minsait

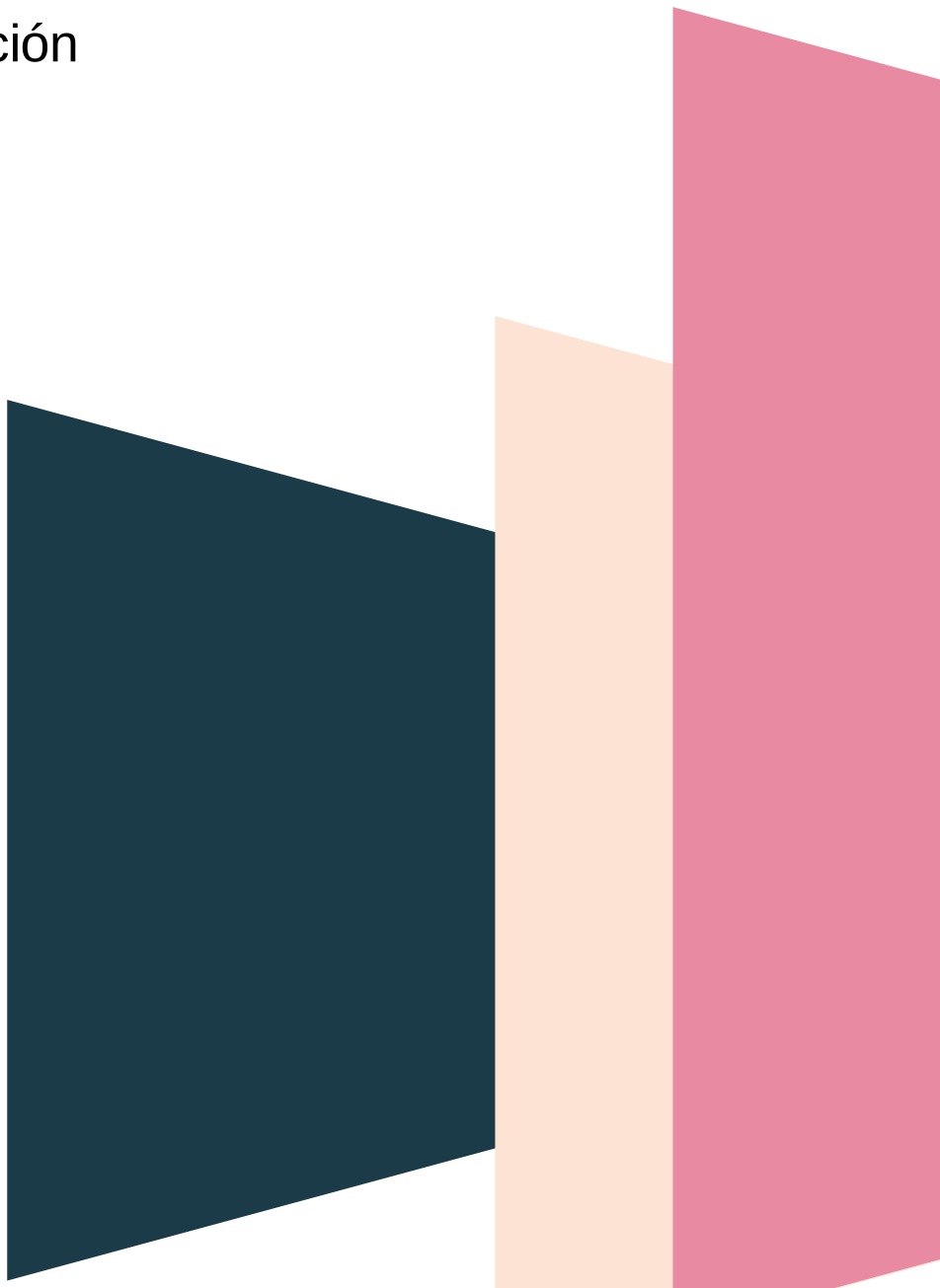
An Indra company

EDITRAN/GT 2.1

Gestor de Transmisiones Web de EDITRAN
Windows/Unix
Manual de Instalación

mayo de 2019

ecosystems
communications



1. Introducción

EDITRAN/GT (Gestor de Transmisiones) permite gestionar las trasmisiones de EDITRAN para poder realizar Emisiones y / o Recepciones de Presentaciones de EDITRAN eficientemente y de una manera desatendida.

EDITRAN/GT es ampliamente parametrizable para conseguir el máximo de flexibilidad, dispone de una aplicación web de configuración que permite definir ventanas de emisión, intervalos de búsquedas, procesos posteriores, reintentos, etc.

En emisión, EDITRAN/GT busca los archivos que están dados de alta en las Presentaciones de EDITRAN/G para transmitirlos de manera que cuando aparece un archivo de emisión asociado a una Presentación de EDITRAN/G y si se encuentra dentro de la ventana horaria seleccionada, se desencadenará automáticamente la emisión de dicha Presentación. Esto es muy útil para automatizar la transmisión de los datos cuando éstos existan aumentando la eficiencia del sistema.

Otra faceta, no menos importante, es la conexión existente entre EDITRAN/GT y el resto de EDITRAN. EDITRAN/GT ha sido diseñado como Gestor de Transmisiones de EDITRAN y como tal, es capaz de leer los perfiles de EDITRAN/G y de EDITRAN/P, de esta manera se hace más sencilla la configuración de EDITRAN/GT y permite la consulta de características de EDITRAN, como el estado de las presentaciones, que podrán ayudar a determinar el motivo de una posible incidencia en un envío o recepción por EDITRAN.

Por último destacar que existen un archivo de log y una tabla de eventos propia de EDITRAN/GT que junto a los archivos de log y estados ya existentes en EDITRAN/P y EDITRAN/G, forman un conjunto completo de información importante tanto de transmisiones correctas como de incidencias en alguna de éstas.

2. Instalación y requisitos

2.1. Requisitos de instalación

2.1.1. Requisitos de software

La instalación del nuevo módulo exige tener instalada una versión de EDITRAN 5.1 ó superior con licencia de uso.

Por otra parte, debido a que alguna de las nuevas funcionalidades se han desarrollado en Java, hay requisitos adicionales a los de una instalación estándar de EDITRAN:

- Tener instalado correctamente **Java JRE 6 o superior**. Para tener compatibilidad con EDITRAN debemos usar una versión de Java de la misma arquitectura que la versión de EDITRAN instalada; esto es, una **versión de 64 bits en máquinas HP-Itanium, HP-UX y Linux** y una **versión de 32 bits en máquinas Windows y el resto de máquinas Unix**.
  Si no tuviera Java puede descargarlo desde http://www.oracle.com/technetwork/java.
- Tener instalada la JCE (Java Cryptography Extension) de la versión de Java utilizada. Se puede descargar de http://www.oracle.com/technetwork/java.
- Tener instalado un servidor web J2EE. Por defecto aconsejamos **Tomcat 6**.
- Tener instalada una Base de Datos relacional que soporte SQL y sea accesible por JDBC. Por defecto, aconsejamos **MySQL 5.6**. También se ha probado la compatibilidad con **ORACLE 11g**, **DERBY 10** y **MS SQL SERVER 2008**.

2.1.2. Requisitos de sistema

Para instalar EDITRANGT junto con Java 6, Tomcat 6 y MYQSL 5 es necesario disponer de al menos 1 GB libre de espacio en disco. Para su ejecución, teniendo en cuenta los recursos utilizados por Tomcat 6 y MYSQL 5, es recomendable tener disponible como mínimo 512MB libres de memoria RAM para cada una de las aplicaciones de EDITRAN/GT (al gestor de transmisiones y a la aplicación web).

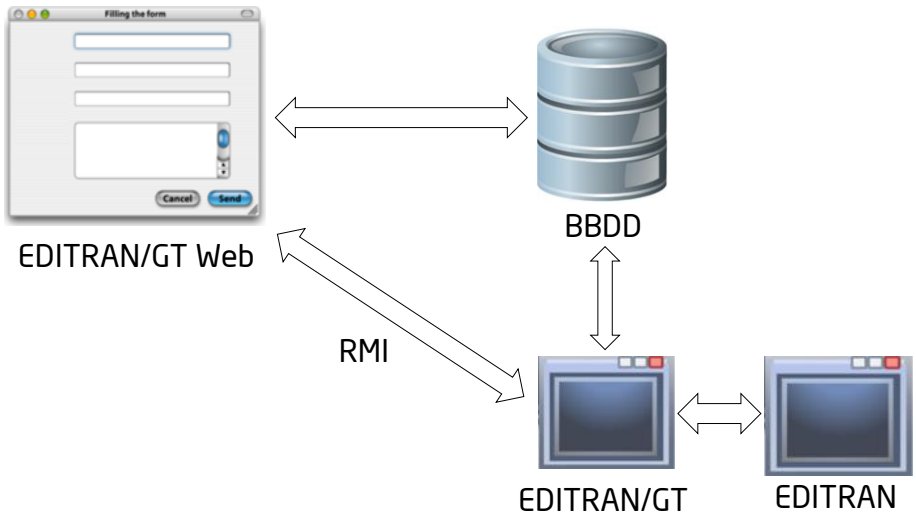
En caso de usar una Base de Datos o un servidor web distinto de los recomendados por defecto, deberá comprobar sus requisitos propios.

