enviados y recibidos sobre cada interfaz durante un período de tiempo, entre otros. De esta forma cada agente de gestión es responsable de notificar de eventos inusuales por medio de traps. Estos eventos pueden ser, funcionamiento incorrecto del agente de gestión, reinicialización del dispositivo gestionado, falla en un enlace, etc.

Este método origina que la red de transmisión no se vea inundada por información de gestión que puede no ser de relevancia. Las redes no fueron diseñadas para transmitir información de gestión que la estación de gestión considere de baja prioridad.

10.2.4 Proxies.

El uso de SNMP requiere que todos los agentes, así como las estaciones de gestión, deben soportar los protocolos UDP e IP. Este hecho limita la gestión a ciertos dispositivos y excluye de la misma a otros, como bridges o módems que no soportan los protocolos mencionados. Más aún, existen pequeños sistemas, como computadoras personales, estaciones de trabajo o controladores programables que no tienen implementados los protocolos presentes en TCP/IP, que requiere de ser gestionados y que no es deseable que sean modificados incorporándoles la lógica de los agentes de gestión y las MIBs.

Para gestionar dispositivos que no implementan SNMP, se ha desarrollado el concepto de Proxy. En este marco, un agente de gestión SNMP actúa como un Proxy para uno o más dispositivos.

La estación de gestión envía pedidos de información hacia el agente Proxy con relación a algún dispositivo. El agente Proxy convierte cada pedido de información en el protocolo de gestión utilizado por el dispositivo en cuestión. Cuando el agente Proxy recibe una respuesta a un pedido de información, este envía la información hacia la estación de gestión de forma análoga. De la misma forma, si una notificación se genera en un dispositivo, el agente Proxy deberá convertirla y enviarla a la estación de gestión utilizando un mensaje de tipo trap.

10.2.5 Descripción de campos del mensaje SNMP

Utilizando SNMPv1 la información es intercambiada entre la estación de gestión y el agente de gestión en forma de mensaje. Cada mensaje de SNMPv1 incluye un número, que indica la versión del protocolo SNMP utilizado, un nombre denominado comunidad y una de cinco unidades de datos de protocolo (PDU). En la siguiente tabla se muestran los diferentes campos del mensaje. Cabe destacar que los mensajes GetRequest, GetNextRequest y SetRequest tienen el mismo formato que el mensaje de GetResponse, con el estatus de error y el índice de error siempre seteados a "0". Esta convención reduce en uno la cantidad de formatos de PDU con los que SNMP debe trabajar.

Principales RFCs que definen SNMPv1

RFC	Título	Fecha
1155	Estructura e identificación de la información de gestión basada en redes TCP/IP	Mayo 1990
1157	SNMP	Mayo 1990
1212	Definición detallada de MIBs	Marzo 1991
1213	Definición de MIB utilizadas sobre dispositivos que funcionen en redes de tipo TCP/IP (MIB II)	Marzo 1991

Tabla 10-1

Los PDUs GetRequest y GetNextRequest son comandos destinados a que el encargado de la gestión obtenga información desde un agente de gestión. La diferencia entre ambos radica, en que GetRequest lista la variable o las variables de las que se quiere obtener información, en cambio GetNextRequest es utilizada para recorrer una MIB estructurada en forma de árbol. En ambos casos, los valores, si están disponibles son respondidos en un PDU de tipo GetResponse. El comando Set se origina en el encargado de la gestión y tiene como objetivo actualizar el valor de variables en el agente. Finalmente, el PDU Trap es encargado de transportar las notificaciones originadas en el agente de gestión.

