Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN INGENIERÍA DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA

TRABAJO FIN DE MÁSTER

SECURIZACIÓN DE EQUIPOS DE PROTECCIÓN Y CONTROL DE SISTEMAS ELÉCTRICOS

Estudiante	*Vallejo García, Javier*
Director	*Perez González, Federico*
Departamento	*Ingeniería de Sistemas y Automática*
Curso académico	*2021-2022*

Bilbao, 19 junio 2022

Resumen

Los dispositivos electrónicos inteligentes o IEDs se han convertido en elementos indispensables dentro de las nuevas redes eléctricas inteligentes al encargarse de su protección y control. Los protocolos de comunicación para los IEDs se establecen bajo la norma IEC 61850 y, entre ellos, MMS (*Manufactoring Message Specification*) es considerado como el de mayor importancia. Dada su relevancia, MMS se ha convertido en objetivo frecuente de ciberataques.

En este proyecto se implementa un sistema de acceso basado en roles o RBAC establecido bajo la norma IEC 62351 como mecanismo de ciberseguridad sobre comunicaciones MMS de servidores IEC 61850 en IEDs.

Para validar la correcta implementación del RBAC se ha implementado una aplicación cliente IEC 61850 que permite comprobar el correcto funcionamiento de los servicios proporcionados por el servidor.

El RBAC implementado sobre los servidores de los IEDs permite el control de las tareas de los clientes sobre dichos dispositivos, al mismo tiempo que proporciona un nivel de seguridad de acceso como mecanismo adecuado para evitar ciberataques.

Abstract

Intelligent electronic devices or IEDs have become indispensable elements in the new smart grids as they are responsible for their protection and control. The communication protocols for IEDs are established under the IEC 61850 standard and, among them, MMS (Manufactoring Message Specification) is considered to be the most important. Given its relevance, MMS has become a frequent target of cyber-attacks.

In this project, a role-based access system or RBAC established under the IEC 62351 standard is implemented as a cybersecurity mechanism over MMS communications of IEC 61850 servers in IEDs.

To validate the correct implementation of the RBAC, an IEC 61850 client application has been implemented to check the correct operation of the services provided by the server.

The RBAC implemented on the IED servers allows the control of the client tasks on these devices, while providing a level of access security as a suitable mechanism to prevent cyber-attacks.

Laburpena

Gailu elektroniko adimendunak edo IEDak ezinbesteko elementu bihurtu dira sare elektriko adimendun berrien barruan, haiek babestu eta kontrolatzeaz arduratzen baitira. IEDetarako komunikazio-protokoloak IEC 61850 arauaren arabera ezartzen dira, horien artean, MMS (*Manufactoring Message Specification*) da garrantzitsuena. Hori dela eta, MMS zibererasoen ohiko helburu bihurtu da.

Proiektu honetan, IEC 62351 araua kontuan izanda roletan edo RBACean oinarritutako sarbide-sistema bat ezartzen da, IEDetan IEC 61850 zerbitzarietako MMS komunikazioen zibersegurtasun-mekanismo gisa.

RBACaren ezarpen egokia baliozkotzeko, IEC 61850 aplikazio bezero bat ezarri da, zerbitzariak emandako zerbitzuen funtzionamendu egokia egiaztatzen duena.

IEDen zerbitzarietan ezarritako RBACari esker, bezeroek gailu horien gainean egiten dituzten lanak kontrolatzea baimentzen du, eta, aldi berean, sarbideen segurtasun-maila bat eskaintzen du zibererasoak saihesteko mekanismo gisa.

Palabras clave:

Control de accesos basado en roles, IEC 61850, IEC 62351, IED, ciberseguridad.

Índice

Lista de tablas

Lista de ilustraciones

Acrónimos

GPS *Global Positioning System*

HMI *Human-Machine Interface*

IED Dispotitivos electrónicos inteligentes o *Intelligent Electronic Devices*

PGA *Power Grid Automation*

USB *Universal Serial Bus*

FTP *File Transfer Protocol*

SFTP *Secure File Transfer Protocol*

SSH *Secure Shell*

RBAC Control de acceso basado en roles o *Role-Based Access Control*

ACSI *Abstract Communication Services Inteface*

MMS *Manufacturing Message Specifications*

VMD *Virtual Manufacturing Device*

GOOSE Generic Object Oriented Substation Events

SCSM Specific Communication Service Mapping

LD Dispositivo Lógico o *Logical Device*

LN Nodo lógico o *Logical Node*

LLN0 *Logical Node-Zero*

GSE Eventos genéricos de subestaciones o *Generic Subtation Event*

SGCB *SETTING-GROUP-CONTROL-BLOCK*

RCB *REPORT-CONTROL-BLOCK*

LCB *LOG-CONTROL-BLOCK*

FC *Functional constraint*

BRCB *Buffered-Report-Control-Block*

URCB *Unffered-Report-Control-Block*)

GOCB *GOOSE-Control-Block*

GSSE *Generic Substation State Event*

GsCB *GSSE Control Block*

MSVCB *Multicast-Sample-Values-Control-Block*

USVCB *Unicast-Sample-Values-Control-Block*

1 Introducción

El sistema electrónico de potencia es un conjunto de elementos que producen, distribuyen y utilizan energía eléctrica. La parte física de este sistema se puede dividir en 4 subsistemas considerando su función [1];

- **Generación** Incluye todos los componentes que producen la energía eléctrica, como centrales nucleares o eólicas.
- **Transmisión**: Red eléctrica que transporta la electricidad desde su centro de generación hasta las zonas de consumo.
- **Distribución**: Red encargada de llevar la electricidad desde el subsistema de transmisión hasta el lugar en la que se va a consumir.
- **Suministro**: Red formada por los hilos de baja tensión (220V en Europa) que hay en cada domicilio y local comercial.

De todo el sistema electrónico de potencia, este proyecto se centra en las distintas redes eléctricas. Aunque estas estén bien diseñadas, siempre se puede producir un fallo en su funcionamiento por un agente externo, como pueden ser condiciones climáticas extremas [2], [3] o interno, por ejemplo el fallo en cascada que puede provocar la caída de un elemento de la red [4]. Para evitar que estos fallos puedan dañar permanentemente todo el sistema o provocar perjuicios en el suministro de electricidad, se cuenta con elementos de protección.

Los componentes básicos para un sistema de protección para las redes eléctricas son [5]:

- **Fusibles**: Elementos que conducen la electricidad y se destruyen solos ante una situación de fallo, abriendo el circuito, para evitar daños mayores. A diferencia de los siguientes elementos, funcionan de forma autónoma.
- **Transformadores de corriente y de tensión**: Bajan los niveles de intensidad y voltaje a unos parámetros con los que pueda trabajar el relé sin que se estropee por sobrecarga.
- **Relés de protección**: Reciben la señal de los transformadores y dan a los interruptores la orden de abrir si hay alguna falta.
- **Interruptores**: Son los elementos que abren el circuito si reciben la orden del relé.

- **Sistema de alimentación**: Sistema independiente del que se está protegiendo y que proporciona electricidad para que funcionen los relés y los interruptores.

Por otro lado, la mayoría de estas redes se construyeron cuando la obtención de la energía era barata y no había tanta concienciación sobre la contaminación del planeta. Por esto, actualmente estas redes se están transformando en redes inteligentes o *Smart Grids*, para hacerlas más eficientes. Un ejemplo de esta transición se tiene en el actual plan de Nepal para la transformación de su red eléctrica [6].

Las redes inteligentes, entre otras cosas, hacen uso de productos y servicios innovadores junto con protocolos de comunicaciones y elementos enfocados al control y monitorización de la red para que sea más eficiente y se tenga un mayor control sobre la misma [7]. En cuanto a los elementos de control y monitorización que usa la red, son los siguientes:

- Medios que permitan la comunicación entre los diferentes elementos del sistema, dando acceso a los diferentes niveles de las subestaciones (centro de control, nivel de subestación y nivel de aparamenta). Esto se hace mediante comunicaciones de radio, ethernet o serie.

- HMIs (*Human-Machine Interface*) para adquisición, monitorización y control del sistema.

- Elementos de sincronización que permitan tener una referencia de tiempo precisa en todo el sistema. Por ejemplo, Sistemas de posicionamiento global o GPS (*Global Positioning System*).

Uno de los elementos más importantes en estas redes eléctricas inteligentes son los equipos de protección y control de sistemas eléctricos, también llamados dispositivos electrónicos inteligentes o IEDs (*intelligent electronic devices*). Estos se instalan en las subestaciones eléctricas para mejorar la capacidad de monitorización, el control, la protección y la adquisición de datos de la red eléctrica [7]. También son capaces de detectar, localizar, comunicar al resto de dispositivos y actuar contra los fallos que puedan ocurrir en la red [8].

Finalmente destacar que, para que los IEDs hagan su trabajo correctamente, se tienen que comunicar continuamente con los demás IEDs y otros elementos de la red [9].Por esto y dada la importancia crítica que tiene la red eléctrica en el mundo actual, la cantidad de formas en las que se las puede realizar un ciberataque [10] y los efectos devastadores que pueden tener en la red (que tiende a aumentar su digitalización [11]) hacen que la ciberseguridad de las comunicaciones de estos dispositivos sea un tema primordial.

1.1 Motivación

Este proyecto se ha realizado en el Departamento de I+D de la Unidad de Negocios PGA (*Power Grid Automation*) de Ingeteam y los equipos con los que se trabaja son los IEDs de

dicha empresa (figura 1.1). Estos equipos se comunican mediante el estándar IEC 61850 (*Communication networks and systems for power utility automation*) y están diseñados para realizar funciones de protección de sistemas eléctricos y para el control y supervisión de los componentes del mismo. Estos equipos tienen integrados tanto trafos de medida de tensión y corriente como tarjetas de entradas y salidas digitales, para poder supervisar y controlar el estado de la aparamenta eléctrica (interruptores, seccionadores y demás automatismos).

Figura 1.1: IED usado en este proyecto

Los equipos de protección y control a nivel de conectividad, para comunicación con otros dispositivos, disponen de puertos serie, ethernet y USB (*Universal Serial Bus*). La configuración de los equipos se realiza a través de ellos. Para la comunicación por puerto Ethernet tiene protocolos basados en la norma IEC (61850, 60870-5-101 o 60870-5-104) y otros como FTP (*File Transfer Protocol*), SFTP (*Secure File Transfer Protocol*) o SSH (*Secure Shell*).

Debido a las posibilidades que ofrecen los IEDs, es necesario securizar su servidor, implementando un control de las acciones que puedan realizar los clientes que se conecten a él. De este modo se evita que se realicen malas prácticas en ellos, ya sea de manera malintencionada (un ciberataque) o por un mal uso de los IEDs. Para ello en este proyecto se va a diseñar e implementar un control de acceso basado en roles o RBAC (*Role-Based Access control*) para la securización de los servicios del servidor. Este método de acceso permitirá el control de los clientes que tienen la capacidad de conectarse a IEDs, limitando las acciones que pueden realizar y la información a la que tienen acceso [12].

1.2 Contexto

Relativo a la ciberseguridad, en los equipos con los que se trabaja en este proyecto ya existen distintas características implementadas para securizar el IED, como son: firewall, listas blancas, bloqueos de acceso, control de sesiones, RBAC y firmware firmado.

Sobre la característica de RBAC implementada en los IEDs, actualmente se dispone de una implementación RBAC a nivel de sistema, que centraliza la gestión de roles y usuarios. Esta implementación está basada siguiendo lo definido tanto en la norma IEEE 1686 como en la IEC 62351, aunque se ha desarrollado de forma que sea configurable y ampliable en el futuro en caso de necesidad. Mediante el RBAC existente se gestiona el acceso autenticado de los usuarios (locales o por protocolo ligero de acceso a directorios) a través de la web, SFTP y display. Una vez el usuario ha accedido correctamente solo tendrá acceso a las funciones que le permitan los roles que tenga asignados.

En cuanto a las comunicaciones IEC 61850, el equipo dispone de un servidor IEC 61850 Ed2 homologado con un certificado de nivel A (UCAIUG). La funcionalidad del IED en este ámbito es muy completa, incluyendo gran parte de los servicios que define la norma mencionada, pero se encuentra en una fase temprana de implementación de las medidas de ciberseguridad definidas por la IEC62351 para el protocolo IEC61850. En la fecha en la que se escribe este texto no existen homologaciones de la UCAIUG que verifiquen la ciberseguridad en IEC 61850, estando previsto que comiencen a lo largo de 2022.

En este contexto, este proyecto está enfocado en robustecer el protocolo IEC 61850 mediante la implementación del RBAC para que sea capaz de securizar el acceso a los servicios disponibles en su servidor. Permitiendo así que la amplia funcionalidad ofrecida por el servidor se pueda limitar a los distintos clientes en base a sus roles asignados. Esto amplifica las características RBAC disponibles, ya que extiende su implementación al nivel de servidor.

Para que ambas implementaciones del RBAC funcionen y se puedan configurar simultáneamente, tienen que usar el mismo archivo de configuración de los roles. Por ello, es necesario un análisis del archivo que usa la implementación RBAC existente y extenderla a la que se hace en este proyecto.

La finalidad es implementar este RBAC en el código fuente del servidor IEC 61850 de los IEDs. Como este está programado en C, todo el desarrollo que se realiza en este proyecto se hace en este mismo lenguaje de programación. En cuanto al entorno de programación, se utiliza *Eclipse*, ya que es este el que tiene las herramientas necesarias para poder depurar el código fuente del servidor.

Alcance y objetivos

2

2.1 Alcance

En este proyecto se pretende diseñar e implementar un sistema de control de accesos basado en roles para la securización de las comunicaciones en los servidores IEC 61850 de un IED.

Las funcionalidades de este sistema deben de ser:

- Identificar a los clientes que desean conectarse al servidor y asignarles roles.
- Permitir que los clientes solo realicen las acciones en el servidor que permiten sus roles asignados.
- Programado mediante funciones de fácil implementación. Se debe de disponer de funciones para:
  - Inicializar toda la configuración del sistema.
  - Comprobar si un cliente puede realizar una cierta acción.
  - Modificar la configuración del sistema una vez inicializada.
- Estar implementado directamente en el código fuente del servidor IEC 61850 de los equipos.
- Poder convivir con el sistema RBAC ya implementado en los equipos sin interferir en sus funcionamientos, pero compartiendo el archivo de configuración de los roles.
- Estar basado en la parte 8 de la norma IEC 62351.

Por otro lado, para comprobar que el sistema funciona correctamente, se debe programar una aplicación cliente IEC 61850 que lance peticiones de los distintos servicios al servidor. De esta forma, asignándole cada vez distintos roles, se puede comprobar si realmente sólo puede acceder a los servicios sobre los que dispone de permiso. Este cliente tiene que cumplir que:

- Permita elegir el servidor al que se desea conectar.
- Se puedan establecer los servicios de los que se envían las peticiones de tres formas distintas:

- Mediante un archivo de configuración.
- Desde la interfaz de la aplicación.
- Mediante los parámetros de ejecución de la aplicación.

2.2 Objetivos

El objetivo principal de este proyecto es implementar en el código fuente de un IED un modelo RBAC basado en la parte 8 de la norma IEC 62351. Para lograr esto, se establecen los siguientes objetivos parciales:

- Analizar los servicios que define la norma IEC 61850 y su mapeo de comunicaciones. Para ello, se revisan las partes de la norma:
  - IEC 61850-7-2, donde se definen los servicios.
  - IEC 61850-8-1, que estandariza el mapeo de casi todos los servicios.
  - IEC 61850-9-2, donde se establece el mapeo de los servicios de trasmisión de valores muestreados.
- Analizar la definición del modelo RBAC que da la parte 8 de norma IEC 62351.
- Estudiar la relación de los permisos definidos en la parte 8 de la norma IEC 62351 con los servicios IEC 61850. Esta relación se da en el reporte técnico 90-19 de la norma IEC 61850.
- Establecer los archivos necesarios para cumplir las funcionalidades que se desean para el control de accesos basado en roles. Se establecen tres archivos cuyas funcionalidades son:
  - Fijar los roles de cada usuario.
  - Indicar los permisos que tiene cada rol. Este será una modificación de un archivo usado en el RBAC ya implementado en los equipos.
  - Identificar los permisos asociados a cada servicio.
- Definir los paquetes de funciones necesarios para que el sistema funcione correctamente. Se definen cuatro paquetes con las siguientes funcionalidades:
  - Gestionar los roles de los usuarios.
  - Gestionar los permisos de los roles.
  - Gestionar los permisos de los servicios.

    - Actualizar la configuración del RBAC.

- Desarrollar las funciones con las funcionalidades establecidas a cada paquete.
- Implementar las funciones en el código fuente de los IEDs.
- Desarrollar una aplicación cliente IEC 61850 para validar la correcta implementación del RBAC.
- Validar la correcta implementación del RBAC.

Para crear la aplicación cliente IEC 61850 que se usa para validar el RBAC implementado, se siguen los siguientes objetivos parciales:

- Descargar y analizar la librería con las funcionalidades cliente IEC 61850.
- Hallar el código necesario para enviar las peticiones de los distintos servicios.
- Crear una función con el código del apartado anterior para cada servicio.
- Crear la aplicación cliente con las funciones anteriores.
- Validar el correcto funcionamiento de la aplicación cliente.

Finalmente, para validar la correcta implementación del RBAC, se realizan las siguientes acciones:

- Validar el código de las funciones programadas para el sistema de control de accesos ejecutándolas en un programa principal.
- Probar los servicios en el código fuente de los equipos en modo de depuración. Para ello se emplea la aplicación *OperationFactory*. Con esto se comprueba en un número limitado de servicios si se ha implementado su gestión de permisos correctamente.
- Ejecutar la aplicación cliente programada directamente sobre el código en modo depuración, para validar la implementación de los permisos de todos los servicios.
- Compilar el código para desplegarlo en un IED y ejecutar el cliente en él, asegurando que el control de accesos funciona correctamente en dicho equipo.

Estudio teórico

3

3.1 Norma IEC 61850

Para diseñar e implementar el control de acceso basado en roles, primero hay que hacer un análisis teórico de la norma IEC 61850 al estar los IEDs con los que se trabaja basados en ésta.

La norma IEC 61850 es el estándar actual para redes de comunicación y sistemas en las subestaciones eléctricas [13]. Aunque este proyecto se centra en sus protocolos de comunicación, ésta abarca múltiples aspectos como[13]:

- Requisitos eléctricos o de calidad.
- Plataformas o perfiles de comunicación.
- Gestión de sistemas y proyectos.

La norma está formada por varias partes, de ellas a lo largo de este proyecto se analizan especialmente:

- **IEC 61850-7-2**: Define los servicios ACSI (*Abstract Communication Services Intefa-ce*) de la norma IEC 61850.
- **IEC 61850-8-1**: Estandariza los mapeos MMS (*Manufacturing Message Specifica-tions*) de los servicios ACSI de la norma (menos los servicios asociados a la trasmisión de valores muestreados).
- **IEC 61850-9-2**: Establece los mapeos MMS de los servicios asociados a la trasmisión de valores muestreados.

3.1.1 Protocolos de comunicación

Al ser el objetivo de este proyecto la ciberseguridad de sus comunicaciones, de todos los aspectos de la norma se analizan sus protocolos de comunicación.

Tal y como se puede ver en la Figura 3.1, la norma define el uso de varios modelos de comunicación. El más importante es el protocolo que hace uso de los servicios del estándar MMS definido en la norma ISO 9506 [14]. Esta comunicación se realiza a través de conexiones TCP/IP (Capa 4 del modelo OSI) para el intercambio de datos o la configuración de los IEDs [15]. Por otro lado, las características del estándar MMS son [16]:

- Especifica la capa de protocolo de aplicación para el control y la monitorización de dispositivos industriales inteligentes.
- Representa las características externas del dispositivo mediante un modelo VMD (*Virtual Manufacturing Device*).
- Define 19 clases distintas de objetos y 86 servicios para actuar sobre ellas.
- Su modelo de comunicación es cliente-servidor.

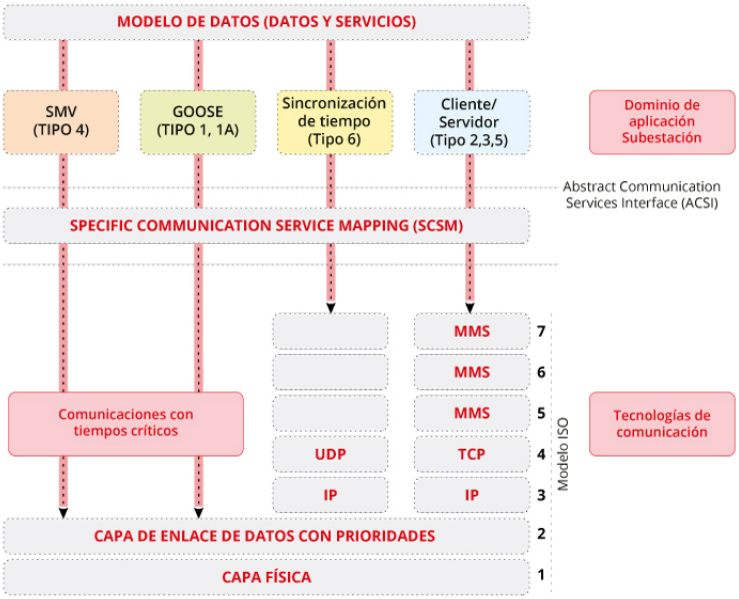


Figura 3.1: Protocolos de comunicación de la norma, obtenido de [17].

Otro protocolo de comunicación importante es el GOOSE (*Generic Object Oriented Substation Events*) que es usado por los IEDs para comunicarse entre sí [18]. Este tiene las siguientes características [19]:

- Se aplica en el envío de datos críticos en tiempo real.
- Envía datos a través de mensajes multicast de ethernet (capa 2 del modelo de OSI).
- Está basado en una arquitectura publicista-subscriptor.

El último protocolo destacable definido en la norma es el de trasmisión de valores muestreados usado para el envío del estado de la red a los IEDs [18]. Sus características son [20]:

- Trasmite valores en bruto obtenidos del muestreo.
- Envía mensajes a través de la capa 2 de ethernet.
- Está basado en una arquitectura publicista-subscriptor.

3.1.2 Modelo ACSI

Una parte importante de los protocolos de comunicación es el modelo ACSI (*Abstract Communication Services Inteface*). Este modelo proporciona un interfaz virtual a un IED que con independencia del perfil de comunicación utilizado ofrece:

- Servicios de comunicación abstractos (se analizan en el apartado 4.1).
- Acceso a variables.
- Transferencia de datos.
- Control de dispositivos.
- Trasmisión de ficheros.

Este modelo permite que una aplicación basada en la norma IEC 61850 funcione independientemente del tipo de red de comunicación [21]. Por otro lado, ACSI permite describir una red eléctrica y sus componentes con un gran grado de detalle abstrayéndose de los distintos fabricantes [22], ya que define un modelo común a todos ellos.

Cabe destacar que el mapeo de los servicios ACSI sobre un protocolo concreto se denomina SCSM (*Specific Communication Service Mapping*) [23]. Los servicios ACSI que no son críticos en el tiempo (siendo estos el grupo mayoritario) mapean a servicios definidos por el protocolo MMS para el intercambio de datos [24]. En el apartado 4.2 se estudia su mapeo MMS. Estos son críticos frente a la ciberseguridad, ya que a través de un mal uso de ellos la información de los IEDs puede ser manipulada y forzada para hacer que los equipos no funcionen correctamente y acceder a sus datos [15]. Por ello, este trabajo se centra en la securización del acceso a estos servicios mediante la implementación del RBAC.

Así mismo dentro del modelo ACSI se establecen las siguientes clases de modelos de datos [25], cuya relación se observa con mayor claridad en la figura 3.2:

- **Servidor o *Server***: Representa el comportamiento visible de un dispositivo mediante el conjunto de datos a los que pueden acceder los clientes a través de los distintos servicios. Un IED puede tener uno o varios servidores y un servidor puede ser el cliente de otro [26].
- **Dispositivo lógico o LD (*Logical Device*)**: Representa un grupo de funciones relacionadas y los servicios asociados a cada una de ellas [27].
- **Nodo lógico o LN (*Logical Node*)**: Representa una función específica. Es la parte más pequeña de una función que intercambia datos. El nodo lógico más importante es el LLN0 (*Logical Node-Zero*). Todos los LD tienen que tener el suyo y contiene su información común [28].

- **Objetos de datos o *Data Objects***: Información contenida en los Nodos Lógicos. Estos representan [29]:
  - Medidas de la red.
  - Valores de configuración y control.
  - Los valores del estado de la red. Por ejemplo, el estado de los interruptores.
- **Atributos de datos o *Data Attributes***: Información contenida dentro de los objetos de datos.

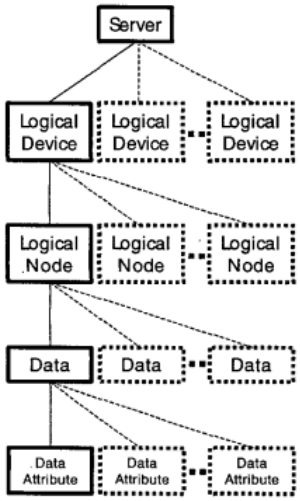


Figura 3.2: Relación entre las clases de modelos de datos del modelo ACSI (obtenido de [25]).

Destacar que para acceder en las comunicaciones a cualquiera de estos objetos, se utilizan las referencias a objetos u *object references*. Tanto la sintaxis como la semántica de éstas están definidas en la norma.

Finalmente, la norma también define diferentes modelos para el intercambio de datos. Estos son [30]:

- **Conjunto de datos o *Data Set***: Hace posible agrupar objetos de datos y atributos de datos.
- **Substitución o *Subtitution***: Con el se pueden cambiar el valor de un proceso.
- **Control del grupo de ajustes o *Setting Group Control***: define como editar los grupos de ajustes y el cambio entre ellos.
- **Control de informes y registros o *Report Control and Logging***: Da la descripción de, dependiendo de los parámetros de configuración establecidos por un cliente, cómo se generan los informes y los registros.

- **Bloques de control para eventos genéricos de subestaciones o GSE (*Generic Substation Event*)**: Permite la distribución a través de todo el sistema de valores de los datos de entrada y salida de manera fiable y rápida.
- **Bloques de control para la trasmisión de valores muestreados**: Envío de valores muestreados de forma rápida y cíclica.
- **Control**: Establece los servicios de control.
- **Tiempo y sincronización temporal**: Proporciona la base temporal para el dispositivo y sistema.
- **Sistema de ficheros**: Establece el intercambio de grandes grupos de datos.
- **Seguimiento**: Da un interfaz de diagnosis para el seguimiento de los servicios.

3.2 Norma IEC 62531

Cuando se creó el estándar IEC 61850 se consideraba "seguro por oscuridad", ya que era muy complejo y especializado. Actualmente, dada la estandarización de los protocolos, éste ha dejado de ser "oscuro" para los ciberatacantes [31]. Para atajar el problema de la ciberseguridad de este estándar y definir los sistemas para securizar los mensajes enviados a través de sus protocolos, se desarrolla el estándar IEC 62351 [31].

IEC 62531 es un estándar que, a la fecha de publicación de este proyecto, aún continúa en desarrollo y aborda los mecanismos para establecer la ciberseguridad de IEDs dentro de una red eléctrica [32]. Actualmente está formado por 16 partes [31]. De todas ellas, este proyecto se centra en la parte 9, ya que es la que define el modelo RBAC para su uso en los IEDs. También se analiza la guía para el manejo del RBAC en los sistemas de potencia, dada en el reporte técnico IEC 62351-90-1 [31].

3.2.1 Modelo RBAC

El RBAC es una tecnología con el potencial de reducir la complejidad y el costo de la administración de seguridad en redes, con un gran número de dispositivos inteligentes. En éste cada usuario que forma parte de una conexión tiene asociados uno a varios roles que a su vez tienen asignado uno o varios permisos [12]. Así a cada usuario sólo se le permite realizar las acciones que definen los permisos de sus roles, controlando la información accesible y las acciones realizables [12]. De esta forma, el RBAC reduce los permisos que se tienen que asignar a cada usuario, ya que sólo tendrán permisos para realizar sus tareas programadas [31]. Esto también asegura que, si se intenta conectar a los equipos un usuario sin ningún rol, al no tener ningún permiso no pueda realizar la conexión.

De esta manera, en el RBAC se tienen que hacer tres asignaciones [33]:

- **Asignación de sujetos (mapeado sujeto a rol)**: Uno o más sujetos pueden ser asignados a uno o más roles. En el caso de este trabajo el usuario se identifica con una IP.
- **Asignación de roles (mapeado rol a permisos)**: Un permiso debe pertenecer al menos a un rol.
- **Asignación de permisos (mapeado de las acciones a los objetos)**: En este proyecto, se refiere a los permisos que tiene asignado casa servicio ACSI.

