Diseño de un sistema delivery-versus-payment empleando Distributed Ledger Technologies

Trabajo Final

Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones

Jose Manuel Panizo Plaza

Directora María Francisca Hinarejos Campos

Enero 2019

Ten siempre a Ítaca en tu mente.

Llegar allí es tu destino.

Mas no apresures nunca el viaje.

Mejor que dure muchos años

y atracar, viejo ya, en la isla,

enriquecido de cuanto ganaste en el camino

sin aguantar a que Ítaca te enriquezca.

Ítaca te brindó tan hermoso viaje.

Sin ella no habrías emprendido el camino.

Pero no tiene ya nada que darte.

Aunque la halles pobre, Ítaca no te ha engañado.

Así, sabio como te has vuelto, con tanta experiencia,

entenderás ya qué significan las Ítacas.

Poema Ítaca, de Konstantinos Petrou Kavafis

Resumen

Las diversas tecnologías de almacenamiento de datos distribuidos, entre las que se encuentra blockchain, representan el nuevo paradigma de la información. Debido a su orientación a transacción y a la posibilidad de verificación y distribución de la información, se convierten en candidatos a proporcionar el ecosistema necesario para los diversos servicios financieros

Este Trabajo Fin de Máster tiene como objetivo el diseño de un sistema de "delivery-versus-payment" (DVP a partir de ahora), empleando como método de intercambio en el entorno financiero europeo, empleando tecnologías de tipo "Distributed Ledger Technologies" (DLT a partir de ahora). Se iniciará con una breve introducción a las infraestructuras de pagos europeas y el concepto de DVP. Posteriormente, se realizará un estudio de los conceptos DLT y las diversas características que ofrecen en cuanto a redes públicas / privadas, método de consenso o mantenimiento de la integridad. A continuación, se seleccionarán las cualidades que deberá poseer una solución DLT para el entorno a estudiar. Posteriormente, se analizarán los diversos componentes y características del framework Hyperledger Fabric, y sobre cada uno de ellos, se tomarán decisiones para el diseño de la arquitectura DVP, teniendo en cuenta la seguridad y la funcionalidad de la arquitectura.

Para terminar el documento actual, se detallará un sistema equilibrado de intercambio entre los actores del sistema DVP, y se incluirá una prueba de concepto sobre el framework Hyperledger Fabric.

Contenido

Tabla de ilustraciones

1. Introducción

1.1. Objetivos del Trabajo Fin de Máster

La tecnología de libro de cuentas distribuido se encuentra presente en la hoja de ruta de numerosos proyectos tecnológicos, debido a sus capacidades de trazabilidad, transparencia e integridad de la información. Una de sus materializaciones, la cadena de bloques o blockchain, es la tecnología disruptiva más popular en los últimos años. Resulta de gran interés analizar las características genéricas de este tipo de tecnologías, y de forma particular, los componentes y mecanismos de una tecnología concreta: Hyperledger Fabric. Dicho análisis se completa y concreta con los detalles de diseño para una industria concreta, la industria financiera, a través del mecanismo entrega-contra-pago de intercambio de liquidez contra valores. Dicho mecanismo se traduce en un sistema de intercambio entre:

- Dos partes interesadas: un inversor y una empresa que pone en el mercado valores.
- Una tercera parte de confianza, que interviene entre las partes interesadas.
- Un organismo auditor, que regula el funcionamiento del sistema.

Resumiendo, el presente Trabajo Fin de Máster tiene como objetivos:

- Establecer un breve marco teórico sobre los sistemas de intercambio de valores contra dinero.
- Analizar las características comunes de las tecnologías de libro de cuentas distribuido, y concluir con las más idóneas para una arquitectura de intercambio de valores.
- Estudiar en detalle los componentes y mecanismos de Hyperledger Fabric, y concluir con un diseño seguro y eficiente para la solución de intercambio de valores.
- Realizar un método de intercambio equilibrado y justo entre los actores involucrados.
- Elaborar una prueba de concepto sobre Hyperledger Fabric.
- Finalizar con las conclusiones del presente estudio.

1.2. Estructura del Trabajo Fin de Máster

Capítulo 1: Se indican los objetivos y la estructura del actual estudio.

Capítulo 2: Se desarrolla el marco teórico sobre métodos de intercambio de valores contra pago, y las características comunes de las tecnologías de libro distribuido.

Capítulo 3: Se estudian los componentes de una implementación concreta como es Hyperledger Fabric y se toman conclusiones de diseño y seguridad para una solución de intercambio de valores.

Capítulo 4: Se diseña un mecanismo equitativo de intercambio de valores y valor líquido.

Capítulo 5: Se detalla una prueba de concepto sobre Hyperledger Fabric.

Capítulo 6: Se finaliza con las conclusiones del actual documento.