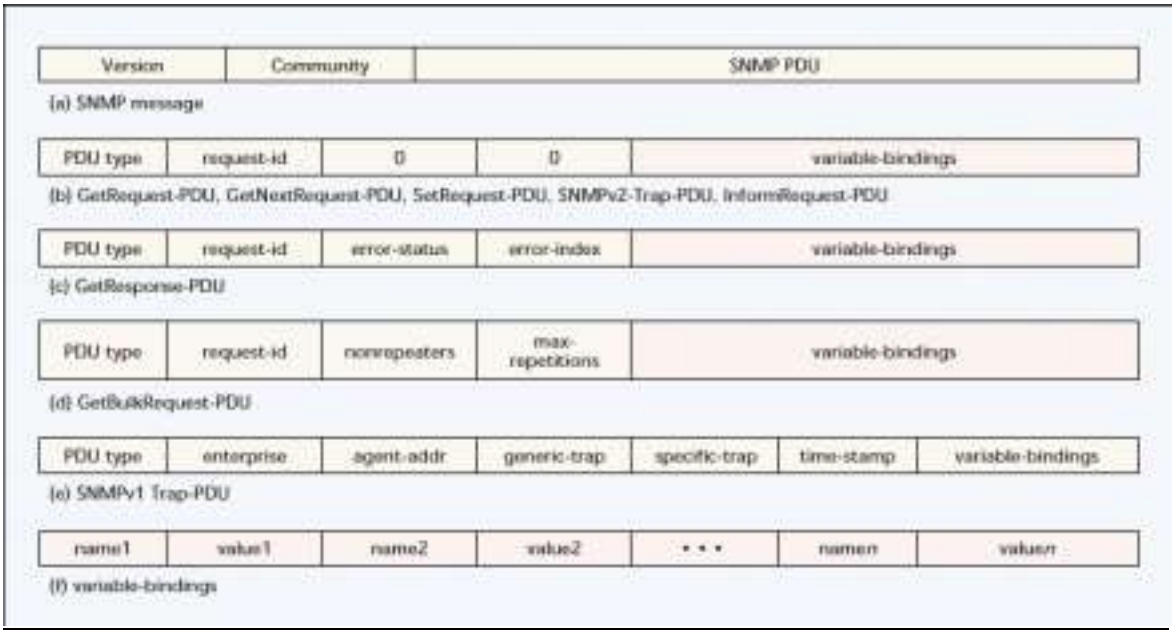


Tabla 10-2

Campo	Descripción
Version	Verisón de SNMP; RFC 1157 es verción 1
Comunidad	Conjunto formado por un agente SNMP y un conjunto de entidades de aplicación de SNMP. El nombre de la comunidad funciona como la contraseña para la autenticación del mensaje de SNMP
Identificación de solicitud	Es utilizado para distinguir entre diferentes solicitudes realizadas.
Estado de error	Utilizado para indicar que ha ocurrido un error procesando una solicitud.
Índice de error	Cuando el estado de error es diferente de cero, este campo indica la variable que produjo el mismo. Una variable es una instancia de un objeto gestionado.
Lista de variables	Lista de nombres de variables y sus correspondientes valores.
Empresa	Basado en sysObjectID.
Dirección del agente	Dirección objeto generador del trap
Trap generico	Tipo genérico de trap.
Trap específico	Codificaicón del trap.
Tiempo	Tiempo transcurrido entre la última inicialización del dispositivo de red y la generación del trap.(sysUpTime)
No repetidores	Indica cuántas variables deben devolver solo un valor cada una.
Máximo-repeticiones	Indica el número de variables que debe ser devuelto por las variables restantes.

Tabla 10-3

10.2.6 Transmisión de un mensaje SNMP

En principio, un dispositivo SNMP realiza las siguientes acciones para transmitir uno de los cinco tipos de PDU a otro dispositivo SNMP:

- Se construye el PDU.
- El PDU es sometido a autenticación. Ésta es realizada utilizando las direcciones IP destino y origen, el nombre de la comunidad. El servicio de autenticación realiza la operación necesaria, que puede ser encriptado o la inclusión de un código de autenticación. El nombre de la comunidad determina el contexto en que se realiza este proceso.
- Se construye el mensaje, con los siguientes campos: la versión, el nombre de la comunidad, y el resultado del paso anterior.
- Luego se invoca al servicio de transporte.

Cabe destacar que generalmente la autenticación no es realizada.

10.2.7 Recepción de un mensaje SNMP

En la recepción de un mensaje SNMP el dispositivo SNMP realiza las siguientes acciones.

- Se realiza un chequeo de la sintaxis del mensaje, y se descarta el mensaje en caso de que sea errónea.

- Se verifica la versión.

- Luego, luego se pasa al servicio de autenticación, el nombre del usuario, la porción del mensaje que corresponde al PDU, y las direcciones IP origen y destino. En este punto pueden ocurrir dos cosas: que la autenticación falle, en cuyo caso se envía un trap informando de este hecho y el mensaje es ignorado; que la autenticación sea exitosa, en cuyo caso el servicio de autenticación responde con el PDU.

- El protocolo realiza un chequeo de sintaxis sobre el PDU y lo ignora en caso de que existan errores. En caso contrario, por medio de la comunidad se elige una determinada política con la que se procesa el PDU.

Todas las operaciones SNMP involucran el acceso a objetos de tipo escalares. Sin embargo, es posible agrupar un número de operaciones de un mismo tipo de comando (Get, Set o Trap), en un único mensaje. De esta forma, si la estación de gestión quiere obtener un grupo de valores de un determinado agente de gestión, con un único mensaje obtiene todos los valores definidos en el grupo. Esta técnica reduce el tráfico que contiene información de gestión en las redes de comunicaciones.