2.2. Arquitectura lógica de EDITRAN/GT

La aplicación EDITRAN/GT se compone de dos módulos independientes entre sí. Estos dos módulos son:

- **EDITRANGT**: Es propiamente el gestor de transmisiones de EDITRAN. Es el encargado de interactuar con EDITRAN y realizar las transmisiones, por lo que **debe estar instalado en una máquina en la que haya una distribución de EDITRAN**.
- **EDITRANGTWeb**: Es la aplicación web de EDITRAN/GT. Muestra una interfaz gráfica web mediante la cual el usuario puede editar y consultar la configuración de EDITRAN/GT. Para poder usar esta aplicación web debemos tener instalado en la máquina un servidor web.

Estas dos aplicaciones se comunican entre sí mediante el sistema JAVA RMI y comparten una misma base de datos, por lo que no es necesario que estas dos aplicaciones estén en la misma máquina. Por tanto el esquema de EDITRAN/GT es el siguiente:

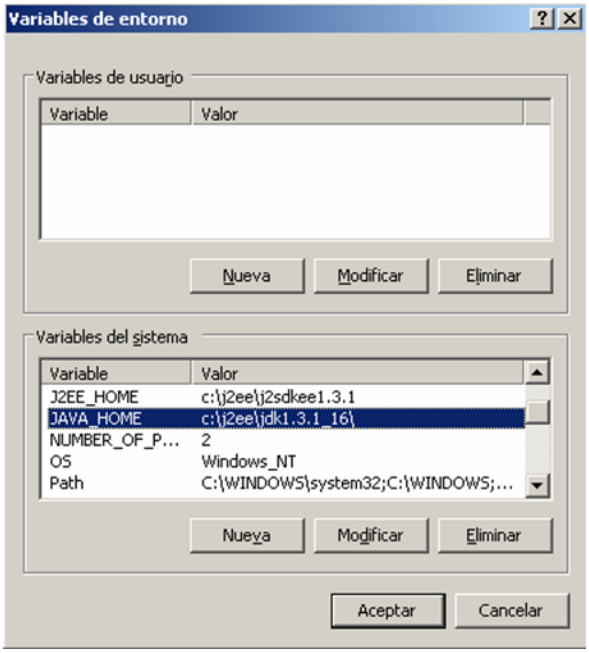


2.3. Instrucciones para la instalación

Para instalar el producto, descomprimiremos el archivo de distribución **zip** en el directorio de **EDITRAN** en los sistemas Windows, o descomprimiremos el archivo **tar.gz** en los sistemas UNIX. Puede ser instalado en otro directorio distinto, pero requiere modificar los archivos de comandos internos para indicar tanto el directorio de **EDITRAN** como el de **EDITRAN/GT**, así que por defecto es recomendable instalarlo dentro de la carpeta raíz de **EDITRAN**.

Como resultado, se habrán creado los directorios **/EDITRANGT** y **/EDITRANGTWeb** en el directorio donde esté instalado EDITRAN, que son la aplicación del gestor de transmisiones EDITRAN/GT el primero, y la aplicación web de EDITRAN/GT el segundo.

Es conveniente que el entorno del usuario esté preparado para ejecutar aplicaciones Java. Si no lo tuviera ya, definir **JAVA_HOME** como variable de entorno y modifique **PATH** para que incluya *$(JAVA_HOME)/bin*.



Para la instalación de la BBDD de la aplicación es recomendable que además se modifique la variable de entorno PATH para que incluya el directorio de aplicaciones (binarios) del servidor de BBDD.

2.4. Descripción de la estructura de archivos instalados

- El directorio base **EDITRAN/GT** contiene los siguientes elementos principales:
  - El script **start_EDItranGT para Windows y Unix/Linux**, que será el utilizado para iniciar la aplicación del buscador de EDITRAN/GT.
  - El subdirectorio de configuración **conf** con los ficheros de configuración de EDITRAN/GT.
  - El subdirectorio **doc**, con la documentación de EDITRAN/GT.
  - El subdirectorio **inst**, con los ficheros para configurar **MySQL5.6** (la BBDD por defecto).
  - El subdirectorio **lib**, con las librerías Java para ejecutar el programa EDITRAN/GT.
  - El subdirectorio **util**, con utilidades para EDITRAN/GT
  - El subdirectorio **Wrapper**, con los archivos que nos permitirán instalar EDITRAN/GT como un servicio de Windows (**solo para Windows**).
- El directorio base de **EDItranGTWeb** contiene los siguientes elementos principales:
  - El archivo de aplicación web **EDItranGT.war**.
  - El subdirectorio de configuración **conf** con los ficheros de configuración de EDITRAN/GT Web.
  - El subdirectorio **doc**, con la documentación de EDITRAN/GT Web.
  - El subdirectorio **inst**, con los ficheros para configurar **MySQL5.6** y **Tomcat 6** (la BBDD y el servidor Web por defecto).
  - El subdirectorio **ssl**, con el almacén de certificados de EDITRAN/GT Web para la conexión segura HTTPS.

3. Base de datos

3.1. Configuración para el gestor de transmisiones EDITRAN/GT

Una vez se ha instalado un servidor de BBDD es necesario crear el esquema de EDITRAN/GT que almacena los datos de la aplicación. Para crearlo se deben realizar los siguientes pasos.

3.1.1. Propiedades de EDITRAN/GT

EDITRAN/GT por defecto espera que la BBDD esté instalada en el mismo servidor donde se ejecuta EDITRAN. En este caso nos conectaremos a una base de datos local **MySQL** arrancada en el **puerto TCP 3306** con el usuario ***editrangt*** y la contraseña ***editrangt,*** que utiliza un esquema llamado **EDITRANGT** donde se crean las tablas que necesita.

Esta configuración por defecto está guardada en el fichero **EDITRANGT_DIR\conf\EDITRANGTconf.properties**. Si se quiere modificar la conexión debe ser editado.

> En el caso de que la BBDD sea diferente a **MYSQL5.6** (como por ejemplo una BBDD ORACLE 11g) se deben configurar las siguientes propiedades: **jpa.database**, **jpa.databasePlatform**, **jpa.database.create**, **jdbc.driverClassName** y **jdbc.url**.

Contenido del archivo:

```
jpa.database=MYSQL
jpa.databasePlatform=org.eclipse.persistence.platform.database.MySQLPlatform
jpa.database.create= MYSQL
jpa.showSql=false
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/EDITRANGT
jdbc.username=ENC(ok1Z5QGnroJsaTAU054GuScvFyGglX+m)
jdbc.password=ENC(ok1Z5QGnroJsaTAU054GuScvFyGglX+m)
planificador.host=127.0.0.1
planificador.port=1099
planificador.alertas=false
```

Las propiedades de este archivo son:

- **Jpa.database**: es el tipo de la BBDD utilizada, según la lista de tipos de BBDD soportados por la especificación de JPA, que por defecto es MYSQL, otros valores son ORACLE, etc...[1].
- **Jpa.databasePlatform**: es la clase de Java que cumple con las especificaciones de JPA para realizar la persistencia de objetos en el tipo de BBDD utilizado[1]. Por defecto se utiliza la clase correspondiente a MYSQL.
- **Jpa.database.create**: especifica si la creación de la BBDD mediante jpa se debe crear de un modo particular. Su valor debe ser MYSQL para las BBDD MYSQL o debe estar vacío para el resto de las BBDD.
- **Jpa.showSql**: especifica si se quieren mostrar en el log las consultas SQL realizadas. Los valores aceptados son true o false.
- **Jdbc.driverClassName**: es la clase de Java que cumple con las especificaciones de JDBC para conectar con la BBDD utilizada. Por defecto se utiliza la clase suministrada por MYSQL[2].
- **Jdbc.url**: es la URL de conexión con la BBDD. Esta cadena de conexión corresponde al formato jdbc:<BBDD>://<SERVIDOR>:<PUERTO>/EDITRANGT. Donde jdbc:<BBDD> indica el protocolo de comunicación, <SERVIDOR> es la IP o el nombre del servidor donde está instalada la BBDD que escucha por el puerto de conexión <PUERTO>. EDITRANGT es el esquema de la BBDD al que nos conectamos.
- **Jdbc.username**: es el usuario de la BBDD que utiliza JDBC para conectar con la BBDD. Su valor por defecto está encriptado y corresponde a la encriptación de la clave **editrangt**[3] con la password de encriptación **secret**.
- **Jdbc.password**: es el password que corresponde al usuario de la BBDD. Su valor por defecto está encriptado y corresponde a la encriptación de la clave **editrangt**[3] con la password de encriptación **secret**.
- **Planificador.host**: es la IP o el nombre del host donde se encuentra el registro de Java RMI que permite la comunicación entre EDITRAN/GT y EDITRAN/GT Web.
- **Planificador.port**: es el puerto de escucha del registro de Java RMI que permite la comunicación entre EDITRAN/GT y EDITRAN/GT Web. Por defecto es el 1099.
- **Planificador.alertas**: indica si que quiere utilizar el servicio de alertas por e-mail. Los valores permitidos son true o false.

[1] JPA (Java Persistence API) es el estándar en Java utilizado para persistir objetos en una base de datos relacional. Los valores aceptados para **jpa.database** y **jpa.databasePlatform** se pueden consultar en:
http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Advanced_JPA_Development/Platform_Specific_Configurations

[2] JDBC (Java Database Connectivity) es el API estándar en Java utilizado para la comunicación de Java con las BBDD. El driver JDBC de comunicación java con la BBDD, debe ser indicado por el usuario. **Por defecto**, la aplicación utiliza el driver de conexión de MYSQL5 *com.mysql.jdbc.Driver*.
En caso de utilizar una BBDD diferente de MYSQL5, como por ejemplo ORACLE 11g, se debe copiar el conector java que suministre el proveedor de BBDD (ORACLE) en el directorio ***/EditranGT/lib***.

