Cybermen - SPC

Public Edition, version 2.5.

Guía de desarrollo

Fecha de generación: 02/01/2015

Autor: Eduardo O. Frigerio.

Contenido

Introducción

Cybermen - SPC (Services Platform in Cluster) es una plataforma de servicios y clúster de procesamiento distribuido, orientado principalmente al desarrollo de aplicaciones de supervisión, control y adquisición de datos (SCADA) y control distribuido (DCS).

Cybermen de por sí, no hace a las aplicaciones SCADA o DCS, pero constituye una base para el desarrollo de este tipo de soluciones, basándose para las mismas en una arquitectura de servicios colaborativos y operando en un clúster de procesamiento.

Conocimientos previos.

Esta guía de desarrollo, se piensa para programadores con conocimientos y experiencia en el desarrollo de soluciones Java SE y Java EE, por lo que no se darán explicaciones de conceptos, herramientas y librerías de estas plataformas, al igual que de herramientas, conceptos y patrones básicos de diseño y desarrollo de aplicaciones Java SE y Java EE.

Se requiere además, conocimientos previos en SOA (Arquitectura Orientada a Servicios) no desde la perspectiva de su implementación desde una herramienta, sino como patrón de diseño de soluciones.

Se recomienda además leer el artículo Workers Control, system pattern publicado en javaHispano para entender conceptualmente la operación de un director de proceso en un clúster variable.

Es de lectura previa recomendable también la Guía de Instalación de Cybermen.

Paquetes y dependencias.

En tiempo de desarrollo un proyecto de Cybermen - SPC requiere de las siguientes librerías.

- jgroups-3.2.12.Final.jar
- CyberLinker.jar
- form4G.jar

Conceptos y características básicas.

Antes de empezar es necesario fijar algunos conceptos y características básicas que hacen a Cybermen como plataforma de servicios al igual que como clúster de procesamiento.

Arquitectura Orientada a Servicios (SOA).

Cybermen es un contenedor de servicios distribuidos, esto significa que cualquier desarrollo para Cybermen, debe plantearse como la construcción de uno o más servicios (sean estos independientes o interconectados), tanto orientados al procesamiento de reglas de negocio, o para ser consumidos por aplicaciones externas al clúster de procesamiento, por medio del cliente de Cybermen.

Manejo de fallas.

En los sistemas de control y automatización industrial se parte del principio de que cualquier desviación posible del modelo normal de comportamiento especificado dentro del requerimiento funcional (falla en la capa de comunicaciones, en los almacenamientos, fallas de negación de servicios, etc., etc., etc., etc. ..), debe tener tratamiento en el sistema tal cual fuese parte del modelo de negocios.

En esta concepción del diseño no existen Exception o errores, si no fallas, al producirse eventos o fracturas fuera de diseño..

Cybermen como plataforma (al igual que sus servicios básicos) se diseñan conforme a esta filosofía.

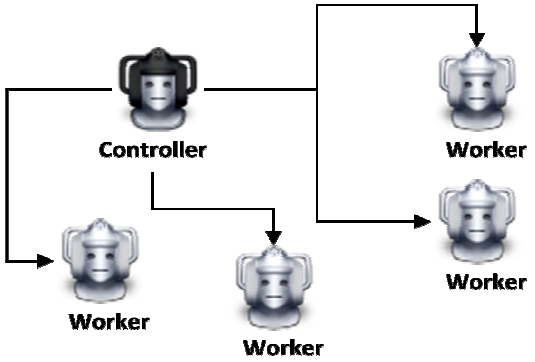
Pero la misma debe ser adoptada por el resto de los desarrollos a ser desplegados en la plataforma para que la misma tenga sentido.

No existen clústeres de un solo nodo.

Si un nodo de procesamiento se ve aislado del resto, Cybermen entiende que el mismo está sufriendo algún problema de conectividad y por ende notificara a los servicios distribuidos, del evento y la duración del mismo, para que estos decidan cual es la mejor acción a realizar.

Workers Control, system pattern.

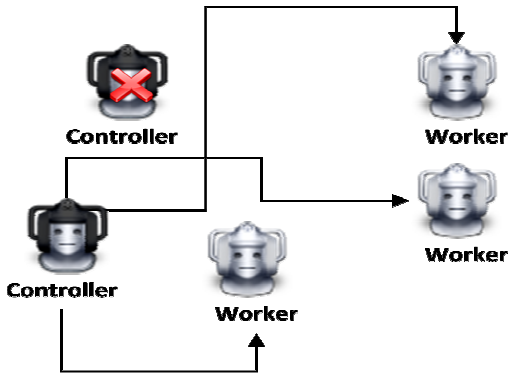
Como mencionado en el artículo de javaHispano, Cybermen (para su definición de servicios distribuidos) adopta de los sistemas SCADA y DCS el concepto de director de procesos, y readapta el mismo, para su operación dentro de un clúster de procesamiento distribuido y de longitud variable.



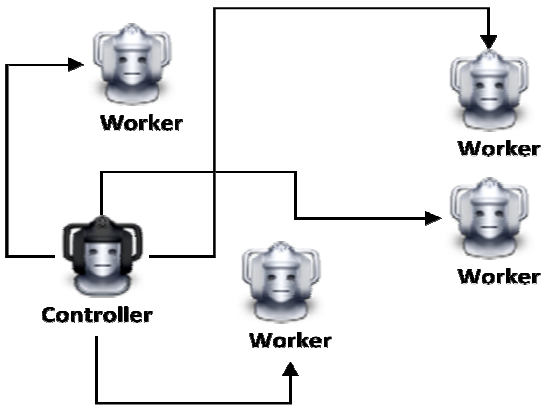
Esto significa que un servicio distribuido hace las veces de director de procesos (Controller) desde uno y solo uno de los nodos del clúster.

Como Controller, su tarea es ordenar la ejecución de los métodos del mismo servicio en el resto de los nodos de procesamiento (Workers) por medio de llamadas RPC.

Si el Controller desaparece del clúster de procesamiento, automáticamente otro tomará su lugar.



Si ese nodo reaparece (o si un nuevo nodo se suma al clúster) lo harán como Worker, no como Controller.



Si un Worker desaparece del clúster (o se suma al clúster), le toca al Controller como director de procesos reordenar las tareas en ejecución conforme a sus propias reglas de negocio y la capacidad de procesamiento total disponible.