Para implementar el intercambio de muchos objetos, todos los PDU de SNMP incluyen un campo llamado grupo de variables. Este campo consiste de una secuencia de referencias a instancias de objetos, acompañadas con los valores de los objetos. Algunos PDU son ocupados por el nombre de una instancia de un objeto, como es el caso de los mensajes Get. En este caso, el campo grupo de objetos es ignorado por el dispositivo que recibe el mensaje.

10.3Perl.

10.3.1 Descripción.

Es un Lenguaje de alto nivel escrito originalmente por ***Larry Wall*** y soportado actualmente por miles de desarrolladores en todo el mundo. La semántica de este lenguaje esta basada en gran medida en el lenguaje de programación ***C,*** incorporando muchas de las mejores características del ***sed*** ,***awk*** , ***the Unix Shell*** y de al menos una docena de herramientas y lenguajes.
Sin querer encasillar al Perl en una función específica, se podría decir que es especialmente fuerte con procesos, archivos y manipulación de texto. Esto lo hace especialmente útil para utilidades del sistema, herramientas de software, tareas de mantenimiento del sistema, acceso a base de datos, programación gráfica, networking, y programación Web.
El Perl esta firmemente plantado sobre plataformas Unix, pero se ha convertido en una herramienta de desarrollo multiplataforma. El Perl es soportado en IBM mainframes; AS/400s; Windows NT, 95, y 98; OS/2; Novell Netware; HP MPE/ix; Mac OS; y todas las distribuciones Unix, incluido Linux.
Larry Wall es un ferviente promotor de software libre y el Perl no es la exención. Perl , incluido el código fuente, las librerías estándar de Perl, los módulos opcionales, y toda la documentación, es provista libremente y soportada enteramente por una gran comunidad de usuarios.
Existe un gran controversia respecto al significado de "Perl" pero la versión mas aceptada es ***"Practical Extraction and Reporting Language"*** aunque existe también una versión muy difundida "***Pathologically Eclectic Rubbish Lister".***

10.3.2 Historia del Perl

El Perl es relativamente un lenguaje viejo, con una primera versión halla por el 1988. En la siguiente tabla se detalla el camino seguido hasta hoy.

Versión	Fecha	Detalles de la versión
Perl 0		Primera presentación del Perl a los compañeros de oficina de Larry Wall.
Perl 1	**Enero de 1988**	El Perl es presentado al mundo.
Perl 2	**Junio de 1988**	Es Introducido el paquete para expresiones regulares de Harry Spencer.
Perl 3	**Octubre de 1989**	Es introducida la habilidad de manejar datos binarios.
Perl 4	**Marzo de 1991**	Es Introducido el primer Libro "Camel" *(Programming Perl*, por Larry Wall, Tom Christiansen, y

		Randal L Schwartz; O'Reilly & Associates). Se publica con el nombre Perl 4 En lugar de Perl 3.
Perl 4.036	**Febrero de 1993**	La última versión estable del Perl 4.
Perl 5	**Octubre de 1994**	La primer versión estable del Perl 5.Esta introduce un gran numero de características.
Perl 5.005_02	**Agosto de 1998**	Otra versión estable.
Perl 5.005_03	**Marzo de 1999**	La última versión estable antes de la 5.6.
Perl 5.6	**Marzo del 2000**	Introduce el soporte de **fork,** mejor threading, un mejorado compilador de Perl y la palabra reservada **our.**
Perl 5.6.1	**Abril del 2001**	Versión estable.
Perl 5.8.0	**Julio del 2002**	Última versión estable.

Tabla 10-4

10.3.3 Características mas Importantes

- **El Perl es Libre (Gratis):** Puede que a primera vista no sea reconocida como una característica importante, pero de hecho es una de las principales. Ya que esto ha posibilitado la expansión del lenguaje y su constante mejoramiento dentro de las comunidades de desarrollo de la Internet. Es mas, esto posibilita que puedan ser "fácilmente" modificadas las funciones base (core) al ser accesible el código fuente del Perl.

- **Es Simple de Aprender, Conciso y fácil de leer:** Cualquier persona con algo de experiencia de programación es capaz de aprender fácilmente este Lenguaje, debido a que integra las mejores características de otros lenguajes muy difundidos.

- **El Perl es Rápido:** El Perl no es un interprete en el sentido estricto; cuando se ejecuta un programa en Perl, este es compilado dentro de un lenguaje altamente optimizado antes de su ejecución.