Capítulo 7: Se indican las posibles líneas de trabajo futuras.

Capítulo 8 y 9: Se muestran las referencias nombradas durante el trabajo, y la relación de anexos necesarios para el entendimiento del mismo.

2. Marco Teórico

2.1. Infraestructura de pago

En este capítulo se introducirán los conceptos financieros necesarios para la comprensión del objetivo del presente TFM. Tal y como se comentaba en la introducción, la pretensión del presente trabajo es estudiar la aplicación de DLT en una aplicación financiera concreta, la entrega contra pago o "delivery versus payment", por lo que es necesario a continuación introducir una serie de conceptos financieros.

2.1.1. Target-2-Securites y Central Securities Depositories

Dentro de la infraestructura de la industria financiera, uno de los elementos clave es la capacidad de registrar y custodiar la propiedad de los diversos activos financieros (valores, fondos, bonos...), y sus posibles traspasos entre particulares u organizaciones, registrando que el activo pasa de las manos de un vendedor a un comprador. Los mercados de valores se crearon con el propósito de juntar a dos tipos de actores: los que tienen un capital para invertir (inversores) y los que quieren recibir prestado ese capital (empresas y organismos), con el sentido de crear un mecanismo de préstamo paralelo a los bancos. Esta sub-industria ha ido evolucionado históricamente con una complejidad creciente. Actualmente, en el continente europeo, esta información está en mano de los Depositarios Centrales de Valores (de ahora en adelante, DCV) nacionales. En el caso de España, el DCV autorizado es IBERCLEAR, el nombre comercial de la "Sociedad de Gestión de los Sistemas de Registro, Compensación y Liquidación de Valores, S.A. Unipersonal". En cada uno de los estados de la Unión Europea existe, al menos, un DCV autorizado.

Las funciones de este organismo pueden ser consultadas en el enlace [11]. De forma muy resumida, es un registro contable de operaciones de valores, permitiendo la transferencia de estos activos entre distintos custodios, siendo el enlace entre emisores de valores e inversores. Los servicios básicos que ofrece son, en primer lugar, ser un registro contable central de los valores negociables, representados por anotaciones en cuenta. En segundo lugar, gestiona la liquidación de los valores, esto es, se encarga del intercambio de valores y dinero efectivo. Para ello, ejecuta las transferencias de valores y las órdenes de pago asociadas.

De forma adicional, IBERCLEAR ofrece otros servicios auxiliares, que podrían centrarse en tres grandes grupos: los relativos al registro contable de valores, a los sistemas de liquidación, y al enlace con otros DCVs. En el primer grupo, se ofrecen servicios como el registro de los titulares de los valores, las emisiones de nuevos valores o la información y cobro de comisiones. En el segundo bloque, se encuentran los servicios de validación y mantenimiento de órdenes de transferencia entre valores y efectivo. Finalmente, se encarga de establecer enlaces como otros DCVs, siendo un DCV emisor o inversor. Un DCV emisor gestiona y refleja en su libro de cuentas los eventos financieros en nombre del emisor de los valores. Un DCV inversor se pone en contacto con otro DCV (el DCV emisor), abre una cuenta en él, y permite la liquidación cruzada entre los dos entes para operar con los valores. A modo de ejemplo, la ilustración 1 muestra los enlaces de IBERCLEAR con otros DCVs.

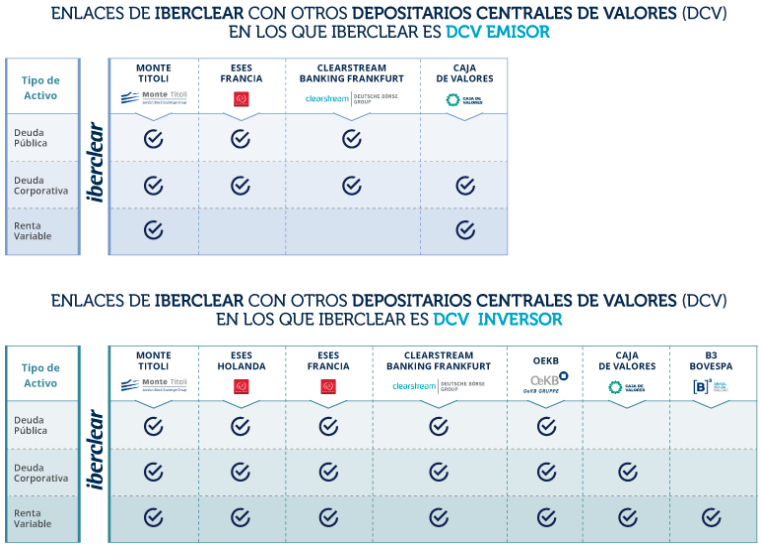

Ilustración 1 – Enlaces de Iberclear con otros DCVS, tomada de [25]