[3] Si se quiere cambiar el usuario y la password con el que se accede a la base de datos, al menos el valor de la password debería estar encriptada, y debe tener el formato: ENC(<PASSWORD_ENCRIPTADA>), donde <PASSWORD_ENCRIPTADA> será el resultado de encriptar la password en claro con una password de encriptación que debe coincidir con el valor de la variable de entorno ENCRYPTION_PASSWORD. La variable de entorno debe ser borrada al iniciarse la aplicación.
Para generar **nuevas claves encriptadas** utilizar la utilidad jasypt. Véase http://www.jasypt.org/cli.html).
Ejemplo de uso en jasypt 1.9:
encrypt input=editrangt password=secret algorithm=PBEWITHMD5ANDTRIPLEDES

3.1.2. Conector Java

Para que EDITRAN/GT se pueda conectar con el servidor de BBDD mediante Java, la aplicación web debe poder acceder al conector Java de la BBDD. Por defecto, la BBDD es

MYSQL5.6, y se distribuye su conector con la instalación de EDITRAN/GT, en **EDITRANGT_DIR/inst/bbdd/mysql5**.

> En el caso de usar otra BBDD deberá utilizar un conector Java para conectar con su base de datos, dependiendo cuál sea ésta.

Para implementar este conector debe copiarse en el directorio **EDITRANGT_DIR/lib** donde esté instalado el gestor de transmisiones de EDITRAN/GT.

3.1.3. Script de instalación de BBDD

El script **EDITRANGT_DIR/util/instalar_BBDD_EDITRANGT** está diseñado para crear un usuario en la BBDD con los permisos necesarios (por defecto con nombre **editrangt** y password **editrangt**), el esquema **EDITRANGT**, las tablas y datos iniciales para su funcionamiento.

En este script se deben modificar adecuadamente los paths de EDITRAN (variable **PWD**) y EDITRAN/GT (variables **LIB** y **SCRIPT_EDITRANGT**), y también el usuario (**--user**) y la password (**--password**) con permisos de administración de la BBDD de MySQL.

> En el caso de usar una BBDD diferente de MySQL, hay que tener en cuenta que el script de instalación de BBDD ejecuta por defecto una operación propia de MYSQL que no es compatible con otras BBDDs:
>
> ```
> mysql --user=root --password=root < %SCRIPT_EDITRANGT% --verbose
> ```
>
> Para conseguir el mismo resultado que esta operación, en el caso de que la BBDD sea diferente a **MYSQL5** se deberán realizar los siguientes pasos utilizando sus herramientas, omitiendo el comando **mysql** del script **instalar_BBDD_EDITRANGT**:
>
> - Creación en la BBDD del esquema **EDITRANGT**.
> - Creación en la BBDD del usuario editrangt y contraseña editrangt.
> - Otorgar al usuario editrangt permisos de definición de datos sobre la BBDD de EDITRAN/GT y permisos para realizar operaciones CRUD sobre las tablas (SELECT, INSERT, DELETE, UPDATE).

Opcionalmente, se pueden editar otros valores de configuración de EDITRAN/GT, como el intervalo de búsqueda de ficheros (**INT_BUSQ**), el intervalo de actualización de las sesiones de EDITRAN (**INT_ACTU**) y el directorio raíz de copia de los ficheros emitidos (**DIR_EMIT**).

Finalmente, se debe **ejecutar** el script **instalar_BBDD_EDITRANGT** para realizar la configuración inicial de la BBDD.

3.2. Configuración para EDITRAN/GT Web

La aplicación web debe estar configurada para conectarse con la misma BBDD que el gestor de transmisiones de EDITRAN/GT realizando los siguientes pasos

3.2.1. Propiedades de EDITRAN/GT Web

Para realizar los cambios necesarios en la configuración de BBDD se procede de la misma manera que para el gestor de transmisiones **EDITRAN/GT,** como se explica en el punto **3.1.1**. El fichero a modificar Debemos editar la información que hay dentro del archivo **EDITRANGTconf.properties** en el directorio EDITRANGTWEB_DIR/conf.

3.2.2. Conector Java

Para que el servidor web se pueda conectar con el servidor de BBDD mediante Java, la aplicación web debe poder acceder al conector Java del proveedor de BBDD. Por defecto, la BBDD es **MYSQL5.6**, y se distribuye su conector con la instalación de EDITRAN/GT, en EDITRANGTWEB_DIR/inst/bbdd/mysql5.

> En el caso de usar otra BBDD, deberá utilizar un conector Java para conectar con su base de datos, dependiendo cuál sea ésta.

En el caso de Tomcat 6 hay que copiar el conector en el directorio ***/lib*** de la instalación. Para otros servidores Web se debe consultar su manual.

4. EDITRAN/GT

La aplicación EDITRAN/GT es un gestor de transmisiones para EDITRAN. Esta aplicación debe ser instalada en un servidor que ya tenga instalado EDITRAN.

4.1. Instalación

Para instalar la aplicación, debemos extraer la carpeta **/EDItranGT** contenida en el archivo de distribución **zip** en los sistemas Windows o **tar.gz** en los sistemas UNIX, en la carpeta raíz donde esté instalado **EDITRAN**. Puede ser instalado en otro directorio, pero requiere modificar los scripts para indicar el directorio de EDITRAN y el de EDITRAN/GT, así que por defecto es recomendable instalarlo dentro de la carpeta raíz de **EDITRAN**.

4.2. Preparación de los scripts

Tanto para Windows (.bat) como para Linux/Unix (.sh) debemos preparar los scripts de la aplicación, que están situados en **EDITRANGT_DIR** y **EDITRANGT_DIR/util.**

El valor de las variables debe ser:

- **PWD**: Directorio de instalación de EDITRAN.
- **LIB**: Directorio de instalación de EDITRAN/GT.
- **QUARTZ**: Directorio de instalación de EDITRAN/GT (sólo para el script start_EDItranGT).
- **-DEDITRANGT_DIR**: Directorio de instalación de EDITRAN/GT.

Es necesario definir la contraseña de cifrado usada para las propiedades **jdbc.username** y **jdbc.password** del fichero **conf/EDITRANGTconf.properties**, siendo su valor por defecto **secret**. Si se quiere modificar esta contraseña deben cifrarse de nuevo las dos propiedades cifradas del fichero **conf/EDITRANGTconf.properties** (punto **3.1.1**).

Si el script se ejecuta manualmente se solicita al usuario introducir este valor. En el caso de ejecutar el script de manera automática (por ejemplo como si fuera un servicio) se debe modificar **start_EDItranGT** para que se defina con este valor la variable **ENCRYPTION_PASSWORD**.

4.3. Iniciar buscador de EDITRAN/GT

Para arrancar el buscador de EDITRAN/GT debemos ejecutar el script **start_EDItranGT** que ha sido previamente configurado. Este script se encarga de iniciar el buscador para permitir a EDITRAN enviar o recibir archivos de forma automática tal como se haya configurado en la Base de Datos.

El buscador de EDITRAN/GT no es el encargado de configurar las presentaciones de EDITRAN (deben realizarse mediante la interfaz gráfica de EDITRAN) sino de la comunicación con él. En un funcionamiento correcto, la ejecución de este programa no termina nunca y no debe acabar, si queremos detenerlo debe cancelarse la ejecución del proceso.

En Windows también se puede instalar EDITRAN/GT como servicio como se explica en el apartado **6.1. Servicio EDITRAN/GT (Windows)**.

4.4. Configuración opcional

4.4.1. Servicio de alertas por e-mail

Cuando el gestor de transmisiones se configura para utilizar el servicio de alertas por e-mail, (mediante la propiedad **planificador.alertas=true** del archivo EDITRANGTconf.properties) el buscador de presentaciones envía al usuario de destino e-mails de alerta cuando las presentaciones se quedan en un estado de error sin reintentos o cuando se produzca alguna situación que pueda provocar que las presentaciones no se transmitan.

Para configurar el servicio de envío de alertas por e-mail, se debe modificar el archivo **EDITRANGT_DIR/conf/servemail.xml**.

Contenido del archivo por defecto:

```
dirEmailDestino=admineditrangt@company.es
dirEmailOrigen=aplicacioneditrangt@company.es
servidorCorreo.host=smtp.company.es
servidorCorreo.puerto=25
servidorCorreo.usuario=aplicacioneditrangt
servidorCorreo.password=aplicacioneditrangt
```

El valor de la propiedad **servidorCorreo.password** se debe cifrar por seguridad mediante el mismo método que se especifica en el punto **3.1.1**. El servicio de alertas es operativo con servidores de correo SMTP que utilizan autenticación mediante usuario y password.

4.4.2. Mensajes de log (log4j)

Se puede cambiar la configuración de los mensajes de log que por defecto usa la aplicación modificando el script **start_EDItranGT** y añadiendo el argumento a la VM **-Dlog4j.configuration** con valor del path al archivo de configuración de log4j seleccionado (por defecto, el archivo /EDItranGT/conf/log4j.properties).

El cambio en los mensajes de log también es aplicable a los scripts de utilidades **migracion_EDItranGT** e **instalar_BBDD_EDItranGT**.

4.4.3. Zona horaria

En el caso de que el buscador, la Base de Datos o el Servidor Web estén instalados en distintas zonas horarias se debe especificar la variable **TZ** del sistema, que indica la zona horaria utilizada dependiendo de la ubicación en la que debemos trabajar.
Por ejemplo, para una instalación en España se debe ejecutar:
export TZ="Europe/Madrid"
Para más información sobre zonas horarias consultar http://www.iana.org/time-zones.