- **El Perl es Extensible:** Es posible escribir paquetes y módulos que extiendan las funcionalidades de lenguaje. Es posible también, llamar un código C externo desde el Perl. La acción reversa es también posible: El interprete de Perl puede ser incorporado directamente en muchos lenguajes, incluido el C. Esto permite que un programa en C pueda llamar funciones del interprete Perl sin llamar a un programa externo.

- **El Perl tiene Tipos de Datos Flexibles:** Se pueden crear variables simples con texto o números , y luego es el Perl el que se ocupa de reconocer el tipo para ser procesado (A diferencia del C). Es posible concatenar strings sin la necesidad de llamar a una función externa. Se pueden además, tratar vectores y valores como listas simples, como típicos vectores indexados e inclusive como staks de información. Es posible también crear vectores asociativos (hashes).Finalmente, el Perl soporta referencias y referencias a objetos. Las referencias permiten crear estructuras de datos complejas adoptando una combinación de hashes, listas y escalares.

- **El Perl es orientado a Objetos:** El Perl soporta todas las características de orientación a objetos (herencia, polimorfismo y encapsulación). No hay limite de cuando y donde se puede hacer uso de estas características.

- **El Perl es colaborativo:** Existe un una gran red mundial de programadores de Perl. Muchos de ellos proveen y usan los módulos y scripts disponibles en CPAN (Comprehensive Perl Archive Network).este es un repositorio de los mejores módulos y scripts disponibles. Con la utilización de los módulos es posible ahorrar horas de programación.

10.3.4 Compilador o Interprete

Existen lenguajes compilados y lenguajes interpretados:

- Un programa en un lenguaje compilado es traducido del código fuente original a código de maquina dependiente de la plataforma. Este código de máquina es referido como un ejecutable. No hay una relación directa entre el código de máquina y el código fuente original, por lo tanto no es posible un proceso de compilación inverso para obtener nuevamente el código fuente a partir del código de máquina. Como consecuencia el ejecutable compilado mantiene a salvo la propiedad intelectual.

- Con un lenguaje interpretado, el interprete lee el codito fuente original e interpreta cada una de las sentencias de forma de realizar las distintas operaciones. Por consiguiente el código fuente es ejecutado en run time.

    - Esto tiene algunas ventajas:

        - Debido a que no existe un proceso de compilación, el desarrollo de código debería ser significativamente mucho mas rápido.
        - El código interpretado tiende a ser mas pequeño y fácil de distribuir.

    - Y Desventajas:

- El código original debe ser provisto para ejecutar un programa.
- Un programa interpretado es generalmente mas lento que un ejecutable compilado, dada la forma de ejecución.

El Perl no encaja exactamente en ninguna de estas descripciones. El funcionamiento interno del Perl es tal que en el tiempo de ejecución del script de Perl, los elementos individuales del script son compilados dentro de un árbol de ***opcodes***. Los Opcodes son similares en concepto al código de máquina. Sin embargo, mientras que el código de máquina es ejecutado directamente por hardware, los opcodes son ejecutados por una Máquina Virtual de Perl (PVM). Los opcodes son objetos altamente optimizados diseñados para realizar funciones específicas.
Cuando un script es ejecutado, se esta ejecutando esencialmente código C compilado, traducción del código fuente en Perl. Esto habilita al Perl a proveer todas las ventajas del un lenguaje de scripts ("lenguaje interpretado") mientras ofrece la ejecución rápida de los programas compilados. Este modo de operación, traducción y luego ejecución por una maquina virtual es como actualmente trabajan los modernos lenguajes de scripts, incluyendo Java y Python .

10.4 Análisis, Diseño e Implementación de Sistemas

10.4.1 Análisis

El Propósito del análisis es establecer y entender el problema y el dominio de la aplicación para que pueda ser diseñada una solución correcta. Un buen análisis captura las características esenciales del problema sin introducir prematuramente restricciones en las decisiones de diseño.

10.4.1.1 Declaración del Problema

Primero es necesario escribir una declaración inicial del problema, en consulta con los interesados (clientes, usuarios, expertos en el dominio del problema). Los requerimientos deben describir que necesita ser hecho y **NO** como será implementado. La declaración inicial del problema podrá ser incompleta, ambigua y errónea ya que es solo el punto de partida y ser completada y corregida durante todo el proceso de Análisis.