Una vez explicado el concepto de un DCV, nos centraremos en el ecosistema que los rodea, TARGET2-Securities (a partir de ahora, T2S). Se trata de una plataforma europea, impulsada por el Banco Central Europeo, y el entorno de servicios financieros propuesto por el Eurosistema, que es la autoridad monetaria de la zona euro. Su objetivo es ofrecer una liquidación centralizada de entrega-contra-pago o en inglés, "delivery versus payment" (DVP a partir de ahora) de valores, en dinero de banco central, en todos los mercados europeos. Con el término "dinero de banco central", expresamos que no se trata de dinero de bancos comerciales, sino que opera directamente con los bancos centrales de los diversos estados. Más adelante, explicaremos con más detalle el concepto de DVP. En consecuencia, con la plataforma T2S se ha conseguido eliminar las barreras entre los países de la Unión Europea a la hora de intercambiar valores y reforzar la idea de crear un mercado único. En dicho proyecto están involucrados todos los DCVs nacionales, mejorando su competitividad. La ejecución de T2S comenzó en 2008 y finalizó en septiembre de 2017, lo que da una idea de la magnitud, complejidad y entes implicados en el mismo. El listado de DCVs autorizados se puede consultar en [24].

Queda fuera del alcance de este estudio describir toda la infraestructura y el ecosistema de los DCVs.

2.1.2. Delivery Versus Payment (DVP)

Introducido en el apartado anterior, este concepto hace referencia a un procedimiento de liquidación de la industria de valores, y ofrecido por los DCVs. Se trata de un sistema de liquidación entre dos partes interesadas, y que estipula que la entrega del valor líquido (cash), ha de realizarse en el instante anterior o de forma simultánea a la entrega del valor (activo). Garantiza, que la entrega de los valores desde el titular al inversor, se produce solamente, si ocurre el pago.

Se trata de un procedimiento que surge como resultado de la prohibición por parte de las instituciones financieras de que la entrega del valor líquido se produjera antes de que el valor estuviera dispuesto a ser traspasado. Garantiza por lo tanto, que la entrega ocurrirá si existe un pago previo. El sistema, de forma resumida, actúa como un enlace entre un comprador y vendedor. A través de dicho enlace, el DCV actúa como un libro de cuentas en que se queda registrada la transacción.

Según la definición del DVP del Banco Central Europeo, disponible en [15], un proceso DVP tiene tres fases:

- Los valores son bloqueados en la cuenta del vendedor, para asegurar que se reservan para la entrega al inversor, y no están disponibles para otra operación.
- El valor líquido es deducido en la cuenta del inversor, y abonado en la cuenta del vendedor.
- Los valores bloqueados son deducidos de la cuenta del emisor, y son abonados en la cuenta del inversor, o si la transferencia de liquidez tiene algún problema, vuelven al emisor.

La ilustración 2 muestra la interacción entre los bancos, DCVs (en inglés, CSDs), y las cuentas de los implicados. Como indicamos, se trata de una abstracción, ya que existen muchas entidades que no aparecen en este gráficos (brokers, custodios intermedios, casa de compensación nacional…).

Ilustración 2 – Interacción entre bancos, DCVs y cuentas de usuarios, tomada de [14]

En el ejemplo anterior, participan los bancos A, B y C, que tienen cuentas en los DCVs 1 y 2 (en la figura aparecen como su nomenclatura en inglés, CSD).

Se expresan dos operaciones: La operación 1 refleja que el banco A compra 100 € en un valor del banco B, y al mismo tiempo, en la operación 2, vende al banco C 100 € en otro valor.

Existen dos tipos de cuentas, de dinero en efectivo (cash accounts), y de valores (securities accounts). Como se observa en el diagrama, de forma atómica, cada operación consta de dos pasos:

- el paso 1a, implica que se produce una transferencia de efectivo entre los bancos A y B.
- el paso 1b, expresa el asentamiento de la compra del valor en los registros del DCV-1.
- el paso 2a, se refiere a la transferencia de efectivo entre los bancos C y A.
- el paso 2b, hace referencia al asentamiento de la transferencia del valor en los registros del DCV-2.

Por claridad, se han eliminado las referencias a agentes en el ejemplo anterior.

Como se observa en el diagrama, la reserva de valor líquido (operaciones 1a y 2a) se producen de forma anterior a la transferencia de valores. Todo ello, queda registrado en los libros de cuentas de los DCVs.

2.1.3. Interés del BCE en tecnologías DLT

El Eurosistema y el Banco Central Europeo se han manifestado muy interesados en la tecnología DLT, como atestigua la creación de un grupo de trabajo denominado "Distributed Ledger Technologies Task Force" o DLT-TF [13]. El propósito de dicho grupo es analizar el impacto potencial de la tecnología DLT y emitir propuestas para futuras acciones e iniciativas de la comunidad T2S. Los miembros de dicho grupo son representantes de bancos comerciales y DCVs, y se pueden consultar en [26].