La parte 8 de la norma IEC 62351 define permisos y roles obligatorios [31], que se analizan en el apartado 4.3. Por otra parte, el informe técnico IEC 61850-90-19 da una relación de los permisos predefinidos y los servicios ACSI que se analizan en el apartado 4.5.

Análisis y diseño

4

4.1 Análisis de los servicios ACSI

Para poder diseñar el sistema de control de accesos basado en roles, hay que analizar la teoría necesaria para que funcione correctamente. Por ello se estudian los servicios ACSI que define la parte 7-2 de la norma 61850. Esta define los servicios ACSI en modelos de las clases definidas en la sección 3.1.2. Estos modelos son [30]:

- Modelo *GenServer*.
- Modelo de asociación.
- Modelo *GenLogicalDeviceCLass*.
- Modelo *GenLogicalNodeClass*.
- Modelo *GenDataObjectClass*.
- Modelo *DATA-SET*.
- Modelo *SETTING-GROUP-CONTROL-BLOCK* (SGCB).
- Modelo *REPORT-CONTROL-BLOCK* (RCB)
- Modelo *LOG-CONTROL-BLOCK* (LCB).
- Modelo GSE.
- Modelo para la trasmisión de valores muestreados.
- Modelo de control.
- Tiempo y sincronización horaria.
- Modelo para la trasferencia de ficheros.

A continuación, se explica detalladamente cada modelo, indicando los servicios que contiene cada uno. Destacar que, en las definiciones de los servicios, los datos visibles para el cliente también son accesibles.

4.1.1 Modelo *GenServer*

Es el modelo asociado al servidor. Debe representar el interfaz de un dispositivo. El único servicio que tiene este modelo es **GetServerDirectory** que devuelve la lista de los nombres de los archivos o nodos lógicos que son accesibles para el cliente a través del servidor.

4.1.2 Modelo de asociación

Este modelo es el encargado de establecer las comunicaciones entre varios tipos de dispositivos. El cliente dispone, en este modelo, de los siguientes servicios:

- **Associate**: Establece una conexión (asociación) entre un cliente y un servidor. Es necesario establecer una conexión para que el cliente pueda enviar solicitudes del resto de servicios ACSI al servidor.
- **Abort**: Desconecta una asociación cliente-servidor de forma abrupta. Es decir, todas las peticiones de servicio emitidas son descartadas.
- **Release**: Desconecta una asociación cliente-servidor de forma ordenada. Lo que significa que las peticiones de servicios emitidas se completan antes de terminar la conexión, pero no se pueden enviar nuevas peticiones.

De estos servicios, los servicios **Abort** y **Release** no se securizan ya que aunque tengan mapeo MMS, al securizar el servicio Assoicate ya se tiene un control de los dispositivos que se conectan al servidor.

4.1.3 Modelo *GenLogicalDeviceCLass*

Este modelo está asociado a los dispositivos lógicos y representa un grupo de nodos lógicos. El servicio que contiene es **GetLogicalDeviceDirectory**, que devuelve una lista de *Object References* de todos los nodos lógicos visibles por el cliente a través del dispositivo lógico.

4.1.4 Modelo *GenLogicalNodeClass*

Este modelo describe los nodos lógicos con los que define servicios para conocer su contenido. Los servicios dentro de este modelo son:

- **GetLogicalNodeDirectory**: Devuelve una lista de *Object References* de las instancias de una clase ACSI accesibles por el cliente a través del nodo lógico deseado.
- **GetAllDataValues**: Obtiene una lista de todos los valores de los atributos de datos con el mismo FC (*Functional Constraint*) de los objetos de datos a los que el cliente tiene acceso por nodo lógico deseado.

El FC es una propiedad de los atributos de datos que se usa en la definición de los objetos de datos. Indica los servicios que pueden hacer operaciones en un determinado atributo y se tiene que especificar con dos caracteres. Por ejemplo, un FC=SG representa que el atributo pertenece a un grupo de ajustes.

4.1.5 Modelo *GenDataObjectClass*

Es el modelo de los objetos de datos con los que sus servicios sirven para manejar los atributos de datos de los objetos. Los servicios de este modelo son:

- **GetDataValues**: Obtiene el valor de los atributos de datos de un objeto de datos accesible por el cliente.
- **SetDatValues**: Establece el valor de los atributos de datos de un objeto de datos accesible por el cliente.
- **GetDataDirectory**: Devuelve una lista de los nombres de todos los atributos de datos de un objeto de datos visibles por el cliente.
- **GetDataDefinition**: Devuelve de un objeto de datos visibles por el cliente una lista de las descripciones de todos sus atributos de datos.

4.1.6 Modelo *DATA-SET*

Este modelo maneja los conjuntos de datos. El modelo ofrece los siguientes servicios:

- **GetDataSetValues**: Devuelve los valores de todos los atributos de datos a los que puede acceder el cliente del conjunto de datos deseado.
- **SetDataSetValues**: Establece los valores de todos los atributos de datos accesibles por el cliente del conjunto de datos deseado.
- **CreateDataSet**: Crea un nuevo conjunto de datos a través de una lista de datos o de atributos de datos.
- **DeleteDataSet**: Solicita al servidor eliminar un conjunto de datos accesible por el cliente.
- **GetDatSetDirectory**: Devuelve la lista de los *object references* de todos los datos miembros un *Data set* a los que el cliente tiene acceso.

Como los servicios **SetDataSetValues**,**CreateDataSet** y **DeleteDataSet** no están implementados en los IEDs con los que se trabaja en este proyecto, no se continúa con su estudio.

4.1.7 Modelo *SETTING-GROUP-CONTROL-BLOCK*

Es el modelo responsable del manejo de los grupos de datos. Los servicios que ofrece son:

- **SelectActiveSG**: Selecciona cuál de los grupos de ajustes es el activo.
- **SelectEditSG**: Elige cuál de los grupos de ajustes es el editable del SGCB al que el cliente puede acceder a través del LLN0 deseado.
- **SetEditSGValue**: Establece el valor del objeto de datos del grupo de control editable del SGCB visible por el cliente a través del LLN0 dado.
- **ConfirmEditSGValues**: Se usa para confirmar que los valores establecidos con la anterior función reescriban a los antiguos.
- **GetEditSGValues**: Devuelve el valor del grupo de ajustes activo o editable al que el cliente puede acceder por el LLN0 deseado.
- **GetSGCBValues**: Sirve para obtener una lista con el valor de los atributos de datos del SGCB visible por el cliente por el LLN0 referenciado.

4.1.8 Modelo *REPORT-CONTROL-BLOCK*

Este modelo maneja los informes, por lo que que tiene que supervisar los procesos de los que se desea poder informar de sus valores. Este grupo de control se divide en dos tipos:

- **BRCB (*Buffered-Report-Control-Block*)**: Controla informes que no son críticos en el tiempo con lo que, si no se pueden mandar, se los inserta en un buffer hasta poder enviarlos.
- **URCB (*Unffered-Report-Control-Block*)**: Controla informes que se tienen que mandar en cuanto se obtienen y se pueden perder si se rompe la conexión o el trasporte no es lo suficientemente rápido para soportarlo.

Los servicios definidos por ambos modelos son:

- **Report** (igual para ambos modelos): Lo usa el RCB para enviar reportes del servidor al cliente. Cómo es un servicio al que no tiene acceso un cliente, no se le tiene en cuenta en el resto de proyecto.
- **GetBRCBValues o GetURCBValues**: Proporciona los valores de los atributos de datos del RCB visible por el cliente a través de un nodo lógico.
- **SetBRCBValues o SetURCBValues**: El cliente lo usa para establecer los valores de los atributos de datos del RCB al que pueda acceder a través de un nodo lógico.

4.1.9 Modelo *LOG-CONTROL-BLOCK*

Es el modelo que controla el correcto funcionamiento de los registros. Se crean dos clases dentro de este modelo:

- **Clase LCB**: Controla los procedimientos para guardar valores de los atributos de datos (la entrada del informe) dentro de un informe (LOG).
- **Clase LOG**: Controla cómo se guardan las entradas de informes dentro del LOG. Lo realiza siguiendo la política del primero que entre es el primero que sale y cuando se llena, para meter un reporte nuevo, se elimina el más antiguo.

En la clase LCB se definen dos servicios:

- **GetLCBValues**: Obtiene los valores de los atributos del LCB visibles por el cliente a través de un nodo lógico.
- **GetLCBValues**: Establece los valores de los atributos del LCB a los que el cliente tiene acceso a través de un nodo lógico.

Por otro lado, la clase LOG establece los siguientes servicios:

- **QueryLogByTime**: Da una lista de las entradas de un registro que haya entre un tiempo inicial y uno final.
- **QueryLogAfter**: Devuelve las entradas de un registro posteriores a un tiempo inicial y una referencia a una entrada. Dentro de un mismo tiempo puede haber distintas entradas de registros.

4.1.10 Modelo GSE

Es el modelo que controla la trasmisión de los valores de entrada y salida por todo el sistema. Se definen dos grupos de control:

- ***GOOSE-Control-Block* (GOCB)**: Es el grupo de control para los mensajes GOOSE. A parte de lo comentado en el apartado 3.1, estos mensajes se basan en el envió de conjunto de datos.
- ***Generic substation state event* (GSSE) Control Block (GsCB)**: Se basa en la trasmisión de cambio de estados (en pares de bits). Al estar esta clase obsoleta y no incluirse en los nuevos IEDs, no se analizan sus servicios.

Los servicios dentro de la clase de GOCB son:

- **SendGOOSEMEssage**: Utilizado por el GOCB del servidor para enviar un mensaje GOOSE.

- **GetGoReference**: Devuelve la referencia a un miembro específico del conjunto de datos de un GOCB.
- **GetGOOSEElementNumber**: Facilita la posición de un miembro seleccionado de un atributo de datos de un GOCB.
- **GetGoCBValues**: Devuelve los valores de los atributos de datos del GOCB visible por el cliente a través del LLN0 deseado.
- **SetGoCBValues**: Establece los valores de los atributos de datos del GOCB visible por el cliente a través del LLN0 deseado.

Al ser **SendGOOSEMEssage** interno del servidor y como **GetGoReference** y **GetGOOSEElementNumber** no usan un mapeo MMS, no se estudian más en este proyecto.

4.1.11 Modelo para la trasmisión de valores muestreados

Dentro del modelo para la trasmisión de valores muestreados, se definen dos bloques de control dependiendo de la conexión usada:

- ***Multicast-Sample-Values-Control-Block* (MSVCB)**: Un único servidor envía datos a través de una conexión con múltiples receptores.
- ***Unicast-Sample-Values-Control-Block* (USVCB)**: El servidor envía datos a través de una conexión con un único receptor.

Dentro de ambos bloques se definen los siguientes servicios:

- **SendMSVMessage y SendUSVMessage**: Envía un mensaje SV a través de una conexión. Como es un servicio interno de los servidores, no tiene mapeo MMS y no se estudia en este trabajo.
- **GetMSVCBValues y GetUSVCBValues**: Devuelve los valores de un MSVCB o un USVCB visible por un cliente a través de un LLNO.
- **GetMSVCBValues y GetUSVCBValues**: Establece los valores de un MSVCB o un USVCB accesible por un cliente a través de un LLNO.

4.1.12 Modelo de control

El objetivo de este modelo es proporcionar que un cliente puede cambiar el estado de procesos internos y externos (por ejemplo, cambiar el valor de una salida digital). Establece los siguientes servicios:

- **Select**: Selecciona un objeto de control.

- **SelectWithValue**: Selecciona un objeto de control y le da un valor.
- **Cancel**: Cancela una operación de control.
- **Operate**: Actúa (da un valor) sobre un objeto de control.
- **TimeActivatedOperate y TimeActivatedOperationTermination**: Usados internamente por el servidor para empezar o terminar respectivamente una operación de control cuando termina un temporizador interno.
- **CommandTermination**: El servidor termina cada secuencia de control con este servicio cuando se trabaja un control con seguridad mejorada.

Los servicios **TimeActivatedOperate**, **TimeActivatedOperationTermination** y **CommandTermination** son internos del servidor, por lo que no se incluyen en el trabajo del resto del proyecto.

4.1.13 Tiempo y sincronización horaria

Este modelo actúa internamente para la sincronización temporal del servidor con el servicio **TimeSynchronization**. Por ser interno del servidor, queda excluido del resto del proyecto.

4.1.14 Modelo para la transferencia de ficheros

El modelo encargado de la transferencia de ficheros (grandes grupos de datos) entre el cliente y el servidor. Este tiene los siguientes servicios:

- **GetFile**: Transfiere el contenido de un archivo del servidor al cliente.
- **SetFile**: Manda el contenido de un fichero del cliente al servidor.
- **DeleteFile**: Elimina un archivo del servidor,
- **GetFileAttributeValues**: Devuelve la información de un archivo del servidor.

4.2 Mapeo MMS de los servicios ACSI

Como se ha comentado, para el intercambio de información los servicios ACSI de estudio en este proyecto hacen uso de mapeos a servicios estandarizados por MMS. Los servicios MMS donde son mapeados los servicios ACSI son (según la definición de los servicios MMS obtenida de [34] y mapeos MMS de los servicios ACSI obtenidos de [35] y [14]):

- **GetNameList**: Devuelve una lista o parte de una lista de los nombres de los objetos definidos en un VMD (*Virtual Manufacturing Device*). Los servicios ACSI

mapeados son GetServerDirectory (para devolver la lista de dispositivos lógicos), GetLogicalDeviceDirectory y GetLogicalNodeDirectory.

- **FileDirectory**: Da los nombres de uno o más ficheros del servidor. Sobre este servicio MMS se mapean los servicios ACSI GetServerDirectory (para devolver la lista de los archivos) y GetFileAttributeValues.

- **Initiate**: Establece la comunicación para el intercambio de información entre dos usuarios MMS mediante el uso de servicos MMS. Únicamente mapea sobre este servicio MMS el servicio ACSI Associate.

- **Read**: Retorna el valor de una o más variables definidas en el VMD. Los servicios ACSi que se mapean son GetAllDataValues, GetDataValues, GetDataSetValues, GetEditSGValue, GetSGCBValues, GetSGCBValues, GetBRCBValues, GetURCBValues, GetMSVCBValues, GetUSVCBValues, GetLCBValues, GetLogStatusValues, GetGoCBValues y Select.

- **Write**: Cambia el valor de una o más variables a un valor deseado. Los servicios ACSI que mapean sobre este servicio MMS son SetDataValues, SelectActiveSG, SelectEditSG, SetEditSGValue, ConfirmEditSGValues, SetBRCBValues,SetURCBValues, SetMSVCBValues, SetUSVCBValues, SetLCBValues, SetGoCBValues, SelectWithValue, Cancel y Operate.

- **GetVariableAccessAttributes**: Devuelve los atributos de un Objeto variable definido en el VMD, los servicios ACSI con mapeo a él son GetDataDirectory y GetDataDefinition.

- **GetNamedVariableListAttributes**: Devuelve los atributos de un objeto de lista variable definido en un VMD, sobre este servicio MMS mapea el servicio ACSi GetDataSetDirectory.

- **ReadJournal**: Da los valores de los campos de un objeto entrada de registro dentro de un objeto registro. Los servicios ACSi QueryLogByTime y QueryLogAfter están mapeados sobre este servicio.

- **ObtainFile**: Se usa para enviar un fichero de un directorio a un servidor. El mapeo del servicio ACSI SetFile se hace sobre este servicio MMS.

- **FileDelete**: Elimina un fichero del servidor. El único servicio que mapea sobre él es el servicio ACSI DeleteFile.

- Como caso especial, el servicio ACSI GetFile mapea a una secuencia de servicios MMS FileOpen, FileRead y FileClose en donde:

    - **FileOpen**: Identifica el archivo que hay que leer y poner en activo el mecanismo de su lectura.

- **FileRead**: Transfiere el contenido o parte del contenido de un archivo abierto del servidor a un cliente,
- **FileClose**: Libera los recursos asociados a la trasferencia del fichero.

4.3 Análisis de los permisos y roles predefinidos del modelo RBAC

Para la correcta implementación del RBAC se analizan los permisos y roles predefinidos del modelo RBAC que establece la norma IEC 62351.

4.3.1 Permisos predefinidos

Los permisos predefinidos que la norma marca como obligatorios en la implementación del modelo RBAC son [33]:

- **LISTOBJECTS**: Permite ver los objetos que se encuentran dentro de un IED si se presenta su identificador.
- **READVALUES**: Posibilita obtener los valores, incluido si identificador, de objetos presentes dentro de un IED.
- **DATASET**: Da acceso a los servicios asociados a conjuntos de datos.
- **REPORTING**: Concede acceso a los servicios de los reportes.
- **FILEREAD**: Permite realizar acciones de lectura sobre archivos.
- **FILEWRITE**: Autoriza llevar a cabo acciones de escritura sobre archivos (contiene el servicio FILEREAD).
- **CONTROL**: Posibilita realizar acciones de control sobre objetos presentes dentro de un IED.
- **CONFIG**: Permite la configuración de objetos presentes dentro de un IED.
- **SETTINGGROUP**: Autoriza configurar de forma remota el SGCB.
- **FILEMGT**: Posibilita eliminar archivos existentes en los IEDs.
- **SECURITY**: Permite llevar a cabo acciones en objetos relacionados con la seguridad.

4.3.2 Roles predefinidos

En cuanto a los roles predefinidos los que establece la norma son [33]:

- **VIEWER**: Sólo tiene permiso para ver los objetos presentes dentro de un IED.
- **OPERATOR**: Es capaz de ver los objetos dentro de un IED y sus valores, así como ejecutar acciones de control.
- **ENGINEER**: Aparte de poder ver los objetos dentro de un IED y sus valores, puede acceder a todos los conjuntos de datos y archivos y puede configurar los equipos.
- **INSTALLER**: Tiene permisos para ver los objetos de un IED y sus valores, escribir ficheros y configurar los equipos.
- **SECADM**: Puede cambiar tanto las asignaciones de roles a permisos y de sujetos a roles, periodos de validación y ajustes de seguridad.
- **RBACMNT**: Es una subfuncionalidad del rol SCADM que puede cambiar la asignación de roles a permisos.
- **SECAUD**: Puede ver un tipo especial de archivos que son los registros de auditoría.

4.4 Mapeo de los permisos y roles predefinidos

El mapeo entre los permisos y roles predefinidos se muestra en la tabla 4.1, los significados de los signos usados son:

- **X**: Tiene asociado ese permiso.
- **C**: Puede ser necesario que sólo tenga acceso de lectura a un tipo determinado de objetos.
- $\mathbf{C_1}$: Acceso condicional de lectura a archivo de datos.
- $\mathbf{X_1}$: Acceso a archivos de datos y configuración.
- $\mathbf{X_2}$: Acceso a archivos de configuración y actualización de firmware.
- $\mathbf{X_3}$: Acceso a archivos de tipo registros de auditorías.
- $\mathbf{X_4}$: Acceso a archivos de seguridad.

Destacar que, en esta primera implementación del sistema de control de acceso basado en roles, los permisos que tienen asociados los roles de forma condicional se consideran asociados siempre. Se deja para una futura versión del mismo la asignación condicional.

Roles	Permisos										
	LISTOBJECTS	READVALUES	DATASET	REPORTING	FILEREAD	FILEWRITE	FILEMGT	CONTROL	CONFIG	SETTINGGROUP	SECURITY
VIEWER	X	C		X	C_1						
OPERATOR	X	X		X	C_1			X		X	
ENGINEER	X	X	X	X	X_1	X_1	X_1		X	X	
INSTALLER	X	X		X	X_2	X_2			X	X	
SECADM	X	X	X		X_4	X_4	X_4		X		X
RBACMNT	X	X					X_4		X		
SECAUD	X	X		X	X_3						

Tabla 4.1: Mapeo de roles a permisos predefinidos del modelo RBAC (obtenida de [33])

4.5 Análisis de la relación de los servicios ACSI con los permisos predefinidos del modelo RBAC

Para implementar los permisos en el RBAC, se necesita conocer el mapeo de permisos a los servicios ACSI que ofrece IEC 61850. Como el informe técnico que define esta relación (IEC 61850-90-19) es un borrador y no se ha publicado su versión final, la relación se puede variar temporalmente para ser usada en este proyecto hasta que se publique su versión definitiva. Por esto, se estudia el mapeo permisos a servicios propuesto por la misma y se le hace unas modificaciones para obtener una versión final que se usa en la implementación del control de acceso.

4.5.1 Relación propuesta por el borrador del reporte técnico

La relación de los permisos con los servicios ACSI que establece la norma se representa en la tabla 4.2. Destacar que para que un rol pueda realizar un servicio basta con que uno de los permisos de ese servicio tenga asociado ese rol [36].

Servicios ACSI	Permisos										
	LISTOBJECTS	READVALUES	DATASET	REPORTING	FILEREAD	FILEWRITE	CONTROL	CONFIG	SETTINGGROUP	FILEMGT	SECURITY
Associate	X	X	X	X	X	X	X	X	X	X	X
Cancel							X				
ConfirmEditSGValues									X		
DeleteFile										X	
GetAllDataValues		X									
GetBRCBValues				X							
GetDataDefinition			X								
GetDataDirectory	X										
GetDataSetDirectory	X										
GetDataSetValues		X	X								
GetDataValues		X									
GetEditSGValues									X		
GetFile					X					X	
GetFileAttributeValues					X					X	
GetGoCBValues		X		X							
GetLCBValues		X		X							
GetLogicalDeviceDirectory	X										
GetLogicalNodeDirectory	X										
GetLogStatusValues		X		X							
GetMSVCBValues	X			X							
GetServerDirectory	X										
GetSGCBValues	X								X		
GetURCBValues	X			X							
GetUSVCBValues	X			X							
Operate							X				
QueryLogAfter	X			X							
QueryLogByTime	X			X							
Select							X				
SelectActiveSG									X		
SelectEditSG									X		
SelectWithValue							X				
SetBRCBValues				X							
SetDataValues								X			

Sigue en la página siguiente.

Servicios ACSI	Permisos										
	LISTOBJECTS	READVALUES	DATASET	REPORTING	FILEREAD	FILEWRITE	CONTROL	CONFIG	SETTINGGROUP	FILEMGT	SECURITY
SetEditSGValues									X		
SetFile						X					
SetGoCBValues								X			
SetLCBValues								X			
SetMSVCBValues								X			
SetURCBValues								X			

Tabla 4.2: Mapeo de permisos a servicios (obtenido de [36])

4.5.2 Modificaciones de la relación propuesta y relación final

Los cambios realizados al mapeo de permisos a servicios establecido por la norma son:

- Incluir el servicio SetUSVCBValues con los mismos permisos que SetMSVCBValues.
- Cambiar el permiso de GetDataDefinition. La norma lo mapea al permiso DATASET y se cambia a LISTOBJECTS.

De esta forma se obtiene que la relación entre permisos que se usa en la implementación del sistema de control de accesos basado en roles es la que se tiene en la tabla 4.3.

Servicios ACSI	Permisos										
	LISTOBJECTS	READVALUES	DATASET	REPORTING	FILEREAD	FILEWRITE	CONTROL	CONFIG	SETTINGGROUP	FILEMGT	SECURITY
Associate	X	X	X	X	X	X	X	X	X	X	X
Cancel							X				
ConfirmEditSGValues									X		
DeleteFile										X	

Sigue en la página siguiente.

Servicios ACSI	Permisos										
	LISTOBJECTS	READVALUES	DATASET	REPORTING	FILEREAD	FILEWRITE	CONTROL	CONFIG	SETTINGGROUP	FILEMGT	SECURITY
GetAllDataValues		X									
GetBRCBValues				X							
GetDataDefinition	X										
GetDataDirectory	X										
GetDataSetDirectory	X										
GetDataSetValues		X	X								
GetDataValues		X									
GetEditSGValues									X		
GetFile					X					X	
GetFileAttributeValues					X					X	
GetGoCBValues		X		X							
GetLCBValues		X		X							
GetLogicalDeviceDirectory	X										
GetLogicalNodeDirectory	X										
GetLogStatusValues		X		X							
GetMSVCBValues	X			X							
GetServerDirectory	X										
GetSGCBValues	X								X		
GetURCBValues	X			X							
GetUSVCBValues	X			X							
Operate							X				
QueryLogAfter	X			X							
QueryLogByTime	X			X							
Select							X				
SelectActiveSG									X		
SelectEditSG									X		
SelectWithValue							X				
SetBRCBValues				X							
SetDataValues								X			
SetEditSGValues									X		
SetFile						X					
SetGoCBValues								X			
SetLCBValues								X			

Sigue en la página siguiente.

Servicios ACSI	Permisos										
	LISTOBJECTS	READVALUES	DATASET	REPORTING	FILEREAD	FILEWRITE	CONTROL	CONFIG	SETTINGGROUP	FILEMGT	SECURITY
SetMSVCBValues								X			
SetURCBValues								X			

Tabla 4.3: Relación final de los servicios con sus permisos

4.6 Análisis del código ya disponible

Para conocer las herramientas para la implementación del RBAC, se analiza el código del que se dispone para este proyecto. Para este fin, se dispone del siguiente código ya programado:

- Código fuente del servidor IEC 61850 de los IEDs.
- Librería opensource para el cliente IEC 61850.

4.6.1 Código fuente del servidor IEC 61850 de los IEDs

El código fuente del servidor IEC 61850 de los IEDs tiene las siguientes características:

- Programado en C. Como se desea implementar directamente el sistema de control de accesos en él, las funciones de este sistema se programan también en C.
- Gestiona las peticiones a los servicios ACSI a través de sus peticiones de servicios MMS. Por ello, para implementar el RBAC en los distintos servicios ACSI hay que saber a qué servicios MMS son mapeados.
- Dispone de herramientas para diferenciar distintos servicios ACSI con el mismo mapeo MMS.
- Puede compilarse con un compilador basado en GGC, tanto como para ser ejecutado en modo de depuración a través del IDE de *eclipse* como para ser implementado en un IED.

4.6.2 Librería opensource para el cliente IEC 61850

En cuanto a la librería de la que se hace uso para la programación del cliente IEC 61850 es una opensource programada en C (es posible descargarla a través del enlace `https://libiec61850.com/`). Esta librería contiene funciones tanto para programar un cliente

como un servidor IEC 61850, utilizando en este trabajo las relativas al cliente. Para este fin la librería incluye funciones para:

- Enviar peticiones de servicios ACSI sobre su mapeo MMS.
- Recibir la respuesta del servidor de la petición.
- Si el servidor devuelve error de la petición, identificar su causa.
- Mostrar los datos de la respuesta del servidor por pantalla.

De esta forma, esta librería se puede usar para programar un cliente IEC 61850 que envíe peticiones de servicios a un servidor y analice su respuesta para validar el RBAC, que se implementa en este proyecto. También al estar esta escrita en el mismo lenguaje de programación que el código fuente del servidor, hace más sencillo el desarrollo del proyecto, ya que se pueden desarrollar y compilar con las mismas herramientas.

4.7 Diseño y análisis de los archivos necesarios para el sistema de gestión de permisos

Para que el RBAC funcione correctamente, necesita una serie de ficheros que se analizan y diseñan en esta sección.

4.7.1 Diseño del archivo .csv para inicializar las IPs con sus roles

El primer archivo que se necesita es un archivo para el mapeado sujeto a rol. En este proyecto, el sujeto (cliente) viene representado por la dirección IP con la se trata de conectar al servidor. Para ello, se decide hacer un archivo .csv.

Este archivo tiene que permitir las siguientes acciones:

- Incluir las direcciones IPs en formato IPv4.
- Asignar los permisos tanto en formato cadena de texto como en formato número entero.
- Dar varios permisos a una misma IP, aunque todos con en el mismo formato.

De esta manera, como cabecera el fichero se diseña siendo su primera línea:

IP;Tipo Rol (1 Int, 0 string);Roles

Después el archivo tiene una línea por cada IP a la que se desea asignar roles. Cada línea tiene el siguiente formato:

IP; 0 o 1; Rol1,Rol2,rol3...

En donde:

- IP: Dirección IP del cliente al que asignar los roles.
- 0 ó 1: Sí es un 0 los roles tendrán formato cadena de texto y sí es 1 entero.
- Rol1,Rol2,rol3...: Los roles asignados a esa IP separados por comas.

Un ejemplo de una implementación de este archivo es:

IP;Tipo Rol (1 Int, 0 string);Roles
192.168.12.1;1;2,3
192.168.16.12;0;VIEWER,OPERATOR

A la hora de implementar este diseño, hay que tener en cuenta que tiene que cumplir las limitaciones con las que se programan las funciones de gestión de las IPs (sección 5.1.1).