5. Aplicación Web de EDITRAN/GT

La aplicación web se utiliza para consultar y configurar las presentaciones que el gestor de transmisiones se encarga de planificar en EDITRAN.

5.1. Instalación

La aplicación web está contenida en el directorio base **/EDItranGTWeb** una vez se ha descomprimido el archivo de distribución **zip** o **tar**. El archivo contenedor de la aplicación web es **EDItranGT.war**. La instalación de la aplicación variará según sea el servidor de aplicaciones en que vayamos a desplegarla.

Para **Tomcat 6** (servidor Web por defecto) se debe copiar el archivo **EDItranGT.war** en el directorio **%TOMCAT_HOME%/webapps**. Una vez copiado el archivo, y realizada la configuración descrita en los puntos siguientes, al arrancar Tomcat se despliega automáticamente la aplicación web y estará lista para acceder a ella.

5.2. Usuarios de EDITRAN/GT Web

Los usuarios de la aplicación web iniciales se crean automáticamente al ejecutar el instalador de la BBDD en el punto **3.1.3**, y corresponden con el usuario de administración **admin** y el usuario de consulta **editrangt**. En la configuración inicial las contraseñas son iguales al nombre de los usuarios **(admin/admin y editrangt/editrangt)**

Las contraseñas se guardan en la BBDD cifradas por motivos de seguridad con el algoritmo SHA y se recomienda cambiar las contraseñas de los usuarios la primera vez que accedan al sistema. Siempre debe existir al menos un usuario administrador en el sistema ya que de otro modo, no habría forma de crear nuevos usuarios o de administrar la aplicación desde la misma.

5.3. Servidor Web

Para configurar el servidor web se deben realizar las siguientes tareas:

- ❑ Configurar la variable de la máquina virtual de Java EDITRANGTWEB_DIR
- ❑ Configurar la variable de entorno del Servidor Web ENCRYPTION_PASSWORD
- ❑ Habilitar la comunicación segura mediante HTTPS.
- ❑ Añadir las librerías necesarias en el servidor

5.3.1. Variable EDITRANGTWEB_DIR

Se debe especificar en las variables de la máquina virtual de Java que arranca el servidor web la variable **EDITRANGTWEB_DIR** con el valor del directorio raíz de instalación de la aplicación web.

Ejemplo: -DEDITRANGTWEB_DIR="C:\EDITRAN\EDItranGTWeb"

En el caso de Tomcat 6 debemos modificar el script que establece el entorno del Servidor (**setenv.bat** o **setenv.sh**) añadiendo este valor a la variable de entorno del servidor Web **JAVA_OPTS**, que controla las opciones de Java:

- ❑ **Sistemas Windows**

```
set JAVA_OPTS=-DEDITRANGTWEB_DIR="C:\EDITRAN\EDItranGTWeb"
```

- ❑ **Sistemas Unix**

```
JAVA_OPTS=-DEDITRANGTWEB_DIR="/opt/editran/EDItranGTWeb"
```

5.3.2. Variable ENCRYPTION_PASSWORD

Se debe especificar en las variables de entorno del servidor web la variable **ENCRYPTION_PASSWORD** con el valor de la contraseña de cifrado utilizado para cifrar las propiedades de configuración. El valor por defecto es **secret**.

En el caso de Tomcat 6 debemos modificar el script que establece el entorno del Servidor (**setenv.bat** o **setenv.sh**) añadiendo esta variable:

- **Sistemas Windows**
  - En el caso de querer ejecutar el servidor Web automáticamente:

```
set ENCRYPTION_PASSWORD=secret
```

  - Si se arranca de forma manual se recomienda solicitar la contraseña de forma segura:

```
set /P ENCRYPTION_PASSWORD = Introduzca la contraseña de cifrado:
```

- **Sistemas Unix**
  - En el caso de querer ejecutar el servidor Web automáticamente:

```
ENCRYPTION_PASSWORD=secret
export ENCRYPTION_PASSWORD
```

  - Si se arranca de forma manual se recomienda solicitar la contraseña de forma segura:

```
echo Introduzca la contraseña de cifrado:
read ENCRYPTION_PASSWORD
export ENCRYPTION_PASSWORD
```

5.3.3. Habilitar la comunicación segura mediante HTTPS y SSL

La aplicación web requiere que la conexión se realice sobre un canal seguro **HTTPS** con codificación **TLS**, ya que si no se habilita no es posible acceder a la aplicación web. Para ello se necesita un certificado que compruebe la veracidad de la conexión, por lo que el cliente deberá crearse uno propio. En caso de no poder crear uno, EDITRAN/GT proporciona uno genérico en **/EDItranGTWeb/ssl/editrangt_tomcat.keystore** con contraseña **editrangt**.

El certificado que proporciona EDITRAN/GT es genérico por lo que el navegador no lo incluye como un certificado de confianza.

La configuración del servidor Web debe editarse para habilitar conexión segura por HTTPS e indicar la ubicación del fichero de claves.

En **Tomcat 6** se debe copiar el fichero de claves (por defecto el proporcionado por EDITRAN/GT) en el directorio **conf** de la instalación.

También hay que buscar en el archivo de configuración del servidor **conf/server.xml** la configuración del conector para TLS, que comienza por:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"…… />
```

Comentar esta etiqueta XML y añadir la siguiente configuración:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
SSLEnabled="true" maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="conf/editrangt_tomcat.keystore" keystorePass="editrangt" />
```

En caso de no usar el fichero de claves por defecto, cambiar los valores de **keystoreFile** y **keystorePass** por los adecuados.

5.3.4. Añadir las librerías necesarias en el servidor

Para que la aplicación web pueda ejecutarse correctamente es necesario cargar en el arranque la librería el-api-2.2.jar, que se encuentra en el directorio **/EDItranGTWeb/inst/web/tomcat6**.

Para Tomcat 6 se debe copiar esta librería en la carpeta **lib** de la instalación del servidor. Además, se debe invalidar el archivo **el-api.jar** eliminando o moviéndolo de directorio.

NOTA: *Esta operación no es necesaria para otros servidores web J2EE que dispongan de una versión de EL igual o superior a la 2.2.*

5.4. Iniciar la web de EDITRAN/GT

Para arrancar la web de EDITRAN/GT debemos ejecutar el script de arranque del servidor web, que en el caso de **Tomcat 6** es **/bin/startup.bat o startup.sh**.

Una vez arrancado el servidor web podemos acceder a la aplicación desde uno de los navegadores soportados solicitando la dirección por defecto:

https://[direccion servidor web]:8443/EDItranGT

Para detener la ejecución de la web de EDITRAN/GT, debemos ejecutar el script de parada del servidor web, que en el caso de **Tomcat 6** es **/bin/shutdown.bat o shutdown.sh**.

5.5. Configuración opcional

5.5.1. Configurar los mensajes de log (log4j)

Se puede cambiar la configuración de los mensajes de log que por defecto usa la aplicación, añadiendo como parámetro de la máquina virtual de Java el parámetro **-Dlog4j.configuration** con valor del path al archivo de configuración de log4j seleccionado (por defecto, /EDItranGTWeb/conf/log4jWEB.properties).

En el caso de Tomcat 6 podemos modificar el script que establece el entorno del Servidor (**setenv.bat** o **setenv.sh**) añadiendo esta variable:

- **Sistemas Windows**

```
set JAVA_OPTS=%JAVA_OPTS% -Dlog4j.configuration=<PATH_FICHERO_LOG4J>
```

- **Sistemas Unix**

```
JAVA_OPTS=$JAVA_OPTS -Dlog4j.configuration=<PATH_FICHERO_LOG4J>
```

Más información sobre la configuración de las propiedades de log4j en:

https://logging.apache.org/log4j/1.2/manual.html

5.5.2. Zona horaria

En el caso de que el buscador, la Base de Datos o el Servidor Web estén instalados en distintas zonas horarias se debe especificar la variable **TZ** del sistema del mismo modo que en el punto **4.4.3 Zona horaria**.

5.5.3. Comunicación con el buscador (RMI)

Es recomendable ajustar los parámetros de configuración de la comunicación con el buscador mediante RMI (la realización de peticiones a la interfaz de métodos remotos). Más concretamente se recomiendan especificar dos parámetros de la VM:

- -Dsun.rmi.transport.tcp.handshakeTimeout=5000
- -Dsun.rmi.transport.tcp.responseTimeout=15000

El primero es el tiempo máximo en milisegundos de establecimiento de la conexión de la aplicación web cuando realiza una petición al buscador y el segundo es el tiempo máximo en milisegundos que la aplicación web espera una respuesta del buscador.

En el caso de Tomcat 6 podemos modificar el script que establece el entorno del Servidor (**setenv.bat** o **setenv.sh**) añadiendo esta variable:

- **Sistemas Windows**

```
set JAVA_OPTS=%JAVA_OPTS% -Dsun.rmi.transport.tcp.handshakeTimeout=5000 -
Dsun.rmi.transport.tcp.responseTimeout=15000
```

- **Sistemas Unix**

```
JAVA_OPTS=$JAVA_OPTS -Dsun.rmi.transport.tcp.handshakeTimeout=5000 -
Dsun.rmi.transport.tcp.responseTimeout=15000
```

Más información de configuración de las propiedades de RMI en el **Anexo: Configuración RMI.**

6. Utilidades

6.1. Servicio EDITRAN/GT (Windows)

6.1.1. Instalación del servicio

Como requisito para poder instalar el servicio de EDITRAN/GT se depende de tener instalado EDITRAN y la Base de Datos como servicio.