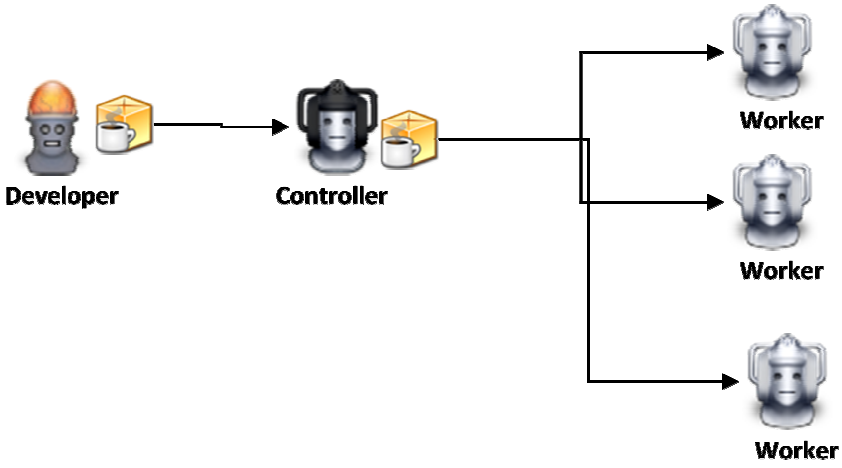
Dado que el modelo de comunicación para las notificaciones de eventos por cambio de constitución en el clúster es multipunto. Todos los nodos del clúster (Controller y Worker's), son notificados del evento de manera independiente.

Un caso práctico.

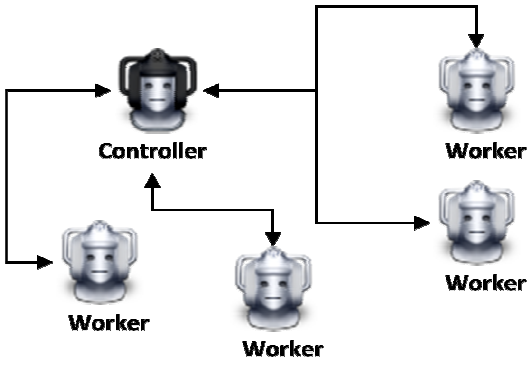
Un buen ejemplo de este "patrón de diseño" puede observarse en el servicio Deploy.

Cuando desde el entorno de desarrollo (por medio de la herramienta correspondiente) un desarrollador pide un despliegue en el clúster, lo está asiendo al Controller del servicio Deploy.

Es este Controller quien redistribuye el desplegable ante el resto de los nodos del clúster (los Worker's)



Por otra parte, permanentemente los Worker's están comparando su lista de desplegables contra el Controller. De existir una diferencia se ejecutará en el Worker la acción correctiva correspondiente (eliminar un desplegable o actualizarlo desde el Controller).



De esta forma, de producirse un error durante una transferencia entre el Controller y un Worker, o si un nodo estaba fuera del clúster al momento en que el desarrollador hiso el despliegue, estos se actualizaran contra el Controller en la primera oportunidad.

Por otra parte si el error ocurre durante la transferencia desde el entorno de desarrollo al Controller (por integridad transaccional) la operación no se realizara y se notificara del fallo al entorno de desarrollo para que el usuario decida si reintenta o no.

De las ediciones de Cybermen - SPC.

Como mencionado en la Guía de Instalación y configuración de la Public Edition, existen tres distribuciones de Cybermen - SPC las cuales son compatibles entre sí, y se diferencian por especialización.

Este documento se limita únicamente a la edición publica de Cybermen - SPC (the Public Edition), pero (y para que los desarrolladores sean consientes de ello) a continuación se realiza una breve descripción de las diferencias básicas entre las distintas ediciones.

La edición pública (the Public Edition).

La edición pública constituye el mínimo común denominador de las tres distribuciones, además de la base de estandarización para el desarrollo de servicios en Cybermen.

Esto significa que cualquier servicio construido y probado en la edición publica debería funcionar correctamente en la edición empresarial, y en la hardware edition.

Siempre que compartan el mismo número de versión del CLPackageInfo (ver referencia al tema en la guía de instalación) y que el hardware / OS lo permita.

La edición por hardware (the Hardware Edition).

Como es evidente, no se trata de un software instalable, si no de una pieza de equipo (mas sus módulos de expansión).

Concretamente de un controlador, a manera de PLR (Relé lógico programable) ordenado al manejo de señales digitales, analógicas, RS-232, display (monitores y carteleras), adquisición de datos (teclados, barcode y demás), entre otros servicios.

Para poder utilizar / desarrollar sobre estos LocalService es necesario adquirir un developer kit que (y además de las APIs y la documentación) se complementa con un emulador por software.

Desde el diseño .

Desde la perspectiva del diseño de soluciones, y además de lo obvio, una diferencia fundamental entre la Public y la Hardware Edition, está en que cada instalación de la Public Edition solo da lugar a un nodo de procesamiento (ver sección "Instalación" de la guía de instalación y configuración). Mientras que cada controlador / emulador de la Hardware Edition está compuesto por dos nodos, uno de procesamiento (esto es la parte en común con Public Edition), y otro de persistencia.

Del nodo de persistencia.

Los nodos de persistencias se ordenan conformando, por un lado, un filesystem compartido, y por otro, un motor de base de datos (HyperSQL) en alta disponibilidad.

Para poder utilizar esta característica (motor de db en alta disponibilidad) se cuenta con un DataSource (para los nodos de procesamiento) y con un driver JDBC (para aplicaciones / clientes externos al clúster) especializados (paquete "CyberJdbc.jar"), y que se distribuyen como parte del developer kit de la Hardware Edition o como componente de los nodos de la Enterprise Edition.

La edición de grado empresarial (the Enterprise Edition).

Esta edición toma de la Hardware Edition el nodo de persistencia pero se desmarca de este, en que los nodos de procesamiento son elásticos.

Los nodos elásticos .

En cada instalación de grado empresarial, se establece una cantidad mínima y máxima de nodos de procesamiento, más los deltas de consumo de memoria, procesador y espacio en disco.

Si los indicadores de memoria, procesador o disco están por encima de lo establecido, el controler de nodos empezará a "cerrar" instancias de nodos de procesamiento hasta que los valores de referencia (memoria, procesador o disco) alcancen el piso establecido, o el número mínimo de nodos para esta instalación.

Por lo contrario, si el hardware / equipo /servidor esta "holgado" (le sobra memoria, procesador y espacio en disco) el controler de nodos empezará a agregar instancias hasta alcanzar las cotas máximas de consumo, o el número máximo de instancias para los nodos de procesamiento establecidos para este equipo.

Este "comportamiento elástico" de las instancias Enterprise Edition, permite adaptar el tamaño del clúster, buscando el mejor rendimiento de los equipos utilizados (ya no se puede hablar de servidores) como "centros de procesamiento" o "puntos neurálgicos" dentro del clúster.

De los servicios de grado empresarial.