4.7.2 Análisis y modificación del archivo RBAC para la relación de los roles con los permisos

Para el mapeo sujeto a rol, se usa un archivo ya definido anteriormente para cuestiones de ciberseguridad de la norma IEEE 1686, donde se hace uso también de los roles definidos en la norma IEC 62351. A este archivo hay que modificarlo para ajustarlo al objetivo del presente trabajo.

4.7.2.1 Análisis del archivo ya existente

El archivo RBAC actual es un documento de texto que contiene la mayor parte de la información para la configuración del RBAC implementado en los IEDs, pero la parte importante en este contexto es la de los roles.

La información que se de cada rol en este archivo de texto es:

- Su nombre en cadena de texto.
- Un número entero que se comporta como su identificador.
- Si son fijos (no pueden ser modificados) o configurables (pueden ser modificados).
- Número entero con signo de 64 bits que representa la máscara de permisos IEEE 1686 del rol. En ella cada permiso está representado por un bit, si el bit es 1 el permiso está contenido en el rol y si es 0 no está contenido en el rol.

- Número entero con signo de 64 bits que representa la máscara de permisos IEEE 1686 de emergencia del rol. Esta es lo mismo que la anterior máscara, pero se tiene en cuenta sólo en situaciones de emergencia.

4.7.2.2 Modificación del archivo para incluir el mapeo de permisos a roles del modelo RBAC

Para la adaptación del archivo RBAC, para que se tenga en cuenta los permisos de la norma IEC 62351, como se van a usar los mismos roles que con la norma IEE 1686, se añade al diseño del fichero la máscara de permisos y de permisos de emergencia IEC 62351 de cada rol. En esta primera implementación del RBAC sólo se hace uso de la máscara de permisos, dejando la de emergencia para futuras implementaciones.

4.7.3 Diseño del archivo .csv para establecer los permisos de cada servicio

En cuanto a la gestión de los permisos de cada rol, igual que en la sección 4.7.1, se opta por un archivo .csv. Este tiene que permitir establecer:

- El bit que ocupa cada permiso dentro de la máscara de permisos.
- Los permisos asociados a cada servicio ACSI.

Para cumplir el propósito, el archivo creado tiene la siguiente cabecera:

"Servicio\Permiso,número bit";,Permiso 1,Numero Bit1;Permiso 2,Numero Bit 2....

Donde el primer elemento (hasta ";") es una cabecera meramente informativa. A continuación habrá un elemento por permiso separado por ";" con el nombre del permiso, el carácter "," y el número de bit del permiso dentro de la máscara.

Después, se establece una fila por cada servicio con el siguiente formato:

Nombre Servicio;0 ó 1;0 ó 1;..

Es decir, el primer elemento de la fila es el nombre del servicio y a continuación, por cada columna correspondiente a un permiso, está el carácter "0" si el servicio no tiene asociado ese permiso o un "1" si lo tiene asociado. Un ejemplo de este archivo, suponiendo que sólo se asignan dos permisos a dos servicios, es:

Servicio\,número de bit;LISTOBJECTS,1;READVALUES,2
Associate;1;1
GetAllDataValues;0;1

4.8 Diseño de los archivos usados en la aplicación cliente IEC 61850

Con el fin de que la aplicación cliente IEC 61850 tenga las funcionalidades deseadas, se diseñan dos archivos .csv distintos con:

- La información del servidor al que conectarse.
- Los servicios sobre los que enviar su petición.

4.8.1 Archivo .csv con la información del servidor al que conectarse

El objetivo de este fichero es dar la información del servidor necesaria para que la aplicación pueda enviarle las peticiones de los distintos servicios MMS. Esta información del servidor incluye:

- Su dirección IP.
- Las referencias a objetos de sus objetos de datos necesarios.
- La dirección de un fichero suyo al que cualquier cliente IEC 61850 tiene acceso de lectura.
- La dirección de un archivo del ordenador donde se ejecuta la aplicación cliente que pueda ser enviado al servidor

Para esto, el archivo tiene la siguiente cabecera puramente informativa:

Objeto Referenciado o IP;Referencia o número de IP;FC (de ser necesaria)

Después, tiene una línea por dato necesario con la siguiente información:

Nombre del dato; información del dato; FC.

Donde la información del dato puede ser la referencia del objeto, la IP o la dirección de un archivo y el FC no es necesario en todos los datos.

4.8.1.1 Datos que debe contener el archivo

Para que este archivo se use de forma correcta, se establecen los datos que son necesarios que incluya para que la aplicación pueda enviar peticiones de todos los servicios. En cuanto a los datos que no necesitan el FC, el nombre de estos datos y la información del dato que tiene que proporcionar son:

- **IP**: Dirección IP del servidor.

- **DataSet**: Referencia del objeto de un conjunto de datos.
- **BRCB**: Referencia de objeto de un *Buffered-report-control-block*.
- **BUCB**: Referencia de objeto de un *Unbuffered-report-control-block*.
- **LogicalDevice**: Referencia a objeto de un dispositivo lógico.
- **RemoteFile**: Dirección de un archivo en el servidor al que el cliente tenga acceso de lectura.
- **LocalFile**: Dirección de un archivo local que se pueda enviar al servidor.
- **GOOSE**: Referencia a objeto de un GoCB.
- **SGCB**: Referencia a objeto del setting-group-control-block del servidor.
- **ControlSelect**: Referencia a objeto de un tipo de objeto de control llamado *sbo-with-normal-security* usado en el servicio Select.
- **ControlSelectWithValue**: Referencia a objeto de un tipo de datos de control de nombre *sbo-with-enhanced-security* usado en el servicio SelectWithValue.
- **ControlOperate**: Referencia a objeto de un tipo de datos de control de nombre *direct-with-normal-security* que se hace uso en el servicio Operate.
- **Log**: Referencia a objeto de un registro.
- **MSVCB**:Referencia a objeto de un Multicast-sample-values-control-block.
- **USVCB**: Referencia a objeto de un Unicast -sample-values-control-block.

Por otro lado, los datos que sí necesitan de FC son:

- **DataObject**: Referencia a un objeto de datos o un atributo de datos.
- **LogicalNode**: Referencia de un nodo lógico.

Con todo esto, un ejemplo de las primeras tres líneas de la implementación de este archivo este archivo es:

Objeto Referenciado o IP;Referencia o número de IP;FC (de ser necesaria)
IP;192.168.180.1;
DataObject;MD3eCTRL/ParGGIO1.Beh;SV

4.8.2 Archivo .csv con los servicios de los que enviar sus peticiones

El fichero .csv con los servicios a enviar sirve para establecer de qué servicios se desea que el cliente lance peticiones. Para ello, tiene la siguiente línea de cabecera:

Nombre del servicio;Mostrar resultados de poder (S o N)

Después, habrá una línea por servicio con la siguiente información separada por el carácter ";":

- El nombre del servicio.
- Sí el servicio no devuelve ningún dato, no pondrá nada. De lo contrario, se puede poner uno de estos caracteres:
  - **S**: Se desea que se muestren por pantalla los datos que devuelve.
  - **N**: No se desea que los muestre. Si se pone cualquier otro carácter no reconocido, por defecto no se muestran los resultados.

Destacar que todos los servicios se llamarán por los nombres establecidos en la sección 4.1, salvo el servicio GetServerDirectory que, como tiene dos mapeos MMS diferentes, se puede llamar con los nombres:

- **GetServerDirectoryL** devuelve los dispositivos lógicos del servidor.
- **GetServerDirectoryF** devuelve los ficheros del servidor.

Un ejemplo de una posible implementación de este archivo es:

Nombre del servicio;Mostrar resultados de poder (S o N)
GetDataValues;S
SetURCBValues;
GetServerDirectoryL;S

4.9 Diseño de los paquetes de funciones necesarios para la implementación del RBAC

Para que el RBAC tenga la funcionalidad deseada, es necesario diseñar correctamente las funcionalidades de los paquetes de funciones que se usarán en su implementación. De esta manera, se diseñan cuatro paquetes cuyas funcionalidades son:

- Gestión de las direcciones IPs.

- Gestión de los roles del archivo RBAC.
- Gestión de los permisos de los servicios ACSI.
- Actualización de la configuración del RBAC.

4.9.1 Paquete para la gestión de las direcciones IPs

El objetivo de este paquete es gestionar un archivo binario donde se guardarán las IPs en formato entero de 32 bits sin signo junto con sus roles (en formato cadena de texto o entero de 16 bits). Este tiene que tener funciones para:

- Añadir una nueva IP al fichero con los roles en cualquiera de los dos formatos.
- Modificar los roles de la IP dentro del fichero.
- Eliminar una IP del fichero.
- Cargar las IPs con sus roles del archivo diseñado en la sección 4.7.1.

4.9.2 Paquete para la gestión de los roles del archivo RBAC

El propósito de este grupo de funciones es la gestión de los roles del archivo RBAC modificado de la sección 4.7.2 para la asignación de los permisos de dichos roles. Las funcionalidades que debe incluir este paquete son:

- Guardar los roles del archivo con su información en una estructura.
- Buscar información de un rol dentro de la estructura anterior.
- Usando la estructura de roles y el archivo con IPs gestionado con el anterior paquete de funciones, crear una estructura de IPs con sus roles y la información de ellos.
- Poder obtener la máscara de permisos de una IP dentro de la estructura de IPs con sus roles.

4.9.3 Paquete para la gestión de los permisos de los servicios ACSI.

La finalidad de este paquete de funciones es saber si una dirección IP puede acceder a un determinado servicio a través de:

- La estructura de IPs con sus roles del paquete anterior.
- El archivo .csv con la relación de los permisos para cada servicio ACSI diseñado en la sección 4.7.3.

4.9.4 Paquete para la actualización de la configuración del RBAC

La funcionalidad de este paquete es que permita que se pueda cambiar la configuración del RBAC mientras está en ejecución. Para ello, debe de tener funciones con capacidad de:

- Buscar en una carpeta para la actualización de la configuración si se ha añadido un archivo nuevo de los definidos en la sección 4.7.
- Comprobar si los archivos nuevos tienen formato correcto y son compatibles con el resto de la configuración del RBAC.
- Si el archivo de actualización es correcto, debe de:
  - Actualizar la configuración del RBAC.
  - Mover el nuevo archivo de configuración a la carpeta de validado, remplazando al que estuviera anteriormente.
- Si el archivo de configuración no es válido, eliminarlo de la carpeta de actualización.
- Dejar grabado en un archivo interno del IED cada vez que se intenta actualizar la configuración del RBAC. Este debe incluir:
  - Si se ha podido actualizar correctamente
  - En caso contrario la razón por la cual no se ha podido actualizar correctamente.

4.10 Diseño de las funciones para la aplicación cliente

El objetivo de las funciones para la aplicación cliente es que sirvan para el envío de las peticiones de los distintos servicios y es necesaria una de ellas por cada servicio. Estás deben cumplir las siguientes características:

- Mostrar por pantalla si el servidor ha procesado correctamente el servicio o no.
- Las funciones asociadas a los servicios que obtienen datos de los equipos tienen que:
  - Disponer de un parámetro booleano de entrada para poder mostrar los resultados por pantalla.
  - Si la función corresponde a un servicio que obtienen datos necesarios para el envío de las peticiones de otros servicios, deben retornar estos datos en una estructura. Estos servicios se analizan en la sección 4.10.1.

- Las funciones asociadas a servicios que necesitan de información para poder enviar su petición (sección 4.10.1) tienen que tener un parámetro de entrada para poder pasar la estructura con esta información.

4.10.1 Diseño de los servicios que necesitan datos para realizar su petición

Para simplificar el envío de las peticiones de estos servicios, como lo que interesa es la parte de comunicaciones y no la realización de esos servicios, se usan los datos obtenidos de otros servicios para lanzar las peticiones de los servicios que necesitan información. Esto evita incluir en el código la estructura de datos para que hagan su funcionamiento, consiguiendo las siguientes ventajas:

- La aplicación programada sirve para cualquier modelo de IED sin tener que modificar su código.
- El código de la aplicación es mucho más ligero y fácil de entender.
- Se evitan fallos que pueden surgir al codificar la estructura de datos.
- Se simplifica la programación de la aplicación.

Con ello, revisando el API de la librería usada para la aplicación [37] y la parte 7-2 de la norma IEC 61850 [30], se obtiene la tabla 4.4 con la relación de los servicios que dan la información necesaria para poder lanzar las peticiones de los servicios que necesitan información para ello.

Servicio	Necesita la información del servicio
SetDataValues	GetDataValues
SetBRCBValues	GetBRCBValues
SetURCBValues	GetURCBValues
SetGoCBValues	GetGoCBValues
SetEditSGValues	GetEditSGValues
SetMSVCBValues	GetMSVCBValues
QueryLogAfter	GetLogStatusValues
QueryLogByTime	GetLogStatusValues
SetLCBValues	GetLCBValues
SetUSVCBValues	GetUSVCBValues

Tabla 4.4: Servicios que necesitan información de otro para enviar su petición (obtenido de analizar [37] y [30])

Por último, se observa que el envío de las peticiones de los servicios del modelo de control también necesita de información del IED para realizarse, pero para obtenerla la propia librería incluye una función.

4.11 Diseño de cómo establecer los servicios de los que la aplicación cliente envía las peticiones

Como se desea que la aplicación cliente cumpla las especificaciones detalladas en la sección 2.1, se diseña que se pueda establecer los servicios de los que envía las peticiones de las siguientes formas:

- **Solicitud de los servicios desde el teclado**: De esta forma, la aplicación hace uso del archivo .csv con la información del servidor de la sección 4.8.1, para que el usuario pueda indicar qué servicio quiere mandar cada vez, introduciendo su nombre por teclado. También la aplicación debe finalizar si se introduce "FIN". Esta está orientada a que el usuario pueda enviar peticiones de ciertos servicios en el momento que se desee.
- **Se envían las peticiones de los servicios que se establece en el archivo de configuración de los servicios de la sección 4.8.2**: Para ello, la aplicación también hace uso del archivo con la información del servidor. Esto sirve para que se puedan diseñar ciertas pruebas en los servidores en los que se lancen ciertas peticiones en un orden concreto y que estas se puedan compartir fácilmente enviando los archivos.
- **Se envía la petición de un único servicio mediante parámetros de ejecución**: Con esto, no se necesitará ningún archivo, ya que toda información del servicio se pasa por parámetros de ejecución de la aplicación. Este último está orientado a automatizar pruebas con scripts que sólo necesiten el ejecutable de la aplicación sin ningún fichero extra.

Desarrollo de la solución

5

En este apartado se presenta la implementación del RBAC basado en la parte 8 de la norma IEC 62351 para el acceso a los servidores de los IEDs, su desarrollo y programación en lenguaje C, así como la aplicación cliente IEC 61850 necesaria para validar el RBAC implementado.

5.1 Programación de las funciones para el RBAC.

El primer paso para el desarrollo de la solución es programar las funciones para que los paquetes diseñados en la sección 4.9 cumplan la funcionalidad deseada.

5.1.1 Funciones para la gestión de las IPs

En esta sección se indica la programación de las funciones del paquete de gestión del archivo binario de las IPs diseñado en 4.9.1. Para gestionar el archivo binario con la información de las IPs, se programan las siguientes funciones:

- GrabarIPFichero.
- ModificarIPFichero.
- EliminarIPFichero.

Estas funciones están declaradas en el fichero de cabecera FuncionesIP.h. Señalar que tienen las siguientes limitaciones:

- Sólo se permite un máximo de 64 IPs en el fichero.
- No se puede guardar una IP si ya existe en el fichero una igual.
- El número de roles asociados a una IP tiene que ser mayor que 1 y menor o igual que 8.
- Si el texto de un rol en formato cadena de texto tiene algún espacio, este se ignora.
- El número máximo de caracteres de una cadena de texto que representa un rol son 15 (16 con el fin de cadena).
- No puede haber roles repetidos.

5.1.1.1 Función GrabarIPFichero

Esta función graba una IP nueva en un archivo binario dado (si no existe se crea). La sintaxis de la función es la siguiente:

```
int GrabarIPFichero(const char sIP[],const char sRoles[],
                    const short int iRoles[], const char ruta[],
                    const int iNumeroRolesInt);
```

En donde sus parámetros de entrada son:

- sIP: La IP que se desea guardar en formato cadena de texto. La IP tendrá forma IPv4. Ejemplo: "192.168.191.167".
- sRoles: Cadena de texto de los roles de la IP separadas por comas si se desea guardar sus roles en formato de cadena de texto. Ejemplo "SECURE,ADMIN,OPERATOR". Si, por el contrario, se desea guardar los roles en entero será un puntero NULL.
- iRoles: Será un array de números de tipo short int correspondientes a los roles de la IP. Si por el contrario se desean guardar los roles en formato string será un puntero NULL. Hay que tener en cuenta que si ambos (sRoles e iRoles) o ninguno son NULL, la función devuelve un error.
- ruta: Dirección del archivo binario donde se desea guardar la IP.
- iNumeroRolesInt: Este parámetro sólo será significativo si los roles se desean guardar en formato short int. En ese caso este parámetro será el número de roles en entero que se quieren guardar.

Esta función devuelve un código de error en formato entero, estos dependen del tipo con el que se quiere guardar los roles. Los comunes a ambos tipos y su significado son:

- 0: No ha habido ningún fallo y la IP se ha guardado correctamente. Hay que tener en cuenta que solo en este caso se guarda la IP en el fichero.
- -1: La IP que se le ha pasado como parámetro de entrada no es válida.
- -2: No es capaz de abrir o crear el archivo en la ruta especificada.
- -3: La IP ya se encuentra guardada.
- -4: Se ha alcanzado el límite de IPs guardadas en el fichero dado.
- -8: sRoles y iRoles tienen ambos valores NULL o ninguno tiene valor NULL.

Los códigos de error para cuando se desea guardar los roles como cadena de texto y su significado son:

- -5. Un rol tiene más de 15 caracteres sin contar el fin de cadena.

- -6: Se intenta guardar una IP con más de 8 roles asociados.
- -7: No se ha pasado ningún rol en la cadena de texto.
- -8: Hay dos roles con el mismo nombre.

Finalmente, el retorno de la función y su significado cuando se desean guardar los roles como short int son:

- -5: Se le han pasado un número de roles para la IP mayor que 8.
- -6: No se le ha pasado ningún rol para guardar.
- -7: Hay dos roles con el mismo número.

5.1.1.2 Función ModificarIPFichero

Esta función modifica los roles que una IP guardada en un fichero dado tiene asociados. Al igual que en la anterior se le pueden pasar los roles con formato short int o como cadenas de texto. Hay que tener en cuenta que esta función cambia los roles que tiene la IP asociados por los nuevos sin tener en cuenta el tipo con el que se tenían esos roles guardados. Su sintaxis es:

```
int ModificarIPFichero(const char sIP[],const char sRoles[],
                       const short int iRoles[],
                       const char ruta[],
                       const int iNumeroRolesInt);
```

Sus parámetros de entrada son los mismos y cumplen la misma función que los de la función anterior. En cuanto a los códigos de error, también tiene valores de retorno distintos dependiendo si los roles se les pasa en formato string o en entero. Mientras que los valores específicos para los roles en cadena o en short int son los mismos que en la función anterior, los generales y su significado son:

- 0: Los roles de la IP se han modificado correctamente. Para cualquier otro valor los roles no se han podido modificar.
- -2: No se ha podido abrir el archivo.
- -3: No se ha encontrado la IP que se desea modificar en el archivo.
- -8: sRoles y iRoles tienen ambos valores NULL o ninguno tiene valor NULL.

5.1.1.3 Función EliminarIPFichero

Esta función sirve para eliminar una IP y su información de un archivo binario dado. Su sintaxis es la siguiente:

```
int EliminarIPFichero(const char sIP[], const char sRuta[]);
```

La dirección IP que se desea eliminar se indica en el parámetro sIP y la ruta donde se encuentra el fichero binario está indicado por sRuta, ambos parámetros en formato cadena de texto.

Para eliminar la IP del fichero, la función realiza el siguiente procedimiento:

- Vuelca las IPs a un archivo temporal sin escribir la IP que se desea eliminar.
- Elimina el archivo original de IPs.
- Renombra el archivo temporal al nombre del fichero original.

Teniendo en cuenta el proceso, los códigos de error que devuelve esta función y lo que representan son:

- 0: No ha habido ningún fallo y se ha podido eliminar la IP correctamente. Sólo en este caso todo el proceso para eliminar la IP se realiza correctamente.
- -1: La IP que se le ha pasado a la función tiene un formato no válido.
- -2: La función no ha sido capaz de abrir el archivo .dat con la información de las IPs.
- -3: No se ha encontrado la IP que se desea eliminar en el archivo. En este caso y los dos anteriores, no se intenta eliminar la IP.
- -4: No se ha podido eliminar el archivo original. En este caso el archivo original quedaría sin fallos y habría un archivo de nombre “temp.dat.[en] el directorio raíz de donde se esté ejecutando la aplicación con todas las IPs menos la que se desea borrar.
- -5: Se ha eliminado el archivo original correctamente pero no se ha podido renombrar el archivo temporal. En este caso el único rastro de las IPs estaría en el archivo "temp.dat”.

5.1.1.4 Función IPsCSV2ArchivoDAT

Esta función configura las IPs con sus roles desde un archivo .csv (diseñado en el apartado 4.7.3) hasta el archivo binario que usa este paquete. La sintaxis de esta función es:

```
int IPsCSV2ArchivoDAT (const char sRutaCSV[],
                       const char sRutaDAT[]);
```

Donde sRutaCSV es la ruta del archivo .csv con la información de las direcciones IPs con sus roles y sRutaDAT es la ruta del archivo binario donde se quiere guardar la información de las IPS. Hay que tener en cuenta que si existe el archivo binario en la ruta se va a eliminar su contenido antes de guardar la información de las IPs.

Por otro lado, los códigos de error de la función son:

- 0: Si el archivo binario se ha creado correctamente con las IPs. En cualquier otro caso, la función no crea ningún archivo binario y escribe por pantalla la información de qué línea del archivo .csv ha producido el error.
- -1: La función no es capaz de abrir el archivo .csv.
- -2: Algunas de las IPs no son válidas.
- -3: Error al guardar los roles en cadena de texto de una IP.
- -4: Error al guardar los roles en formato entero de texto de una IP.
- -5: La cadena de texto de los roles de una IP tiene demasiados caracteres.
- -6: Una IP tiene demasiados roles en formato entero.

5.1.2 Funciones para la gestión de los roles del archivo RBAC

En cuanto al desarrollo de las funciones para la gestión de los roles del archivo RBAC diseñado en la sección 4.9.2, se crean las siguientes funciones:

- CargarRolesFicheroRBAC.
- BuscarInformacionRolEstrcutura.
- VisualizarEstructuraRoles.
- FicheroIp2EstructuraIPRol.
- ObtenerMascaraPermisosIP.
- VisualizarDatosIPEstrucura.
- VisualizarEstructuraIPs.

Estas funciones se encuentran declaradas en el archivo "Funciones_RBAC.h".

5.1.2.1 Función CargarRolesFicheroRBAC

Esta función analiza el archivo RBAC y carga sus roles en una estructura interna del fichero de cabecera del paquete. Esta está formada por un entero con el número de roles guardados y un array de estructuras con la siguiente información de cada rol:

- Un entero que indica si el rol es fijo (definido por #define rolFijo 0) o configurable (definido por #define rolConfigurable 1).
- Una cadena de texto con el nombre del rol.

- Un número de identificación del rol en formato short int.
- Dos long int, uno con la máscara de permisos IEC 61850 del rol y otro con su máscara de permisos de emergencia IEC 61850.

Las limitaciones de esta estructura son:

- No puede tener más de 64 roles, si en el fichero se superan estos la función no los añade a la estructura.
- Ningún rol puede tener el nombre o el ID repetido.

La función tiene la siguiente sintaxis:

```
int CargarRolesFicheroRBAC(const char sRuta[]);
```

Donde sRuta es la cadena de texto con la ruta donde se encuentra el archivo RBAC.

Los códigos de error de esta función son:

- 0: Los roles se han guardado correctamente.
- -1: No se ha podido acceder al archivo RBAC.config.

Por otro lado, si algún rol no se puede guardar por no cumplir las limitaciones de la estructura, se escribe su información en la pantalla sin afectar al resto de roles.

5.1.2.2 Función BuscarInformacionRolEstrcutura

Esta función busca en la estructura anterior la información de un rol por su nombre en cadena de texto o por su ID como numérico. La función con sus parámetros es:

```
int BuscarInformacionRolEstrcutura (char sNombreRol[],
                                    short int* piIDRol,
                                    long int* iMascaraPermisos,
                                    long int* iMascaraEmergencia,
                                    int* iTipoRol );
```

Si el parámetro sNombreRol es una cadena de rol nulo (establecida con el define _61850_STRING_ROL_NULO), para buscar el rol en la estructurase usará el entero apuntado por piIDRol como ID del rol. De lo contrario, se busca el rol por nombre igual a la cadena de texto sNombreRol.

De encontrarse el rol en la estructura, se devuelve su información en los correspondientes punteros que se la pasa como parámetros de entrada (incluyendo su ID si se le llama por nombre o su nombre si se le llama por ID). Los códigos de error que devuelve la función son:

- 0: Ha encontrado el rol y ha copiado la información en los punteros correspondientes.
- -1: No se ha creado la estructura de roles con la función CargarRolesFicheroRBAC.

- -2: No se ha encontrado el rol en la estructura.

5.1.2.3 Función VisualizarEstructuraRoles

La finalidad de esta función es mostrar por pantalla la información contenida en la estructura anterior. Su sintaxis es la siguiente:

```
void VisualizarEstructuraRoles();
```

Como se observa, la función no devuelve ningún dato ya que se usa solamente al depurar el código.

5.1.2.4 Función FicheroIp2EstructuraIPRol

Esta función, a través del fichero binario creado por las funciones de gestión de IP y la estructura de roles creada en la anterior función, crea una estructura formada por un array y el número de elementos del array en entero (esta estructura es interna también del archivo de cabecera). Cada elemento del array corresponde a una IP del fichero y se incluye la siguiente información:

- El número de IP en long unsigned int.
- El número de roles que tiene la IP.
- Un array de estructuras (una por rol) con la misma información que la estructura creada en CargarRolesFicheroRBAC.

La sintaxis de la función es la siguiente:

```
int FicheroIp2EstructuraIPRol (const char ruta);
```

Donde la dirección del archivo con las IPs se le indica a la función a través de la cadena de texto ruta. Los códigos de error de la función son:

- 0: No ha habido ningún problema y las IPs se han guardado correctamente.
- -1: La función no es capaz de abrir el archivo de IPs.
- -2: No se ha inicializado la estructura de roles.
- -3: Algún rol de alguna IP no se ha podido encontrar en la estructura de roles. En este caso se muestra el rol por pantalla y se borra toda la información de la estructura de IPs.
- -4 El archivo de IPs no contenía ninguna.

5.1.2.5 Función ObtenerMascaraPermisosIP

Esta función busca la máscara de Permisos y la máscara de permisos de emergencia asociada a una dirección IP. La función con sus parámetros es:

```
long int ObtenerMascaraPermisosIP(const char sIP[],
                                  long int* piMascaraEmergencia);
```

Donde sIP es la IP en forma de cadena de texto en formato IPv4 y piMascaraEmergencia un puntero a la zona de memoria donde guardar la máscara de permisos de emergencia de la IP. Los valores que devuelve la función son:

- La máscara de permisos de la IP: no ha habido ningún fallo, en este caso devolvería la máscara de permisos de emergencia en piMascaraEmergencia.
- 0: No se ha podido encontrar la máscara de permisos de la IP.

5.1.2.6 Función visualizarDatosIPEstrucura

Esta función visualiza por pantalla los datos de los roles de una IP que se encuentran dentro de la estructura de IPs. Su sintaxis es:

```
int visualizarDatosIPEstrucura(const char sIP[]);
```

Donde sIP es la dirección de IP en cadena de texto. Los códigos de error de la función son:

- 0: no ha habido ningún error y se ha visualizado correctamente los roles de la IP.
- -1: La estructura no tiene ninguna IP guardada.
- -2: LA IP no es válida.
- -3: No se ha encontrado la IP especificada.

5.1.2.7 Función VisualizarEstructuraIPs

La finalidad de esta función es mostrar por pantalla la información contenida en la estructura anterior de IPs con sus roles. Su sintaxis es la siguiente:

```
void VisualizarEstructuraIPs();
```

Esta función no devuelve ningún dato y su uso está limitado a depurar el código.