10.4.1.2 Modelado de Objetos

El modelo de objetos deberá mostrar la estructura estática del dominio del Problema. Primero se identificara las clases de objetos que intervienen, luego sus relaciones incluyendo todo tipo de agregación, definido por lo distintos niveles detalle según sea preponderante para el análisis. Luego se deberán reconocer los atributos y relaciones entre clases de objetos y quedarse con los que se consideren importantes. El considerar la utilización del concepto de herencia podría simplificar el modelo. El ultimo paso es agrupar en módulos las clases de objetos relacionados y describir cada una de las entidades resultantes.

10.4.1.3 Modelado Dinámico

El modelo dinámico muestra el comportamiento del sistema , esencialmente secuencias de interacciones. Para la concepción de este modelo, habrá que preparar escenarios de sesiones típicas y excepcionales. Entonces se identificara eventos externos entre el sistema y el mundo exterior. Se construirán diagrama de estados representando el comportamiento del sistema ante los eventos externos.

10.4.1.4 Modelado funcional

El modelo funcional muestra las derivaciones funcionales de los valores significativos del sistema sin tener en cuenta cuando fueron computados. En la construcción de este modelo, primero deberán identificarse los valores de entrada y salida del sistema como parámetros de los eventos externos. Luego se construirá

los diagramas de flujo mostrando el cálculo de cada valor de salida en relación con otros valores del sistema.

10.4.1.5 Conclusión del Análisis

Aunque todos los modelos anteriormente descriptos sirven para una definición mas completa del problema que lleva al mejor entendimiento del mismo, no todos los problemas son iguales. Por lo tanto la forma de encararlos para su análisis puede ser variada, dependiendo de muchos factores, como la experiencia, el tiempo, los costos, etc.
Sin embargo esta metodología de análisis constituye una muy buena forma de encarar problemas complejos que justifique el esfuerzo. Claro esta, que una versión reducida de esta metodología sea la opción a tomar.
En todo caso, el resultado de la fase de Análisis deberá ser la definición adecuada del problema para encarar la próxima fase, el Diseño.

10.4.2 Diseño

Luego de la fase de Análisis y antes de empezar con un diseño detallado, el desarrollador del sistema deberá realizar una aproximación a la solución. Para ello deberá partir del modelo de estructuras de alto nivel del sistema e incluso partirlas en subsistemas. Luego deberá identificar todos los recursos de software y hardware requeridos por cada uno de ellos y su interacción. Esto es definir la arquitectura del Sistema.
Para facilitar el diseño del sistema se podrá dividir en capas horizontales y particiones verticales. Donde cada capa definirá un nivel de abstracción completamente diferente a otra capa. De esta forma cada capa brindara un servicio a capas superiores y recibirá el servicio de capas inferiores (al igual que el modelo OSI).
Cada subsistema deberá cumplir con funciones específicas que en conjunto atenderán a cumplir con los requisitos especificados en la etapa de Análisis. La fase de Diseño deberá documentarse de manera de ser una guía que puedan leer y entender los que construyan el código y los que prueban y mantienen el Software.
El Diseño debe proporcionar una completa idea de lo que es el Software, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista de la Implementación.
Para evaluar la calidad de una presentación del diseño, se deben establecer criterios técnicos para un buen diseño, como por ejemplo son:

- Un diseño debe presentar una organización jerárquica que haga un uso inteligente del control entre los componentes del software.
- El diseño debe ser modular, es decir, se debe hacer una partición lógica del Software en elementos que realicen funciones y subfusiones especificas.
- Un diseño debe contener abstracciones de datos y procedimientos.

- Debe producir módulos que presenten características de funcionamiento independiente.

- Debe conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.

- Debe producir un diseño usando un método que pudiera repetirse según la información obtenida durante el análisis de requisitos de Software.

Estos criterios no se consiguen por casualidad. El proceso de Diseño del Software exige buena calidad a través de la aplicación de principios fundamentales de Diseño, Metodología sistemática y una revisión exhaustiva.

10.4.3 Implementación

La próxima fase luego del análisis y el diseño de una solución es naturalmente su implementación. ¿Que abarca esta fase? Bueno, simplemente durante esta fase se deberán hacer realidad el diseño planteado con los recursos estipulados para cubrir los requisitos del problema. Como es natural suponer, esto es difícil. Pero el planeamiento metódico y la revisión exhaustiva durante las fases anteriores darán la medida del éxito en esta empresa.
Por lo general se considera al desarrollo de software como un ciclo de ***Demming***[9] en el que se debe realizar un continuo ir y venir de revisiones y mejoras mientras dure el ciclo de vida del producto. Esto es muy sano y hasta deseable, ya que da trabajo a mucha gente. Y partiendo de hecho de que el éxito inmediato y completo para alcanzar metas es humanamente imposible. Pero esto no significa, repito, no significa que lanzar al mercado un producto de software que no es funcional sea válido, como nos tiene acostumbrados la grandes compañías de desarrollo de Software.