Para instalar el servicio del buscador de EDITRAN/GT debemos entrar en el subdirectorio **Wrapper** que hay en la carpeta **EDItranGT**. Dentro de él veremos 3 scripts:

- **EDItranGT.bat**: Este script nos va a servir para comprobar que el servicio instalado funcionaría correctamente por lo que es recomendable hacer una prueba inicial. En caso de probar este programa, conviene hacerlo durante un par de minutos para comprobar que no saltan ninguno de los timeouts que existen para el servicio.
- **InstallEDItranGT.bat**: Este script instala el programa EDITRAN/GT como servicio de Windows.
- **UninstallEDItranGT.bat**: Este script desinstala el servicio de Windows EDITRAN/GT.

6.1.2. Configuración del servicio

Por defecto, el servicio tiene unos ciertos parámetros ya definidos que pueden ser cambiados. Esta configuración del servicio viene reflejada en el archivo **Wrapper/conf/wrapper.conf**.

En el caso de cambiar la configuración, aconsejamos probar el script **EDItranGT.bat** para asegurar el correcto funcionamiento. Los campos que pueden ser interesantes son:

- **wrapper.ntservice.dependency.1:** Nos indica la dependencia con la Base de Datos y con el servicio de EDITRAN.

En el caso de utilizar una Base de Datos distinta de la usada por defecto debemos indicar el servicio que la controla.

- **wrapper.ntservice.starttype:** Modo en el cual el servicio instalado es iniciado. Se puede escoger entre AUTO_START, DELAY_START o DEMAND_START. Por defecto, el inicio del servicio es AUTO.
- **wrapper.working.dir:** Por defecto la instalación de EDITRAN/GT está instalada dentro de EDITRAN en el directorio **EDItranGT**. En caso de no ser así el valor debe ser el directorio de EDITRAN.
- **set.ENCRYPTION_PASSWORD=secret**: Aquí definimos el valor de la variable de entorno. En caso de cambiar la contraseña de cifrado como en los puntos punto **3.1.1** debe cambiarse también este valor.
- **wrapper.java.additional.1= -DEDITRANGT_DIR=EDItranGT:** La ubicación de la instalación de EDITRAN/GT. Por defecto está instalado dentro de EDITRAN en el directorio **EDItranGT**. En caso de no ser así el valor debe ser el directorio de EDITRAN/GT.

6.2. Migración de la configuración de EDITRAN/GT

En el caso de tener una instalación de EDITRAN/GT de Windows anterior, podemos migrar el archivo de configuración anterior para esta nueva versión. Para ello debemos ejecutar el script **util/migracion_EDItranGT**. Debe estar situado el antiguo archivo **editrangt.cfg** en la carpeta donde esté instalado **EDITRAN** para migrar la configuración a la Base de Datos.

6.3. Borrado de eventos de log

Si es necesario eliminar eventos antiguos para evitar un crecimiento de la BBDD podemos utilizar el script **util/borrar_Eventos_EDItranGT** para el borrado de eventos antiguos de EDITRAN/GT.

El script nos pide una fecha con formato **DD/MM/YYYY** para eliminar de la BBDD los eventos anteriores. Esta herramienta se debe usar con especial precaución ya que los datos borrados no pueden ser recuperados.

7. Anexo: Configuración RMI

Podemos personalizar y optimizar el comportamiento de la comunicación entre la aplicación web y el buscador (realizado vía Java RMI) configurando las propiedades RMI al ejecutar la máquina virtual de Java.

La aplicación web de EDITRAN/GT representa la parte cliente del entorno de ejecución de Java RMI, que realiza llamadas a métodos remotos del buscador de presentaciones, mientras que la aplicación de búsqueda de presentaciones de EDITRAN/GT representa la parte servidora que expone los métodos remotos que puede invocar la aplicación web de GT.

Cada vez que un usuario realiza una solicitud de una página en la aplicación web, se hace una llamada al buscador para recuperar su estado, y en caso de producirse un error de comunicación, se mostrará en la parte superior de la aplicación WEB, un mensaje de alerta de **ERROR COMUNICACIÓN**.

En caso de producirse un error de comunicación se debe revisar el log para analizar cuál ha sido su tipo y su causa. Si la excepción lanzada es del tipo java.rmi.RemoteException se debe revisar la configuración de RMI, es decir, que las propiedades de EDItranGT.properties planificador.host y planificador.port sean correctas y que el entorno de ejecución de Java RMI esté operativo.

Dada la naturaleza interactiva de la aplicación web, y teniendo en cuenta la experiencia de usuario, se debe especificar la propiedad **sun.rmi.transport.tcp.handshakeTimeout** en la aplicación web con un valor distinto de 0, ya que 0 indica espera infinita, y si esta propiedad tiene ese valor, la aplicación web se quedará bloqueada cada vez que se solicite una carga de una página y no se pueda establecer una comunicación RMI con el buscador.

Por lo tanto, se recomienda establecer este valor a una espera máxima de **5 segundos**, que será el tiempo mínimo que tardará una página web en cargarse cuando existan problemas de comunicación con el buscador.

Por otro lado, y en el mismo sentido con respecto a la experiencia de usuario, se recomienda especificar la propiedad **sun.rmi.transport.tcp.responseTimeout** en la aplicación web, con un valor distinto de 0, ya que 0 indica espera infinita, y si esta propiedad tiene ese valor, la aplicación web se quedará bloqueada cada vez que se solicite una operación al buscador que no se resuelva.

Por esto, se recomienda establecer este valor a una espera máxima de **15 segundos**, que será el tiempo máximo que tardará la aplicación web en recuperar el control de la navegación cuando solicite una operación al buscador que tarda demasiado en resolverse.

Para hacer uso de las propiedades RMI, se deberá modificar el script de arranque del buscador de presentaciones (start_EDItranGT.bat o start_EDItranGT.sh) y/o el de arranque del servidor de aplicaciones web J2EE (catalina.bat o catalina.sh para Tomcat), y especificar a continuación del comando de ejecución de java el nombre de la propiedad RMI y su valor (si es una propiedad con valor), o solo el nombre de la propiedad (si es una propiedad sin valor) como por ejemplo:

java -Djava.rmi.activation.port=1100 -Djava sun.rmi.dgc.logLevel ...

7.1. Propiedades java.rmi

Documentación original disponible en: http://docs.oracle.com/javase/6/docs/technotes/guides/rmi/javarmiproperties.html

Propiedades útiles para utilizar en máquinas virtuales de Java que exportan objetos remotos (apropiadas para el buscador de presentaciones)

- **java.rmi.activation.port**. Esta propiedad se usa para establecer el número del Puerto TCP en el cual esta VM debería comunicarse con rmid (por defecto, rmid escucha en el puerto 1098, pero se puede configurar para escuchar en un puerto diferente usando la opción -port en la línea de comandos de rmid). El valor por defecto de esta propiedad es 1098, así que esta propiedad solo necesita ser establecida en las VMs que necesiten comunicarse con una instancia de rmid que se esté ejecutando en un puerto diferente del 1098.
- **java.rmi.dgc.leaseValue**. El valor de esta propiedad representa la duración (en milisegundos) de uso concedida a otras VMs que mantengan referencias remotas a objetos que hayan sido exportados por esta VM. El valor por defecto de esta propiedad es 60000 milisegundos (10 minutos).
- **java.rmi.server.codebase**. Esta propiedad especifica las localizaciones desde las cuales las clases publicadas por esta VM (por ejemplo: clases stub, clases personalizadas que implementen el tipo de retorno declarado de una llamada a un método remoto, o interfaces usadas por un proxy o una clase stub) que pueden ser descargadas. El valor de esta propiedad es una cadena en formato URL o una lista de URLs separadas por comas que serán la localización del código para todas las clases cargadas desde el CLASSPATH de esta VM.
- **java.rmi.server.hostname**. El valor de esta propiedad representa la cadena del nombre del host que debería estar asociada con stubs remotos para objetos remotos creados localmente, con el fin de permitir a los clientes invocar métodos sobre el objeto remoto. El valor por defecto de esta propiedad es la dirección IP del host local, en formato de dobles comillas.
- **java.rmi.server.logCalls**. Si este valor es true, las llamadas entrantes y las excepciones lanzadas de llamadas entrantes serán registradas en System.err. Si se establece esta propiedad a true se facilitará la depuración de los programas RMI. Consultar también sun.rmi.server.exceptionTrace.
- **java.rmi.server.randomIDs**. Si este valor es true, los identificadores de los objetos remotos exportados por esta VM serán generados usando un generador de números aleatorios. El valor por defecto es false.
- **java.rmi.server.useCodebaseOnly**. Si este valor es true, se prohíbe la carga automática de clases excepto para el CLASSPATH local y desde la propiedad java.rmi.server.codebase establecida en esta VM. El uso de esta propiedad evita que las VMs clientes descarguen dinámicamente el código de nuestros codebases.
- **java.rmi.server.useLocalHostname**. Java RMI usa una dirección IP para identificar al host local cuando la propiedad java.rmi.server.hostname no está especificada y no se puede obtener un nombre de dominio calificado completamente para el localhost. Con el objeto de forzar a Java RMI para que use un nombre de dominio calificado completamente por defecto, se debe establecer esta propiedad a true.