5.1.3 Funciones para la gestión de los permisos de los servicios ACSI

En cuanto a la implementación de las funciones para la gestión de los permisos de los servicios ACSI (paquete diseñado en la sección 4.9.3), estas tienen el fin de poder establecer

si una determinada IP puede tener permiso para acceder a un determinado servicio. Estas funciones son:

- InicializarPermisos.
- visualizarServiciosGuardados.
- AccesoServicioConcedido.
- PermisoRealizarAccion.

Éstas se encuentran declaradas en el fichero de cabecera “Funciones_gestion_permisos.h”.

5.1.3.1 Función InicializarPermisos

Esta función carga la relación de los permisos de cada servicio ACSI desde el archivo .csv diseñado en la sección 4.7.3 a una estructura interna. Ésta es usada por el resto de funciones de este paquete para funcionar correctamente. La sintaxis de la función es:

```
int InicializarPermisos(const char sRutaCSV[]);
```

Donde la ruta del archivo .csv se le indica a través del parámetro sRutaCSV en formato cadena de texto. Los códigos de error de esta función son:

- 0: La estructura se ha generado correctamente.
- -1: No se ha podido abrir el archivo .csv.
- -2: No había ningún servicio en el archivo .csv.

5.1.3.2 Función visualizarServiciosGuardados

La finalidad de esta función es visualizar los servicios y su máscara de permisos de la estructura creada con la función anterior. Su sintaxis es:

```
void visualizarServiciosGuardados();
```

Esta función no tiene valores de retorno, ya que sólo se usa para la depuración del código.

5.1.3.3 Función AccesoServicioConcedido

Esta función permite comprobar que una determinada máscara de permisos puede acceder a un determinado servicio. La sintaxis de la función es la siguiente:

```
int AccesoServicioConcedido (const char sServicio[],
                             const long int
                                iMascaraPermisosSolicitante);
```

Donde sServicio es el nombre del servicio al que se desea acceder en forma de cadena de texto e iMascaraPermisosSolicitante es la máscara de permisos que se quiere comprobar si tiene acceso a un determinado servicio. Los valores que devuelve la función son:

- _61850_ACCION_DENEGADA: El solicitante no puede acceder al servicio.
- _61850_ACCION_PERMITIDA: El solicitante puede acceder al servicio.

Donde ambos son enteros cuyos valores son definidos en el propio fichero de cabecera de las funciones.

Destacar que si no se puede encontrar el servicio en el archivo .csv, se indica que se puede acceder a él, pero sale un aviso por pantalla. Por otro lado, para que la máscara pueda acceder a un servicio basta con que tenga un permiso común al servicio que se desea acceder.

5.1.3.4 PermisoRealizarAccion

Esta función tiene las siguientes funcionalidades:

- Comprueba si una determinada IP puede tener acceso a un determinado servicio ACSI.
- Es capaz en diferenciar entre distintos servicios con un mismo mapeo MMS, mediante la referencia a objeto o una referencia al servicio (número entero).
- Si la dirección IP no tiene permiso de acceder al servicio solicitado,lo escribe con el nombre del servicio y su número de dirección IP en un archivo interno del IED. De esta manera se controla cuando un cliente ha intentado realizar una acción para la que no tiene permiso.

La sintaxis de la función es la siguiente:

```
int PermisoRealizarAccion(const char sIP[],char sObjRef[],
                          int iMapeoMMS,int iRefServicio);
```

En donde los parámetros de la función son:

- sIP: cadena de texto con la dirección IP que solicita acceder al determinado servicio ACSI.
- sObjRef: cadena de texto con el ObjReference de una determinada llamada a un servicio (no es necesario introducirlas en caso de que con el mapeo MMS y/o la referencia al servicio se pueda diferenciar el servicio ACSI).
- iMapeoMMS: un entero que representa un determinado mapeo a un servicio MMS.
- iRefServicio: un entero que sirve de referencia de un determinado servicio ACSI (No es necesario en todos los casos).

Los valores de retorno de la función son:

- _61850_ACCION_PERMITIDA: El solicitante tiene permiso para acceder al servicio ACSI.

- _61850_ACCION_DENEGADA: El solicitante no tiene permiso para acceder al servicio ACSI.
- -4: No se puede obtener la máscara de permisos de la IP.
- -5: No se ha podido obtener el servicio asociado a la petición.

5.1.4 Función para la inicialización del RBAC

Para hacer más sencilla la implementación del RBAC, se programa una función que permita inicializar todo lo necesario para el sistema, haciendo uso del resto de funciones. La función que se crea tiene de nombre inicializarConfiguracionRBAC y su sintaxis es:

```
int inicializarConfiguracionRBAC(const char sRutaCSVIPs[],
                                 const char sRutaDATIPs[],
                                 const char sRutaRBAC[],
                                 const char sRutaCSV[]);
```

En donde:

- sRutaCSVIPs: Ruta del archivo .csv con los roles asociados a cada IP.
- sRutaDATIPs: Ruta donde se quiere guardar el archivo binario de la relación de las IPs con sus roles. Si existe un archivo en esa misma ruta sus datos se borrarán antes de grabar en él las relaciones entre las IPs.
- sRutaRBAC: Ruta donde se encuentra el archivo RBAC.config".
- sRutaCSV: Ruta donde se encuentra el archivo .csv para establecer los permisos asociados a cada servicio ACSI.

Los valores que puede retornar esta función son:

- 0: Ningún fallo en su ejecución.
- -1: Fallo al crear el archivo binario de IPs con sus roles.
- -2: Fallo al cargar los roles del archivo RBAC.
- -3: Fallo al crear la estructura de IPs con sus roles.
- -4: Fallo al crear la estructura de servicios con sus permisos.

De esta manera, para poder usar el sistema de control de acceso, basta con usar esta función para inicializarlo y la función PermisoRealizarAccion en los accesos a cada servicio para controlar las IPs que tienen acceso a ellos. Por otro lado, el resto de funciones de los paquetes se emplean para hacer modificaciones y consultas del sistema una vez inicializado.

Finalmente, destacar que esta función se encuentra declarada en el fichero de cabecera "Funciones_datos_comunes.h".

5.1.5 Paquete para la actualización de la configuración del RBAC

Para poder modificar la configuración del RBAC una vez inicializado, se programa el paquete de funciones para actualizar esta configuración. Las funciones programadas en este paquete, una por cada archivo de configuración usado en la configuración del RBAC, son:

- ComprobarActualizacionCSVIP.
- ComprobarActualizacionArchivoRBAC.
- ComprobarActualizacionCSVServicios.

Todas ellas tienen los siguientes códigos de errores:

- _61850_ACCESO_IMPOSIBLE: No se ha encontrado un nuevo archivo de configuración en la carpeta de actualización.
- _61850_ACTUALIZACION_FALLO: Se ha encontrado un archivo de configuración nuevo, pero este tiene formato incorrecto o es incompatible con la configuración actual del RBAC.
- _61850_ACTUALIZACION_OK: El archivo de configuración es correcto y se ha actualizado la configuración correctamente.

Destacar que todas ellas cumplen los requisitos establecidos en el diseño del paquete (sección 4.9.4) y se encuentran declaradas en el archivo "Funciones_actualizacion_configuracion.h".

5.1.5.1 Función ComprobarActualizacionCSVIP

Esta función sirve para comprobar si se ha cargado algún archivo .csv para especificar la relación de las IPs con sus roles en la carpeta de actualización. Su sintaxis es la siguiente:

```
int ComprobarActualizacionCSVIP (const char sRutaCSVIPUpdate[],
                                 const char sRutaCSVIPValidado[],
                                 const char sRutaDat[],
                                 const char sRutaRBAC[]);
```

En donde sus parámetros de entrada son:

- sRutaCSVIPUpdate: Dirección del archivo .csv de IPs con sus roles en la carpeta de actualización.
- sRutaCSVIPValidado:Dirección el archivo .csv de IPs validado.

- sRutaDat: Dirección del archivo .dat (que gestiona el paquete de la sección 5.1.1) con la información de las IPs con sus roles de la configuración validada.
- sRutaRBAC: Dirección del archivo RBAC con la relación de los roles y permisos validado.

Para que el nuevo archivo .csv de las direcciones IPs sea compatible con la configuración anterior, todos los roles que aparecen en él tienen que estar definidos en el archivo RBAC validado. Finalmente la función, aparte de actualizar la configuración del RBAC, también lo hace con el archivo .dat con la información de las IPs con sus roles.

5.1.5.2 Función ComprobarActualizacionArchivoRBAC

La finalidad de esta función es comprobar si hay un nuevo archivo RBAC para la relación de los roles y permisos en la carpeta de actualización. La sintaxis de esta función es:

```
int ComprobarActualizacionArchivoRBAC (const char sRutaDat[],
                                       const char sRutaRBACUpdate[],
                                       const char sRutaRBACValidado[]);
```

Los parámetros de entrada de la función son:

- sRutaDat: Dirección del archivo .dat con la información de las IPs con sus roles validadas.
- sRutaRBACUpdate: Dirección del archivo RBAC en la carpeta de actualización.
- sRutaRBACValidado: Dirección del archivo RBAC validado.

Para que el archivo RBAC nuevo sea compatible con la configuración del RBAC anterior, tiene que contener la información de todos los roles que aparecen en el archivo .dat.

5.1.5.3 Función ComprobarActualizacionCSVServicios

El objetivo de esta función es comprobar si se encuentra un nuevo archivo .csv con los permisos de cada servicio en la carpeta de actualización. La sintaxis de esta función es:

```
int ComprobarActualizacionCSVServicios (const char
                                         sRutaCSVUpdate[],
                                        const char
                                         sRutaCSVValidado[]);
```

Los parámetros de entrada de la función son:

- sRutaCSVUpdate: Dirección del archivo .csv con la relación de los permisos de cada servicio en la carpeta de actualización.
- sRutaCSVValidado: Dirección del archivo .csv con la relación de los permisos de cada servicio en la carpeta de validado.

Indicar que no es necesario que esta función compruebe que el archivo .csv con los permisos de cada servicio es compatible con la configuración actual, ya que éste es independiente de los demás.

5.2 Implementación del RBAC en el código fuente de los equipos

Mediante las funciones programadas, se implementa el sistema de control de accesos basado en roles en el código fuente de los equipos. Para que éste funcione correctamente, se implementan las siguientes funcionalidades:

- Inicialización de la configuración del RBAC.
- Securización de los diferentes servicios.
- Cambio en la configuración del RBAC una vez inicializado.

5.2.1 Inicialización de la configuración del RBAC.

La primera funcionalidad a implementar es la carga de toda la configuración del RBAC mediante información de los archivos de la sección 4.7.2, dejando el sistema listo para la securización de los servicios. Para ello, se localiza la parte del código fuente del servidor IEC 61850 que se ejecuta al arrancar el IED y se añade el siguiente código:

```
1 inicializarConfiguracionRBAC(_61850_DIRECCION_SCV_IP_ROLES,
    _61850_DIRECCION_DATOS_IP, _61850_DIRECCION_RBAC,
    _61850_DIR_CSV_PERMISOS_SERVICIOS);
```

Como se observa, basta con usar la función inicializarConfiguracionRBAC y pasarle la dirección de los archivos necesarios mediante identificadores definidos en los ficheros de cabecera de la sección 5.1.2.

Destacar que no se realiza una gestión de los errores de la función por las siguientes razones:

- La carpeta donde se guardan los ficheros no tiene acceso el usuario final del IED, sólo los trabajadores de PGA en la fase de desarrollo de los equipos.
- Si el usuario del IED quiere hacer un cambio en algún fichero de configuración lo tiene que hacer a través de los mecanismos implementados en la sección 5.2.3. Estos comprueban que los archivos son correctos antes de enviarlos a la carpeta donde se guardan los ficheros de configuración que usa el RBAC.

5.2.2 Securización de los diferentes servicios.

En cuanto a la securización de los diferentes servicios, ésta se realiza detectando cuándo un cliente envía una petición de un servicio y sólo dejando acceder a él si su dirección IP tiene permiso.

Primero se tiene que programar la gestión de la dirección IP del cliente que hace petición. Para esto, cuando se detecta una petición de un servicio se realiza el siguiente proceso (figura 5.1):

- Obtención de la dirección IP del cliente en formato entero largo sin signo.
- Conversión de la IP a cadena de texto para que pueda ser usada por la función PermisoRealizarAccion.
- Comprobar que el cliente tiene permiso para acceder al servicio mediante la función PermisoRealizarAccion.

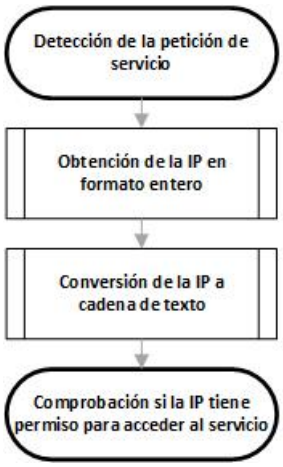


Figura 5.1: Diagrama del proceso para la obtención la IP que realiza la petición de un servicio

Para implementar la función PermisoRealizarAccion (figura 5.2), se la llama pasándole los datos necesarios para que identifique el servicio al que desea acceder el cliente y su dirección IP. Una vez llamada, si la función retorna:

- **_61850_ACCION_DENEGADA**: No se permite que el cliente acceda al permiso.
- **_61850_ACCION_PERMITDA**: Se permite al cliente acceder al servicio.
- **Otro valor**: En esta primera implementación no se permite al cliente acceder al servicio, Queda como proyecto de futuro añadir el código para gestionar estos valores de retorno de forma alternativa.

Destacar que todo este procedimiento se tiene que implementar para cada servicio. Como el código fuente del servidor IEC 61850 de los IEDs gestiona las peticiones en distintas funciones dependiendo de su mapeo MMS, se usa el estudio de la sección 4.2 para detectar

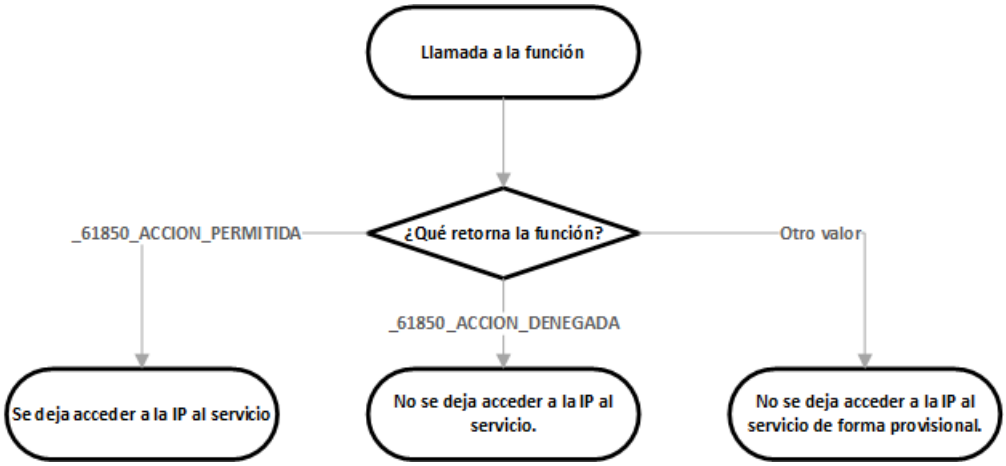


Figura 5.2: Diagrama de la implementación de la función InicializarPermisos

la parte del código donde se llama a cada función. También se usa información de toda la sección 4.1 para detectar a qué servicio exacto desea acceder el cliente. El siguiente código es un ejemplo de esta implementación para el servicio Operate:

```
1  //Se obtiene la IP que llama
2  long unsigned int callingIP=(*(netInfo)).ass_ind_info.
       calling_paddr.netAddr.ip;
3  //Se pasa la IP a string para ser usada por las funciones que
       obtienen si dicha IP tiene permiso para acceder al servicio
4  struct in_addr ip_addr;
5  ip_addr.s_addr=callingIP;
6  const char *sIP= inet_ntoa(ip_addr);
7  //Aquí se llega cuando se obtiene una petición del servicio
       Operate, se comprueba si la IP tiene permiso para acceder a él
8  if(PermisoRealizarAccion(sIP,NULL, MMSOP_WRITE,_61850_OPERATE)!=
       _61850_ACCION_PERMITIDA){
9      //Se le pasa NULL a la cadena de texto ya que se conoce al
           servicio al que se quiere acceder
10   //Si no tiene permiso para acceder al objgeto se devuelve que
         su acceso a él está denegado
11   mvluWrVaCtrl->wrVaCtrl->failure =  ARE_OBJ_ACCESS_DENIED;
12   retcode=SD_FAILURE;
13   mvlu_wr_prim_done (mvluWrVaCtrl, retcode);
14   return;
15 }
```

5.2.3 Actualización de la configuración RBAC una vez inicializado

El último paso para la implementación del RBAC es introducir en el código fuente del IED la funcionalidad de actualizar su configuración después de haber inicializado el sistema.

Para ello se tiene que comprobar de forma periódica si se ha introducido un nuevo archivo a la carpeta de actualización y actualizar la configuración si se encuentra uno.

Para esto se utilizan las funciones programadas en la sección 5.1.5. Localizada una zona del código fuente de los equipos que se ejecuta de forma periódica, se introduce en ella el siguiente código para conseguir la funcionalidad deseada:

```
1 //Comprobacion del archivo .csv que relaciona las IPs y sus roles
2 ComprobarActualizacionCSVIP(_61850_UPDATE_CSV_IP_ROLES,
      _61850_DIRECCION_SCV_IP_ROLES, _61850_DIRECCION_DATOS_IP,
      _61850_DIRECCION_RBAC);
3 //Comprobacion del archivo RBAC.config
4 ComprobarActualizacionArchivoRBAC(_61850_DIRECCION_DATOS_IP,
      _61850_UPDATE_RBAC, _61850_DIRECCION_RBAC);
5 //Comprobacion del archivo .csv que asigna los permisos a cada
      servicio
6 ComprobarActualizacionCSVServicios(
      _61850_UPDATE_CSV_PERMISOS_SERVICIOS,
      _61850_DIR_CSV_PERMISOS_SERVICIOS);
```

Destacar que en esta primera implementación del sistema RBAC no se gestionan los valores de retorno de las funciones, ya que ellas realizan todas las acciones necesarias. Esto deja la gestión de estos valores para versiones futuras de la implementación del RBAC.

5.3 Programación de las funciones para el envío de las peticiones de los servicios ACSI

Para validar el RBAC implementado en el código fuente de los IEDs, se programa la aplicación cliente IEC 61850. Para ello, se comienza programando las funciones para el envío de las peticiones de servicios. Estas se programan con las siguientes características:

- En caso de rechazar la petición, muestra un código sobre el motivo del rechazo en formato entero.
- El código de rechazo se escribe en un puntero a entero pasado como parámetro de la función. Aunque no se usa en la aplicación, es útil para futuros usos de estas funciones.
- Los servicios que no necesitan devolver ningún valor, devuelven un booleano que indica si se ha realizado bien el servicio (true) o mal (false). Esto no se utiliza en la aplicación pero es útil para futuros usos de las funciones.

Además de estas funciones, se programa una función para obtener estructuras con los objetos de control para usarlos en sus servicios. De esta manera, se programan todas las funciones

del fichero de declaraciones Funciones_Servicios_Cliente.h (anexo A.1) para ser usadas en la programación de la aplicación cliente UIEC 61850.

5.4 Programación de la aplicación cliente IEC 61850 para la validación del RBAC

Con las funciones para el envío de las peticiones se programa la aplicación cliente IED 61850.En su programación los dos aspectos más importantes para su funcionamiento correcto son:

- Implementación del envío de las peticiones de servicios.
- Implementación de los distintos procedimientos a seguir para establecer de qué servicios debe enviar las peticiones.

El código fuente de esta aplicación se encuentra en el anexo A.2.

5.4.1 Implementación del envío de las peticiones de servicios

La forma en la que se deben enviar las peticiones depende del tipo de servicio. En la programación de la aplicación se distinguen cinco tipos de servicios:

- Servicios del modelo de asociación.
- Servicios que para lanzar sus peticiones se necesita información que se obtiene con otros servicios.
- Servicios del modelo de control.
- Servicio DeleteFile (borra un fichero).
- Servicios que se pueden enviar directamente.

5.4.1.1 Servicios del modelo de asociación

Como se necesita de una conexión para poder enviar las peticiones del resto de los servicios, la aplicación envía automáticamente las peticiones de los servicios pertenecientes a este modelo cada vez que se ejecuta. Para ello se realiza el siguiente procedimiento:

- Al principio de la ejecución de la aplicación se envía una petición de servicio Associate. Si el servidor:
    - No la procesa correctamente, finaliza su ejecución.

- La procesa correctamente:
    - Envía todas las peticiones de servicios que se le indica.
    - Envía la petición del servicio Release para desconectarse del cliente y finaliza el programa.

Este procedimiento viene descrito en el diagrama de la figura 5.3.

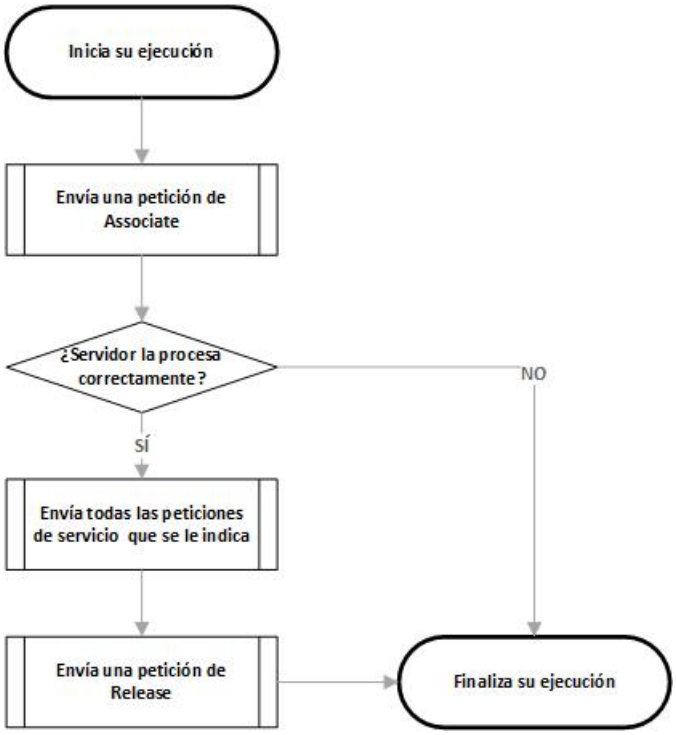


Figura 5.3: Diagrama del envío de las peticiones de los servicios del modelo de asociación

Como la aplicación lanza las peticiones de los servicios de este modelo automáticamente, no se puede indicar que las lance en otro momento.

5.4.1.2 Servicios para los que es necesario disponer de datos que se obtienen con otros servicios para ser enviados

En cuanto a los servicios que necesitan disponer de datos que se obtienen con otros servicios para ser enviados (tabla 4.4), antes de enviar la petición de éstos se comprueba si se ha ejecutado anteriormente el servicio que obtiene la información correctamente y si:

- Se ha ejecutado anteriormente, envía la petición del servicio deseado.
- No se ha ejecutado anteriormente, envía la petición de ese servicio y si la respuesta del servidor es que:
    - Lo ha ejecutado correctamente, envía la petición del servicio deseado.
    - No lo ha ejecutado correctamente, no envía la petición del servicio deseado.

El procedimiento que se sigue para el correcto envío de las peticiones de estos servicios se representa en el diagrama de la figura 5.4.

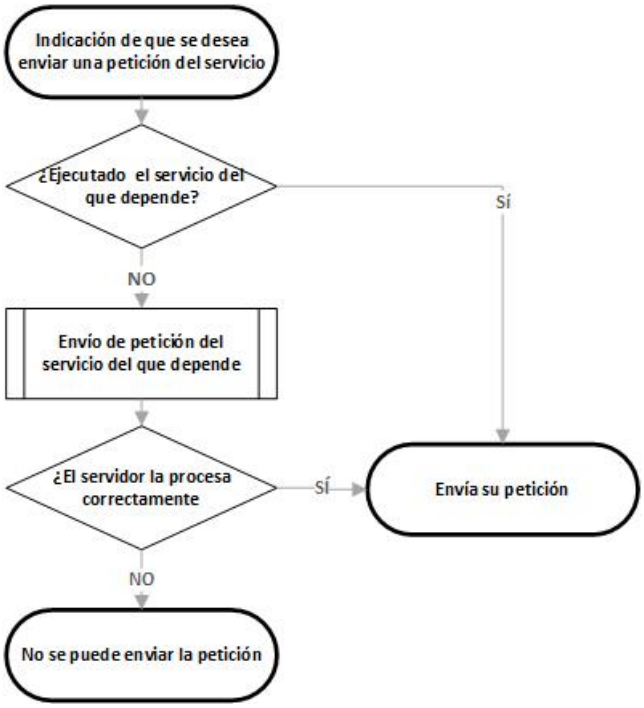


Figura 5.4: Diagrama del envío de la petición de los servicios que para ser enviadas necesitan de la información de otros servicios

5.4.1.3 Servicios del modelo de control

El problema con los servicios del modelo de control es que para enviar sus peticiones se necesita cierta información de los objetos de control en los que se desea llevarlos a cabo. Para ello, dentro de las funciones de la sección 5.3 se ha creado una que, a través de varias peticiones de distintos servicios, obtiene esta información y la devuelve en una estructura definida en la librería. De esta forma, se necesitan tres de estas estructuras para la información de los objetos de control sobre los cuales enviar las peticiones de los servicios de este modelo. Sus nombres son:

- **ContSelect**: Usado por los servicios Select y Cancel.
- **ContSelectWithValue**: Asociado al servicio SelectWithValue.
- **ContOperate**: Usado por el servicio Operate.

Por esto, cuando se quiere enviar la petición de cualquiera de estos servicios, si no se ha obtenido con anterioridad la información de la estructura, se hace uso de la función para obtenerla y sólo se continúa con la petición de los servicios si esta información se ha obtenido correctamente.

Por otro lado, al igual que cuando se selecciona un objeto de control, hasta que no pasa un cierto intervalo de tiempo y se cancela automáticamente, no se puede realizar cualquier otra acción de control sobre otro objeto de control. Para evitar esto, para enviar las peticiones de los servicios Select y SelectWithValue, aparte de asegurarse de tener la estructura de control, se realiza la siguiente secuencia de acciones:

- Se realiza la petición de servicio sobre el objeto de control correspondiente.
- Si el servicio se ha realizado correctamente, se envía una petición del servicio cancel para poder continuar realizando acciones de control sobre otros objetos.

Con todo lo comentado, el envío de la petición de los servicios Select y SelectWithValue se hace siguiendo el procedimiento dado por el diagrama de la figura 5.5.

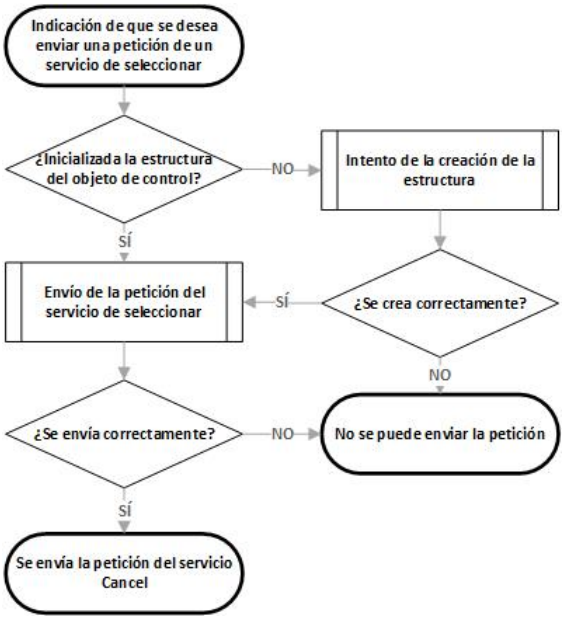


Figura 5.5: Diagrama del envío de la petición de los servicios Select y SelectWithValue

También, como para cancelar una acción de control se ha tenido que seleccionar con anterioridad, para enviar una petición de Cancel la aplicación sigue los siguientes pasos a parte de comprobar la estructura de información:

- Se envía una petición de Select en el objeto de control cuya información se encuentra en la estructura ContSelect.
- Si el Select se ha realizado correctamente, se envía una petición de servicio Cancel sobre el mismo objeto de control.

De esta forma, el envío de la petición del servicio Cancel se realiza siguiendo el diagrama de la figura 5.6.