Como parte del paquete de grado empresarial se agregan servicios orientados a cubrir requerimientos propios de las áreas de seguridad informática (LoginService para LDAP y Active Directory, por ejemplo), de infraestructura, (notificación de eventos por perdida de conectividad o eliminación de nodos, por ejemplo), o de desarrollo de soluciones (servicios de integración con correo, entre otros).

Las herramientas de la Public Edition.

Apache Ant.

Cybermen provee de una serie de herramientas desarrolladas a partir de tareas de ANT y orientadas tanto al desarrollo de soluciones, como a una administración básica del clúster.

Para su utilización (y además de las librerías de Cybermen) se deberá contar dentro del build con los siguientes elementos.

- La pila protocolar de JGroups del clúster (archivo "jgroups.xml"), ver sección "Configuración del canal de comunicaciones" de la guía de instalación para más información.

- Nombre del clúster, ver sección "Configuración de un clúster" de la guía de instalación para más información.

- Usuario (y clave) con permisos para la tarea a ejecutar, ver sección "Servicio Deploy, security context & security policy" de la guía de instalación para más información.

A continuación se detallan cada una de las tareas / acciones permitidas, además de su declaración en el build

AntDeploy.

Permite desplegar y o actualizar un descriptor de servicios (archivo XML) o un paquete de servicios (archivo SAR).

```
<target name="deploy" depends="sar">
	<path id="toolslib">
		<path location="${lib}/CyberLinker.jar"/>
	</path>

	<taskdef name="deploy"
		classname="org.cyberLinker.tools.AntDeploy"
		classpathref="toolslib"/>

	<deploy clusterName="cybermen"
		jgroupsConfig="${basedir}/jgroups.xml"
		source="mysql-ds.xml"
		IPStack ="java.net.preferIPv4Stack"
		userName ="developer"
		password ="XXXX"
	/>
</target>
```

En el ejemplo anterior se observa el deploy de un DataSource para una base MySQL.

El parámetro "IPStack" corresponde a la configuración de JGroups y debe ser el mismo que el fijado para el clúster de procesamiento. Ver la documentación de JGroups para mayor información.

AntUndeploy.

Permite retirar / eliminar del clúster un descriptor de servicios (archivo XML) o un paquete de servicios (archivo SAR).

```
<target name="unDeploy">
	<path id="toolslib">
		<path location="${lib}/CyberLinker.jar"/>
	</path>

	<taskdef name="unDeploy"
		classname=" org.cyberLinker.tools.AntUndeploy"
		classpathref="toolslib"/>

	<deploy clusterName="cyberman"
		jgroupsConfig="${basedir}/jgroups.xml"
		source="mysql-ds.xml"
		IPStack ="java.net.preferIPv4Stack"
		userName ="developer"
		password ="XXXX"
	/>
</target>
```

En el ejemplo anterior se observa el unDeploy de un DataSource para una base MySQL.

AntGet.

Obtiene un descriptor de servicios (archivo XML) o un paquete de servicios (archivo SAR) del clúster.

```
<target name="get">
	<path id="toolslib">
		<path location="${lib}/CyberLinker.jar"/>
	</path>

	<taskdef name="get"
		classname=" org.cyberLinker.tools.AntGet"
		classpathref="toolslib"/>

	<deploy clusterName="cyberman"
		jgroupsConfig="${basedir}/jgroups.xml"
		source="mysql-ds.xml"
		IPStack ="java.net.preferIPv4Stack"
		userName ="developer"
		password ="XXXX"
	/>
</target>
```

En el ejemplo anterior se recupera el archivo mysql-ds.xml desplegado en el clúster.

StubCodeGenerator.

Esta tarea permite generar codigo java de un cliente de Cybermen - SPC personalizado para un servicio especifico del tipo RemoteInvokerService, conforme a la interface remota definida en el mismo

```
<target name="codeGenerator">
	<path id="toolslib">
		<path location="${lib}/CyberLinker.jar"/>
	</path>

	<taskdef name="codeGenerator"
		classname="org.cyberLinker.tools.StubCodeGenerator"
		classpathref="toolslib"/>

	<codeGenerator clusterName="cybermen"
		jgroupsConfig="${basedir}/jgroups.xml"
		IPStack ="java.net.preferIPv4Stack"
		userName ="developer"
		password ="XXX"
		srcdir="${src}"
		package="testing.text"
		accessService="ClientCodeGenerator"
	/>
</target>
```

En el ejemplo anterior creará el código fuente de un cliente dedicado para el servicio ClientCodeGenerator en el paquete testing.text del proyecto

AntReset.

Esta tarea permite reiniciar todos los nodos del clúster al mismo tiempo y solo debería emplearse en el caso de instalar o actualizar librerías de terceros dentro del directorio "include" de todos los nodos del clúster, para que aquellos servicios que la emplean puedan acusar el cambio.

```
	<target name=" reset">
		<path id="toolslib">
			<path location="${lib}/CyberLinker.jar"/>
		</path>

		<taskdef name="reset"
			classname="org.cyberLinker.tools.AntReset"
			classpathref="toolslib"/>

		<deploy clusterName="cyberman"
			jgroupsConfig="${basedir}/jgroups.xml"
			IPStack ="java.net.preferIPv4Stack"
			userName ="admin"
			password ="XXXX"
		/>
	</target>
```

AntShutdown.

Ojo con esto. Esta instrucción detiene todos los nodos del clúster.
Es decir, detiene el clúster, por lo que después para reactivarlo es necesario levantar nodo por nodo.

```
<target name="shutdown">
	<path id="toolslib">
		<path location="${lib}/CyberLinker.jar"/>
	</path>

	<taskdef name="shutdown"
		classname="org.cyberLinker.tools.AntShutdown"
		classpathref="toolslib"/>

	<deploy clusterName="cyberman"
		jgroupsConfig="${basedir}/jgroups.xml"
		IPStack ="java.net.preferIPv4Stack"
		userName ="admin"
		password ="XXXX"
	/>
</target>
```

Descriptores y paquetes de servicios.

En esta sección explicaremos como declarar y empaquetar los servicios para su despliegue, al igual que su estructura de encapsulamiento.

El descriptor de servicios.