[9] Agregar rerferencia de Demming.

10.5 ASN.1_Abstract Syntax Notation One

El ASN.1 es un lenguaje desarrollado y estandarizado por la CCITT (Internatinal Consultative Committe on Telegraphy and Telephony) e ISO (International Organization for Standarization). Este es un lenguaje utilizado para definir sintaxis abstracta para datos de aplicación.
El ASN.1 es usado para definir estructuras de aplicación y presentación las unidades de datos de protocolo (Protocol Data Units - PDUs).Tambien el ASN.1 es usado para definir las base de datos de información de gestión (MIBs) para tanto sistemas de gestión SNMP como sistemas de gestión OSI.
La siguiente tabla muestra algunos de las principales definiciones relevantes al tratamiento del ASN.1.

Tabla de Términos Relevantes en ASN.1

Sintaxis Abstracta	Describe la estructura genérica de datos, independientemente de cualquier técnica utilizada para representar los datos. La sintaxis permite que los tipos de datos sean definidos y que sus posibles valores especificados.
Tipos de Datos	Un conjunto definido de datos. Un tipo puede ser simple, en cuyo caso es definido por el conjunto de sus valores, o estructurado, en cuyo caso es definido en termino de otros tipos.
Codificación	La secuencia completa de octetos usada para representar un dato.
Reglas de Codificación	Una especificación de mapeo entre una sintaxis especifica y alguna otra. Específicamente, las reglas de codificación determinan algorítmicamente, para cualquier grupo de datos definido en la sintaxis abstracta, la representación de esos valores en la sintaxis de transferencia.
Sintaxis de Transferencia	Es la manera en la que los datos son representados en términos de patrones de bits mientras se realiza la transferencia entre entidades.

Tabla 10-5

10.5.1 Sintaxis Abstracta

Se pueden considerar dos grandes componentes en una comunicación entre sistemas finales (end to end).

- El **componente de Transferencia de Datos** es considerada como el mecanismo encargado de transferir los datos entre los sistemas finales. En el caso de la suite TCP/IP este componente consistiría de los protocolos TCP o UDP.

- El **componente de Aplicación** es el usuario del componente de transferencia de datos y esta compuesto de las aplicaciones de los usuarios finales de la aplicación. En el caso de la suite TCP/IP consistiría de aplicaciones como SNMP, FTP, SMTP, Telnet , etc.

Cuando cruzamos la frontera entre la aplicación y el componente de transferencia de datos, hay un cambio significante en la manera en la cual los datos son presentados. Para el componente de transferencia de datos, los datos recibidos desde la aplicación son especificados como valores binarios de secuencias de octetos. Estos valores binarios pueden ser directamente ensamblados en unidades de datos de servicio (SDUs) para ser pasados entre capas y ensamblados dentro de unidades de datos de protocolo (PDUs) para ser pasados entre entidades en la misma capa. El Componente de Aplicación, por otro lado se encarga de cómo los usuarios finales perciben los datos. En general, el usuario esta principalmente preocupado por la semántica de los datos. Entonces el Componente de Aplicación deberá proveer una presentación de datos que pueda ser convertida a valores binarios. Esto significa que deberá preocuparse por la sintaxis de los datos.
En el siguiente gráfico se ilustra este funcionamiento.