Propiedades útiles para establecer en VMs que realizan llamadas a métodos remotos (apropiadas para la aplicación WEB)

- **java.rmi.server.codebase.** Esta propiedad especifica las localizaciones desde las que las clases publicadas por esta VM (por ejemplo, clases personalizadas que implementan unan interfaz que es el tipo de parámetro declarado para una llamada a un método remoto) pueden ser descargadas. El valor de esta propiedad es una cadena en formato URL (o una lista de URLs separadas por espacios) que será la localización del código para todas las clases cargadas desde el CLASSPATH de esta VM.
- **java.rmi.server.disableHttp**. Si este valor es true, se deshabilita el HTTP tunneling, incluso cuando http.proxyHost esté establecido. El valor por defecto es false. Si sabes que tu programa nunca necesitará usar HTTP tunneling, entonces deshabilitando HTTP tunneling, los tiempos de espera para las conexiones fallidas serán más cortos.
- **java.rmi.server.useCodebaseOnly**. Si este valor es true, se prohíbe la carga automática de clases excepto para el CLASSPATH local y desde la propiedad java.rmi.server.codebase establecida en esta VM. EL uso de esta propiedad previene a las VMs clientes de descargar dinámicamente el código ejecutable de otras localizaciones de código.

7.2. Propiedades sun.rmi

Las propiedades de sun.rmi deben utilizarse sabiendo que no forman parte del estándar de las propiedades de las máquinas virtuales de Java y que por lo tanto, su funcionamiento es dependiente de la implementación de la VM.

Documentación original disponible en:

http://docs.oracle.com/javase/6/docs/technotes/guides/rmi/sunrmiproperties.html

Propiedades que son útiles para establecer en VMs que exportan objetos remotos (apropiadas para el buscador de presentaciones)

- **sun.rmi.dgc.ackTimeout (1.4 y posterior).** El valor de esta propiedad representa la cantidad de tiempo (en milisegundos) que el lado servidor del entorno de ejecución de Java RMI referenciará a un objeto remoto (o a una referencia a un objeto remoto) que haya sido retornado por la actual máquina virtual como parte del resultado de una llamada a un método remoto, hasta que reciba una confirmación del cliente de que la referencia remota ha sido recibida y procesada. El valor máximo es Long.MAX_VALUE. El valor por defecto es 300000 (**cinco minutos**).
- **sun.rmi.dgc.checkInterval (1.1 y posterior).** El valor de esta propiedad representa (en milisegundos) con qué frecuencia el entorno de ejecución de Java RMI comprueba si han caducado las referencias DGC (Distributed Garbage Collector). El valor por defecto es la mitad del valor de la propiedad java.rmi.dgc.leaseValue.
- **sun.rmi.dgc.logLevel (1.1 y posterior).** Esta propiedad controla el registro de las llamadas entrantes y salientes relacionadas con la garantía, la renovación y la expiración de las referencias DGC. Envía la salida al log de "dgc".
- **sun.rmi.dgc.server.gcInterval (1.2 y posterior).** Cuando sea necesario asegurar que los objetos remotos inalcanzables sean eliminados y recogidos

por el recolector de basura cada cierto tiempo, el valor de esta propiedad representa el máximo intervalo (en milisegundos) que el entorno de ejecución de Java RMI permitirá entre las recolecciones de basura de la pila local. El valor por defecto es 3600000 milisegundos (**una hora**).

- **sun.rmi.loader.logLevel (1.2 y posterior).** Esta propiedad controla el registro del nombre y la localización de cada clase, cuando el entorno de ejecución de Java RMI intente cargar una clase como resultado de realizar un unmarshalling ya sea de un argumento o de un valor de retorno. Esta propiedad envía la salida al log "loader".
- **sun.rmi.server.exceptionTrace (1.2 y posterior**). Esta propiedad controla la salida de las trazas de la pila de excepciones y errores del lado del servidor que han sido lanzadas por llamadas remotas entrantes. Si el valor es true, se imprimen las trazas de la pila de excepciones. Por defecto (**false**), no se imprimen las trazas de la pila de excepciones.
- **sun.rmi.server.suppressStackTraces (1.4 y posterior).** Si este valor es true, la parte servidora del entorno de ejecución de Java RMI limpiará las trazas de la pila de todas las excepciones lanzadas desde la actual máquina virtual que hayan resultado de las llamadas remotas.
- **sun.rmi.transport.logLevel (1.1 y posterior).** Esta propiedad controla el registro detallado del flujo de información de la capa de transporte. Se envía la salida al log "transport".
- **sun.rmi.transport.tcp.localHostNameTimeOut (1.1.7 y posterior).** El valor de esta propiedad representa el tiempo (en milisegundos) que el entorno de ejecución de Java RMI esperará para obtener un nombre de dominio completamente calificado para el host local. El valor por defecto es 10000 milisegundos (**10 segundos**).
- **sun.rmi.transport.tcp.logLevel (1.1 y posterior).** Esta propiedad indica que se realice un registro detallado de la subcapa de transporte específica TCP. Se envía la salida al log "tcp".
- **sun.rmi.transport.tcp.readTimeout (1.2.2 y posterior).** El valor de esta propiedad representa el tiempo (en milisegundos) usado como tiempo de espera de las conexiones TCP entrantes sin utilizar que usa el entorno de ejecución de Java RMI. El valor se pasa a java.net.Socket.setSoTimeout. Esta propiedad se usa solo para los casos donde un cliente no ha liberado una conexión sin utilizar como debería (ver sun.rmi.transport.connectionTimeout). El valor por defecto es 2*3600*1000 milisegundos (**dos horas**).
- **sun.rmi.transport.tcp.maxConnectionThreads (6 y posterior).** El valor de esta propiedad controla el tamaño máximo del pool de hilos usado para manejar las conexiones entrantes, y por lo tanto establece el límite superior del número de invocaciones de métodos remotos entrantes que pueden ejecutarse concurrentemente. Si se establece esta propiedad a un valor menor se puede mejorar el rendimiento de un servidor de aplicaciones Java RMI bajo condiciones de mucha carga, pero si se establece con un valor demasiado bajo (dependiendo de la naturaleza de los patrones de invocación remota de la aplicación) puede llevar a una situación de bloqueo o de inanición. El valor por defecto es el máximo, Integer.MAX_VALUE, lo que significa que **no hay un límite** establecido.

- **sun.rmi.transport.tcp.threadKeepAliveTime (6 y posterior).** El valor de esta propiedad controla la cantidad de tiempo que un hilo en el pool de hilos usado para manejar conexiones entrantes permanecerá sin utilizar antes de terminar. El valor por defecto es 60000 milisegundos (**un minuto**).

Propiedades útiles para establecer en VMs que realizan llamadas a métodos remotos (apropiadas para la aplicación WEB)

- **sun.rmi.client.logCalls (1.4 y posterior).** Si el valor de esta propiedad es true, el logger sun.rmi.client.call se establecerá con el nivel Level.FINER. Las llamadas remotas son registradas con el nivel Level.FINER, y las excepciones lanzadas desde llamadas remotas son registradas con el nivel Level.FINE.
- **sun.rmi.dgc.cleanInterval (1.1 y posterior).** El valor de esta propiedad representa la máxima duración de tiempo (en milisegundos) que el entorno de ejecución de Java RMI esperará antes de reintentar un fallo de una llamada "clean" al DGC. El valor por defecto es de 180000 milisegundos (**tres minutos**).
- **sun.rmi.dgc.client.gcInterval (1.2 y posterior**). Cuando sea necesario asegurar que las llamadas a limpiar de DGC para las referencias remotas inalcanzables sean realizadas cada cierto tiempo, el valor de esta propiedad representa el intervalo (en milisegundos) que el entorno de ejecución de Java RMI establecerá entre recolecciones de basura del heap local. El valor por defecto es de 3600000 milisegundos (**una hora**).
- **sun.rmi.loader.logLevel (1.2 y posterior).** Esta propiedad controla el registro de cada nombre de clase y de su localización, cada vez que el entorno de ejecución de Java RMI intente cargar una clase como resultado de realizar el unmarshalling ya sea de un argumento o de un valor de retorno. Esta propiedad envía la salida al log "loader".
- **sun.rmi.server.logLevel (1.1 y posterior).** Esta propiedad controla el registro de la información relacionada con llamadas salientes, incluyendo información de reutilización de conexiones. Envía la salida al log "transport".
- **sun.rmi.transport.connectionTimeout (1.1.6 y posterior).** El valor de esta propiedad representa el periodo (en milisegundos) en el que las conexiones de socket pueden permanecer en un estado "unused", antes de que el entorno de ejecución de Java RMI permita a estas conexiones ser liberadas (cerradas). El valor por defecto es 15000 milisegundos (**15 segundos**). Ver también sun.rmi.transport.tcp.readTimeout.
- **sun.rmi.transport.logLevel (1.1 y posterior).** Esta propiedad controla el registro detallado de la actividad de la capa de transporte. Envía la salida al log "transport".
- **sun.rmi.transport.proxy.connectTimeout (1.1 y posterior).** El valor de esta propiedad representa la máxima cantidad de tiempo (en milisegundos) que el entorno de ejecución de Java RMI esperará a un intento de conexión (createSocket) hasta ser completado, antes de intentar contactar con el servidor usando HTTP. Esta propiedad solo se usa cuando la propiedad http.proxyHost sea establecida y java.rmi.server sea false. El valor por defecto es 15000 milisegundos (**15 segundos**).
- **sun.rmi.transport.proxy.eagerHttpFallback (1.4.1 y posterior).** Si este valor es true y la propiedad del sistema java.rmi.server.disableHttp no está establecida, entonces la RMISocketFactory por defecto realizará HTTP

tunneling cuando cualquier java.net.SocketException sea lanzada desde un intento de conexión inicial (directo), de forma opuesta al comportamiento por defecto de realizar HTTP tunneling solo si un intento de conexión inicial lanza java.net.UnknownHostException o java.net.NoRouteToHostException. Esta configuración es útil cuando existen firewalls que deniegan (en lugar de ignorar) los intentos de conexión a puertos no autorizados, con el resultado de obtener java.net.ConnectionExceptions en la RMISocketFactory global por defecto del cliente. Si esta propiedad del sistema no está activada, entonces esas ConnectionExceptions no realizarán comunicación vía HTTP, ya que no son UnknownHostExceptions ni NoRouteToHostExceptions. De todas formas, si esta propiedad del sistema está activada, entonces las ConnectionExceptions causarán HTTP fallback, porque ConnectionException es subclase de SocketException.