Para que un servicio de Cybermen - SPC pueda ser desplegado dentro de un clúster de procesamiento debe contar con un descriptor que indique sus características y configuración de arranque.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<entry key="services">JDBC_MYSQL</entry>

<entry key="JDBC_MYSQL.classtName">org.cyberPacket.service.sar.DataSource</entry>
<entry key="JDBC_MYSQL.driverClass">com.mysql.jdbc.Driver</entry>
<entry key="JDBC_MYSQL.connectionURL">jdbc:mysql://db-server-one/</entry>
<entry key="JDBC_MYSQL.userName">root</entry>
<entry key="JDBC_MYSQL.password"></entry>
<entry key="JDBC_MYSQL.validationQuery">select 1 from dual</entry>

</properties>
```

El ejemplo anterior corresponde al archivo " mysql-ds.xml " y describe a un DataSource para una base MySQL.

A continuación se detallan los parámetros y atributos comunes a todos los servicios.

services

Lista de nombres separados por espacio " " de cada servicio del descriptor.
En el ejemplo anterior solo posee una entrada "JDBC_MYSQL"

<serviceName>.classtName

Este atributo es obligatorio, e indica la clase a ser cargada como servicio
En el ejemplo anterior?. "org.cyberPacket.service.sar.DataSource"

<serviceName>.depends

Este atributo es optativo, contiene una lista de nombres separados por espacio " " de los servicios que deberán estar operativos en el nodo antes de que este esté disponible.

<serviceName>.transDepends

Este atributo es optativo, contiene una lista de nombres separados por espacio " " de servicios de tipo RemoteInvokerService (servicios de acceso remoto) cuyos controler´s deben estar operativos dentro del clúster, antes de que este esté disponible para ser utilizado.

En el ejemplo siguiente el LoginService (la política de seguridad) "CyberPolicy" no estará operativo hasta que el servicio "HsqlServer" este accesible de forma remota para el nodo.

```
<entry key="CyberPolicy.classtName">org.cyberCoreHE.sar.DataBase.HsqlLogin2Service</entry>
<entry key="CyberPolicy.serverDB">HsqlServer</entry>
<entry key="CyberPolicy.transDepends">HsqlServer</entry>
```

De la dependencia a la trans dependencia

Mientras que la dependencia entre servicios / EJB´s es un concepto normal para el desarrollo en Java EE (servicios / EJB´s desplegados en el mismo AS). La trans dependencia es la evolución de este concepto llevado a un escenario de servicios colaborativos operando de manera distribuida dentro del mismo clúster, pero donde no todos los servicios están desplegados (o están disponibles) en todos los nodos o al mismo tiempo.

isDeploy

Optativo, este atributo puede establecerse tanto a nivel de descriptor de despliegue como de servicio (<serviceName>.isDeploy).

Se trata de una expresión regular (devuelve verdadero o falso) escrita en JavaScript Java embebido que permite tomar decisiones en tiempo de deploy.

En el siguiente ejemplo el isDeploy establecido a nivel de descriptor de despliegue. No permitirá que ninguno de los servicios del descriptor se carguen en el nodo si el mismo no está operando en un hardware con arquitectura ARM y el usuario de ejecución del nodo a nivel OS no sea "PI"

```
<entry key="isDeploy">
( ( System.getProperty("os.arch", "").equalsIgnoreCase("ARM") ) and
  ( node.userName.equalsIgnoreCase("pi") )
)</entry>
```

Si la misma restricción, en lugar de establecerse a nivel de un descriptor de despliegue, se establece a nivel descriptor de arranque (archivo "cyber-service.xml" del directorio "conf" del nodo) por falso, se apagara el nodo, sin que esto constituya o reporte una falla al resto del clúster.

En el siguiente ejemplo el isDeploy del servicio "IOService", no permitirá que el mismo se cargue dentro del nodo si este no está operando en un hardware con arquitectura ARM, el paquete "pi4j-core.jar" esté disponible dentro del classPath y el usuario de ejecución del nodo a nivel OS no sea "PI"