Por otro lado, en un informe tecnológico de 2016, disponible en [12], reconoció el posible papel de DLT dentro de la arquitectura T2S, pero manifestó que DLT es una tecnología muy reciente y habría que estudiar su madurez para servicios financieros, en términos de seguridad y eficiencia.

En diciembre de 2016, el Banco Central Europeo y su homólogo el Banco de Japón, anunciaron el proyecto Stella. Se trata de una iniciativa conjunta de investigación para evaluar la aplicabilidad de tecnología DLT en el área de las infraestructuras financieras. El resultado de dicho proyecto puede ser consultado en [16], y se centra en el estudio del rendimiento de DLT para una solución de alta demanda y en tiempo real, como puede ser la requerida por un entorno financiero como T2S, y en las posibilidades de DLT para ofrecer resiliencia al sistema. A modo de resumen, las conclusiones de dicho estudio fueron las siguientes:

- Las soluciones basadas en DLT pueden aportar el rendimiento requerido por un sistema liquidatario en tiempo real, como es el BCE y el Bank of Japan.
- El rendimiento de una DLT está influenciado por la distancia entre nodos y el tamaño de la red.
- Las soluciones DLT tienen el potencial para aportar soluciones que refuercen la resiliencia y la fiabilidad del sistema.

2.2. Distributed Ledger Technologies

2.2.1. Introducción

Distributed Ledger Technologies (DLT a partir de ahora) es el nuevo paradigma de almacenamiento y compartición de la información, situado por Garnett dentro de las 10 tendencias de estrategia tecnológicas para 2018 [1]. Antes de detallar las distintas características e implementaciones de esta tecnología, es necesario detallar los rasgos comunes a todas las redes DLT. Este término engloba sub-categorías como Blockchain, o implementaciones como Ethereum, Hyperledger o Bitcoin.

En su definición, DLT hace referencia a un libro de cuentas (ledger) distribuido. El concepto de ledger indica que se representan transacciones entre cuentas o usuarios del DLT. Esta información no se almacena de forma centralizada, sino distribuida y descentralizada entre los nodos de la red. Estos nodos son equipos o servidores independientes que pueden registrar, compartir, sincronizar y validar transacciones. Lo anterior permite que los datos se almacenen de forma:

- Independiente de un tercero de confianza que valide las transacciones, una autoridad central que regule el funcionamiento de la red y tenga mecanismos de control y/o censura.
- Resiliente, ya que al incorporar varios nodos se elimina la existencia de un único punto de fallo.
- Verificable y transparente, debido a la disponibilidad de la información.
- distribuida en cuanto a la gobernanza de la red, ya que los nodos necesitan alcanzar un consenso para validar los nuevos datos que van a formar parte de la DLT.

No obstante, DLT no tiene la necesidad de ser público o descentralizado. Los matices que se definirán en la implementación de la DLT puede convertirse en una red privada, centralizada o sometida a la validación de un organismo regulador, tal y como detallaremos más adelante.

La primera implementación de una DLT fue la plataforma Bitcoin, la cual se desarrolló en forma de una cadena de bloques o blockchain. Se tratan de las implementaciones más conocidas, pero existen otras que abandonan completamente la idea de la cadena de bloques como la red Tangle de IOTA o HashGraph. Frecuentemente, se emplean los términos DLT y blockchain como sinónimos. Esta confusión no debería causar una falta de entendimiento del lector, ya que ambos son conceptos y no implementaciones como Bitcoin, Ethereum o Hyperledger, siempre y cuando se tenga claro que una blockchain es un tipo de DLT, y existen DLTs que no son blockchains.

A continuación se detallarán características a tener en cuenta a la hora de elegir una implementación DLT para una aplicación determinada.

2.2.2. Mecanismos de consenso

La naturaleza distribuida de los nodos que componen una DLT requiere que se alcance un acuerdo entre ellos para definir las reglas que se emplearán para validar los nuevos datos que van a formar parte de la información contenida. Dicho acuerdo se denomina “mecanismo de consenso”, y puede ser ejecutado por todos los nodos de la red, o por solo algunos nodos. El objetivo del mismo es determinar si una transacción es legítima o no, empleando métodos de

validación criptográfica. También es necesario para evitar los conflictos que puedan surgir de las transacciones, por ejemplo, que un usuario quiera cometer fraude y emplee los fondos de una cuenta repetidas veces.