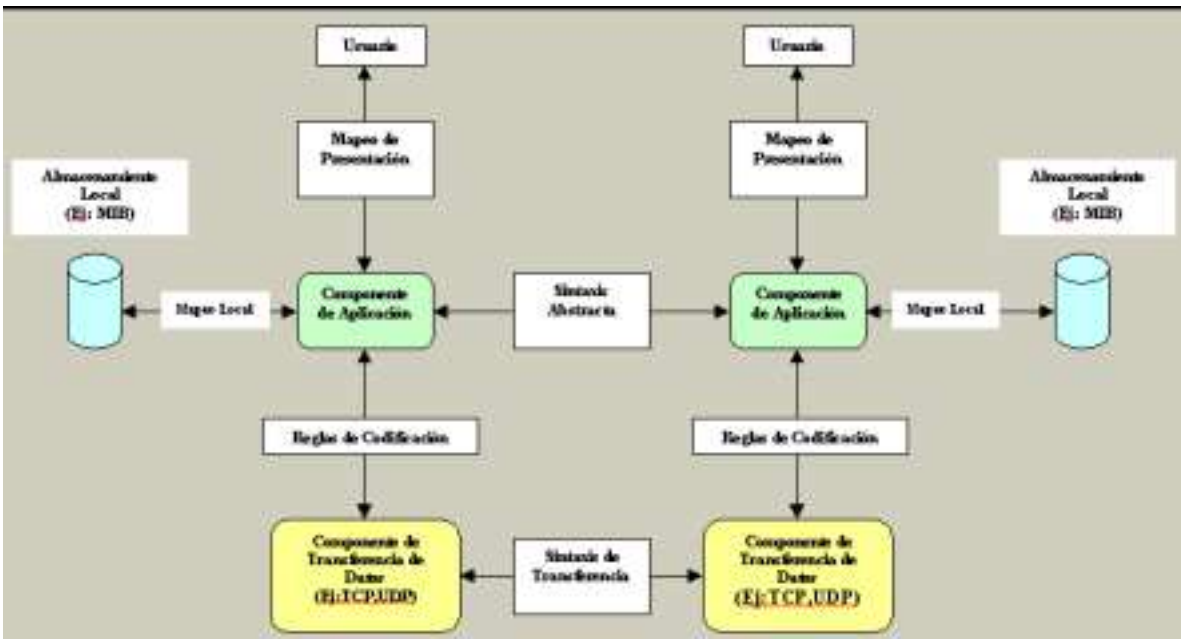


Figura 10-9

Para la componente de Aplicación , la información es representada en una sintaxis abstracta que esta de acuerdo con los tipos y valores de los datos. La sintaxis Abstracta formalmente especifica los datos independientemente de cualquier otra representación.

La sintaxis Abstracta es usada para intercambiar información entre componentes de Aplicación de diferentes sistemas. Este intercambio consiste de PDUs.
Dentro de un sistema, la información representada usando sintaxis abstracta debe mapearse dentro de algún formato para la presentación al usuario. Similarmente, esta sintaxis abstracta debe mapearse dentro de alguna forma de almacenamiento local.
En el caso de MIB como almacenamiento local este mapeo es transparente ya que las estructura de datos dentro de la MIB esta compuesta por la misma sintaxis abstracta.
La componente de Aplicación debe también traducir entre la sintaxis abstracta de la aplicación y la sintaxis de transferencia que describe los valores de datos con formato binario, adecuándolo para la interacción con la componente de transferencia de datos.

10.5.2 Conceptos de ASN.1

10.5.2.1 Definición de Módulos

El ASN.1 es un lenguaje que puede ser utilizado para definir estructura de datos.
Una definición de estructura estará dada, en ASN.1, por Módulos referenciados por nombre.
Un Módulo tiene la siguiente forma básica:

<*referenciademodulo*> DEFINITIONS ::=
BEGIN
EXPORTS
IMPORTS
AssignmentList
END

La **<*referenciademodulo*>** es el nombre del módulo seguido opcionalmente por un **object identifier** (identificador de objeto) para identificar al módulo.
La construcción **EXPORTS** indica que definiciones en este módulo puede ser importado por otros módulos.
La construcción **IMPORTS** indica que definiciones de tipos y de valores pueden ser importados de otros módulos en este módulo.
La lista de asignación consiste de tipos de asignación, valores de asignación y definición de macros (explicado mas adelante). Los tipos y valores de asignaciones tienen la siguiente forma:
<nombre> *::=* ***<descripción>***

10.5.2.2 Convenciones de Notación

- El Layout no es importante; múltiples espacios y líneas en blanco pueden ser consideradas como espacios simples.

- Los comentarios están delimitados por pares de guiones medio (--) al comienzo del comentario.

- Los identificadores (nombres de valores y campos), tipos de referencia (nombre de los tipos), y nombre de módulos pueden consistir de mayúsculas, minúsculas, dígitos y guiones.

- Un identificador debe comenzar con minúscula.

- Una referencia de tipo o nombre de módulo comienza con mayúscula.

- Un tipo propio de la notación debe estar totalmente con mayúsculas (Ej: BEGIN).

10.5.2.3 Tipos de Datos Abstracto

Los tipos se pueden clasificar en cuatro categorías:

1. **Simple:** estos tipos no poseen componentes (INTEGER, BOOLEAN, BIT STRING, etc.).
2. **Estructurado:** estos tipos poseen componentes (SEQUENCE, SEQUENCE OF, SET, SET OF).

3. **Etiquetados(Tagged):** estos tipos derivan de otros componentes.

4. **Otros:** Incluye los tipos **CHOICE** y **ANY.**

Todo tipo de datos en ASN.1. con excepción de **CHOICE** y **ANY** posee una etiqueta asociada. Esta etiqueta cosiste en un nombre de clase y un entero no negativo como número de etiqueta. Hay cuatro clases de tipos de datos, o cuatro clases de etiquetas:

1. **UNIVERSAL:** Generalmente útil, tipo independiente de la aplicación y mecanismo de construcción.