```
<entry key="IOService.classtName">org.cyberCoreHE.sar.io.IOService</entry>
<entry key="IOService.isDeploy">
( ( System.getProperty("os.arch", "").equalsIgnoreCase("ARM") ) and
  ( System.getProperty("java.class.path", "").toLowerCase().indexOf("pi4j-core.jar") > -1) and
  ( node.userName.equalsIgnoreCase("pi") )
)</entry>
```

El isDeploy no es una dependencia

En el caso de las dependencias, o de las trans dependencias las mismas, se validaran una, otra y otra vez, hasta que los servicios referenciados como dependencias estén disponibles. Solo entonces se procederá con la carga del servicio dependiente.

En el caso del isDeploy esta función solo se ejecuta una vez, durante el proceso de carga o de deploy, teniendo que redeployar el paquete de servicios, o reiniciar el nodo para que esta función vuelva a ejecutarse.

Si el isDeploy devuelve falso o se produce un error de ejecución en el mismo (genera un ScriptException), se tomará por malo y el nodo actuara en consecuencia.

Del contexto de ejecución.

A nivel de ScriptEngine se fijan como importPackage las librerías "java.lang.*" y "org.cyberLinker.dto.*" teniendo que referenciar de manera completa cualquier otra clase java que no esté en estos paquetes.

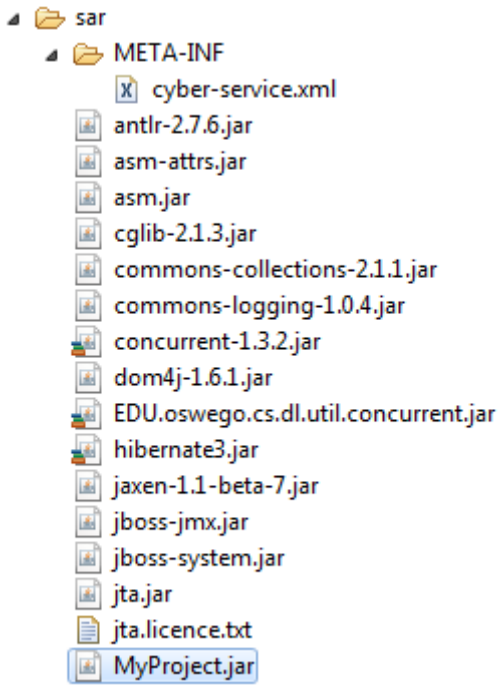
La única variable de entorno definida (dentro de la Public Edition) es el objeto node que representa al nodo actual.

De la sintaxis a nivel de JavaScript

por problemas de incompatibilidad de formatos (entre XML y JavaScript) los operadores " || " y " && " fueron reemplazados por las palabras reservadas " or " y " and " respectivamente, teniendo que dejar un espacio en blanco " " como separador entre los términos.

El paquete de servicios (SAR).

Cybermen - SPC soporta y promueve la utilización de ficheros SAR como mecanismo de encapsulamiento para los servicios, sus recursos, dependencias y descriptor de despliegue correspondientes al mismo desarrollo.



Estructura interna del archivo MyProject.sar.
La librería MyProject.jar mostrada en la imagen corresponde a los vinarios del proyecto.

Nota: dentro de un archivo SAR el descriptor de despliegue se encuentra siempre en el directorio "META-INF" con el nombre de "cyber-service.xml"

Importante

La forma correcta de crear un paquete de servicios es por medio de la herramienta jar del JDK o su equivalente en ANT.

Del encapsulamiento y la estructura de ClassLoader.

Cada nodo de procesamiento de Cybermen - SPC (y al igual que algunos Application Server) emplea una estructura de ClassLoader en tres niveles.

Root Classloader

Correspondiente a la variable CLASSPATH, al classloade raíz, y al comportamiento estándar para las aplicaciones Java desde la versión 1.2 de la JVM.

Include

Corresponde al directorio "include" de la instalación de un nodo de Cybermen. Toda librería de objetos Java (archivos "jar" o "zip") a ser utilizado por los servicios pero que NO se desee (o no sea conveniente) redistribuir como parte del paquete de servicios (driver's jdbc, Hibernate, jakarta commons, librerías J2EE, etc..) deberán incluirse en este directorio.

Nota: al igual que con algunos Application Server, es necesario reiniciar el Nodo para que el mismo acuse un cambio en el directorio.

SarClassloader

Correspondiente al paquete de servicios (los servicios, sus recursos, dependencias y descriptor de despliegue encapsulados en este), existe una instancia de SarClassloader por cada descriptor de servicios, que estén activo en el nodo.

Por regla general los servicios que están declarados en el mismo descriptor comparten la misma instancia de SarClassloader, por lo que no existen problemas de pasaje de referencia o de serialización entre ellos.

Este Classloader está atado al contexto de ejecución y es accesible por medio de este.

La estructura de servicios de Cybermen - SPC

Como mencionado anteriormente, Cybermen es un contenedor de servicios distribuidos, lo que obliga a plantear los desarrollos para esta plataforma, bajo una arquitectura orientada a servicios (SOA) y contemplando la utilización de llamadas a procedimientos remotos (RPC) entre pares (el mismo servicio en otros nodos de procesamiento) o por aplicaciones externas al clúster por medio del cliente de Cybermen.

Recomendación

Desde este punto, es conveniente seguir esta guía con el apiDoc de Cybermen a la mano, para consultar la documentación de desarrollo relacionada a los temas tratados.

La interfaz Service

Perteneciente al paquete "org.cyberLinker.service.model", es una abstracción de la que heredan las interfaces operativas a ser utilizadas por los desarrollos.
Como consta en la apiDoc de Cybermen la misma define los siguientes métodos.

close

Este método es invocado por el nodo desde el contexto de ejecución de forma previa al sierre de un servicio.

Importante

Si un servicio desea serrarse como parte de su lógica de comportamiento deberá hacerlo por medio del contexto, y no por invocación directa a este método.
Ver LocalContext en esta guía.

init

Este método es invocado por el nodo desde el contexto de ejecución de forma previa al despliegue como parte de la inicialización del servicio.

LocalContext

Esta interface es recibida por el servicio como parte de su inicialización y representa al contexto de ejecución. Por lo que es recomendable conservar la referencia dentro del servicio para su utilización conforme a conveniencia.
Sus métodos...

stopService

Detiene el servicio en ejecución previa llamada al método close del mismo.

restartService

Detiene el servicio (previa llamada al método close del mismo) y vuelve a lanzarlo (volviendo a llamar al método init del mismo) .

getName

El nombre del servicio, tal como declarado en el descriptor de despliegue.

getService

Dado un nombre de servicio, obtiene una referencia a dicho servicio dentro del mismo nodo de procesamiento.
Dicho de otra manera, es el canal de comunicaciones entre servicios del mismo nodo.

getStub

Dado un nombre de servicio del tipo RemoteInvokerService (de acceso remoto), devuelve un canal RPC contra el Controler de ese servicio, sea este local o remoto, dentro del mismo clúster de Cybermen - SPC.
Nota: Esto es una implementación dentro del nodo de procesamiento del cliente de Cybermen, que hereda la configuración del nodo y el esquema de permisos del servicio solicitante.

getClassLoaderService

Regresa una referencia al ClassLoader correspondiente al despliegue de un descriptor XML o de un paquete SAR, este es el mecanismo a utilizar para acceder a los recursos disponibles para un servicio como parte de su workspace, disponibles en el directorio include del nodo de procesamiento, y del resto de librerías de Cybermen

getNode

Regresa una instancia de la clase Node conteniendo información sobre el nodo desde el cual se está ejecutando el servisio.

getClient

Regresa una instancia de la clase Client conteniendo información sobre el punto de origen del thread de ejecución, sea este un nodo de procesamiento (local o remoto) o un cliente de Cybermen.

> ***Nota***
>
> Por "estar atado" al thread de ejecución desde el punto de origen se entiende que este método varía su respuesta entre entradas.

getStartTimeMillis

Devuelve el tiempo en milisegundos que el servicio lleva operando en este contexto sin variaciones.

LocalService

Esta interfaz se piensa para ser implementada por aquellos servicios que son consumidos dentro del mismo nodo de procesamiento, ya sea desde otros servicios o desde cualquiera de sus subclases.

Un ejemplo claro de este tipo de servicios es la clase "DataSource" del paquete "org.cyberPacket.service.sar" responsable del soporte de DataSource para Cybermen.

```
import java.io.PrintWriter;

 * Cybeversion of sql.DataSource..
public class DataSource implements LocalService, javax.sql.DataSource
      { private static Logger log = Logger.getLogger(DataSource.class.getName());
```

La variante TimerTask

Si un servicio local hereda de la clase java.util.TimerTask. es posible agregar en su descriptor de despliegue el atributo "period".

Si este atributo se marca con un valor superior a los 100 milisegundos. El método run será invocado desde el contexto según el intervalo de tiempo fijado.

Como resulta evidente, esta variante se piensa para la ejecución de tareas programadas o de ejecución croneada.

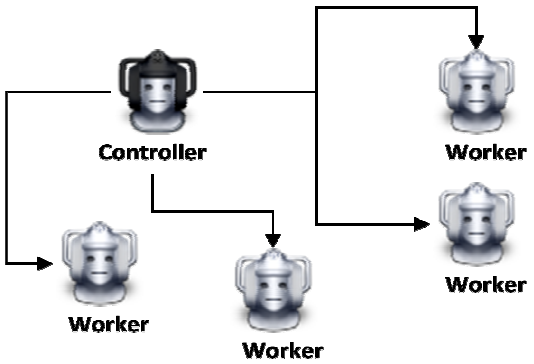
Un ejemplo de esta variante, puede encontrarse en el servicio LocalDeploy dentro del descriptor de arranque (archivo "cyber-service.xml" en el directorio "conf") de los nodos de procesamiento.