Antes de entrar a analizar los métodos de consenso, hay que explicar el "Problema de los generales bizantinos", como parte de la comparativa de dichos métodos. Se trata de un problema clásico de las redes distribuidas, y en general de cualquier conjunto de equipos informáticos que quieren llegar a un objetivo común. Dicho problema se ejemplifica a través de una situación en la que un ejército se encuentra asediando una ciudad. Sus tropas están organizadas en torno a generales. Los generales se comunican entre ellos a través de mensajes, cuyo contenido puede ser "atacar" o "retirarse". Los generales pueden ser tenientes o comandantes, que es una figura única que dicta la orden. Los tenientes reciben la orden y la emiten a los demás. Cualquiera, tanto el comandante como los tenientes, pueden cometer fraude y ser traidores al ejército. En la ilustración 3 se observa el paso de mensajes entre el comandante, y los tenientes, siendo el teniente número 3 un general traidor.

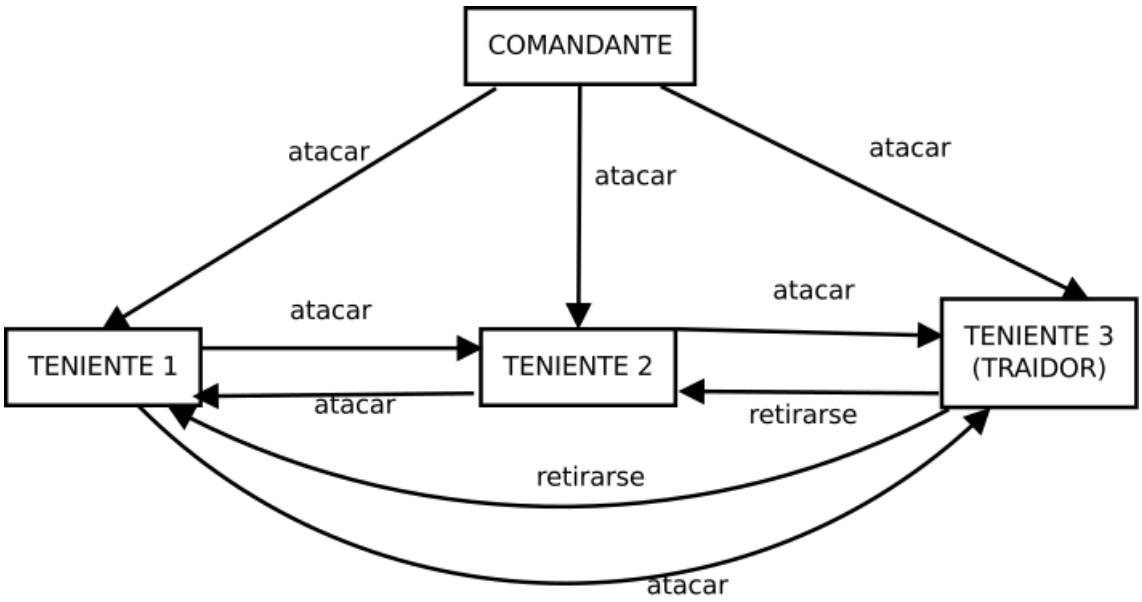

Ilustración 3 –Problema de los generales bizantinos, tomada de [43]

Para solucionar este problema, los tenientes escucharán los mensajes de los demás, y tomarán la decisión mayoritaria, es decir, la que envíen el mayor número de generales. Esta solución no será válida si el número de generales traidores es superior a la mitad (ataque del 51%).

Se indican a continuación varios sistemas de consenso empleados o propuestos teóricamente para sistemas distribuidos.

2.2.2.1. Prueba de trabajo

Este método de consenso también es conocido por su denominación “Proof of Work”. Es un mecanismo para alcanzar el consenso en una DLT de forma distribuida, en el que no se requiere la confianza previa entre nodos, ni en una tercera parte. Se basa en la fuerza de computación de los nodos que participan en la red. Se propone un reto a todos los nodos, en forma de operación criptográfica. El nodo que primero consiga solucionar dicho reto, agrega un bloque con todas las transacciones pendientes y se lleva una determinada recompensa. De esta forma, los elementos de la DLT compiten en recursos por ser los primeros en realizar la operación criptográfica establecida. Cuando lo consigue, incorpora los nuevos datos al ledger e informa al resto. Una implementación de este tipo de consenso existe en Bitcoin. En dicha red el ledger es una cadena de bloques. Se parte de conjunto de transacciones, y el reto consigue en agregar una cadena variable (llamada “nonce”) a ese conjunto, para que el hash del valor total sea una cadena que comience por un número determinado de ceros. El nodo que consiga ese reto, se lo comunica a los demás, consigue un premio por ese trabajo, y agrega el bloque a la cadena de bloques. Dicho bloque contendrá un hash del bloque anterior. En la ilustración 4 se observa un ejemplo de la cadena de bloques de Bitcoin. Cada uno de los bloques contiene el llamado árbol de Merkle de las transacciones. Explicaremos este concepto sin entrar en demasiados detalles. Cada transacción tiene un hash asociado, y el árbol de Merkle es una representación de los hashes de las transacciones, incluyendo un hash final. Dicho árbol está presente en la cabecera de un bloque. Así, es posible verificar si una transacción ha sido aceptada por la red descargando las cabeceras de los bloques y los árboles de Merkle.