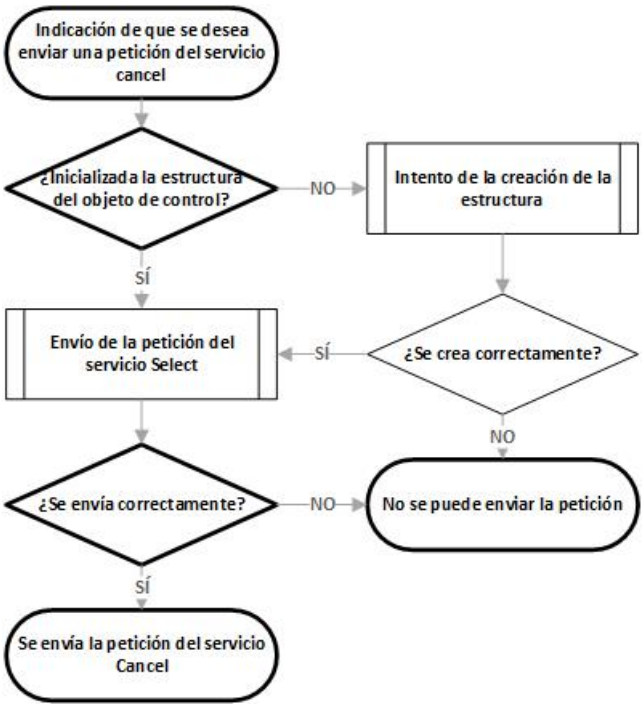


Figura 5.6: Diagrama del envío de la petición del servicio Cancel

Finalmente, para el servicio Operate no hay que tener ninguna consideración aparte de tener la estructura con la información del objeto de control. Por ello, una vez obtenida esta estructura la aplicación envía la petición directamente. Esto se muestra en el diagrama de la figura 5.7.

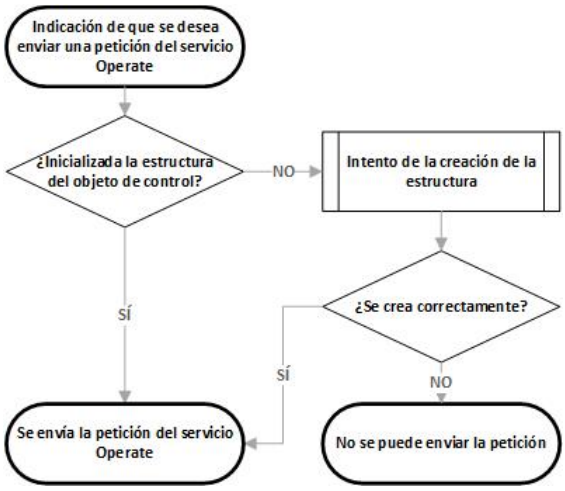


Figura 5.7: Diagrama del envío de la petición del servicio Operate

5.4.1.4 Servicio DeleteFile

El problema con el servicio DeleteFile es que la única carpeta en la que tiene permiso de borrar archivos un cliente IEC 81850 es una de nombre *NotValidated*. Esta carpeta sirve para enviar archivos de configuración al IED, con lo que cualquier archivo que se envía a ella el IED lo analiza y lo elimina al finalizar. Por esto el archivo dura en esta carpeta unos segundos.

Por ello, para poder realizar un servicio DeleteFile correctamente en el servidor, la aplicación cliente realiza la siguiente secuencia (diagrama de la figura 5.8):

- Realiza una petición de SetFile para enviar un archivo local a la carpeta *NotValidated*.
- Si el SetFile se ha realizado correctamente, envía la petición de DeleteFile del archivo que se mueve a la carpeta en el paso anterior.

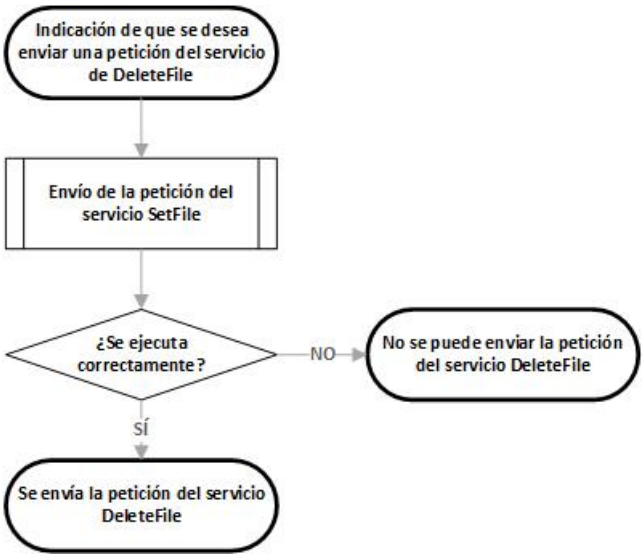


Figura 5.8: Diagrama del envío de la petición del servicio DeleteFile

De esta forma, se envía la petición DeleteFile antes de que al equipo le haya dado tiempo a eliminar el archivo de la carpeta y se puede ejecutar este servicio de forma correcta.

5.4.1.5 Servicios que se pueden enviar directamente

Dentro de los servicios que se pueden enviar directamente se encuentran el resto que no se han incluido en los otros grupos. Para enviar peticiones de éstos no se necesita de ningún otro servicio, así que la aplicación lo hace directamente cuando se le indica.

5.4.2 Implementación de cómo establecer los servicios de los que la aplicación envía las peticiones

Para que se pueda establecer las peticiones de los servicios que la aplicación envía se implementas las formas diseñadas en la sección 4.11.

5.4.2.1 Solicitud de las peticiones de los servicios a enviar por teclado

Esta opción se implementa para que se ejecute cuando se lanza el ejecutable de la aplicación (de nombre *aplicacion_cliente*) sin ningún parámetro de ejecución. Así, el programa se comporta de la siguiente forma cada vez que requiera de un nuevo servicio para lanzar su petición (diagrama de la figura 5.9):

- Muestra todos los servicios disponibles por pantalla.
- Lee el nombre del servicio que se introduce por teclado.
- Si el servicio elegido puede mostrar resultados, la aplicación pide introducir “N” para no mostrarlos y “S” para mostrarlos. Con cualquier otro carácter la aplicación lo vuelve a preguntar hasta que se introduzca uno de estos dos.
- Si el nombre del servicio introducido es erróneo se muestra por pantalla y se pide otra vez el servicio.
- Finaliza cuando se introduce la cadena de texto “FIN” por teclado.

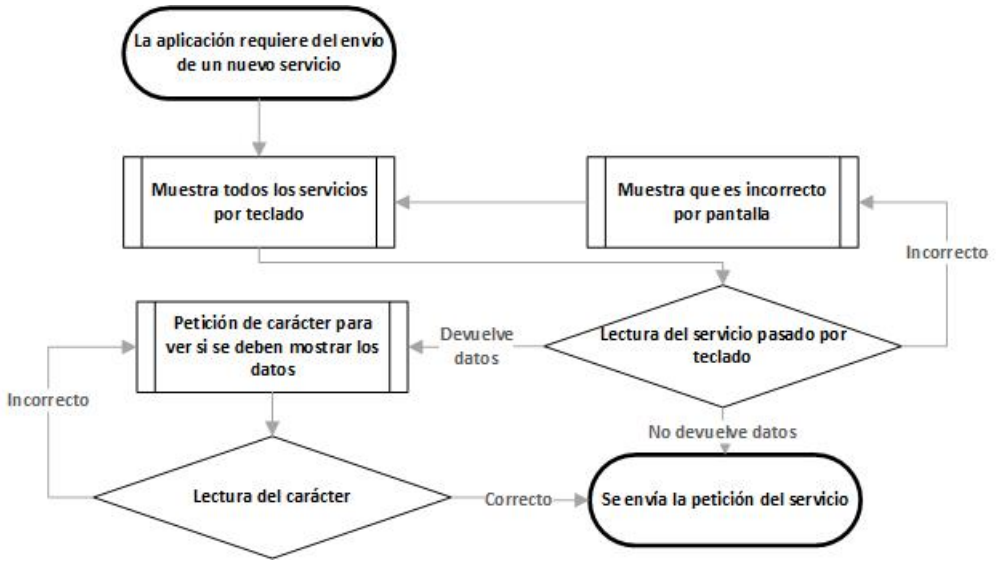


Figura 5.9: Diagrama del funcionamiento de la aplicación cliente cuando envía las peticiones de los servicios que se la pasa por teclado

Para poder enviar de esta forma los servicios, hace falta que se encuentre en la misma carpeta que el ejecutable un archivo .csv de nombre *Informacion_Equipo.csv* con el formato definido en 4.8.1, para tener la información necesaria del equipo. Como ejemplo de este tipo de ejecución de la aplicación se tiene la figura 5.10.

Figura 5.10: Aplicación cliente funcionando enviando servicios por teclado

5.4.2.2 Envío de las peticiones de los servicios del archivo .csv

La funcionalidad de poder enviar los servicios que aparecen en un archivo .csv con el formato establecido en la sección 4.8.2 se implementa para que se ejecute cuando la aplicación se llame con los siguientes comandos:

./aplicacion_cliente -f <nombre archivo.csv>.

En donde <archivo.csv> es el nombre del archivo con el formato definido en la sección 4.8.2, que contiene los servicios que se desea enviar. Este modo de funcionamiento tiene las siguientes características:

- Si hay algún nombre de servicio no reconocido se muestra un mensaje por pantalla, se ignora y se pasa al siguiente.
- Si el carácter de mostrar resultados no es reconocido, por defecto no se muestran.
- La aplicación envía todos los servicios en el orden que aparecen en el fichero, pudiendo enviar la petición de un mismo servicio varias veces. La ejecución del programa finaliza cuando se envían todos los servicios del fichero.

Para obtener la información del equipo es necesario el archivo *Informacion_Equipo.csv*. Un ejemplo de la aplicación ejecutándose de esta forma se muestra en la figura 5.11.

Figura 5.11: Aplicación cliente enviando los servicios que establece el archivo

5.4.2.3 Envío de las peticiones del servicio pasado por parámetros

La funcionalidad de que la aplicación envíe el permiso de un único servicio del que se le pasa la información por parámetros de ejecución se implementa para que funcione cuando la aplicación se ejecuta con los siguientes comandos:

./aplicacion_cliente -p <IP><servicio><ObjRef><Mostrar resultado (S o N)><FC>

En donde:

- ***<IP>***: Dirección IP en formato IPv4 del servidor al que se desean enviar las peticiones.
- ***<servicio>***: Nombre del servicio del que se desea enviar su petición.
- ***<ObjRef>***: Referencia del objeto de datos en el que se quiere realizar el servicio. Este parámetro es necesario para todos los servicios menos para GetServerDirectory.

- ***<Mostrar resultado (S o N)>***: Carácter con el que indicar si se muestran los resultados. Es sólo necesario en los servicios que pueden mostrar los datos obtenidos.

- ***<FC>***: *Functional constraint* del objeto de datos al que se desea enviar el servicio. Este parámetro sólo es necesario para los servicios GetDataDefinition, GetDataValues, SetDataValues, GetLogicalNodeDirectory y GetAllDataValues.

El funcionamiento de la aplicación en este modo es el siguiente:

- La aplicación (a parte de los servicios para crear y terminar la conexión) sólo envía la petición del servicio indicado por parámetros.

- Si el nombre del servicio no es correcto, se indica por pantalla y se termina la aplicación.

- En caso de mostrar resultados el servicio, si no se ha introducido el carácter de mostrar resultado o se ha introducido uno diferente de los establecidos, no se muestra la información de forma predeterminada.

Indicar que como toda la información se pasa por parámetros, la aplicación no necesita de ningún archivo. Un ejemplo de la aplicación enviando un único servicio que se la pasa por parámetro aparece en la figura 5.12.

Figura 5.12: Aplicación cliente enviando la petición del servicio pasado por parámetro

6 Validación

En este apartado se presentan las pruebas realizadas para validar que cada paso del proyecto se realiza de forma correcta.

6.1 Pruebas de la programación e implementación del RBAC.

A lo largo de todo el desarrollo del RBAC se ha comprobado el correcto funcionamiento de:

- Las funciones programadas.
- La implementación del RBAC en el código fuente del servidor IEC 61850 ejecutándolo en:
  - Modo depuración desde el entorno de programación *Eclipse*.
  - Directamente en un IED.

6.1.1 Prueba de las funciones programadas.

Con el fin de que el RBAC cumpla sus objetivos, se tiene que validar el correcto funcionamiento de las funciones que lo componen. Para ello se prueban las funciones de cada paquete mediante distintos programas. El código empleado para validar cada grupo de funciones se encuentra en los siguientes anexos:

- Paquete para la gestión de las IPs: Anexo A.3.1.
- Paquete para la gestión de los roles del archivo RBAC: Anexo A.3.2 .
- Paquete para la gestión de los permisos de los servicios ACSI: Anexo A.3.3. En este caso no se ha validado la función PermisorRealizarAccion, ya que ésta se implementa sobre funciones ya validadas y se ha decidido comprobar su correcto funcionamiento mediante la validación de su implementación.
- Función para la inicialización del RBAC: Anexo A.3.4.
- Paquete para la actualización de la configuración del RBAC: Anexo A.3.5.

6.1.2 Pruebas en el código fuente del Servidor IEC 61850 en modo depuración

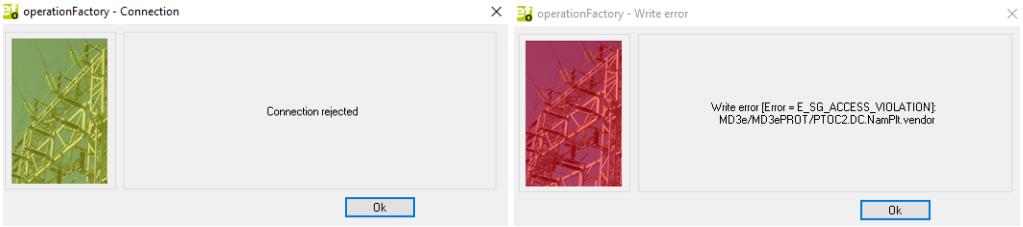
Para validar la implementación del RBAC en el código fuente del servidor IEC 61850 antes de implementarlo en los IEDs, se ejecuta en modo depuración desde el entorno de programación *Eclipse*. Para esto se usan dos herramientas:

- **Aplicación *Operation Factory***: Validación de la securización de servicios puntuales.
- **Aplicación cliente IEC 61850**: Validación de todo el RBAC.

6.1.2.1 Prueba de servicios puntuales mediante el uso de *Operation Factory*

La aplicación *Operation Factory* de *Ingeteam* es una simulación de un cliente IEC 61850 usado para facilitar las pruebas de comunicación con cualquier servidor IEC 61850. La aplicación tiene la posibilidad de conectarse y obtener y modificar datos del servidor mediante el uso de algunos de los servicios ACSI [38].

Dada la limitación de esta herramienta, sólo se usa cuando se han securizado unos pocos servicios del servidor para probarlos y comprobar que el RBAC funciona correctamente en ellos. En la figura 6.1 se observan las pruebas para intentar acceder sin el permiso correspondiente con esta aplicación a los servicios Associate (a) y SetDataValues (b).



(a) Associate (b) SetDataValues

Figura 6.1: Prueba para intentar realizar un servicio en el servidor sin permiso con *Operation Factory*

De esta forma, se comprueba que el RBAC no deja acceder correctamente a los servicios a los que un cliente no tiene permiso.

6.1.2.2 Prueba de todos servicios mediante el uso de la aplicación cliente 61850

Para validar el RBAC cuando se ha implementado completamente, se prueba que securiza correctamente todos los servicios ACSI siguiendo los siguientes pasos con la aplicación cliente IEC 61850:

1. No darle permiso para realizar el servicio Associate a la aplicación cliente.

2. Comprobar la falta de conexión al servidor, con lo que no puede ejecutar ningún servicio.

3. Asignarle el permiso de realizar sólo el servicio Associate.

4. Comprobar la conexión con el servidor, pero no se puede acceder a ningún otro servicio.

5. Asignarle el permiso para realizar el servicio Asocciate junto a otro servicio que no necesite de ningún otro para realizarse.

6. Probar que sólo puede conectarse al servidor y realizar el servicio del que se dispone el permiso.

7. Repetir los pasos 5 y 6 para todos los servicios que no necesiten de otro servicio.

8. Repetir los pasos 5 y 6 pero esta vez con los servicios que necesitan de otro para realizarse. Hay que tener en cuenta que:

    - Se le tiene que asignar también el permiso del servicio necesario para obtener los datos y poder ejecutar el que se desea probar.

    - Se deben poder ejecutar los servicios Associate, el que obtiene los datos y el que se desea probar.

A la hora de asignar al cliente el permiso para realizar los servicios deseados se establecen los archivos de configuración del RBAC de la siguiente forma:

- **Archivo .csv para establecer los permisos de cada servicio de la sección 4.7.3**: Crear un nuevo permiso. Este permiso se modificará para que permita realizar a la aplicación cliente los servicios que sean necesarios en cada paso, dejando el resto de archivos intactos.

- **Archivo RBAC de la sección 4.7.2**: Crear un rol cuya máscara de permisos sólo incluya el creado en el punto anterior.

- **Archivo .csv para establecer los roles de una dirección IP**: Asignar a la dirección IP de la aplicación cliente el rol creado en el punto anterior.

De esta forma se agilizan mucho las pruebas, ya que basta con modificar un solo archivo.

6.1.3 Prueba de todo el RBAC implantado en un IED mediante la aplicación cliente IEC 61850

Para validar que el RBAC sigue funcionando cuando se ha implantado en un IED, se repiten los pasos del apartado anterior. De esta forma se asegura que cuando se implemente el sistema diseñado en un IED dentro de un entorno de trabajo real, funcionará correctamente.

6.2 Pruebas de la aplicación cliente IEC 61850

Para realizar correctamente la validación de la implementación del RBAC hay que validar también el funcionamiento de la aplicación IEC 61850. Para ello se lanzan sus peticiones de servicio a un IED sin el RBAC implementado y se estudia si se hace correctamente con el analizador de protocolos *Wireshark*. De esta manera se validan los siguientes aspectos:

- Ejemplos de la librería.
- Código obtenido para el lanzamiento de las peticiones de servicios.
- Funciones programadas con el código anterior.
- Aplicación final.

7 Planificación

7.1 Actividades

En la tabla 7.1 se muestran las actividades en las que se divide este proyecto

Fase	Actividad	Descripción	Duración
1.Estudio teórico	1.1 Norma IEC 61850	Búsqueda de la información básica sobre la norma IEC 61850, los protocolos de comunicación que define y el modelo ASCI	1 semana
	1.2 Norma IEC 62351	Análisis de los aspectos básicos de la norma IEC 62351 y estudio del modelo RBAC que define	1 semana
2.Análisis y diseño	2.1 Análisis de los servicios ACSI	Estudio de la parte 7-2 de la norma IEC 61850 para la obtención de la información de todos los servicios ACSI	2 semanas
	2.2 Análisis del mapeo MMS de los servicios ACSI	Lectura de las partes 8-1 y 9-2 de la norma IEC 61850 para el estudio del mapeo MMS de los distintos servicios ACSI	2 semanas
	2.3 Análisis de los roles y servicios predefinidos del modelo RBAC	Estudio de la parte 9 de la norma IEC 62351 para obtener los roles y permisos predefinidos y su relación.	1 semana
	2.4 Análisis de la relación de permisos predefinidos y servicios ACSI	Estudio del mapeo de servicios a permisos predefinidos que establece el reporte técnico IEC 61850-90-19 y su modificación para ajustarlo a este proyecto	1 semana
	2.5 Análisis del código ya disponible	Estudio del código fuente del servidor IEC 61850 de los IEDs y de la librería con las funciones para la programación de la aplicación cliente IEC 61850	3 semanas

Sigue en la página siguiente

Fase	Actividad	Descripción	Duración
	2.6 Diseño y análisis de los archivos necesarios para el RBAC	Diseño de los archivos .csv para establecer los roles de cada dirección IP y los permisos de cada servicio ACSI. Análisis y modificación del archivo RBAC para establecer los permisos de cada rol	4 dias
	2.7 Diseño de los archivos usados en la aplicación cliente IEC 61850	Diseño de los archivos .csv con la información del servidor y con los servicios de los que lanzar peticiones	3 dias
	2.8 Diseño de los paquetes de funciones del RBAC	Diseño de los paquetes con las funciones necesarias para la implementación del RBAC	2 semanas
	2.8 Diseño de las funciones para la aplicación cliente	Diseño de las funciones que envían las peticiones de servicios para ser usadas por la aplicación cliente	1 semana
	2.9 Diseño de cómo establecer los servicios en la aplicación cliente	Diseño de las tres formas en las que se puede establecer de qué servicios lanza peticiones la aplicación cliente y los requisitos que debe cumplir	2 días
3.Desarrollo	3.1 Programación de las funciones para el RBAC	Programación de las funciones necesarias para que los distintos paquetes de funciones cumplan las funcionalidades deseadas	8 semanas
	3.2 Implementación del RBAC en el código fuente del IED	Implementación de las funciones programadas en el código fuente del servidor IEC 61850 de los IEDS	8 semanas
	3.3 Programación de las funciones para la aplicación cliente IEC 61850	Programación de las funciones para el envío de peticiones de los distintos servicios siguiendo el diseño realizado	4 semanas

Sigue en la página siguiente

Fase	Actividad	Descripción	Duración
	3.4 Programación de la aplicación cliente IEC 61850	Programación de la aplicación cliente IEC 61850 con las funciones para el envío de las peticiones de los servicios y cumpliendo las formas en las que establecer de qué servicio lanza las peticiones diseñadas	2 semanas
4.Validación	4.1 Validación de las funciones programadas para el RBAC	Validación del correcto funcionamiento de los paquetes de funciones programados	2 semanas
	4.2 Validación de los primeros servicios securizados por el RBAC	Validación mediante el uso de *Operation Factory* y el código fuente del servidor IEC 61850 ejecutándose en modo debug de los primeros servicios securizados	2 semanas
	4.3 Validación de las funciones para la aplicación cliente IEC 61850	Validación del correcto funcionamiento de la aplicación IEC 61850 mediante *Wireshark* y un IED sin el RBAC implementado	2 semanas
	4.4 Validación de la aplicación cliente IEC 61850	Validación del correcto funcionamiento de la aplicación IEC 61850 mediante *Wireshark* y un IED sin el RBAC implementado	1 semana
	4.5 Validación de toda la implementación del RBAC	Validación mediante el uso de la aplicación IEC 61850 de toda la implementación del RBAC. Tanto con el código fuente del servidor IEC 61850 (ejecutándose en modo depuración) como implementado en un IED	7 semanas
5.Documentación	5.1 Desarrollo de la documentación	Desarrollo de todo el trabajo realizado en este proyecto para que pueda ser replicable	76 semanas

Tabla 7.1: Actividades en las que se divide este proyecto

7.2 Distribución de actividades

La distribución de las distintas actividades se muestra en el diagrama Gantt de la figura 7.1.

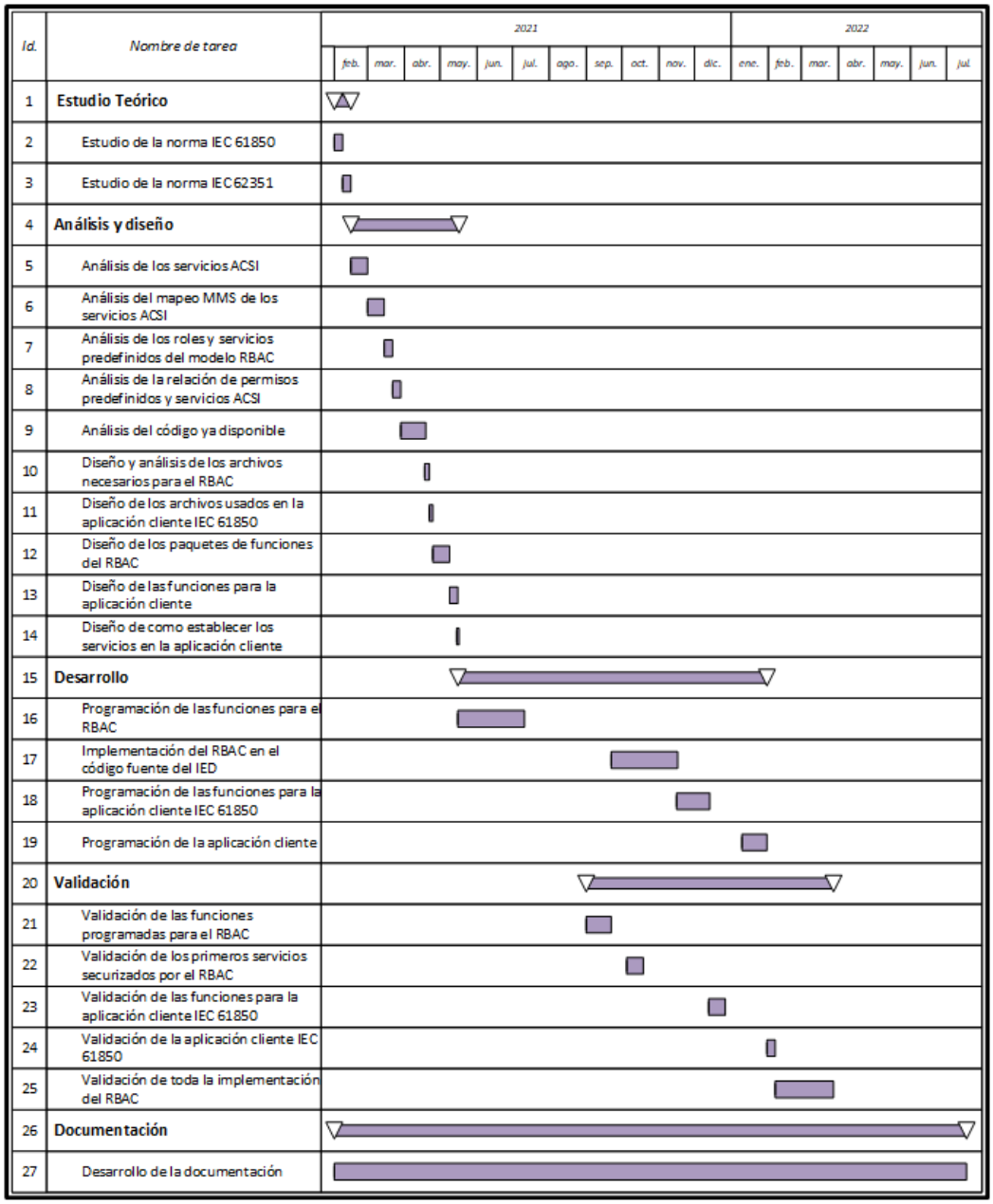


Figura 7.1: Diagrama Gantt con la distribución de las distintas actividades

8 Presupuesto

En esta sección, se detalla el presupuesto de este proyecto.

8.1 Lista de precios

Hardware			
Elemento	**Cantidad**	**Precio unitario (€)**	**Precio total (€)**
PC portátil *Dell Inspiron 15-5000 Intel Core i5-1135G7/8GB/256GB SSD/15.6"*	1	718,99	718,99
IED *Ingepac EF MD*	1	3495[1]	3(495
Software			
Elemento	**Cantidad**	**Precio unitario (€)**	**Precio total (€)**
OperationFactory	1	990	990
Máquina virtual basada en *Linux*	1	0	0
Editor de de textos en LaTeX *Overleaf*	1	0	0
Entorno de programación *Eclipse 2019-03*	1	0	0
Librería de C *libiec61850*	1	0	0
Recursos humanos			
Elemento	**Cantidad (h)**	**Precio unitario (€)**	**Precio total (€)**
Ingeniero Técnico Superior	100	80	8000
Ingeniero Graduado	1000	50	50000

Tabla 8.1: Lista de precios de este proyecto

[1] dependiendo del modelo y opciones de hardware

8.2 Inmovilizado material

Hardware			
Tiempo de amortización (años)	**Cuota de amortización semanal (€)**	**Tiempo de uso (semanas)**	**Inmovilizado (€)**
7	11,58	60	694,8
Software			
Tiempo de amortización (años)	**Cuota de amortización semanal (€)**	**Tiempo de uso (semanas)**	**Inmovilizado (€)**
5	3,81	60	228,46

Tabla 8.2: Inmovilizado material de este proyecto

8.3 Presupuesto total

Elemento	**Coste (€)**
Amortizaciones de hardware	694,8
Amortizaciones de software	228,46
Recursos humanos	58000
Presupuesto de Ejecución Material (PEM) 59223,26€	
Elemento	*Coste (€)*
IVA (21 % sobre el PEM)	12436,88
Presupuesto Total 71660,14€	

Tabla 8.3: Presupuesto total de este proyecto

9 Conclusiones

En este proyecto se ha realizado el diseño, programación e implementación de un control de accesos basado en roles (RBAC), en el código fuente del servidor IEC 61850 de un IED, para:

- Asegurar que los clientes conectados al servidor sólo realicen lo que sus roles le permiten.
- Hacer que los clientes sin ningún rol asignado no se puedan conectar al servidor, securizándolo frente a ciberataques.
- Controlar, mediante los archivos de registros, a los clientes que intenten acceder a servicios sin permiso.