```
<entry key="LocalDeploy.classtName">org.cyberCore.service.sar.LocalDeploy</entry>
<entry key="LocalDeploy.depends">Deploy</entry>
<entry key="LocalDeploy.period">5000</entry>
```

En el ejemplo anterior, se fija una latencia de 5 segundos, para que los worker's revisen su lista de servicios y la comparen contra el controler.

ClusterService

Este es el primero de los servicios distribuidos y se corresponde directamente con lo descripto en la sección Workers Control, system pattern de esta guía.
Además de los métodos close e init de la interfaz Service, agregan los métodos runController, runWorker, los cuales se ejecutaran uno u otro en un nodo de procesamiento conforme al papel que desempeñe el servicio en ese nodo.
También ofrece el método disconnectedEvent del que hablaremos más adelante.



Estos métodos (runController, runWorker y disconnectedEvent), se ejecutan varias veces dentro del siclo de vida de un servicio en un nodo de procesamiento, conforme a dos criterios distintos.

Por cambio de constitución del clúster

Cada vez que una instancia del mismo servicio se ponga disponible o desapareciese del clúster de procesamiento, cada nodo del clúster notificará a su propia instancia del mismo servicio del evento y el papel que le corresponde. Si desde el evento, pasa a ser un controller se llamará al método runController, de lo contrario se llamará al método runWorker.

Por tarea programada

Como con la variante TimerTask de LocalService, es posible especificar en el descriptor de arranque el parámetro "period".
De realizar esta configuración con un valor superior a los 100 milisegundos. El método run correspondiente (runController o runWorker), será invocado desde el contexto según el intervalo de tiempo fijado y de manera independiente del evento de cambio de constitución del clúster.

Del método disconnectedEvent

Este método es invocado desde el contexto ante un evento de desconexión del servicio del resto del clúster.
Este método será llamado regularmente (mientras subsista el evento de desconexión) conforme al intervalo de tiempo definido en el parámetro "period" o con un intervalo de 3 segundos en caso de que dicho parámetro no esté especificado.

Importante:
Se considera que un ClusterService esta desconectado del resto, cuando permanece sin comunicación con el testo de los nodos por un periodo de tiempo superior a los 3 secundas.

Nota:
Esta limitación de tiempo se establece para enviras falso eventos de desconexión provocados al utilizar la pila protocolar TCP-IP de JGroups en redes inestables o de baja ganancia.

El método getStartTimeMillis en el contexto de un ClusterService

En el caso de los ClusterService el método getStartTimeMillis de la interface LocalContext. se resetea ante un evento de cambio de contexto.

Es decir que si en un nodo, un servicio, está asiendo las beses de Controller o Worker el método getStartTimeMillis le estará indicando cuanto tiempo lleva ejerciendo ese rol.

Lo mismo ocurre ante un evento de desconexión. Por lo que, de ser llamado desde el cuerpo del método disconnectedEvent le indicará al servicio cuanto tiempo lleva desconectado.

ContextService

Esta interfaz, extiende en funcionalidad a LocalContext, agregando el soporte necesario para el procesamiento distribuido. La misma es recibida por el servicio como parte de las llamadas a los métodos runController, runWorker y disconnectedEvent, y (al igual que con LocalContext) representa al contexto de ejecución. Por lo que es recomendable conservar la referencia dentro del servicio para su utilización conforme a conveniencia.

Sus métodos...

amITheController

Verdadero si la instancia del servicio en ese momento y nodo de procesamiento esta asiendo de Controller. Falso por lo contrario.

getController

Regresa una instancia de la clase Node correspondiente al nodo sobre el cual se está ejecutando el rol de Controller del servicio.

getCollectionNodes

Regresa una colección de la clase Node sobre los cuales se están ejecutando el mismo servicio en ese instante.
Nota: la colección no incluye el nodo desde el cual se llama al método, solo los nodos remotos, e independientemente del rol de ejecución del servicio en cada nodo (controller o worker).

callMethod

Estos métodos permiten la ejecución de un método del mismo servicio en el nodo especificado vía RPC, y obtienen el resultado

Importante.

Se entiende que cada parámetro enviado al método remoto, al igual que el valor de retorno de dicho método, deben implementar Serializable (o Externalizable) del paquete java.io para su utilización vía RPC.

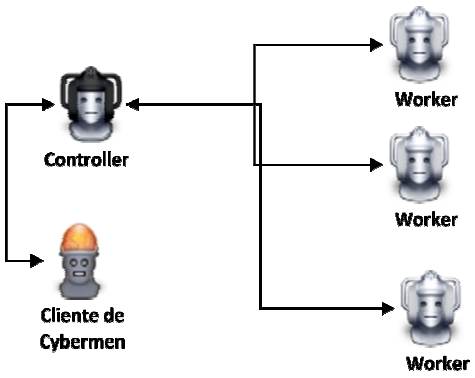
callCollectionMethod

A diferencia de callMethod, este método permite la ejecución de un método del mismo servicio, en todos los nodos especificados en la colección, y obtiene una instancia de Map<Node,Object> identificando la respuesta de la ejecución en cada nodo de procesamiento

RemoteInvokerService

Esta interfaz, extiende las capacidades de ClusterService para su invocación por parte de aplicaciones externas al clúster de procesamiento, ya sea desde el cliente genérico de Cybermen o de los generados por medio de la herramienta StubCodeGenerator.

Es de destacar que el cliente de Cybermen no habla directamente con el clúster de procesamiento, si no que se comunica con el controller, siendo este (y conforme a su propia lógica) quien decide responder al cliente, o redirigir la llamada a uno o mas worker's. Ver el ejemplo descripto en la sección un caso práctico de esta guía.



getRemoteInterface

Este método debe ser implementado desde el servicio, y corresponde a la definición de la interface a ser empleada por las aplicaciones externas.

También es el punto de partida para la creación del código fuente generado por la herramienta StubCodeGenerator.

> ***Importante***
>
> Canto el cliente de Cybermen, como el StubCodeGenerator no encestan implementar esta interface ni tenerla como parte de sus classpath,
> Lo único que un desarrollador o el StubCodeGenerator necesita de esta es conocer sus métodos (el apiDoc) para saber cómo emplearlos.

Restricción de acceso por roles.

Al tratarse de un servicio visible desde el mundo exterior, el mismo debe estar bajo el contexto de una política de seguridad (ver sección LoginService en esta guía, y "Servicio Deploy, security context & security policy" de la Guía de Instalación de Cybermen).
Para cumplir con esto, Cybermen adopta el uso de las anotaciones @RunAs (a nivel de clase) y @RolesAllowed (a nivel de método) pertenecientes a la especificación "Specifying the runAs Security Identity" de java EE.

```
@RunAs("developer administrator")
public class Deploy implements RemoteInvokerService, RemoteDeploy
       { private ContextService context = null;
```

Ejemplo de restricción de acceso, tomado del servicio Deploy. El usuario del cliente de Cybermen, debe poseer el rol de "administrator" o el de "developer" para poder acceder al servisio.