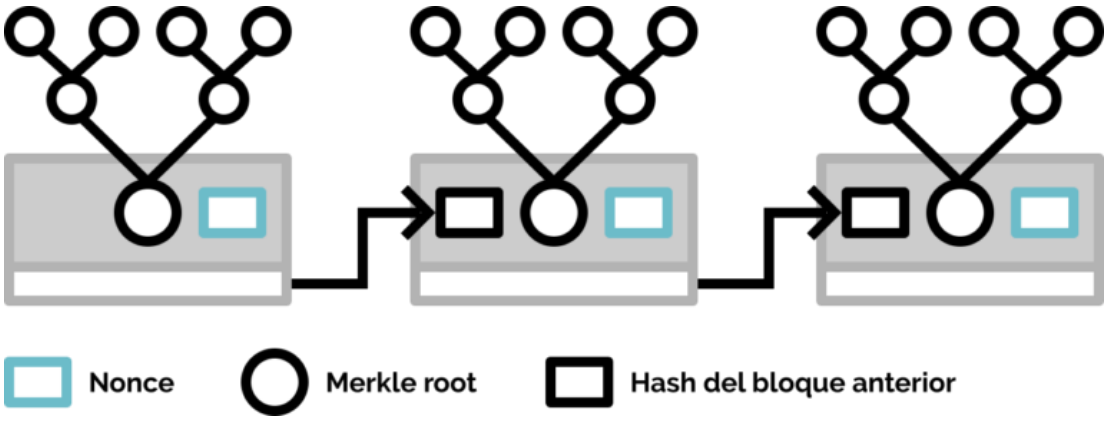

Ilustración 4 –cadena de bloques en Bitcoin, tomada de [44]

Este modelo de consenso es tolerante al problemas de “los generales bizantinos”, que aplicado en una red DLT, significa que es capaz de soportar que los nodos quieran cometer un fraude e informen al resto de una información errónea con respecto a una transacción. Si un nodo desea cometer fraude, incorporará un bloque incorrecto a la cadena de bloques, y la dará por buena. Se creará una bifurcación de la cadena, una con contenido correcto, y otra con incorrecto. Si los nodos honestos mantienen la mayor parte de la capacidad de proceso de la red, la cadena que tiene el contenido correcto crecerá más rápido y enseguida se alcanzará el consenso sobre la misma. Se puede obtener más información de este problema en los enlaces [3] y [4].

En las implementaciones de este consenso, como en Bitcoin, se requiere que:

- Todos los nodos se comuniquen entre sí.
- Todos los mensajes estén firmados para dotar de autenticidad, integridad y no repudio.
- Todos los mensajes tienen un sello de tiempo para evitar duplicidad de información.

Este sistema tiene varias desventajas implícitas. Una de ellas es que se malgastan muchísimos recursos, ya que todos los nodos tienen que realizar una computación criptográfica costosa, y sólo el trabajo de uno de los nodos será útil, en el sentido que agregará el resultado de su trabajo a la red. Los recursos energéticos empleados son enormes, en [5] se puede comprobar que actualmente se están empleando cerca de 73 Teravatios-hora al año para el mantenimiento de la red Bitcoin.

Este sistema es sensible al ataque del 51%. Si se consigue controlar al 51% de los nodos, conseguirían gobernar el funcionamiento del ledger ya que poseen la potencia computacional mayoritaria. En el caso de Bitcoin, tendrían la cadena más larga de bloques y podrían controlar las transacciones. Este ataque requiere de tal coste computacional que no es factible.

Además, en el caso de redes basadas en cadenas de bloques, requiere que un bloque se confirme un número determinado de veces para dar la información como válida y marcar el bloque como final, lo que ralentiza su funcionamiento.

Por otro lado, favorece a los nodos más potentes, ya que serán los que tiene más recursos. Actualmente, 7 pools de nodos (grupos de nodos cooperadores) se reparten más del 75% del minado de Bitcoin. [2]. Esto es una contradicción al objetivo de la plataforma de que se trata de un conjunto de nodos con gobernanza descentralizada.

2.2.2.2. Prueba de participación

También conocido como “Proof of Stake”, este mecanismo de consenso se basa en que el nodo que tenga más participaciones en la red distribuida, tendrá más probabilidades de incorporar la nueva información a la red. Se razona en que tiene más intereses en que el funcionamiento de la red sea correcta. En este caso, el premio será inferior que en el mecanismo “prueba de trabajo”, o nulo. De hecho, en este método los nodos que colaboran no se llaman “mineros” sino “forjadores”. Los nodos candidatos deberán demostrar la posesión de sus participaciones en la red.