Para lograr este objetivo se ha:

- Analizado los servicios ACSI y su mapeo mediante las partes 7-2,8-1 y 9-2 de la norma IEC 61850.
- Estudiado los roles, permisos y sus relaciones definidas en la parte 8 de la norma IEC 62351.
- Diseñado y establecido los archivos necesarios para que el sistema RBAC funcione correctamente.
- Definido y programado los paquetes de funciones para la funcionalidad del RBAC.
- Implementado el RBAC en el código fuente de un servidor IEC 61850 de un IED.
- Validado la implementación realizada.

Para la validación del RBAC implementado, se ha programado una aplicación cliente IEC 61850 para comprobar su correcto funcionamiento. Asegurando que el RBAC funciona correctamente y se pueda implementar en un sistema real. Adicionalmente esta aplicación servirá también para validar nuevas funcionalidades que se añadan a los IEDs en su servidor IEC 81850.

9.1 Futuros proyectos

Todo lo trabajado en este proyecto ha abierto la puerta a futuros trabajos:

- Diseño de nuevos permisos y roles para que el RBAC se ajuste más a las funcionalidades de los IEDs.
- Añadir una capa de ciberseguridad basada en llaves, definida en la parte 9 de la norma IEC 62351 [31]. De esta forma habría que cambiar el RBAC implementado para que reconozca a cada usuario por su llave en vez de por su dirección IP.
- Introducir la asignación condicional de los permisos a los roles. De esta manera el RBAC podrá cumplir el mapeo de roles a permisos de la tabla 4.1.
- Incluir en el RBAC el uso de la máscara de permisos de emergencia del archivo para el mapeo de los roles a permisos (sección 4.7.2). Así se podrán asignar los permisos de emergencia a los roles ante cualquier problema con las comunicaciones del servidor IEC 61850 de un IED.
- Incluir el manejo de todos los valores de retorno en la implementación de la función para comprobar si un cliente tiene acceso a un servicio y de las funciones para la actualización del RBAC.
- Creación de una interfaz gráfica para la aplicación cliente IEC 61850, facilitando futuras funcionalidades.

Referencias bibliográficas

[1]A. J. Conejo y L. Baringo, *Power Electronics and Power Systems Power System Operations*. Springer, 2018 (vid. pág. 1).

[2]M. Kezunovic, M. Kezunovic, I. Dobson e Y. Dong, „Impact of Extreme Weather on Power System Blackouts and Forced Outages: New Challenges“, sep. de 2008, págs. 1-5 (vid. pág. 1).

[3]M. Panteli y P. Mancarella, „Influence of extreme weather and climate change on the resilience of power systems: Impacts and possible mitigation strategies“, *Electric Power Systems Research*, vol. 127, págs. 259-270, oct. de 2015 (vid. pág. 1).

[4]R. Kinney, P. Crucitti, R. Albert y V. Latora, „Modeling cascading failures in the North American power grid“, *The European Physical Journal B - Condensed Matter and Complex Systems 2005 46:1*, vol. 46, págs. 101-107, 1 ago. de 2005 (vid. pág. 1).

[5]L. G. Hewitson, M. Brown y R. Balakrishnan, *Practical power system protection*. 2006, vol. 22 (vid. pág. 1).

[6]T. N. Bhattarai, S. Ghimire, B. Mainali y col., „Applications of smart grid technology in Nepal: status, challenges, and opportunities“, *Environmental Science and Pollution Research*, vol. 1, págs. 1-25, feb. de 2022 (vid. pág. 2).

[7]G. Dileep, „A survey on smart grid technologies and applications“, *Renewable Energy*, vol. 146, págs. 2589-2625, feb. de 2020 (vid. pág. 2).

[8]N. H. Ali y M. M. Eissa, „Accelerating the protection schemes through IEC 61850 protocols“, *International Journal of Electrical Power & Energy Systems*, vol. 102, págs. 189-200, nov. de 2018 (vid. pág. 2).

[9]D. Mlakić, H. R. Baghaee, S. Nikolovski, M. Vukobratović y Z. Balkić, „Conceptual design of IoT-based AMR systems based on IEC 61850 microgrid communication configuration using open-source hardware/software IED“, *Energies*, vol. 12, 22 nov. de 2019 (vid. pág. 2).

[10]P. Eder-Neuhauser, T. Zseby, J. Fabini y G. Vormayr, „Cyber attack models for smart grid environments“, *Sustainable Energy, Grids and Networks*, vol. 12, págs. 10-29, dic. de 2017 (vid. pág. 2).

[11]T. Krause, R. Ernst, B. Klaer, I. Hacker y M. Henze, „Cybersecurity in Power Grids: Challenges and Opportunities“, *Sensors 2021, Vol. 21, Page 6225*, vol. 21, pág. 6225, 18 sep. de 2021 (vid. pág. 2).

[12]A. B. Fadhel, D. Bianculli y L. Briand, „A comprehensive modeling framework for role-based access control policies“, *Journal of Systems and Software*, vol. 107, págs. 110-126, sep. de 2015 (vid. págs. 3, 13).

[13]A. Volkova, M. Niedermeier, R. Basmadjian y H. D. Meer, „Security challenges in control network protocols: A survey“, *IEEE Communications Surveys and Tutorials*, vol. 21, págs. 619-639, 1 ene. de 2019 (vid. pág. 9).

[14]„Communication networks and systems for power utility automation. Part 8-1, Specific communication service mapping (SCSM)–mappings to MMS (ISO 9506-1 AND ISO 9506-2) and to ISO/IEC 8802-3.“, International Electrotechnical Commission, Standard, 2011 (vid. págs. 9, 21).

[15]T. S. Ustun y S. M. Hussain, „An Improved Security Scheme for IEC 61850 MMS Messages in Intelligent Substation Communication Networks“, *Journal of Modern Power Systems and Clean Energy*, vol. 8, págs. 591-595, 3 mayo de 2020 (vid. págs. 9, 11).

[16]H. Shang, Z. Zhao y R. Thorn, „Implementing Manufacturing Message Specifications (MMS) within collaborative virtual environments over the Internet“, *INTERNATIONAL JOURNAL OF COMPUTER INTEGRATED MANUFACTURING*, vol. 16, págs. 112-127, 2 ene. de 2010 (vid. pág. 9).

[17]*Estándar IEC 61850, todos para uno y uno para todos | INCIBE-CERT*, `https://www.incibe-cert.es/blog/estandar-iec-61850-todos-uno-y-uno-todos` (vid. pág. 10).

[18]B. Lee, D. K. Kim, H. Yang y H. Jang, „Role-based access control for substation automation systems using XACML“, *Information Systems*, vol. 53, págs. 237-249, oct. de 2015 (vid. pág. 10).

[19]M. C. Magro, P. Pinceti, L. Rocca y G. Rossi, „Safety related functions with IEC 61850 GOOSE messaging“, *International Journal of Electrical Power & Energy Systems*, vol. 104, págs. 515-523, ene. de 2019 (vid. pág. 10).

[20]S. M. Hussain, S. M. Farooq y T. S. Ustun, „A Method for Achieving Confidentiality and Integrity in IEC 61850 GOOSE Messages“, *IEEE Transactions on Power Delivery*, vol. 35, págs. 2565-2567, 5 oct. de 2020 (vid. pág. 10).

[21]L. Ling, Y. Hongyong y C. Xia, „Model differences between IEC 61970/61968 and IEC 61850“, *Proceedings of the 2013 3rd International Conference on Intelligent System Design and Engineering Applications, ISDEA 2013*, págs. 938-941, 2013 (vid. pág. 11).

[22]J. A. Oliveira-Lima, V. Delgado-Gomes, J. F. Martins y C. Lima, „Standard-based service-oriented infrastructure to integrate intelligent buildings in distributed generation and smart grids“, *Energy and Buildings*, vol. 76, págs. 450-458, jun. de 2014 (vid. pág. 11).

[23]B. Li, R. Zhao, J. Lu y col., „CoAP Protocol Communication Mapping for Power Distribution Internet of Things Based on IEC 61850“, *Proceedings - 2021 6th Asia Conference on Power and Electrical Engineering, ACPEE 2021*, págs. 562-566, abr. de 2021 (vid. pág. 11).

[24]S. M. Hussain, M. A. Aftab e I. Ali, „IEC 61850 Modeling of DSTATCOM and XMPP Communication for Reactive Power Management in Microgrids“, *IEEE Systems Journal*, vol. 12, págs. 3215-3225, 4 dic. de 2018 (vid. pág. 11).

[25]A. Apostolov y B. Muschlitz, „Object Modeling of Measuring Functions in IEC 61850 Based IEDs“, *Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference*, vol. 2, págs. 471-476, 2003 (vid. págs. 11, 12).

[26]T. Kostic y C. Frei, „Modelling and using IEC 61850-7-2 (ACSI) as an API“, *2007 IEEE Lausanne POWERTECH, Proceedings*, págs. 713-719, 2007 (vid. pág. 11).

[27]D. Li e Y. Zhang, „Information Model of ARC Protection System Based on IEC 61850“, *Advanced Materials Research*, vol. 383-390, págs. 2540-2544, 2012 (vid. pág. 11).

[28]„Communication networks and systems for power utility automation-Part 7-4: Basic communication structure-Compatible logical node classes and data object classes“, International Electrotechnical Commission, Standard, 2010 (vid. pág. 11).

[29]N. Higgins, V. Vyatkin, N. K. C. Nair y K. Schwarz, „Distributed power system automation with IEC 61850, IEC 61499, and intelligent control“, *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 41, págs. 81-92, 1 2011 (vid. pág. 12).

[30]„Communication networks and systems for power utility automation. Part 7-2, Basic information and communication structure–abstract communication service interface (ACSI)“, International Electrotechnical Commission, Standard, 2010 (vid. págs. 12, 15, 38).

[31]S. M. Hussain, T. S. Ustun y A. Kalam, „A Review of IEC 62351 Security Mechanisms for IEC 61850 Message Exchanges“, *IEEE Transactions on Industrial Informatics*, vol. 16, págs. 5643-5654, 9 sep. de 2020 (vid. págs. 13, 14, 78).

[32]R. Schlegel, S. Obermeier y J. Schneider, „A security evaluation of IEC 62351“, *Journal of Information Security and Applications*, vol. 34, págs. 197-204, jun. de 2017 (vid. pág. 13).

[33]„Power systems management and associated information exchange – Data and communications security – Part 8: Role-based access control for power system management“, International Electrotechnical Commission, Standard, 2020 (vid. págs. 13, 23, 25).

[34]„ISO 9506-1 Industrial automation systems-Manufacturing Message Specification-Part 1: Service definition“, international standard organization, Standard, 2000 (vid. pág. 21).

[35]„Communication networks and systems for power utility automation. Part 9-2, Specific communication service mapping (SCSM)–sampled values over ISO/IEC 8802-3.“, International Electrotechnical Commission, Standard, 2011 (vid. pág. 21).

[36]„Proposal to develop IEC TR 61850-90-19. Communication networks and systems for power utility automation – Part 90-19: Using Role Based Access Control (RBAC) and IEC 61850“, International Electrotechnical Commission, inf. téc., 2020 (vid. págs. 25, 27).

[37]*libiec61850: API Reference Manual*, `https://support.mz-automation.de/doc/libiec61850/c/latest/` (vid. pág. 38).

[38]„Ficha tecnica INGESYS EFS“, Ingeteam, inf. téc. (vid. pág. 68).

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN INGENIERÍA DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA

ANEXO A: PROGRAMAS FUENTE

SECURIZACIÓN DE EQUIPOS DE PROTECCIÓN Y CONTROL DE SISTEMAS ELÉCTRICOS

Estudiante *Vallejo García, Javier*
Director *Perez González, Federico*
Departamento *Ingeniería de Sistemas y Automática*
Curso académico *2021-2022*

Bilbao, 19 junio 2022

Esquemas y programas fuente

A

A.1 Archivo *Funciones_Servicios_Cliente.h* con las funciones para lanzar las peticiones de los distintos servicios programadas

```
/*
 * Funciones_Servicios_Cliente.h
 *
 *  Created on: 27 Oct. 2021
 *      Author: Javier.Vallejo
 */

#ifndef FUNCIONES_SERVICIOS_CLIENTE_H_
#define FUNCIONES_SERVICIOS_CLIENTE_H_

#include "iec61850_client.h"

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <libgen.h>

#define C_61850_N_CARCACTERES_OBJETO_MAX 129

//Estructura para guardar los resultados de GetLogStatusValues y
   usarlos en los servicios de Query
typedef struct LogStatusValues{
  bool bLogStatusValido;
  MmsValue* OldEntrTm;
  MmsValue* NewEntrTm;
  MmsValue* OldEnt;
  MmsValue* NewEnt;
}LogStatusValues;

//Estructura para guardar los resultados de GetLCBValues y
   usarlos en SetLCBValues
typedef struct LCBValues{
  bool bLCBValido;
```

```
34    bool bLogEna;
35    char sLogRef[C_61850_N_CARCACTERES_OBJETO_MAX+1];
36    char sDatSetRef[C_61850_N_CARCACTERES_OBJETO_MAX+1];
37    MmsValue* TrgOps;
38    uint32_t iIntgPd;
39  }LCBValues;
40
41
42  /*
43   * Implementación de las funciones para generar peticiones
         cliente IEC 61850 de servicios ACSI. Todas las que no tienen
         que retornar
44   * nada importante, retornan un boleano que si es true es que se
         ha relizado correctmanete el servicio de la petición y si
         devuelve false
45   * es que no se ha podido realizar.
46   */
47
48  bool CrearConexion(IedConnection* pCon,IedClientError* pError,
       char* hostname,int tcpPort);
49
50  bool GetDataSetDirectory(IedConnection con, IedClientError*
       pError, const char* dataSetReference, bool bMostrarResultado);
51
52  bool GetDataSetValues(IedConnection con, IedClientError* pError,
       const char* dataSetReference, bool bMostrarResultado);
53
54
55  ClientReportControlBlock GetBRCBValues(IedConnection con,
       IedClientError* pError, const char* brcbReference, bool
       bMostrarResultado);
56
57  ClientReportControlBlock GetURCBValues(IedConnection con,
       IedClientError* pError, const char* urcbReference, bool
       bMostrarResultado);
58
59  /*
60   * La función SetRCBValues sirve tanto para realizar un servicio
         SetBRCBValues como un SetUCBValues. El mismo reconoce que
         tipo de report es
61   * y realiza peticiones de todos los settings.
62   */
63  bool SetRCBValues(IedConnection con, IedClientError* pError,
       ClientReportControlBlock rcb);
64
65  /*
```

```
66   * La función GetServerDirectory sirve tanto para usarlo con
         FileDirectoryClass (bEsFileDirectory=true) como para
67   * LogicalDeviceClas (bEsFileDirectory=false)
68   */
69  bool GetServerDirectory(IedConnection con, IedClientError* pError
        , bool bEsFileDirectory, bool bMostrarResultado);
70
71  /*
72   * La siguiente función implementa getLogicalNodeDirectory. Según
         la descipción de la función del cliente que hace uso
73   * para implementar este servicio, para que no use información
         obtenida anteriormente y lance una petición se tiene que
74   * pedir mostrar Data-sets o reports. En este caso se elige que
         muestre los data sets por esta razón se le tiene que pasar
75   * la referencia de un nodo lógico que contenga data sets
76   */
77  bool GetLogicalNodeDirectory(IedConnection con, IedClientError*
        pError, const char* sLogicalNode, bool bMostrarResultado);
78
79
80  bool GetLogicalDeviceDirectory(IedConnection con, IedClientError*
         pError, const char* sLogicalDevice, bool bMostrarResultado);
81
82  bool GetFile(IedConnection con, IedClientError* pError, const
        char* sDireccionFicheroServidor);
83
84  /*
85   * La función SetFile envía el contenido del archivo que se
         encuentra sDireccionFicheroLocal a la dirrección
         sDireccionFicherServidorDestino
86   */
87  bool SetFile(IedConnection con, IedClientError* pError, const
        char* sDireccionFicheroLocal, const char*
        sDireccionFicheroServidorDestino);
88
89  bool DeleteFile(IedConnection con, IedClientError* pError, const
        char* sDireccionFicheroServidorBorrar);
90
91  ClientGooseControlBlock GetGoCBValues(IedConnection con,
        IedClientError* pError, const char* sGoCB, bool
        bMostrarResultado);
92
93  bool SetGoCBValues(IedConnection con, IedClientError* pError,
        ClientGooseControlBlock GOOSECLiente);
94
```

```
95  MmsValue* GetDataValues(IedConnection con, IedClientError* pError
        , const char* sObjRef,const char* sFC, bool bMostrarResultado)
        ;
96
97  bool SetDataValues(IedConnection con, IedClientError* pError,
        const char* sObjRef,const char* sFC,MmsValue* ValorEscribir);
98
99  ControlObjectClient CrearObjetoControl(IedConnection con, const
        char* sRefObjetoControl);
100
101 bool Select(ControlObjectClient control);
102
103 bool Cancel(ControlObjectClient control);
104
105 bool SelectWithValue(ControlObjectClient control, MmsValue*
        ctlVal);
106
107 bool Operate(ControlObjectClient control, MmsValue* ctlVal);
108
109 bool SelectActiveSG(IedConnection con, IedClientError* pError ,
        const char* sRefSGBC, const int iNumeroSGActivo);
110
111 bool SelectEditSG(IedConnection con, IedClientError* pError ,
        const char* sRefSGBC, const int iNumeroSGEditable);
112
113 MmsValue* GetEditSGValues(IedConnection con, IedClientError*
        pError, const char* sRefEditSG, bool bMostrarResultado);
114
115 bool SetEditSGValues(IedConnection con, IedClientError* pError,
        const char* sRefEditSG ,MmsValue* EditSGValuesEscribir);
116
117 bool ConfirmEditSGValues(IedConnection con, IedClientError*
        pError, const char* sRefSGBC);
118
119 bool GetSGCBValues(IedConnection con, IedClientError* pError,
        const char* sRefSGBC, bool bMostrarResultado);
120
121 bool GetAllDataValues(IedConnection con, IedClientError* pError,
        const char* sLNRef,const char* sFC, bool bMostrarResultado);
122
123 bool GetFileAttributeValues(IedConnection con, IedClientError*
        pError, const char* sDireccionFicheroServidor, bool
        bMostrarResultado);
124
125 bool GetDataDefinition(IedConnection con, IedClientError* pError,
         const char* sObjRef, const char* sFC, bool bMostrarResultado)
        ;
```

```
126
127 LogStatusValues GetLogStatusValues (IedConnection con,
        IedClientError* pError, const char* sRefLog, bool
        bMostrarResultado);
128
129 bool QuerLogAfter(IedConnection con, IedClientError* pError,
        const char* sRefLog, LogStatusValues* pLogStatusLeido, bool
        bMostrarResultado);
130
131 bool QueryLogByTime(IedConnection con, IedClientError* pError,
        const char* sRefLog, LogStatusValues* pLogStatusLeido, bool
        bMostrarResultado);
132
133 LCBValues GetLCBValues (IedConnection con, IedClientError* pError
        , const char* sRefLog, bool bMostrarResultado);
134
135 bool SetLCBValues(IedConnection con, IedClientError* pError,
        const char* sRefLog, LCBValues* pLCBLeido);
136
137 /*
138  * La función GetSVCBValues sirve tanto para un GetMSVCBValues
        como para GetUSVCBValues ya que a la hora de mostrar los
        resultados
139  * reconoce de que tipo de SV se trata
140  */
141 ClientSVControlBlock GetSVCBValues(IedConnection con, const char*
        sRefSVCB, bool bMostrarResultado);
142
143 /*
144  * La función SetSVCBValues sirve tanto para un SetMSVCBValues
        como para SetUSVCBValues ya que a la hora de mostrar los
        resultados
145  * reconoce de que tipo de SV se trata
146  */
147 bool SetSVCBValues(ClientSVControlBlock SVCBModificar);
148
149 #endif
```