```
@RolesAllowed("undeploy")
public void delete(String name) throws .....
{ File source = new File(System.getProperty("cyberman.deploy"));
```

Ejemplo de restricción de acceso al método delete del servicio Deploy. El usuario del cliente de Cybermen, debe poseer el rol de "undeploy" para poder ejecutar dicho método.

LoginService

Esta interfaz, identifica a un servicio (sea este del tipo Local, Cluster o RemoteInvoker) como política de seguridad a ser utilizada desde el contexto de un RemoteInvokerService.

Quienes quieran implementar sus propias políticas de seguridad (integración con LDAP o con Active Directory, por ejemplo) deberán implementar desde un servicio de Cybermen esta interfaz, desplegar dicho servicio, para (recién entonces) poder mencionarlo como política de seguridad en el descriptor de un RemoteInvokerService.

Ver sección "Servicio Deploy, security context & security policy" de la Guía de Instalación de Cybermen

El canal de comunicaciones y la clase ChannelAccess

Como mencionado en la sección "Configuración del canal de comunicaciones" de la Guía de Instalación, Cybermen utiliza JGroups v3.2.12.Final como toolkit de mensajería fiable, para las llamadas RPC, tanto entre los servicios dispuestos en un clúster de procesamiento, como con los clientes de Cybermen.

Para ello no solo mantiene un único archivo de declaración de la pila protocolar, si no que, además, y para mantener la separación entre clústeres que estén en la misma infraestructura, agrega al nombre del canal de JGroups, el nombre del clúster de procesamiento.

Aquellos desarrollos que empleen JGroups, ya sea de manera directa como mecanismo de mensajería asincrónica, o por medio de herramientas de terceros, deberán gestionar el JChannel de JGroups por medio de la clase ChannelAccess del paquete " org.cyberLinker.client" y el método getChannel(String channelName). Ver el ejemplo siguiente

```
File config = new File(System.getProperty("cybermen.config"));
     config = new File(config, "jgroups.xml");

ChannelAccess channelAccess = new ChannelAccess( System.getProperty("cybermen.clusterName"),
                                                 config.toURI().toURL()
                                               );
JChannel channel = channelAccess.getChannel("cMessenger");
myListModel = new MyListModel(channel);
```

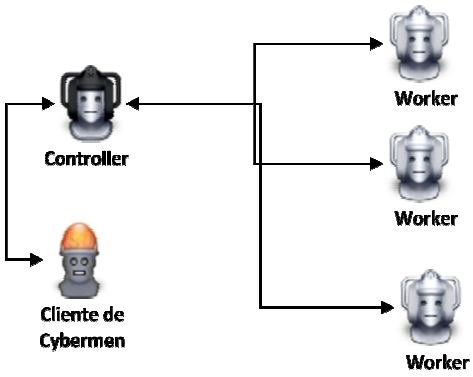
Ejemplo de ostensión de un canal de JGroups, desde un servicio local.

Como resulta evidente al observar el codujo de muestra, el objetivo de acceder a la clase JChannel de JGroups por medio de un ChannelAccess, y no directamente desde JGroups, está en que el mismo comparte la separación de canales / clúster de Cybermen - SPC,
Además de que la clase ChannelAccess es el puente de acceso a los StubService de los RemoteInvokerService.

El cliente de Cybermen

Como mencionado en varias oportunidades a lo largo de esta guía y de la Guía de Instalación, Cybermen - SPC ofrece un mecanismo de conexión remota con el clúster de procesamiento, tanto para el canal de comunicaciones (ver sección "El canal de comunicaciones y la clase ChannelAccess"), como para los servicios (RemoteInvokerService) vía RPC

Recordar que en el caso de RemoteInvokerService los clientes dialogan con el nodo que esté haciendo las veces de controller al momento de una llamada RCP



Pudiendo variar este de una invocación RCP a otra, por lo que las mismas deberán manejarse de forma transaccional y auto contenida.

La clase ChannelAccess

Del paquete "org.cyberLinker.client" este es el punto de entrada al clúster desde el mundo exterior. El mismo se diseña pensando en los siguientes escenarios de integración.

Comunicación inter clúster

Entre distintos clúster de procesamiento. Uno emplea el cliente del otro. Por lo que se deberá contar con el nombre y la pila protocolar del otro clúster para poder establecer la comunicación remota.

Desde aplicaciones externas, al clúster

Desde un punto de vista práctico, es exactamente igual al caso anterior.

getStubService

Los métodos getStubService, son los encargados de obtener el stub correspondiente al RemoteInvokerService solicitado.