Por lo tanto, el nodo que tenga más participaciones en la red (o monedas en caso de una red de criptomonedas), será el que agregue una nueva información al ledger. Se congelarán por un periodo determinado sus participaciones (monedas), para que, en caso de que haya generado un bloque inválido cometiendo fraude, se le aplicarán multas.

Existen diversas modalidades para elegir al nodo que agrega el bloque a la red. Un método es elegir el nodo que más participaciones tenga, como se ha explicado, siguiendo un algoritmo de prueba de participación pura. Otro método, como el empleado en la red de la criptomoneda Peercoin, es el llamado “Proof-of-coin-age”, en el que se tiene en cuenta tanto el número de participaciones como el tiempo que el nodo las posee.

Comparativamente con “proof-of-work”, este método es más rápido, ya que no requiere de hallar una solución criptográfica, y la latencia de la DLT será inferior.

Este sistema también tiene sus desventajas. En primer lugar, los sistemas basados en consenso "Prueba de participación", los nodos no pueden alcanzar un consenso si la cadena se bifurca como resultado de que un nodo cometa fraude. Como agregar bloques en las cadenas no supone una prueba de esfuerzo, los nodos continuarán haciéndolo en las dos cadenas. Por lo tanto, si la red no provee de mecanismos adicionales, es computacionalmente más sencillo cometer fraude, frente a la posibilidad del mismo en "proof-of-work". En las implementaciones de este método de consenso se emplea un mecanismo de seguridad como es el bloqueo de las participaciones de dicho nodo en la red, para su decomiso en caso de operaciones fraudulentas.

Por otra parte, al no tener incentivos para la validación de la información, no tiene el mecanismo de seguridad que proporciona el hecho de que varios nodos se encuentren pugnando por la recompensa mediante la validación de los datos.

En esta red se reduce el riesgo del ataque del 51%, ya que requiere que el atacante tenga que hacerse con el 51% de las participaciones o tokens. En cambio, si lo consiguiera, ya podría controlar las transacciones si la plataforma no posee más mecanismos de seguridad.

La primera plataforma que incorporó este mecanismo fue Peercoin en 2012.

Ethereum se ha mostrado muy preocupado muy el consumo energético del mecanismo "proof-of-work" y está desarrollando un hard-fork para implementar el mecanismo proof-of-stake mediante el protocolo Casper [6]. En dicho protocolo, los nodos se incorporan a un pool de forgers, y reciben una pequeña recompensa por validar las transacciones. Para incorporarse al pool deberán enviar un smart contract junto con una cantidad determinada de Ether (la criptomoneda de Ethereum). El mecanismo para evitar comportamientos fraudulentos consistirá en que Casper es capaz de hacer que dicho nodo pierda su depósito.

2.2.2.3. Mecanismos de consenso mediante firmas múltiples

Otra forma de que un sistema distribuido alcance un consenso para agregar nuevos datos es mediante un esquema de firmas múltiples. Si consideramos que existen un conjunto de nodos como candidatos a ser validadores de la información, hay que conseguir la firma de un subconjunto de esos nodos. Para que sea pueda ser tolerante a la actuación de varios nodos fraudulentos, el número de firmas necesarias ha de ser superior a la mitad del número total de nodos. En el ejemplo de una blockchain, siguiendo este método, se conseguirá un consenso si el último bloque a añadir contiene una firma múltiple de la mayoría de los nodos. Este mecanismo requiere que existe una infraestructura de clave pública detrás de los nodos, lo suficientemente confiable para dotar de seguridad a la plataforma. Este hecho, a su vez, difiere parcialmente de uno de los objetivos de una DLT: eliminar la autoridad central de confianza.

La ventaja de este método es que no tiene prácticamente latencia, en especial si se compara con el método "proof-of-work".

2.2.2.4. Federated Byzantine Agreement

El origen de los mecanismos de consenso basados en acuerdos bizantinos (byzantine agreement), es un simple acuerdo entre la mayoría de nodos para lograr un consenso frente a

actores maliciosos. Es un sistema limitado a un número de nodos inicial y cerrado (permissioned).

El acuerdo federado Bizantino (o FBA, por sus siglas en inglés) es uno de los mecanismos que aparecieron en los orígenes de Blockchain. Los nodos validan bloques de información cuando llegan a un consenso, alcanzado cuando un mínimo de nodos consideran que la información del bloque es correcta. El protocolo determina cual es el quorum de nodos necesarios para validar la información.