2. **APPLICATION:** Es relevante para aplicaciones particulares.

3. **Context-specific:** También relevante solamente a ciertas aplicaciones particulares, pero aplicado en un contexto limitado.

4. **Private:** tipos definidos por el usuario.

10.5.3 Definición de Macros en ASN.1

Incluido en la especificación de ASN.1 esta la notación de macro de ASN.1. esta notación permite al usuario extender la sintaxis del ASN.1 para definir nuevos tipos con sus valores.
Hay tres niveles que deben tenerse en cuenta:

- La notación de Macros, usada para definir macros.

- Una definición de macro, expresada en la notación del macro y usada para definir un grupo de instancias del macro.
- Una instancia de macro, generada desde la definición del macro por la sustitución de valores por variables.

Un definición de macro tiene la siguiente forma:

<*nombredelmacro*> MACRO ::=
BEGIN
TYPE NOTATION ::= <*nuevo tipo de sintaxis*>
VALUE NOTATION ::= <*nuevo valor de sintaxis*>
<*reglas adicionales*>
END

<*nuevo tipo de sintaxis*> describe el nuevo tipo de sintaxis que define el macro.
<*nuevo valor de sintaxis*> describe el nuevo valor de sintaxis que define el macro.
<*reglas adicionales*> describe cualquier regla gramatical adicional para cualquier tipo o valor de sintaxis.

10.5.4 Reglas Básicas de Codificación (BER)

Las reglas básicas de codificación (BER) son especificaciones de codificación desarrolladas por la CCITT (X.209) e ISO (ISO 8825). Estas especificaciones describen un método para la codificación de los valores de cada tipo de ASN.1 como string de octetos para poder ser pasados al componente de Transporte de datos.
Existen tipos de codificación alternativos, tales como PER (Packed encoding rules), DER (Distinguished encoding rules), CER (Cononical encoding rules), y LWER (Lightweight encoding rule). Sin embargo la codificación Ver es la mas utilizada.

11 Bibliografía y Referencias

1. *Nicholas Negroponte* - ***Begin digital****, Media Lab MIT,1995.*
2. *James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy y Willlian Lorensen* **– *Object-Oriented Modeling And Design****,General Electric and Development Center,1991.*
3. *William Stalling* – ***The Practical Guide to Network-Management Standars****,1993.*
4. *International Engineering Consortium* ***(http://www.iec.org) -TMN,****2002.*
5. *Capacitación Siemens****, Introducción al TMN****.*
6. *William Stallings* – ***SNMP,SNMPv2, And CMIP. The practical Guide to Network-Management Standards****, 1996.*
7. ***RFC821-SMTP****.*

8. *David J. Sidor* - ***TMN Standards: Satisfying Today's Needs While Preparing for Tomorrow*** *, Nortel IEEE Communications Magazine, Marzo 1998.*
9. *ITU-T Recommendation M.3010 —* ***Principles for a Telecommunications Management Network*** *, Julio 1996.*
10. *RFC1155 –* ***Structure and Identification of Management Information Base for TCP/IP-Based Internets.***
11. *RFC1213-* ***MIB-II.***
12. *RFC1157-* ***Simple Network Management Protocol****.*
13. ***ASN.1****: X.208 (ITU) and ISO 8824.*
14. *Perl Tutoriales - http://www.oreilly.com.*
15. *Apache Web Server - http://www.apache.org.*
16. *Codigo, Manuales y librerías de Perl - http://www.perl.org.*
17. *Base de Datos Myslq - http://www.mysql.com.*
18. *Free Software Foundation (FSF) - http://www.gnu.org.*
19. *Linux Mandrake - http://www.linux-mandrake.com.*
20. *Gustavo Rossi (UNLP), Daniel Schwabe y Robson Mattos Guimarães (PUC do Rio de Janeiro) -* ***Cohesive Design of Personalized Web Applications****, IEEE Internet Computing Abril 2002.*
21. *Lakshmi Raman, ADC Telecommunications -* ***OSI Systems and Network Management****, IEEE Communications Magazine, March 1998.*
22. *William Stallings -* ***SNMP and SNMPv2: The Infrastructure for Network Management****, IEEE Communications Magazine, March 1998.*
23. *Esteban Javier Próspero y Marcelo V. Frances Vall (UBA) -* ***SNMP-IMS*** *(Simple Network Management Protocol Internet Management System).*