```
try { ChannelAccess channelAccess =
            new ChannelAccess(clusterName, jgroupsConfig.toURI().toURL());
      stub = channelAccess.getStubService("Deploy", userName, password);
     } catch(Exception e)
             {
```

Ejemplo invocación al servicio "Deploy" dese el código fuente de las herramientas de ANT de Cybermen.

Nota

De no especificarse el userName and password, el cliente de Cybermen, tomara el nombre de usuario del sistema operativo, y la palabra "anonymous" como password

Comunicación remota entre servicios

La clase ChannelAccess está desaprobada para este escenarios de integración. En su logar se debe utilizar el método getStub provisto por el contexto de ejecución del servicio.

StubService

Del paquete "org.cyberLinker.client", esta clase representa a un canal RPC contra un servicio RemoteInvokerService operando como controler, dentro del clúster de Cybermen - SPC. Conforme a los métodos expuestos por medio del "getRemoteInterface" del servicio.

Además de los métodos "callMethod", "close", " getController", heredados de la interface "BasicContext"

Métodos desaprobados (Deprecated)

Los métodos "**block**", "**suspect**", "**viewAccepted**" y "**unblock**" provenientes de la interface " MembershipListener".de JGroups, están desaprobados, y no deberían utilizarse.

getClient

Identifica a la entidad cliente de un servicio pero dese la perspectiva del propio cliente, no desde la del servicio.

isClosed

Este método indica si el canal de comunicaciones con el clúster esta activo.

isAccessMethod

Este método es complementario a los "callMethod" mientras que los primeros permiten la ejecución de un método del servicio dentro del clúster por RPC, "isAccessMethod" permite saber si el cliente tiene los permisos necesarios para poder realizar la llamada al método del servicio (callMethod).

De la estructura de proyectos de Cybermen y sus licencias

Como mencionado anteriormente, existen tres distribuciones de Cybermen - SPC, las cuales son compatibles entre sí, y se diferencian por especialización (Public, Hardware y Enterprise Edition).

Además de esta distinción, el proyecto fue dividido en bloques para facilitar su desarrollo y mantenimiento, Siendo los siguientes los relevantes para la edición pública.

CyberLinker.jar

Es el bloque fundacional de Cybermen, contiene las herramientas de desarrollo descriptas en esta guía, los clientes de Cybermen y todos los objetos (clases e interfaces) necesarios para crear servicios y poder desplegarlos en el clúster.

Cualquier edición / distribución de Cybermen (sean las tres actuales u otras) están obligados a guardar compatibilidad y dependencia con todos los elementos del paquete CyberLinker para ser operativos y compatibles con el resto de las ediciones.

De la licencia

CyberLinker se desarrolla y mantiene bajo licencia LGPL y su código fuente está disponible bajo los términos generales de esta licencia.

Dado que CyberLinker constituye el bloque fundacional y la base para las distintas implementaciones de Cybermen, cualquier cambio, modificación, alteración, ampliación o restricciones, sobre sus objetos, funcionalidades o modelos de comportamientos, deberá ser informado y o solicitado al equipo de desarrollo de CyberLinker para que el mismo sea evaluado e incorporado en futuras versiones de esta librería, conforme a lo establecido en la Lesser General Public License, version 3.0.

CyberCore.jar

Constituye el centro de un nodo de procesamiento de la edición pública y forma parte (a nivel de componente) de las Hardware Edition y Enterprise Edition. Por lo que no se permite su utilización por fuera de las mismas.

CyberCore no es de código abierto, y hereda el esquema de licenciamiento de las distintas ediciones de Cybermen.
La razón por la que el código de esta librería es de acceso restringido (solo a aquellos que colaboran en el mantenimiento y evolución del mismo). es que se trata de una pieza de software compacta, muy integrada, en donde no hay mucho espacio para que demasiadas personas metan mano.

De la licencia en la Public Edition

La edición publica de Cybermen - SPC es de software libre.
Libre para bajarse, para ser redistribuido (sin alteraciones, claro), para ser instalado, para ser empleado. No existen limitaciones (por parte de Cybermen) en cuanto a la cantidad de nodos que pueden conformar un clúster, ni la cantidad de clústeres que se puedan conformar dentro de la misma red, o de los servicios a ser desplegados en un clúster.

CyberPacket.jar

Corresponde al grupo de servicios básicos disponibles en las distintas ediciones de Cybermen (Public, Hardware y Enterprise Edition).

La idea de colocar estos servicios en un paquete separado (y además de facilitar el desarrollo, mantenimiento y actualizaciones independiente de las ediciones), está en que CyberPacket sea el receptor de las donaciones de la comunidad.

Es decir que si alguien considera que uno o más servicios que haya desarrollado, pueden ser de utilidad para un tercero y desea donarlos a la comunidad.

El equipo de desarrollo de CyberPacket (después de evaluarlo, y con los agradecimientos del caso) incorporará estos servicios dentro del CyberPacket

De la licencia

CyberPacket se distribuye como parte de las ediciones de Cybermen bajo licencia LGPL y su código fuente está disponible bajo los términos generales de esta licencia.

Al igual que con la licencia de CyberLinker, cualquier cambio, modificación, alteración, ampliación o restricciones, sobre sus servicios, objetos, funcionalidades o modelos de comportamientos, deberá ser informado y o solicitado al equipo de desarrollo de CyberPacket para que el mismo sea evaluado e incorporado en futuras versiones de esta librería, conforme a lo establecido en la Lesser General Public License, version 3.0.

De los servicios básicos

De los servicios disponibles en el CyberPacket a continuación hablaremos de su funciones y rasgos principales.

UsersRolesLogin

Es la política de seguridad por defecto, disponible como parte de la Public Edition y su descriptor es el siguiente.

```
<entry
key="CyberPolicy.classtName">org.cyberPacket.service.security.UsersRolesLogin</en
try>
<entry key="CyberPolicy.usersProperties">login-users.properties</entry>
<entry key="CyberPolicy.rolesProperties">login-roles.properties</entry>
```

Del atributo usersProperties

Contiene el nombre de un archivo en el directorio "conf" de tipo properties, que contiene el par nombre de usuario y clave de acceso de cada uno de los usuarios registrados a ser validados por esta política.

Del atributo rolesProperties

Contiene el nombre de un archivo en el directorio "conf" de tipo properties, que contiene para cada nombre de usuario, una lista separada por espacios " " de los roles asignados al mismo dentro de esta política.

Nota: como con cualquier servicio de Cybermen, se pueden declarar N instancias del mismo tipo en el mismo nodo, siempre y cuanto cada instancia tenga distinto nombre de servicio dentro del descriptor.

CyberMap

Implementa la interface Map de JavaSE como un objeto distribuido dentro del clúster de procesamiento.

De los métodos put y remove

El "como un objeto distribuido" significa que las operaciones de put y remove (e independientemente de en que nodo del clúster ocurran) tendrán impacto en todo el clúster, dado que las mismas se realizan contra el controler, y desde este a los workers en una única unidad transaccional

Nota: si un workers se suma al clúster su primer tarea es actualizar el mapa de objetos desde el controler antes de volverse disponible para el resto del nodo.

CyberMaps, no es thread-safe.

Dado que CyberMaps no implementa ningún mecanismo de bloqueo de registros o patrón de dominio de elementos, de realizarse múltiples operaciones simultáneas sobre la misma clave, solo tendrá valor la última operación realizada por el controlar. lo que puede generar inconsistencias entre el mapa y los iterator del mismo.

CyberProperties

Entiende las funciones de CyberMap agregando soporte para dataFile.

Del atributo dataFile

Contiene el nombre de un archivo en el directorio "conf" de tipo properties, utilizado para guardad el set de datos.

De los métodos flush

Cada vez que este método sea llamado (en cualquier parte del clúster) se procede a guardad el set de datos en el dataFile de cada nodo (en todo el clúster).

Del patrón Workers Control en este servicio

Cuando un servicio es activado, el control obtendrá el dataSet de su dataFile e impondrá ese set de datos a los Workers.

DataSource

Implementa la interface DataSource, CommonDataSource y Wrapper de JavaSE, como un servicio local.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<entry key="services">JDBC_MYSQL</entry>

<entry key="JDBC_MYSQL.classtName">org.cyberPacket.service.sar.DataSource</entry>
<entry key="JDBC_MYSQL.driverClass">com.mysql.jdbc.Driver</entry>
<entry key="JDBC_MYSQL.connectionURL">jdbc:mysql://db-server-one/</entry>
<entry key="JDBC_MYSQL.userName">root</entry>
<entry key="JDBC_MYSQL.password"></entry>
<entry key="JDBC_MYSQL.validationQuery">select 1 from dual</entry>

</properties>
```

El ejemplo anterior corresponde al archivo " mysql-ds.xml " y describe a un DataSource para una base MySQL.

Del pool de conexiones

Un defecto aun sin corregir en este servicio, es que los objetos Connection no regresan al pool de conexiones una vez entran en desuso.
Por lo que es necesario "serrallos" antes de ser descartados.