A.2 Código de la aplicación cliente IEC 61850

```
1  /*
2   * aplicacion_cliente.c
3   *
4   * Esta aplicación sirve para lanzar peticiones de servicio IEC
        61850.
5   */
6
7  #include "iec61850_client.h"
8  #include "../../funciones_servicios_cliente/
       Funciones_Servicios_Cliente.h" //Donde están implementadas las
        funciones de las peticiones de servicios
9
10 #include <stdlib.h>
11 #include <stdio.h>
12 #include <signal.h>
13 #include <time.h>
14 #include <libgen.h>
15 #include <signal.h>
16 #include <unistd.h>
17
18 #define C_61850_N_CARACTERES_IP 15
19 #define C_61850_NOMBRE_ARCHIVO_EQUIPO "Informacion_Equipo.csv"
20 #define C_61850_C_RETORNO_CARRO '\r' //Caracter de retorno de
       carro
21 #define C_61850_C_NUEVA_LINEA '\n'  //Caracter de fin de linea
22 #define C_61859_FIN_CADENA '\0'  //Caracter de fin de cadena
23 #define C_61850_N_CARCACTERES_SERVICIO_MAX 30
24 #define C_61850_N_CARACTERES_FC 2
25 #define C_61850_ARCHIVO_BORRAR_REMOTO "/SCL/notvalidated/borrar.
       txt"
26
27
28 static bool bPararAplicacion=false;
29
30 //Signal handler usado
31 void sigint_handler(int signalId)
32 {
33   puts("\nComenzado el proceso de parar la aplicación:");
34   bPararAplicacion=true;
35 }
36 //El tipo referenciaObjeto se crea para guardar la referebcia a
       objetos
37 typedef char referenciaObjeto [C_61850_N_CARCACTERES_OBJETO_MAX
       +1];
38
```

```
39 //La estructura InfoObjetoFC se crea para guardar la referencia
       de un objeto y su FC (se usa si es necesario el FC)
40 typedef struct InfoObjetoFC{
41   referenciaObjeto sObjRef;
42   char sFC[C_61850_N_CARACTERES_FC+1];
43 }InfoObjetoFC;
44
45 int LeerCadenaArchivoConSaltoFinLinea(FILE* fArchivo,const char
       cSeparador, char* sCadenaOut){
46   //FUnción que lee de un archivo una cadena de texto hasta el
         carcater cSeparador o un fin de linea y la guarda en
         sCadenaOut
47   //También está función salta a la sigueinte linea cuando se
         encuentra un fin de linea y detecta cuando se ha llegado al
         final del archivo
48   char cLeido=fgetc(fArchivo);
49   int iNCaracter=0;
50   while (cLeido!=cSeparador && !bPararAplicacion){
51     if (feof(fArchivo) || cLeido==EOF){
52       return-1; //Si llega a fin de archivo antes de llegar al
             separador devuelve -1
53     }
54     if(cLeido!=' '){
55       //Ignora los espacios en blanco
56       sCadenaOut[iNCaracter] = cLeido;
57       iNCaracter++;
58     }
59     cLeido = fgetc(fArchivo);
60     if (cLeido == C_61850_C_NUEVA_LINEA) {
61       //Se ha llegado al fin de la linea
62       break;
63     } else if (cLeido == C_61850_C_RETORNO_CARRO) {
64       //Entra aquí si detecta un retorno de carro
65       cLeido = fgetc(fArchivo); //Lee el siguiente carcater
66       if (cLeido == C_61850_C_NUEVA_LINEA) {
67         //De esta manera se llega a leer el final de linea de los
               archivo de windows ("\r\n")
68         break;//Se sale del ciclo while
69       }
70     }
71   }
72   sCadenaOut[iNCaracter]='\0'; //Para marcar el fin de la cadena
         de texto
73   return 0;//Si no ha habido ningún problema devuelve 0
74 }
75
76 void saltoLineaArchivo (FILE* fArchivo){
```

```
77    //Esta función sierve para pasar a leer la siguiente linea del
          archivo o el final del mismo
78    char cLeido=fgetc(fArchivo);
79    while (cLeido!=C_61850_C_NUEVA_LINEA && !bPararAplicacion){
80      if (feof(fArchivo)) {
81        return; //Ha llegado al final de la linea
82      }
83      //Este ciclo while lee hasta que encuentre el caracter de
            nueva linea
84      if (cLeido==C_61850_C_RETORNO_CARRO){
85        //Entra aquí si detecta un retorno de carro
86        cLeido=fgetc(fArchivo);//Lee el siguiente carcater
87        if(cLeido==C_61850_C_NUEVA_LINEA){
88          //De esta manera se llega a leer el final de linea de los
                archivo de windows ("\r\n")
89          break; //Se sale del ciclo while
90        }
91      }
92      cLeido=fgetc(fArchivo);
93    }
94  }
95
96  void leerServicioTeclado(char* sServicio){
97    //Esta función sirve para leer un servicio por teclado.
98    if(!bPararAplicacion){
99      puts("Servicios disponibles:GetDataValues, GetDataDefinition,
            SetDataValues, GetDataSetValues, GetDataSetValues,
          GetDataSetDirectory, GetBRCBValues, GetURCBValues,");
100     puts("SetBRCBValues, SetURCBValues, GetServerDirectoryL (lo
          hace en logical Device class), GetServerDirectoryF (lo
          hace en File Directory class),");
101     puts("GetLogicalDeviceDirectory, GetFile,
          GetFileAttributeValues, SetFile, DeleteFile, GetGoCBValues
          , SetGoCBValues, SelectActiveSG, SelectEditSG,
          GetEditSGValues, ");
102     puts("GetSGCBValues, SetEditSGValues, ConfirmEditSGValues,
          GetAllDataValues, Select, SelectWithValue, Cancel, Operate
          , GetLogStatusValues, QueryLogAfter");
103     puts("QueryLogByTime, GetLCBValues, SetLCBValues,
          GetMSVCBValues, SetMSVCBValues, GetUSVCBValues y
          SetUSVCBValues");
104     puts("Introduce el servicio que deseas lanzar o FIN para
          finalizar.");
105     scanf("%s", sServicio);
106   }
107 }
108
```

```
109 bool comprobacionMostrarResultados(bool bUsarArchivoServicio,FILE
        * fArchivoServicios){
110   char cLeido;
111   if (bUsarArchivoServicio) {
112     cLeido = fgetc(fArchivoServicios);
113     if (cLeido =='N') {
114       return false;
115     } else if (cLeido == 'S') {
116       return true;
117     } else {
118       printf("Caracter ");
119       putchar(cLeido);
120       puts(" de petición de mostrar resultados no reconocido, por
              defecto no se muestran.");
121       return false;
122     }
123   } else {
124     while (true) {
125       puts("Introduce S si deseas ver los resultado y N sí no");
126       fflush(stdout);//Para que no de error la lectura del
             caracter
127       scanf(" %c", &cLeido);
128       if (cLeido == 'N') {
129         return false;
130       } else if (cLeido == 'S') {
131         return true;
132       } else {
133         puts("Caracter no reconocido.");
134       }
135     }
136   }
137 }
138
139 void mostrarOpciones(){
140   puts("los parámetros del programa son:");
141   puts("./aplicacion_cliente [opciones]  <Resto de parametros>");
142   puts("  opciones:");
143   puts("    -f lanzar servicios por ficheros:");
144   puts("      <Resto de parametros>=[Nombre archivo .csv
        servicios]");
145   puts("    -p lanzar servicio de parámetros de ejecución.");
146   puts("      <Resto de parametros>=[ip] [servicio] [ObjRef] ([
        Mostrar reusltados (S o N)] [FC]) esatos últimos no siempre
        necesarios");
147   puts("Si no se pone parámetros la aplicación los pedirá por
        teclado.");
148 }
```

```
149
150
151 int main(int argc, char** argv) {
152   int tcpPort = 102;
153   InfoObjetoFC ObjetoServicio;
154   char sIP[C_61850_N_CARACTERES_IP+1], sServicioLanzar[
          C_61850_N_CARCACTERES_SERVICIO_MAX+1];
155   char* sNombreArchivoServicios;
156   /*
157    * La variable bUsarArchivoServicio dice si los servicios se
           lanzan por el archivo,
158    * bUsarInformacionParametros por los parámetros de ejecución y
           bMostrarResultado dice si
159    * se muestran los resultados al lanzar el servicio
160    */
161   bool bUsarArchivoServicio=false, bUsarInformacionParametros=
          false, bMostrarResultado=false;
162   //Inicio del gestor de señales
163   signal(SIGINT, sigint_handler);
164   //Lectura de los parámetros de ejecución de la función
165   if(argc>1){
166     if(strcmp(argv[1], "-f") == 0){
167       //Si el primer parámetro es -f significa que se lanza por
              fichero
168       sNombreArchivoServicios = argv[2];
169       bUsarArchivoServicio = true;
170       printf("Los servicios se lanzarán del archivo %s.\n",
              sNombreArchivoServicios);
171     }else if(strcmp(argv[1], "-p") == 0){
172       if(argc<5){
173         puts("Error: número de parámetros insufciente para lanzar
                servicio por parámetros.");
174         mostrarOpciones();
175         goto Fin_programa;//Se va a finalizar el programa
176       }
177       //Si el primer parámetro es -p lanzarán el servicio pasado
              por parámetros de ejecución
178       bUsarInformacionParametros=true;
179       puts("Se lanzará el servicio pasado por parámetros de
              ejecución.");
180       strncpy(sIP,argv[2],C_61850_N_CARACTERES_IP+1);
181       strncpy(sServicioLanzar,argv[3],
              C_61850_N_CARCACTERES_SERVICIO_MAX+1);
182       strncpy(ObjetoServicio.sObjRef,argv[4],
              C_61850_N_CARCACTERES_OBJETO_MAX+1);
183       if(argc>5){
```

```
184           //Si el número de parámetros es mayor que 4 significa que
                  se incluye el mostrar resultados
185           if(strcmp(argv[5], "S") == 0){
186             bMostrarResultado=true;
187           }else if(strcmp(argv[5], "N") == 0){
188             bMostrarResultado=false;
189           }else{
190             printf("El parámetros %s de mostrar reusltados no es
                    correcto, por defencto no se muestran.\n",argv[5]);
191             bMostrarResultado=false;
192           }
193           if(argc>6){
194             //Si el número de parámetros es mayor que 6 significa
                    que se incluye el FC
195             strncpy(ObjetoServicio.sFC,argv[6],
                    C_61850_N_CARACTERES_FC+1);
196           }
197         }
198       }else{
199         puts("Parametros introducido de forma incorrecta.\n");
200         mostrarOpciones();
201         goto Fin_programa;//Se va a finalizar el programa
202       }
203     }else{
204       //Si no hay parámetros se lanzan por lectura de teclado
205       puts("Los servicios se lanzarán por lectura de teclado.");
206     }
207     //Se inicializan las variables donde irá la información de las
          referencias de los objetos
208     InfoObjetoFC DataObject,LogicalNode;
209     referenciaObjeto sObjetoLeido, sDataSet, sBRCB, sURCB,
          sLogicalDevice,sRemoteFile, sLocalFile,sGOOSE, sSGCB,
210               sEditSG, sControlSelect, sControlSelectWithValue,
                    sControlOperate, sLog, sMSVCB, sUSVCB ;
211     int iRespuestaLectura;
212     if (!bUsarInformacionParametros) {
213       //Si no se lanza por parámetros se parsea primero el archivo
            para obtener la información del equipo
214       FILE* fArchivoEquipo = fopen(C_61850_NOMBRE_ARCHIVO_EQUIPO, "
            r");
215       if (fArchivoEquipo == NULL) {
216         printf("No se puede abrir el archivo %s de la información
              del equipo.\n",C_61850_NOMBRE_ARCHIVO_EQUIPO);
217         goto Fin_programa;
218         //Va a finalizar el main
219       }
```

```
220       saltoLineaArchivo(fArchivoEquipo); //Se salta la lineas de
              las cabeceras
221       while (!feof(fArchivoEquipo) && !bPararAplicacion) {
222         //El while lee comto máximo hasta el final del archivo
223         iRespuestaLectura = LeerCadenaArchivoConSaltoFinLinea(
224             fArchivoEquipo, ';', sObjetoLeido);
225         if (iRespuestaLectura != 0) {
226           break; //Se ha llegado al final del archivo
227         }
228         //Se determina de que tipo de objeto es la referncia y se
              guardan sus datos
229         if (strcmp(sObjetoLeido, "IP") == 0) {
230           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                sIP);
231           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                elemento, se salta a la siguiente linea
232         } else if (strcmp(sObjetoLeido, "DataObject") == 0) {
233           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                DataObject.sObjRef);
234           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                DataObject.sFC);
235         } else if (strcmp(sObjetoLeido, "DataSet") == 0) {
236           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                sDataSet);
237           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                elemento, se salta a la siguiente linea
238         }else if(strcmp(sObjetoLeido, "BRCB") == 0){
239           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                sBRCB);
240           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                elemento, se salta a la siguiente linea
241         }else if(strcmp(sObjetoLeido, "URCB") == 0){
242           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                sURCB);
243           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                elemento, se salta a la siguiente linea
244         }else if(strcmp(sObjetoLeido, "LogicalDevice") == 0){
245           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                sLogicalDevice);
246           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                elemento, se salta a la siguiente linea
247         }else if(strcmp(sObjetoLeido, "LogicalNode") == 0){
248           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                LogicalNode.sObjRef);
249           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                LogicalNode.sFC);
250         } else if(strcmp(sObjetoLeido, "RemoteFile") == 0){
```

```
251           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sRemoteFile);
252           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
253         }else if(strcmp(sObjetoLeido, "LocalFile") == 0){
254           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sLocalFile);
255           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
256         }else if(strcmp(sObjetoLeido, "GOOSE") == 0){
257           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sGOOSE);
258           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
259         }else if(strcmp(sObjetoLeido, "SGCB") == 0){
260           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sSGCB);
261           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
262         }else if(strcmp(sObjetoLeido, "EditSG") == 0){
263           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sEditSG);
264           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
265         }else if(strcmp(sObjetoLeido, "ControlSelect") == 0){
266           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sControlSelect);
267           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
268         }else if(strcmp(sObjetoLeido, "ControlSelectWithValue") ==
                0){
269           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sControlSelectWithValue);
270           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
271         }else if(strcmp(sObjetoLeido, "ControlOperate") == 0){
272           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sControlOperate);
273           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
274         }else if(strcmp(sObjetoLeido, "Log") == 0){
275           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sLog);
276           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
277         }else if(strcmp(sObjetoLeido, "MSVCB") == 0){
```

```
278           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sMSVCB);
279           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
280         }else if(strcmp(sObjetoLeido, "USVCB") == 0){
281           LeerCadenaArchivoConSaltoFinLinea(fArchivoEquipo, ';',
                  sUSVCB);
282           saltoLineaArchivo(fArchivoEquipo); //Como Solo tiene un
                  elemento, se salta a la siguiente linea
283         }else {
284           printf("EL objeto de nombre %s no se reconoce.\n",
                  sObjetoLeido);
285           saltoLineaArchivo(fArchivoEquipo); //Se salta a la
                  siguiente linea
286         }
287       }
288       fclose(fArchivoEquipo);
289     }
290     //Se inicia la conexión con el servidor
291     IedClientError error;
292     IedConnection con;
293     CrearConexion(&con, &error, sIP, tcpPort);
294
295     if (error == IED_ERROR_OK) {
296       //Si se puede conectar, se empieza con el lanzamiento de
              servicios
297       //Se inicializan los elementos necesarios
298       MmsValue* ValorObjetoDatos=NULL;
299       MmsValue* EditSG=NULL;
300       ClientReportControlBlock brcb = NULL,urcb = NULL;
301       ClientGooseControlBlock GOOSE=NULL;
302       ControlObjectClient ContSelect=NULL,ContSelectWithValue=NULL,
              ContOperate=NULL;
303       LogStatusValues LogStatusResultado;
304       LogStatusResultado.bLogStatusValido=false;//Se inicializa
              como que no es válido
305       LCBValues LCBResultado;
306       LCBResultado.bLCBValido=false;
307       ClientSVControlBlock MSVCB=NULL, USVCB=NULL;
308       //Ciclo while infinito para el lanzamiento de las peticiones
              de los servicios
309       FILE* fArchivoServicios;
310       if(bUsarArchivoServicio){
311         //De usarse, se abre el archivo con los servicios a mandar
312         fArchivoServicios= fopen(sNombreArchivoServicios, "r");
313         if(fArchivoServicios==NULL){
```

```
314           printf("No se puede abrir el archivo %s de los servicios
                  .\n",sNombreArchivoServicios);
315           goto  Fin_programa;
316         }
317         saltoLineaArchivo(fArchivoServicios);//Se salta la lineas
                de las cabeceras
318       }
319       while(!bPararAplicacion){
320         //En este ciclo while se lanzarán todas las peticiones de
                servicios
321         if(bUsarArchivoServicio){
322           iRespuestaLectura=LeerCadenaArchivoConSaltoFinLinea(
                  fArchivoServicios,';', sServicioLanzar);
323           if(iRespuestaLectura!=0){
324             puts("Se ha llegado al final del archivo de servicios."
                    );
325             break;
326           }
327         }else if(!bUsarInformacionParametros){
328           leerServicioTeclado(sServicioLanzar);
329         }
330         //Se determina el servicio que se deasea lanzar
331         if (strcmp(sServicioLanzar, "GetDataValues") == 0) {
332           puts("###Realización de GetDataValues###");
333           if(bUsarInformacionParametros){
334             ValorObjetoDatos = GetDataValues(con, &error,
                    ObjetoServicio.sObjRef, ObjetoServicio.sFC,
                    bMostrarResultado);
335           }else{
336             bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
337             ValorObjetoDatos = GetDataValues(con, &error,DataObject
                    .sObjRef, DataObject.sFC,bMostrarResultado);
338           }
339         }else if (strcmp(sServicioLanzar, "GetDataDefinition") ==
                0) {
340           puts("###Realización de GetDataDefinition###");
341           if(bUsarInformacionParametros){
342             GetDataDefinition(con, &error,ObjetoServicio.sObjRef,
                    ObjetoServicio.sFC, bMostrarResultado);
343           }else{
344             bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
345             GetDataDefinition(con, &error,DataObject.sObjRef,
                    DataObject.sFC,bMostrarResultado);
346           }
347         }else if (strcmp(sServicioLanzar, "SetDataValues") == 0) {
```

```
348           puts("###Realización de SetDataValues###");
349           if(bUsarInformacionParametros){
350             DataObject=ObjetoServicio;
351           }
352           if(ValorObjetoDatos==NULL){
353             puts("Como no se tienen los valores del objeto de datos
                    se obtienen con un GetDataValues.");
354             ValorObjetoDatos = GetDataValues(con, &error,DataObject
                  .sObjRef, DataObject.sFC, false);
355           }
356           //Después de realizar el GetDataValues se vuelve a
                comprobar si se ha obtenido correctamente
357           if(ValorObjetoDatos!=NULL){
358             SetDataValues(con, &error,DataObject.sObjRef,
                  DataObject.sFC, ValorObjetoDatos);
359           }else{
360             puts("Cómo no se ha realizado correctamente el
                  GetDataValues no se puede relizar el SetDataValues.\
                  n");
361           }
362         }else if(strcmp(sServicioLanzar, "GetDataSetValues") == 0){
363           puts("###Realización de GetDataSetValues###");
364           if(bUsarInformacionParametros){
365             GetDataSetValues(con, &error, ObjetoServicio.sObjRef,
                  bMostrarResultado);
366           }else{
367             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
368             GetDataSetValues(con, &error, sDataSet,
                  bMostrarResultado);
369           }
370         }else if(strcmp(sServicioLanzar, "GetDataSetDirectory") ==
              0){
371           puts("###Realización de GetDataSetDirectory###");
372           if (bUsarInformacionParametros) {
373             GetDataSetDirectory(con, &error, ObjetoServicio.sObjRef
                  ,bMostrarResultado);
374           } else {
375             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
376             GetDataSetDirectory(con, &error, sDataSet,
                  bMostrarResultado);
377           }
378         }else if(strcmp(sServicioLanzar, "GetBRCBValues") == 0){
379           puts("###Realización de GetBRCBValues###");
380           if (bUsarInformacionParametros) {
```

```
381             brcb = GetBRCBValues(con, &error, ObjetoServicio.
                    sObjRef,bMostrarResultado);
382           } else {
383             bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
384             brcb = GetBRCBValues(con, &error, sBRCB,
                    bMostrarResultado);
385           }
386         } else if(strcmp(sServicioLanzar, "GetURCBValues") == 0){
387           puts("###Realización de GetURCBValues###");
388           if (bUsarInformacionParametros) {
389             urcb = GetURCBValues(con, &error, ObjetoServicio.
                    sObjRef,bMostrarResultado);
390           } else {
391             bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
392             urcb = GetURCBValues(con, &error, sURCB,
                    bMostrarResultado);
393           }
394         }else if(strcmp(sServicioLanzar, "SetBRCBValues") == 0){
395           puts("###Realización de SetBRCBValues###");
396           if (bUsarInformacionParametros) {
397             strncpy(sBRCB,ObjetoServicio.sObjRef,
                    C_61850_N_CARCACTERES_OBJETO_MAX+1);
398           }
399           if (brcb == NULL) {
400             puts("Como no se tiene guardado el BRCB se obtiene con
                    un GetBRCBValues.");
401             brcb = GetBRCBValues(con, &error, sBRCB, false);
402           }
403           //Después de realizar el GetBRCBValues se vuelve a
                comprobar si se ha obtenido correctamente
404           if (brcb != NULL) {
405             SetRCBValues(con, &error, brcb);
406           } else {
407             puts("Cómo no se ha realizado correctamente el
                    GetBRCBValues no se puede relizar el SetBRCBValues.\
                    n");
408           }
409         }else if(strcmp(sServicioLanzar, "SetURCBValues") == 0){
410           puts("###Realización de SetURCBValues###");
411           if (bUsarInformacionParametros) {
412             strncpy(sURCB,ObjetoServicio.sObjRef,
                    C_61850_N_CARCACTERES_OBJETO_MAX+1);
413           }
414           if (urcb == NULL) {
```

```
415             puts("Como no se tiene guardado el BRCB se obtiene con
                    un GetBRCBValues.");
416             urcb = GetURCBValues(con, &error, sURCB, false);
417           }
418           //Después de realizar el GetBRCBValues se vuelve a
                 comprobar si se ha obtenido correctamente
419           if (urcb != NULL) {
420             SetRCBValues(con, &error, urcb);
421           } else {
422             puts("Cómo no se ha realizado correctamente el
                    GetURCBValues no se puede relizar el SetURCBValues.\
                    n");
423           }
424         }else if(strcmp(sServicioLanzar, "GetServerDirectoryL") ==
               0){
425           puts("###Realización de GetServerDirectory en logical
                  Device class###");
426           if (!bUsarInformacionParametros) {
427             bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
428           }
429           GetServerDirectory(con, &error, false, bMostrarResultado)
                  ;
430         }else if(strcmp(sServicioLanzar, "GetServerDirectoryF") ==
               0){
431           puts("###Realización de GetServerDirectory en file
                  Directory class###");
432           if (!bUsarInformacionParametros) {
433             bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
434           }
435           GetServerDirectory(con, &error, true, bMostrarResultado);
436         }else if(strcmp(sServicioLanzar, "GetLogicalDeviceDirectory
               ") == 0){
437           puts("###Realización de GetLogicalDeviceDirectory###");
438           if (bUsarInformacionParametros) {
439             GetLogicalDeviceDirectory(con, &error, ObjetoServicio.
                    sObjRef, bMostrarResultado);
440           }else{
441             bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
442             GetLogicalDeviceDirectory(con, &error, sLogicalDevice,
                    bMostrarResultado);
443           }
444         }else if(strcmp(sServicioLanzar, "GetLogicalNodeDirectory")
                == 0){
445           puts("###Realización de GetLogicalNodeDirectory###");
```

```
446           if (bUsarInformacionParametros) {
447             GetLogicalNodeDirectory(con, &error, ObjetoServicio.
                  sObjRef, bMostrarResultado);
448           }else{
449             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
450             GetLogicalNodeDirectory(con, &error, LogicalNode.
                  sObjRef, bMostrarResultado);
451           }
452         }else if(strcmp(sServicioLanzar, "GetAllDataValues") == 0){
453           puts("###Realización de GetAllDataValues###");
454           if (bUsarInformacionParametros) {
455             GetAllDataValues(con, &error, ObjetoServicio.sObjRef,
                  ObjetoServicio.sFC, bMostrarResultado);
456           }else{
457             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
458             GetAllDataValues(con, &error, LogicalNode.sObjRef,
                  LogicalNode.sFC, bMostrarResultado);
459           }
460         }else if(strcmp(sServicioLanzar, "GetFile") == 0){
461           puts("###Realización de GetFile###");
462           if (bUsarInformacionParametros){
463             GetFile(con, &error, ObjetoServicio.sObjRef);
464           }else{
465             GetFile(con, &error, sRemoteFile);
466           }
467         }else if(strcmp(sServicioLanzar, "GetFileAttributeValues")
              == 0){
468           puts("###Realización de GetFileAttributeValues###");
469           if (bUsarInformacionParametros){
470             GetFileAttributeValues(con, &error, ObjetoServicio.
                  sObjRef,bMostrarResultado);
471           }else{
472             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
473             GetFileAttributeValues(con, &error, sRemoteFile,
                  bMostrarResultado);
474           }
475         }else if(strcmp(sServicioLanzar, "SetFile") == 0){
476           puts("###Realización de SetFile###");
477           if (bUsarInformacionParametros){
478             SetFile(con, &error, ObjetoServicio.sObjRef,
                  C_61850_ARCHIVO_BORRAR_REMOTO);
479           }else{
480             SetFile(con, &error, sLocalFile,
                  C_61850_ARCHIVO_BORRAR_REMOTO);
```

```
481           }
482         }else if(strcmp(sServicioLanzar, "DeleteFile") == 0){
483           puts("###Realización de DeleteFile###");
484           if (bUsarInformacionParametros){
485             strncpy(sLocalFile,ObjetoServicio.sObjRef,
                  C_61850_N_CARCACTERES_OBJETO_MAX+1);
486
487           }
488           /*
489            * Como la única carpeta en la que se puede hacer un
                DeleteFile en la de NotValidated
490            * y el equipo elimina al poco tiempo su contenido, antes
                de poder hacer un DeleteFile
491            * hay que hacer un SetFile, por esta razón al querer
                hacer un DeleteFile por parémetros
492            * hay que introducir también la dirección de un archivo
                local.
493            */
494           puts("Antes de hacer el DeleteFile hay que hacer un
                SetFile.\n");
495           if(SetFile(con, &error, sLocalFile,
                C_61850_ARCHIVO_BORRAR_REMOTO)){
496             DeleteFile(con, &error, C_61850_ARCHIVO_BORRAR_REMOTO);
497           }else{
498             puts("Al no poder hacerse el SetFile, no se puede hacer
                  el DeleteFile.");
499           }
500         }else if(strcmp(sServicioLanzar, "GetGoCBValues") == 0){
501           puts("###Realización de GetGoCBValues###");
502           if (bUsarInformacionParametros) {
503             GOOSE = GetGoCBValues(con, &error,ObjetoServicio.
                  sObjRef, bMostrarResultado);
504           } else {
505             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
506             GOOSE = GetGoCBValues(con, &error,sGOOSE,
                  bMostrarResultado);
507           }
508         }else if(strcmp(sServicioLanzar, "SetGoCBValues") == 0){
509           puts("###Realización de SetGoCBValues###");
510           if (bUsarInformacionParametros) {
511             strncpy(sGOOSE,ObjetoServicio.sObjRef,
                  C_61850_N_CARCACTERES_OBJETO_MAX+1);
512           }
513           if(GOOSE==NULL){
514             puts("Como no se tiene la información del GOOSE, se
                  tiene que hacer un GetGoCBValues.");
```

```
515             GOOSE = GetGoCBValues(con, &error,sGOOSE,false);
516           }
517           if(GOOSE!=NULL){
518             SetGoCBValues(con, &error, GOOSE);
519           }else{
520             puts("Como no se ha realizado correctamente el
                  GetGoCBValues, no se puede hacer el SetGoCBValues.")
                  ;
521           }
522         }else if(strcmp(sServicioLanzar, "SelectEditSG") == 0){
523           puts("###Realización de SelectEditSG###");
524           if (bUsarInformacionParametros) {
525             SelectEditSG(con, &error, ObjetoServicio.sObjRef, 4);
526           } else {
527             SelectEditSG(con, &error, sSGCB, 4);
528           }
529         }else if(strcmp(sServicioLanzar, "SelectActiveSG") == 0){
530           puts("###Realización de SelectActiveSG###");
531           if (bUsarInformacionParametros) {
532             SelectActiveSG(con, &error, ObjetoServicio.sObjRef, 3);
533           } else {
534             SelectActiveSG(con, &error, sSGCB, 3);
535           }
536         }else if(strcmp(sServicioLanzar, "GetEditSGValues") == 0){
537           puts("###Realización de GetEditSGValues###");
538           if (bUsarInformacionParametros) {
539             EditSG=GetEditSGValues(con, &error, ObjetoServicio.
                  sObjRef, bMostrarResultado);
540           }else{
541             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
542             EditSG=GetEditSGValues(con, &error, sEditSG,
                  bMostrarResultado);
543           }
544         }else if(strcmp(sServicioLanzar, "GetSGCBValues") == 0){
545           puts("###Realización de GetSGCBValues###");
546           if (bUsarInformacionParametros) {
547             GetSGCBValues(con, &error, ObjetoServicio.sObjRef,
                  bMostrarResultado);
548           }else{
549             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
550             GetSGCBValues(con, &error, sSGCB, bMostrarResultado);
551           }
552         }else if(strcmp(sServicioLanzar, "SetEditSGValues") == 0){
553           puts("###Realización de SetEditSGValues###");
554           if (bUsarInformacionParametros) {
```

```
555             strncpy(sEditSG,ObjetoServicio.sObjRef,
                    C_61850_N_CARCACTERES_OBJETO_MAX+1);
556           }
557           if(EditSG==NULL){
558             puts("Como no se tiene el valor del EditSG, se hace un
                    GetEditSGValues");
559             EditSG=GetEditSGValues(con, &error, sEditSG, false);
560           }
561           if(EditSG!=NULL){
562             SetEditSGValues(con, &error, sEditSG, EditSG);
563           }else{
564             puts("Como no se ha podido realizar el GetEditSGValues,
                     no se puede realizar el SetEditSGValues.\n");
565           }
566         }else if(strcmp(sServicioLanzar, "ConfirmEditSGValues") ==
                0){
567           puts("###Realización de ConfirmEditSGValues###");
568           if (bUsarInformacionParametros) {
569             ConfirmEditSGValues(con, &error, ObjetoServicio.sObjRef
                    );
570           }else{
571             ConfirmEditSGValues(con, &error, sSGCB);
572           }
573         }else if(strcmp(sServicioLanzar, "Select") == 0){
574           puts("###Realización de Select###");
575           if (bUsarInformacionParametros) {
576             strncpy(sControlSelect,ObjetoServicio.sObjRef,
                    C_61850_N_CARCACTERES_OBJETO_MAX+1);
577           }
578           if(ContSelect==NULL){
579             puts("Como no se ha obtenido anteriormente, se obtiene
                    el objeto de control (operaciones de lectura) al que
                     se le realiza Select y Cancel .");
580             ContSelect=CrearObjetoControl(con, sControlSelect);
581           }
582           if(ContSelect!=NULL){
583             Select(ContSelect);
584             puts("Se realiza Cancel para poder seguir haciendo
                    operaciones de control si se desea.");
585             Cancel(ContSelect);
586           }else{
587             puts("Cómo no se ha podido crear el objeto de control,
                    no se ha podido realizar Select.");
588           }
589         }else if(strcmp(sServicioLanzar, "SelectWithValue") == 0){
590           puts("###Realización de SelectWithValue###");
591           if (bUsarInformacionParametros) {
```

```
592             strncpy(sControlSelectWithValue,ObjetoServicio.sObjRef,
                    C_61850_N_CARCACTERES_OBJETO_MAX+1);
593           }
594           if(ContSelectWithValue==NULL){
595             puts("Como no se ha obtenido anteriormente, se obtiene
                    el objeto de control (operaciones de lectura) al que
                     se le realiza SelectWithValue.");
596             ContSelectWithValue=CrearObjetoControl(con,
                    sControlSelectWithValue);
597           }
598           if(ContSelectWithValue!=NULL){
599             SelectWithValue(ContSelectWithValue,MmsValue_newBoolean
                    (true));
600             puts("Se realiza Cancel para poder seguir haciendo
                    operaciones de control si se desea.");
601             Cancel(ContSelectWithValue);
602           }else{
603             puts("Cómo no se ha podido crear el objeto de control,
                    no se ha podido realizar SelectWithValue.");
604           }
605         }else if(strcmp(sServicioLanzar, "Cancel") == 0){
606           puts("###Realización de Cancel###");
607           if (bUsarInformacionParametros) {
608             strncpy(sControlSelect,ObjetoServicio.sObjRef,
                    C_61850_N_CARCACTERES_OBJETO_MAX+1);
609           }
610           if(ContSelect==NULL){
611             puts("Como no se ha obtenido anteriormente, se obtiene
                    el objeto de control (operaciones de lectura) al que
                     se le realiza Select y Cancel .");
612             ContSelect=CrearObjetoControl(con, sControlSelect);
613           }
614           if(ContSelect!=NULL){
615             puts("Se realiza Select para poder hacer Cancel después
                    .");
616             if(Select(ContSelect)){
617               Cancel(ContSelect);
618             }else{
619               puts("Cómo no se ha podido realizar Select, no se
                      puede realizar Cancel.");
620             }
621           }else{
622             puts("Cómo no se ha podido crear el objeto de control,
                    no se ha podido realizar Select.");
623           }
624         }else if(strcmp(sServicioLanzar, "Operate") == 0){
625           puts("###Realización de Operate###");
```

```
626           if (bUsarInformacionParametros) {
627             strncpy(sControlOperate,ObjetoServicio.sObjRef,
                    C_61850_N_CARCACTERES_OBJETO_MAX+1);
628           }
629           if(ContOperate==NULL){
630             puts("Como no se ha obtenido anteriormente, se obtiene
                    el objeto de control (operaciones de lectura) al que
                     se le realiza Operate .");
631             ContOperate=CrearObjetoControl(con, sControlOperate);
632           }
633           if(ContOperate!=NULL){
634             Operate(ContOperate,MmsValue_newBoolean(true));
635           }else{
636             puts("Cómo no se puede crear el objeto de control, no
                    se puede realizar Operate.");
637           }
638         }else if(strcmp(sServicioLanzar, "GetLogStatusValues") ==
                0){
639           puts("###Realización de GetLogStatusValues###");
640           if (bUsarInformacionParametros) {
641             LogStatusResultado = GetLogStatusValues(con, &error,
                    ObjetoServicio.sObjRef,bMostrarResultado);
642           } else {
643             bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
644             LogStatusResultado = GetLogStatusValues(con, &error,
                    sLog, bMostrarResultado);
645           }
646         }else if(strcmp(sServicioLanzar, "QueryLogAfter") == 0){
647           puts("###Realización de QueryLogAfter###");
648           if (bUsarInformacionParametros) {
649             strncpy(sLog,ObjetoServicio.sObjRef,
                    C_61850_N_CARCACTERES_OBJETO_MAX+1);
650           }
651           if(!LogStatusResultado.bLogStatusValido){
652             puts("Como no se han obtenido anteriormente los Log
                    Status, se hace un GetLogStatusValues.");
653             LogStatusResultado = GetLogStatusValues(con, &error,
                    sLog, false);
654           }
655           if(LogStatusResultado.bLogStatusValido){
656             if(!bUsarInformacionParametros){
657               bMostrarResultado = comprobacionMostrarResultados(
                      bUsarArchivoServicio, fArchivoServicios);
658             }
659             QuerLogAfter(con, &error, sLog,&LogStatusResultado,
                    bMostrarResultado);
```

```
660           }else{
661             puts("Como no se han obtenio los Log Status, no se
                  puede realizar el QuerLogAfter");
662           }
663         }else if(strcmp(sServicioLanzar, "QueryLogByTime") == 0){
664           puts("###Realización de QueryLogByTime###");
665           if (bUsarInformacionParametros) {
666             strncpy(sLog,ObjetoServicio.sObjRef,
                  C_61850_N_CARCACTERES_OBJETO_MAX+1);
667           }
668           if(!LogStatusResultado.bLogStatusValido){
669             puts("Como no se han obtenido anteriormente los Log
                  Status, se hace un GetLogStatusValues.");
670             LogStatusResultado = GetLogStatusValues(con, &error,
                  sLog, false);
671           }
672           if(LogStatusResultado.bLogStatusValido){
673             if(!bUsarInformacionParametros){
674               bMostrarResultado = comprobacionMostrarResultados(
                    bUsarArchivoServicio, fArchivoServicios);
675             }
676             QueryLogByTime(con, &error, sLog,&LogStatusResultado,
                  bMostrarResultado);
677           }else{
678             puts("Como no se han obtenio los Log Status, no se
                  puede realizar el QueryLogByTime");
679           }
680         }else if(strcmp(sServicioLanzar, "GetLCBValues") == 0){
681           puts("###Realización de GetLCBValues###");
682           if (bUsarInformacionParametros) {
683             LCBResultado = GetLCBValues(con, &error, ObjetoServicio
                  .sObjRef,bMostrarResultado);
684           } else {
685             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
686             LCBResultado = GetLCBValues(con, &error, sLog,
                  bMostrarResultado);
687           }
688         }else if(strcmp(sServicioLanzar, "SetLCBValues") == 0){
689           puts("###Realización de SetLCBValues###");
690           if (bUsarInformacionParametros) {
691             strncpy(sLog,ObjetoServicio.sObjRef,
                  C_61850_N_CARCACTERES_OBJETO_MAX+1);
692           }
693           if(!LCBResultado.bLCBValido){
694             puts("Como no se han obtenido anteriormente los
                  LCBValues, se hace un GetLCBValues.");
```

```
695             LCBResultado = GetLCBValues(con, &error, sLog, false);
696           }
697           if(LCBResultado.bLCBValido){
698             SetLCBValues(con, &error, sLog,&LCBResultado);
699           }else{
700             puts("Como no se han obtenio los Log Status, no se
                  puede realizar el QueryLogByTime");
701           }
702         }else if(strcmp(sServicioLanzar, "GetMSVCBValues") == 0){
703           puts("###Realización de GetMSVCB###");
704           if (bUsarInformacionParametros) {
705             MSVCB = GetSVCBValues(con,ObjetoServicio.sObjRef,
                  bMostrarResultado);
706           } else {
707             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
708             MSVCB = GetSVCBValues(con,sMSVCB,bMostrarResultado);
709           }
710         }else if (strcmp(sServicioLanzar, "SetMSVCBValues") == 0) {
711           puts("###Realización de SetMSVCBValues###");
712           if(bUsarInformacionParametros){
713             strncpy(sMSVCB,ObjetoServicio.sObjRef,
                  C_61850_N_CARCACTERES_OBJETO_MAX+1);
714           }
715           if(MSVCB==NULL){
716             puts("Como no se tienen los valores del MSVCB se
                  obtienen con un GetMSVCBValues.");
717             MSVCB = GetSVCBValues(con,sMSVCB,false);
718           }
719           if(MSVCB!=NULL){
720             SetSVCBValues(MSVCB);
721           }else{
722             puts("Como no se ha podido realizar correctamente el
                  GetMSVCBValues, no se puede realizar el
                  SetMSVCBValues");
723           }
724         }else if(strcmp(sServicioLanzar, "GetUSVCBValues") == 0){
725           puts("###Realización de GetUSVCB###");
726           if (bUsarInformacionParametros) {
727             USVCB = GetSVCBValues(con,ObjetoServicio.sObjRef,
                  bMostrarResultado);
728           } else {
729             bMostrarResultado = comprobacionMostrarResultados(
                  bUsarArchivoServicio, fArchivoServicios);
730             USVCB = GetSVCBValues(con,sUSVCB,bMostrarResultado);
731           }
732         }else if (strcmp(sServicioLanzar, "SetUSVCBValues") == 0) {
```

```
733           puts("###Realización de SetUSVCBValues###");
734           if(bUsarInformacionParametros){
735             strncpy(sUSVCB,ObjetoServicio.sObjRef,
                  C_61850_N_CARCACTERES_OBJETO_MAX+1);
736           }
737           if(USVCB==NULL){
738             puts("Como no se tienen los valores del USVCB se
                  obtienen con un GetUSVCBValues.");
739             USVCB = GetSVCBValues(con,sUSVCB,false);
740           }
741           if(USVCB!=NULL){
742             SetSVCBValues(USVCB);
743           }else{
744             puts("Como no se ha podido realizar correctamente el
                  GetUSVCBValues, no se puede realizar el
                  SetUSVCBValues");
745           }
746         }else if(strcmp(sServicioLanzar, "FIN") == 0){
747           break;
748         }else {
749           printf("###El nombre %s no es un servicio válido###\n",
                sServicioLanzar);
750         }
751         if(bUsarArchivoServicio){
752           //Si se usa el archivo de lanzar servicios se le pasa a
                la siguiente linea.
753           saltoLineaArchivo(fArchivoServicios);
754         }
755         if(bUsarInformacionParametros){
756           //Si se lanza el servicio lanzado por parámetro sólo se
                hace una iteración
757           break;
758         }
759       }
760       puts("###Lanzamiento de servicios finalizado###");
761       if(bUsarArchivoServicio){
762         //Se cierra el archivo de lanzamiento de servicios de
              haberse usado
763         fclose(fArchivoServicios);
764       }
765       //Se cierra la conexión
766       IedConnection_close(con);
767       //Se libera la memoria asociada de las variables
768       if(ValorObjetoDatos){
769         MmsValue_delete(ValorObjetoDatos);
770       }
771       if (brcb) {
```

```
772         ClientReportControlBlock_destroy(brcb);
773       }
774       if (urcb) {
775         ClientReportControlBlock_destroy(urcb);
776       }
777       if(GOOSE){
778         ClientGooseControlBlock_destroy(GOOSE);
779       }
780       if(EditSG){
781         MmsValue_delete(EditSG);
782       }
783       if(ContSelect){
784         ControlObjectClient_destroy(ContSelect);
785       }
786       if(ContSelectWithValue){
787         ControlObjectClient_destroy(ContSelectWithValue);
788       }
789       if(ContOperate){
790         ControlObjectClient_destroy(ContOperate);
791       }
792       if(MSVCB){
793         ClientSVControlBlock_destroy(MSVCB);
794       }
795       if (USVCB) {
796         ClientSVControlBlock_destroy(USVCB);
797       }
798
799     } else {
800       printf("No se a podido conectar a la IP %s por el puerto TCP
              %i\n", sIP, tcpPort);
801     }
802
803     IedConnection_destroy(con);
804     Fin_programa:
805     return 0;
806   }
```