- **sun.rmi.transport.proxy.logLevel (1.1 y posterior).** Esta propiedad controla el registro de eventos (createSocket y createServerSocket) cuando se usa la clase por defecto RMISocketFactory. Este tipo de registro es útil para las aplicaciones que utilizan Java RMI sobre HTTP. Los eventos de factorías de sockets personalizadas no son registrados por esta propiedad. Envía algunos mensajes al log "proxy" y otros al log "transport".
- **sun.rmi.transport.tcp.handshakeTimeout (1.4 y posterior).** El valor de esta propiedad representa la cantidad de tiempo (en milisegundos) que la parte cliente del entorno de de ejecución de Java RMI usará como tiempo de espera de lectura de socket en la lectura inicial de los datos de handshake (protocolo de reconocimiento) cuando se establece una nueva conexión JRMP. Esta propiedad es usada para configurar por cuánto tiempo el entorno de ejecución de Java RMI esperará antes de decidir que una conexión TCP aceptada por un servidor remoto no puede ser usada, ya sea porque la entidad que está escuchando en el puerto del host remoto no es un servidor Java RMI, o porque el servidor de algún modo no está funcionando correctamente. El valor máximo es Integer.MAX_VALUE, y un valor 0 indica un tiempo de espera infinito. El valor por defecto es 60000 (**un minuto**).
- **sun.rmi.transport.tcp.responseTimeout (1.4 y posterior).** El valor de esta propiedad representa la cantidad de tiempo (en milisegundos) que la parte cliente de un entorno de ejecución Java RMI usará como tiempo de espera de lectura de socket en una conexión JRMP establecida cuando se leen los datos de respuesta para una invocación a un método remoto. Por lo tanto, esta propiedad puede ser usada para establecer un tiempo de espera de los resultados a invocaciones remotas; si este tiempo de espera expira, la invocación asociada fallará con una java.rmi.RemoteException. La configuración de esta propiedad debe hacerse con la debida consideración, ya que en efecto establece un límite superior de la duración permitida de cualquier invocación remota saliente con éxito. El valor máximo es Integer.MAX_VALUE, y un valor 0 indica un tiempo de espera infinito. El valor por defecto es 0 (**espera infinita**).
- **sun.rmi.transport.tcp.logLevel (1.1 y posterior).** Esta propiedad provee un registro detallado de la subcapa de transporte TCP. Envía la salida al log "tcp".

7.3. Cambio de la librería RMI

Opcionalmente en el script **start_EDItranGT** se puede especificar el valor de la variable **JAR** para que indique el path absoluto a la librería Java que contiene el código de la aplicación del buscador, que será utilizado como valor del argumento de la VM -**Djava.rmi.server.codebase**.

Ejemplo para Windows: set JAR=file:./EDItranGT/lib/EDItranGTBuscador.jar

8. Anexo: Configuración del pool de conexiones de la BBDD

Por defecto, el pool de conexiones de la BBDD se gestiona mediante la librería c3p0. De forma opcional, en caso de querer configurar este pool de conexiones podemos realizar distintas alternativas.

8.1. Configuración de c3p0

Podemos personalizar y optimizar el comportamiento de la comunicación entre las aplicaciones y la base de datos configurando las propiedades del pool de conexiones con la BBDD C3P0. Para ello, deberemos editar los ficheros **datasource.xml** que se encuentran en:

- El directorio **EDITRANGT_DIR/conf** para el gestor de transmisiones EDITRAN/GT.
- El directorio **EDITRANGTWEB_DIR/conf** para el servidor Web de EDITRANGT.

Documentación original en inglés disponible en: http://www.mchange.com/projects/c3p0/#configuration_properties.

En el archivo datasource.xml especificamos las propiedades de C3P0 con el siguiente formato XML:

```
<bean     id="dataSource"     class="com.mchange.v2.c3p0.ComboPooledDataSource"
destroy-method="close" >
      <property name="driverClass" value="${jdbc.driverClassName}"/>
      <property name="jdbcUrl"                value="${jdbc.url}"/>
      <property name="user"                   value="${jdbc.username}"/>
      <property name="password"               value="${jdbc.password}"/>
      <property name="minPoolSize"           value="5"/>
      <property name="maxPoolSize"           value="20"/>
      <property name="maxIdleTime"           value="300"/>
      <property name="maxStatements"           value="50"/>
      <property name="testConnectionOnCheckin"   value="true"/>
      <property name="idleConnectionTestPeriod"  value="3000"/>
  </bean>
```

A continuación se describen las propiedades existentes de C3P0 (que consideramos que pueden ser interesantes y que C3P0 no desaconseja modificar sus valores por defecto).

- **acquireIncrement.** Por defecto: **3**. Determina cuantas conexiones de una vez c3p0 intentará adquirir cuando el pool haya utilizado todas las conexiones.
- **acquireRetryAttempts.** Por defecto: **30**. Define cuantas veces c3p0 intentará adquirir una nueva conexión de la base de datos antes de abandonar. Si este valor es menor o igual a 0, c3p0 intentará conseguir una conexión indefinidamente.
- **acquireRetryDelay.** Por defecto: **1000**. Milisegundos, el tiempo que c3p0 esperará entre reintentos de adquisición.
- **checkoutTimeout.** Por defecto: **0**. El número de milisegundos que un cliente esperando a adquirir una conexión esperará cuando el pool esté agotado. Cero significa una espera indefinida. Si se establece un valor positivo y se supera el tiempo indicado se lanzará una excepción del tipo SQLException.
- **driverClass.** Por defecto: **null**. El nombre de la clase totalmente calificado de la JDBC driverClass que debe conseguir las conexiones.
- **idleConnectionTestPeriod.** Por defecto: **0**. Si es un número mayor que 0, c3p0 comprobará todas las conexiones del pool no usadas, y no verificadas, cada vez que pase este número de segundos.

- **initialPoolSize.** Por defecto: **3**. El número de conexiones que el pool intentará adquirir en su arranque. Debería estar entre minPoolSize y maxPoolSize.
- **jdbcUrl.** Por defecto: **null**. La URL JDBC de la base de datos de la que se pueden y se deben adquirir las conexiones.
- **maxConnectionAge.** Por defecto: **0**. Segundos, el tiempo de vida de una conexión. Una conexión con una duración mayor que maxConnectionAge será destruida y eliminada del pool. Cero significa que no hay una duración máxima para el tiempo de vida de las conexiones.
- **maxIdleTime.** Por defecto: **0**. Segundos que una conexión puede permanecer en el pool y sin ser utilizada antes de ser descartada. Cero significa que las conexiones nunca se sacan del pool.
- **maxIdleTimeExcessConnections.** Por defecto: **0**. El número de segundos que las conexiones que excedan el minPoolSize tienen permitido permanecer inutilizadas en el pool antes de ser sacadas. Esta propiedad es útil para las aplicaciones que desean minimizar de una forma agresiva el número de conexiones abiertas, disminuyendo el tamaño del pool hacia el minPoolSize, si a continuación de un pico, el nivel de carga disminuye y las conexiones adquiridas no se necesitan más. Si se establece maxIdleTime, maxIdleTimeExcessConnections debería ser más pequeño para que tenga efecto. Cero significa que las conexiones en exceso no se sacan del pool.
- **maxPoolSize.** Por defecto: **15**. El número máximo de conexiones que controlará el pool a la vez.
- **maxStatements.** Por defecto: **0**. El tamaño de la cache global de PreparedStatement de c3p0. Si maxStatements y maxStatementsPerConnection son cero, el cacheo de sentencias no estará activado. Si maxStatements es cero pero maxStatementsPerConnection no es cero, el cacheo de sentencias estará activado, pero no existirá un límite global, solo un máximo por conexión. maxStatements controla el número total de sentencias cacheadas para todas las conexiones. Si se establece, debería ser un número bastante grande, ya que cada conexión del pool requiere su propio grupo de sentencias cacheadas. Un valor apropiado, sería considerar cuantas PreparedStatements distintas son usadas frecuentemente en la aplicación, y multiplicar ese número por el maxPoolSize.
- **maxStatementsPerConnection.** Por defecto: **0**. El número de PreparedStatements que c3p0 cachea para una conexión del pool. Si maxStatements y maxStatementsPerConnection son cero, el cacheo de sentencias no estará activado. Si maxStatementsPerConnection es cero pero maxStatements es distinto de cero, el cacheo de sentencias estará activado, y el límite global forzado, pero no habrá límite establecido en cuanto al número de sentencias preparadas para una conexión. Si se establece, maxStatementsPerConnection debería establecerse con un valor que sea el número de las distintas PreparedStatements que son usadas frecuentemente en la aplicación, más dos o tres extras para que las sentencias infrecuentes no fuercen que las sentencias cacheadas más comunes sean sacrificadas.
- **minPoolSize.** Por defecto: **3**. El número mínimo de conexiones que el pool mantendrá en cualquier momento.

- **numHelperThreads.** Por defecto: **3**. C3p0 es muy asíncrono. Las operaciones JDBC más lentas son normalmente realizadas por hilos auxiliares no bloqueantes. La propagación de estas operaciones sobre múltiples hilos puede mejorar sensiblemente el rendimiento permitiendo que múltiples operaciones se realicen simultáneamente.
- **Password.** Por defecto: **null**. Para las aplicaciones que usen ComboPooledDataSource o cualquier Datasource no orientado a pool implementado por c3p0, define el password que será usado por el método por defecto getConnection() del Datasource.
- **privilegeSpawnedThreads.** Por defecto: **false**. Si es true, los hilos creados por c3p0 tendrán el java.security.AccessControlContext asociado a las clases de la librería c3p0. Por defecto, los hilos creados por c3p0 (hilos auxiliares, hilos java.util.Timer) heredan su AccessControlContext del hilo cliente que inicializa el pool. Esto a veces puede ser un problema, especialmente en servidores de aplicaciones que soportan el redespliegue en caliente de las aplicaciones clientes. Si los hilos de c3p0 mantienen una referencia a AccessControlContext del primer cliente que los utiliza, puede ser imposible recolectar la basura para un ClassLoader asociado con el cliente cuando este replegado en una VM en ejecución. Además, es posible que los hilos del cliente puedan carecer de los permisos suficientes para realizar las operaciones que requiere c3p0. Establecer esta propiedad a true puede resolver estas situaciones.
- **propertyCycle.** Por defecto: **0**. Máximo tiempo en segundos antes de que las restricciones de configuración del usuario tengan efecto. Determina con qué frecuencia maxConnectionAge, maxIdleTime, maxIdleTimeExcessConnections, unreturnedConnectionTimeout son comprobadas. C3P0 periódicamente comprueba la duración de las conexiones para ver si han expirado. Este parámetro determina el periodo. Cero significa automático: un periodo correcto determinado por c3p0.
- **testConnectionOnCheckin.** Por defecto: **false**. Si es true, una operación se realizará de forma asíncrona con cada registro de una nueva conexión para comprobar que es válida. Se puede usar junto con idleConnectionTestPeriod para una comprobación de las conexiones de confianza y asíncrona.
- **user.** Por defecto: **null.** Para las aplicaciones que usen ComboPooledDataSource o cualquier Datasource no orientado a pool implementado por c3p0, define el usuario que será usado por el método por defecto getConnection() del Datasource.

8.2. Cambio de la librería del pool de conexiones

En caso de querer cambiar la librería del pool de conexiones que se usa por defecto, se debe cargar la nueva librería jar. Para ello se debe copiar en:

- ❑ El directorio **EDITRANGT_DIR/lib** para el gestor de transmisiones EDITRAN/GT.
- ❑ El directorio **/lib** de la instalación de Tomcat 6 en el caso de usar el servidor web por defecto. Si se usa otro servidor web debe moverse a un directorio en el que se carguen las librerías al iniciar el servidor.

9. Anexo: Configuración QUARTZ

Podemos personalizar y optimizar el comportamiento del planificador, configurando las propiedades del archivo de configuración de QUARTZ.

En el archivo **conf/quartz.properties** especificamos las propiedades de QUARTZ con el siguiente formato:

```
org.quartz.scheduler.instanceName: DefaultQuartzScheduler
org.quartz.scheduler.rmi.export: false
org.quartz.scheduler.rmi.proxy: false
org.quartz.scheduler.wrapJobExecutionInUserTransaction: false
org.quartz.threadPool.class: org.quartz.simpl.SimpleThreadPool
org.quartz.threadPool.threadCount: 10
org.quartz.threadPool.threadPriority: 5
org.quartz.threadPool.threadsInheritContextClassLoaderOfInitializingThread:
true
org.quartz.jobStore.misfireThreshold: 60000
org.quartz.jobStore.class: org.quartz.simpl.RAMJobStore
```

A continuación se describen las propiedades existentes de QUARTZ (que consideramos que pueden ser interesantes y que QUARTZ no desaconseja modificar sus valores por defecto).

- **org.quartz.scheduler.instanceName.** Puede ser una cadena de texto, y su valor no tiene significado para el planificador en sí mismo - pero sirve para que el código cliente pueda distinguir entre varios planificadores cuando se utilizan varias instancias en un mismo programa.
- **org.quartz.scheduler.makeSchedulerThreadDaemon.** Un valor true o false que especifica si el hilo principal del planificador debería ser un hilo de tipo demonio o no.
- **org.quartz.scheduler.threadsInheritContextClassLoaderOfInitializer.** Un valor true o false que especifica si los hilos creados por Quartz heredarán el contexto ClassLoader del hilo inicializador (el hilo que inicializa la instancia de Quartz). Esto afectará al hilo principal de planificación de Quartz y a los hilos de SimpleThreadPool (si está siendo usado). Establecer este valor a true puede ayudar con problemas de carga de clases, y otros relacionados con usar Quartz en un servidor de aplicaciones.
- **org.quartz.scheduler.batchTriggerAcquisitionMaxCount.** Es el número máximo de disparadores que un planificador tiene permitido adquirir (para disparar) a la vez. Por defecto su valor es 1. Cuanto mayor sea el número, más eficiente será al disparar (en situaciones donde hay muchos disparadores que necesitan ser disparados al mismo tiempo) - pero con el coste de una posible carga no balanceada entre nodos de un cluster.
- **org.quartz.scheduler.batchTriggerAcquisitionFireAheadTimeWindow.** La ventana de tiempo en milisegundos en la que un disparador tiene permitido para adquirir y disparar a partir de su hora de disparo. Por defecto es 0. Cuanto mayor sea el número, más procesos por lotes de adquisición de disparadores serán capaces de seleccionar y disparar más de 1 disparador de una vez - con el coste de que la planificación de disparadores no sea precisa (los disparadores pueden disparar antes de esta cantidad). Esto puede ser útil (para beneficio del rendimiento) en situaciones donde el planificador tiene un gran número de disparadores que necesitan ser disparados en el mismo momento (o casi).

- **org.quartz.scheduler.wrapJobExecutionInUserTransaction.** Se debe establecer a true si se desea que Quartz comience una UserTransaction antes de llamar a execute de una tarea. La transacción se confirmará después de que se complete el método execute de la tarea y después de que el JobDataMap se actualice (si es un StatefulJob). El valor por defecto es false.
- **org.quartz.threadPool.class.** Es el nombre de la implementación del ThreadPool que se desea usar. El pool de hilos que incluye Quartz es org.quartz.simpl.SimpleThreadPool, y este contiene un pool de hilos de tamaño fijo que existen durante el tiempo de vida del planificador.
- **org.quartz.threadPool.threadCount.** Puede ser cualquier número positivo, aunque los valores entre 1 y 100 son los más útiles. Es el número de hilos disponibles durante la ejecución de tareas concurrentes. Si solo tienes unas pocas tareas que se disparan unas pocas veces al día, entonces1 hilos es suficiente. Si tienes decenas de miles de tareas, que se disparan varias veces por minuto, entonces se necesitan entre 50 y 100 (este valor dependerá de la naturaleza del trabajo que realizan las tareas, y de los recursos del sistema).
- **org.quartz.threadPool.threadPriority.** Puede ser cualquier número positivo entre 1 y 10. El valor por defecto es 5.
- **org.quartz.threadPool.makeThreadsDaemons.** Se puede establecer a true para que los hilos del pool sean creados como hilos demonios. Su valor por defecto es false.
- **org.quartz.threadPool.threadsInheritGroupOfInitializingThread.** Puede ser true o false y por defecto es true.
- **org.quartz.threadPool.threadsInheritContextClassLoaderOfInitializingThread**. Puede ser true o false y por defecto es false.
- **org.quartz.scheduler.rmi.export.** Si se quiere exportar el planificador de Quartz vía RMI como servidor entonces se debe establecer esta propiedad a true.
- **org.quartz.scheduler.rmi.proxy.** Si se quiere conectar a un planificador remoto, entonces debe establecerse esta propiedad a true. Además, en este caso, se debe especificar un host y un puerto para el proceso de registro de RMI – el cual típicamente es localhost en el puerto 1099. No tiene sentido especificar un valor true a la vez para las propiedades org.quartz.scheduler.rmi.export y org.quartz.scheduler.rmi.proxy en el mismo archivo de configuración – en este caso, se ignorará la opción export. Un valor false para las opciones export y proxy es válido, si no se utiliza Quartz vía RMI.
- **org.quartz.jobStore.class.** RAMJobStore se utiliza para almacenar la información de planificación (tareas, disparadores y calendarios) en memoria. RAMJobStore es rápida y ligera, pero toda la información de planificación se pierde cuando el proceso termina.
- **org.quartz.jobStore.misfireThreshold.** El número de milisegundos que el planificador permitirá a un disparador pasarse de la hora de su siguiente disparo, antes de que se considere un disparo fallido. El valor por defecto es 60000 (60 segundos).

minsait

An Indra company

Contacto
editran@indra.es
T +34 91 480 80 80

Avda. de Bruselas 35
28108 Alcobendas,
Madrid, España
T +34 91 480 50 00
F +34 91 480 50 80

www.minsait.com