A.3 Programas Main para la validación de las funciones del RBAC

A.3.1 Validación del paquete para la gestión de las IPs

```
int main (void){
  const char* sIps[]={
    "12.234.23.23",
    "123.23.45.34",
    "255.255.255.255",
    "Esto no es una IP",
    "186.255.145.123",
    "173.34.22.12",
    "267.56.56.56",
    "123.23.45.34",
    "123.23.45.34",
    "123.33.25.88",
    "123.233.233.2"
  };
  const char* sRols[]={
      "rol,valido",
      "EsteRolEsMuyGrandeeeeee,no,se,guarda",
      "DA,IGUAL,LA,IP,ES,MALA",
      "ESTA,CADEANA,TIENE,MUCHOS,ROLES,PARA,VER,EL,FALLO,DE,
         MUCHOS,ROLES",
      "ROLES,VALIDOS,REPET,ROLES",
      "DA,IGUAL,IP,MALA",
      "   ROLES ,   VALIDOS, espacios, eliminar",
      "VA,A,DAR,ERROR,IP,REPETIDA",
      "  ",
      "ip,buena"
  };
  const char* iIps[]={
        "1.234.23.23",
        "12.23.45.34",
        "25.255.255.255",
        "Esto no es una IP",
        "18.255.145.123",
        "17.34.22.12",
        "26.56.56.56",
        "12.23.45.34",
        "12.23.45.34",
        "12.33.25.88",
        "12.233.233.2"
      };
  short int iRoles[12][11];
```

```
41    int i,j;
42    for(i=0; i<12; i++){
43      for (j=0; j<11; j++){
44        iRoles[i][j]=i*j;
45      }
46    }
47    int iNumRoles[11]={2,3,2,0,9,2,3,4,0,2,5};
48    const char ruta[]=sDireccionIP;
49    int nElementos=sizeof(sIps)/sizeof(const char*);
50    puts("Comprobación del guardado de las IPs una a una.");
51    for(i=0;i<nElementos;i++){
52      switch( GrabarIPFichero(sIps[i],sRols[i],NULL,ruta,1)){
53        case 0:{
54          printf("La IP %s se ha guardado bien.\n",sIps[i]);
55        }; break;
56        case -1:{
57          printf("La IP %s no es válida.\n",sIps[i]);
58        };break;
59        case -2:{
60          printf("En la IP %s no ha habido un problema con el
                archivo.\n",sIps[i]);
61        }; break;
62        case -3:{
63          printf("La IP %s ya se encuentra guardada.\n",sIps[i]);
64        }; break;
65        case -4 :{
66          printf("En la IP %s se ha alcanzado el número máximo de
                IPs guardadas.\n",sIps[i]);
67        }; break;
68        case -5 : {
69          printf("En los roles %s se encuentra uno con demasiados
                carácteres.\n", sRols[i]);
70        }; break;
71        case -6: {
72          printf("Los roles %s son demasiados.\n",sRols[i]);
73        };break;
74        case -7: {
75          printf("La ip %s no tiene roles.\n",sIps[i]);
76        };break;
77        case -8: {
78          printf("Los roles %s tiene algunos repetidas.\n",sRols[i
                ]);
79        }
80      }
81      switch( GrabarIPFichero(iIps[i],NULL,iRoles[i],ruta,iNumRoles
            [i])){
82        case 0:{
```

```
83          printf("La IP %s se ha guardado bien.\n",iIps[i]);
84        }; break;
85        case -1:{
86          printf("La IP %s no es válida.\n",iIps[i]);
87        };break;
88        case -2:{
89          printf("En la IP %s no ha habido un problema con el
                archivo.\n",iIps[i]);
90        }; break;
91        case -3:{
92          printf("La IP %s ya se encuentra guardada.\n",iIps[i]);
93        }; break;
94        case -4 :{
95          printf("En la IP %s se ha alcanzado el número máximo de
                IPs guardadas.\n",iIps[i]);
96        }; break;
97        case -5 : {
98          printf("Los roles de la IP %s son demasiados.\n", iIps[i
                ]);
99        }; break;
100       case -6: {
101         printf("La IP %s no tiene roles guardados.\n",iIps[i]);
102       };break;
103       case -7:{
104         printf("La IP %s tiene dos o más roles iguales.\n",iIps[i
                ]);
105       }
106     }
107   }
108   puts("\n \nComprobación de elementos guardados.\n");
109   FILE *fArchivo=fopen(ruta,"rb");
110   if (fArchivo!=NULL){
111     struct informacionIP inforLeida;
112     char *some_addr;
113     struct in_addr ip_addr;
114     int i;
115     while (fread(&inforLeida,sizeof(inforLeida),1,fArchivo)!=0){
116       ip_addr.s_addr =inforLeida.IPNum;
117       some_addr = inet_ntoa(ip_addr);
118       printf("La IP de número %lu corresponde a la IP %s.\n",
              inforLeida.IPNum,some_addr);
119       puts("Sus roles son:");
120       for (i=0; i<inforLeida.iNumeroRoles;i++){
121         if (inforLeida.iInfoIP==RolesString){
122           puts(inforLeida.roles.cadenas[i]);
123         }else if(inforLeida.iInfoIP==RolesInt){
124           printf("%d.\n",inforLeida.roles.enteros[i]);
```

```
125           }
126         }
127         puts("\n");
128       }
129     }else{
130       puts("No se ha podido leer el archivo para escribir.");
131     }
132     fclose(fArchivo);
133     char sRolesModificados[]="ROLES,MODIFI,CADOS,BIEN";
134     short int iRolesMod[]={1,66,99,232};
135     ModificarIPFichero("17.34.22.12", sRolesModificados, NULL, ruta
          , 4);
136     ModificarIPFichero("255.255.255.255", sRolesModificados, NULL,
          ruta, 4);
137     ModificarIPFichero("25.255.255.255",NULL,iRolesMod, ruta, 4);
138     ModificarIPFichero("173.34.22.12",NULL, iRolesMod, ruta, 4);
139     puts("\n \nComprobación de elementos guardados después de la
          modificación.\n");
140     fArchivo=fopen(ruta,"rb");
141     if (fArchivo!=NULL){
142       struct informacionIP inforLeida;
143       char *some_addr;
144       struct in_addr ip_addr;
145       while (fread(&inforLeida,sizeof(inforLeida),1,fArchivo)!=0){
146         ip_addr.s_addr =inforLeida.IPNum;
147         some_addr = inet_ntoa(ip_addr);
148         printf("La IP de número %lu corresponde a la IP %s.\n",
              inforLeida.IPNum,some_addr);
149         puts("Sus roles son:");
150         for (i=0; i<inforLeida.iNumeroRoles;i++){
151           if (inforLeida.iInfoIP==RolesString){
152             puts(inforLeida.roles.cadenas[i]);
153           }else if(inforLeida.iInfoIP==RolesInt){
154             printf(" %d.\n",inforLeida.roles.enteros[i]);
155           }
156         }
157         puts("\n");
158       }
159     }else{
160       puts("No se ha podido leer el archivo para escribir.");
161     }
162     fclose(fArchivo);
163     printf(" %d\n", EliminarIPFichero("17.34.22.12",ruta));
164       printf(" %d\n",EliminarIPFichero("255.255.255.255",ruta));
165       ModificarIPFichero("255.255.255.255", "COMP,NO,MOD,IP,ELIMI",
            RolesString, ruta, 4);
166       printf(" %d\n",EliminarIPFichero("123.23.45.34",ruta));
```

```
167       printf("%d\n",EliminarIPFichero("123.233.233.2",ruta));
168       puts("\n \nComprobación de elementos guardados después de
              eliminar una IP.\n");
169       fArchivo=fopen(ruta,"rb");
170       if (fArchivo!=NULL){
171         struct informacionIP inforLeida;
172         char *some_addr;
173         struct in_addr ip_addr;
174         while (fread(&inforLeida,sizeof(inforLeida),1,fArchivo)!=0)
                {
175           ip_addr.s_addr =inforLeida.IPNum;
176           some_addr = inet_ntoa(ip_addr);
177           printf("La IP de número %lu corresponde a la IP %s.\n",
                  inforLeida.IPNum,some_addr);
178           puts("Sus roles son:");
179           for (i=0; i<inforLeida.iNumeroRoles;i++){
180             if (inforLeida.iInfoIP==RolesString){
181               puts(inforLeida.roles.cadenas[i]);
182               }else if(inforLeida.iInfoIP==RolesInt){
183                printf("%d.\n",inforLeida.roles.enteros[i]);
184             }
185           }
186           puts("\n");
187         }
188       }else{
189         puts("No se ha podido leer el archivo para escribir.");
190       }
191     fclose(fArchivo);
192     remove(ruta);
193     puts("Comprobación del guardado de las IPs mediante la función
            IPsCSV2ArchivoDAT.");
194     if(IPsCSV2ArchivoDAT("IP_Roles.csv",ruta)==0){
195       FILE *fArchivoNuevo=fopen(_61850_DIRECCION_DATOS_IP,"rb");
196       if (fArchivoNuevo!=NULL){
197         struct informacionIP inforLeida;
198         char *some_addr;
199         struct in_addr ip_addr;
200         int i;
201         while (fread(&inforLeida,sizeof(inforLeida),1,fArchivoNuevo
                )!=0){
202           ip_addr.s_addr =inforLeida.IPNum;
203           some_addr = inet_ntoa(ip_addr);
204           printf("La IP de número %lu corresponde a la IP %s.\n",
                  inforLeida.IPNum,some_addr);
205           puts("Sus roles son:");
206           for (i=0; i<inforLeida.iNumeroRoles;i++){
207             if (inforLeida.iInfoIP==_61850_ROLES_STRING){
```

```
208               puts(inforLeida.roles.cadenas[i]);
209             }else if(inforLeida.iInfoIP==_61850_ROLES_INT){
210               printf(" %d.\n",inforLeida.roles.enteros[i]);
211             }
212           }
213           puts("\n");
214         }
215         fclose(fArchivoNuevo);
216         remove(ruta);
217       }else{
218         puts("No se ha podido leer el archivo de las IPs.");
219       }
220     }else{
221       puts("No se ha podido pasar correctamente las IPs del archivo
              .csv al .dat.");
222     }
223       return 0;
224   }
```

A.3.2 Validación del paquete para la gestión de los roles del archivo RBAC

A.3.2.1 Funciones correspondientes únicamente a los roles del archivo RBAC.

```
1   int main (void){
2     int iResultado=CargarRolesFicheroRBAC(sDirecionRBAC);
3     if(iResultado==0){
4       puts("Los roles guadaddos son:");
5       VisualizarEstructuraRoles();
6     }else{
7       switch(iResultado){
8         case -1:{
9           puts("No ha sido posible abrir el archivo.");
10        };break;
11        case -2:{
12          puts("Hay un rol demasiado grande.");
13        };break;
14        case -3:{
15          puts("En el archivo hay demasiados roles.");
16        };break;
17        case -4: {
18          puts("Hay dos roles con el mismo nombre.");
19        };break;
20        case -5: {
21          puts("Hay dos roles con el mismo id");
22        }
23      }
```

```
24      }
25      char* sRols[]={
26            "VIEWER",
27            "OPERATOR",
28            "NOEXISTE",
29        };
30      puts("\n");
31      short int iRols[]={4,-16,27};
32      short int iID;
33      int tipoRol;
34      char sNombre[nMaxCaracteres+1];
35      long int iMascaraPermiso, iMascaraEmergencia;
36      puts("Buscar la información rol por ID y por nombre.");
37      for (int i=0; i<3; i++){
38        iResultado=BuscarInformacionRolEstrcutura(sRols[i],&iID,&
              iMascaraPermiso,&iMascaraEmergencia,&tipoRol);
39        switch (iResultado){
40          case -1:{
41            puts("No se ha inicializado la estructura de los roles.")
                  ;
42          }; break;
43          case -2:{
44            printf("El rol %s no se ha encontrado.\n",sRols[i]);
45          }; break;
46          case 0:{
47            printf("El rol %s tiene la máscara de permisos %ld y la
                  mascara de permisos de emergencia %ld. Su id es %hd.\n
                  ",sRols[i],iMascaraPermiso,iMascaraEmergencia,iID);
48          }
49        }
50        strcpy(sNombre,_61850_STRING_ROL_NULO);
51        iResultado=BuscarInformacionRolEstrcutura(sNombre,&iRols[i],&
              iMascaraPermiso,&iMascaraEmergencia,&tipoRol);
52        switch (iResultado){
53          case -1:{
54            puts("La estructura está vacía.");
55          }; break;
56          case -2:{
57            printf("El rol %d no se ha encontrado.\n",iRols[i]);
58          }; break;
59          case 0:{
60            printf("El rol %d tiene la máscara de permisos %ld y la
                  mascara de permisos de emergencia %ld. Su nombre es %s
                  .\n",iRols[i],iMascaraPermiso,iMascaraEmergencia,
                  sNombre);
61          }
62        }
```

```
63      }
64      return 0;
65    }
```

A.3.2.2 Funciones para los roles de las direcciones IPs

```
1   int main (void){
2     puts("Inicialización del archivo RBAC.\n");
3     int iResultado=CargarRolesFicheroRBAC(sDirecionRBAC);
4     if(iResultado==0){
5         puts("\n\n Comprobación los roles RBAC guardados.\n\n");
6             VisualizarEstructuraRoles();
7     }
8     puts("\n\nInicialización del archivo IPS.\n");
9     const char* sIps[]={
10        "12.234.23.23",
11        "123.23.45.34",
12        };
13    const char* sRols[]={
14        "VIEWER,OPERATOR",
15        "INSTALLER,RBACMGMT,ADMIN",
16      };
17    const char* iIps[]={
18      "1.234.23.23",
19      "12.23.45.34",
20      };
21    short int iRoles[3]={1,2,3};
22    int iNumRoles[2]={2,3};
23    const char ruta[]=sDireccionIP;
24    int nElementos=sizeof(sIps)/sizeof(const char*);
25    for(int i=0;i<nElementos;i++){
26      switch( GrabarIPFichero(sIps[i],sRols[i],NULL,ruta,1)){
27        case 0:{
28          printf("La IP  %s se ha guardado bien.\n",sIps[i]);
29        };
30      }
31      switch( GrabarIPFichero(iIps[i],NULL,iRoles,ruta,iNumRoles[i
            ])){
32        case 0:{
33          printf("La IP  %s se ha guardado bien.\n",iIps[i]);
34        };
35      }
36    }
37    puts("Creacción de la estructura de direcciones IP con sus
        roles.");
38    iResultado= FicheroIp2EstructuraIPRol (ruta);
39    if(iResultado==0){
40        puts("La estructura se ha creado correctamente.\n");
```

```
41          puts("Comprobación de la información de los roles de cada
                IP yendo uno por uno:\n");
42          for(int i=0;i<nElementos;i++){
43              visualizarDatosIPEstrucura(sIps[i])
44            visualizarDatosIPEstrucura(iIps[i])
45          }
46          puts("Comprobación de la información de los roles de cada
                IP mediante la función VisualizarEstructuraIPs:\n");
47          VisualizarEstructuraIPs();
48      }else{
49          printf("Fallo en la creación de la estructura con código %d
                .",iResultado);
50      }
51      remove(ruta);
52      return 0;
53  }
```

A.3.3 Funciones para la gestión de los permisos de los servicios.

```
1   int main (void){
2     InicializarPermisos(sDirecionCSV);
3     puts("*******Comprobación servicios guardados*****************\
          n");
4     puts("Los servicios guardados son:\n");
5     visualizarServiciosGuardados();
6     const char* sServicios[]={
7           "Cancel",
8           "DeleteDataSet",
9           "GetFile",
10          "GetSGCBValues",
11          "Associate",
12          "Release",
13          "Release",
14          "DeleteDataSet",
15          "iaia",
16          };
17    const int iBitPermiso[]={7, 1, 5, 2, 3,11, 10,8, 24};
18    puts("\n\n\n*******Comprobación de acceso a servicios
          Directamente*****************\n");
19    int i;
20    for (i=0; i<sizeof(sServicios)/sizeof(const char*); i++){
21      int iResultado=AccesoServicioConcedido (sServicios[i],
            potencia(2,iBitPermiso[i]));
22      switch(iResultado){
23        case _61850_ACCION_DENEGADA:{
```

```
24          printf("El solicitante con el bit de permiso %d, puede
                acceder al servicio %s.\n",iBitPermiso[i],sServicios[i
                ]);
25        };break;
26        case _61850_ACCION_PERMITIDA:{
27          printf("El solicitante con el bit de permiso %d, no puede
                 acceder al servicio %s.\n",iBitPermiso[i],sServicios[
                i]);
28        };break;
29      }
30    }
31    return 0;
32  }
```

A.3.4 Validación de la función para la inicialización del RBAC

```
1   int main (void){
2     inicializarConfiguracionRBAC(sDireccionCSVIPs, sDireccionIP,
          sDirecionRBAC,sDirecionCSV);
3     puts("Comprobación de las IPs guardadas.\n");
4     FILE *fArchivo=fopen(sDireccionIP,"rb");
5     if (fArchivo!=NULL){
6       struct informacionIP inforLeida;
7       char *some_addr;
8       struct in_addr ip_addr;
9       int i;
10      while (fread(&inforLeida,sizeof(inforLeida),1,fArchivo)!=0){
11        ip_addr.s_addr =inforLeida.IPNum;
12        some_addr = inet_ntoa(ip_addr);
13        printf("La IP de número %lu corresponde a la IP %s.\n",
              inforLeida.IPNum,some_addr);
14        puts("Sus roles son:");
15        for (i=0; i<inforLeida.iNumeroRoles;i++){
16          if (inforLeida.iInfoIP==RolesString){
17            puts(inforLeida.roles.cadenas[i]);
18          }else if(inforLeida.iInfoIP==RolesInt){
19            printf(" %d.\n",inforLeida.roles.enteros[i]);
20          }
21        }
22        puts("\n");
23      }
24    }else{
25      puts("No se ha podido leer el archivo para escribir.");
26    }
27    puts("\n\n Comprobación los roles RBAC guardados.\n\n");
28    VisualizarEstructuraRoles();
```

```
29    puts("\n\n Comprobación de la estructura de IPs.\n\n");
30    VisualizarEstructuraIPs();
31    puts("\n\n Comprobación de la estructura de servicios con sus
          permisos.\n\n");
32    visualizarServiciosGuardados();
33    return 0;
34  }
```

A.3.5 Validación del paquete para la actualización de la configuración del RBAC

```
1   int main (void){
2       int iResultadoActualizacionCSVIP,IResultadoActualizacionRBAC,
            IResultadoActualizarPermisos, iIteracion=0;
3     puts("Carga de datos inicial.");
4     inicializarConfiguracionRBAC(_61850_DIRECCION_SCV_IP_ROLES,
          _61850_DIRECCION_DATOS_IP, _61850_DIRECCION_RBAC,
          _61850_DIR_CSV_PERMISOS_SERVICIOS);
5     puts("\n\nComprobación los roles RBAC guardados.\n\n");
6     VisualizarEstructuraRoles();
7     puts("\n\nComprobación de la estructura de IPs.\n\n");
8     VisualizarEstructuraIPs();
9     puts("\n\nComprobación de la estructura de servicios con sus
          permisos.\n\n");
10    visualizarServiciosGuardados();
11    puts("Comienzo del ciclo while para comprobar actualizaciones")
          ;
12    while(iIteracion<40){
13      //Hace 40 comprobaciones
14      printf("Comprobacion numero %d.\n",iIteracion+1);
15      iResultadoActualizacionCSVIP=ComprobarActualizacionCSVIP(
            _61850_UPDATE_CSV_IP_ROLES,_61850_DIRECCION_SCV_IP_ROLES,
            _61850_DIRECCION_DATOS_IP, _61850_DIRECCION_RBAC);
16      if(iResultadoActualizacionCSVIP==_61850_ACTUALIZACION_OK){
17        puts("Configuracion actualizada correctamente con el nuevo
              fichero de IPs.");
18        puts("Comprobacion de las nuevas IPs guardadas.");
19        puts("\n\nComprobación de la estructura de IPs.\n\n");
20        VisualizarEstructuraIPs();
21        puts("");
22      }else if(iResultadoActualizacionCSVIP!=
            _61850_ACCESO_IMPOSIBLE){
23        printf("Fallo en la actualizacion del archivo de IPs con
              codigo %d.\n",iResultadoActualizacionCSVIP);
24        puts("\n\nComprobacion de la estructura de IPs.\n\n");
25        VisualizarEstructuraIPs();
26      }
```

```
27      IResultadoActualizacionRBAC=ComprobarActualizacionArchivoRBAC
            (_61850_DIRECCION_DATOS_IP, _61850_UPDATE_RBAC,
            _61850_DIRECCION_RBAC);
28      if (IResultadoActualizacionRBAC == _61850_ACTUALIZACION_OK) {
29        puts("Configuracion actualizada correctamente con el nuevo
              fichero RBAC.");
30        puts("\n\nComprobacion los roles RBAC guardados.\n\n");
31        VisualizarEstructuraRoles();
32        puts("\n\nComprobacion de la estructura de IPs.\n\n");
33        VisualizarEstructuraIPs();
34        puts("");
35      } else if (IResultadoActualizacionRBAC !=
            _61850_ACCESO_IMPOSIBLE) {
36        printf("Fallo en la actualizacion del archivo de RBAC con
              codigo %d.\n", iResultadoActualizacionCSVIP);
37        puts("\n\nComprobacion los roles RBAC guardados.\n\n");
38        VisualizarEstructuraRoles();
39        puts("\n\nComprobacion de la estructura de IPs.\n\n");
40        VisualizarEstructuraIPs();
41        puts("");
42      }
43      IResultadoActualizarPermisos=
            ComprobarActualizacionCSVPermisosServicios (
            _61850_UPDATE_CSV_PERMISOS_SERVICIOS,
            _61850_DIR_CSV_PERMISOS_SERVICIOS);
44      if (IResultadoActualizarPermisos == _61850_ACTUALIZACION_OK)
            {
45        puts( "Configuracion actualizada correctamente con el nuevo
              fichero csv de permisos.");
46        puts("\n\nComprobación de la estructura de servicios con
              sus permisos.\n\n");
47        visualizarServiciosGuardados();
48        puts("");
49      } else if (IResultadoActualizarPermisos !=
            _61850_ACCESO_IMPOSIBLE) {
50        printf("Fallo en la actualizacion del archivo csv de
              permisos con codigo %d.\n", iResultadoActualizacionCSVIP
              );
51        puts("\n\nComprobación de la estructura de servicios con
              sus permisos.\n\n");
52        visualizarServiciosGuardados();
53        puts("");
54      }
55      iIteracion++;
56      sleep(1);//Duerme 1 segundo
57    }
58    puts("Fin programa");
```

```
59      return 0;
60  }
```