Cada participante elige confiar en una serie de nodos, forman un círculo de confianza (o federación, de ahí el nombre del mecanismo) dentro del cual se alcanza de forma sencilla en consenso. En el momento en el que un nodo de un círculo confíe en otro nodo de otro círculo, se irá expandiendo el consenso en la red. Por lo tanto, es necesario que los círculos de confianza tengan nodos en común y no sean conjuntos disjuntos. Si los círculos son disjuntos, se podría llegar a que dos islas de la misma no logren un acuerdo, dando lugar a un bloqueo.

Los nodos pueden unirse a la red de forma abierta (permissionless) y el control es descentralizado. Cada nodo elige en qué nodos confía, y los distintos nodos no tienen que poseer la misma combinación de nodos de confianza.

Se emplea en la blockchain Ripple, y en Stellar desarrollaron este algoritmo, al definir un protocolo FBA seguro, el llamado Stellar Consensus Protocol (SCP). Este protocolo implementa un sistema de votación entre los nodos, en el que los miembros de un quorum votan para aceptar una afirmación. Este sistema de votación permite que el protocolo SCP minimice la posibilidad de que existan bloqueos de acuerdos entre nodos. Se puede obtener más información de este protocolo en [8].

2.2.2.5. Practical Byzantine Fault Tolerance

También conocido por sus siglas PBFT, se trata de otro mecanismo para facilitar el consenso de los nodos y evitar que nodos maliciosos quieran propagar información incorrecta. Se deriva del método anterior. De forma circular, uno de los nodos es elegido como líder y el resto como nodos de respaldo. Supone que funcionará correctamente si los nodos maliciosos no superan el 33% de los nodos.

Cada nodo implementa una máquina de estados para desarrollar el algoritmo de consenso en 4 pasos:

- Se inicia cuando un cliente solicita una petición al nodo líder.
- Se reenvía la petición a los nodos de respaldo.
- Los nodos ejecutan la petición y envían la respuesta al cliente.
- Cuando el nodo haya recibido un número determinado de respuestas con el mismo resultado, dará por válido el consenso.

Todos los mensajes entre los nodos contienen mecanismos de autenticidad e integridad. En este método de consenso, por lo tanto, es crítica la autoridad que proporciona la identidad de los nodos.

La ventaja de este mecanismo es que es más rápido que los anteriores. Puede ser el método de consenso para un sistema que responde a transacciones que requieren una respuesta

inmediata, frente a los modelos de prueba de trabajo como Bitcoin, que tienen que esperar a que se confirme la transacción. En PBFT, si un bloque ha sido confirmado por una serie de nodos, es final.

Una vez que ha llegado a un consenso, el bloque resultante es elegido como final. Además, el consumo de energía es sustancialmente inferior, como se ha explicado anteriormente con otros mecanismos.

El problema que presenta este protocolo, es que debido a la cantidad de mensajes que se reenvían, no es válido para redes con una gran cantidad de nodos. Los mensajes presentan firmas digitales y códigos de integridad, que sería ineficiente en una red grande.

Para más información de este método, se puede consultar la referencia [9].

2.2.2.6. Prueba de importancia

Este tipo de consenso es un subconjunto de la prueba de partipación o "proof-of-stake". No solo tiene en cuenta el peso o importancia de cada uno de los nodos, sino también el comportamiento entre ellos, ya sea por cantidad o magnitud de las transacciones. Si fuera el caso de una red cuyos activo son criptomonedas, estaríamos hablando que que la importancia de una cuenta se mediría en función del número de monedas que posee, las transacciones que ha realizado y la cantidad invertida en las mismas.

Un ejemplo de aplicación de este consenso sería la red "NEM", o blockchain de activos inteligentes. Después de que un bloque se incorpore a la cadena, la "prueba de importancia" de un usuario determina quién realmente cosechó un bloque basado en factores como el tiempo en que las monedas han estado en la cuenta del usuario, si los usuarios realizan de forma activa transacciones, o el número de transacciones en los últimos días.

Se puede consultar más información de esta red en [42].

2.2.2.7. Delegated Harvesting

Se trata de un mecanismo para minar bloques de forma delegada o remota. La forma tradicional de minado, sería el minado "local", ya que requiere que un nodo se encuentre activado. Una forma alternativa sería que un servidor remoto realizara el minado por nosotros, pero tendría que poseer nuestra clave privada para realizar la firma de los bloques minados. El sistema Delegated Harvesting permite que nodos remotos minen por nosotros, mediante la autorización remota empleando un tipo de transacción y una aportacíon económica. Mediante esta transacción se delega el minado en el nodo remoto, sin necesidad de transferir la clave de nuestro nodo. Este sistema también se emplea en la blockchain NEM.

2.2.3. Métodos de participación en una DLT

Otra de las opciones a tener en cuenta a la hora de diseñar una DLT o elegir una implementación existente es la forma en la que los nodos, o potenciales usuarios (incluyendo ciudadanos u organizaciones) pueden participar en ella. Esta elección tiene sus consecuencias, ya que si no existe restricción en su participación, las medidas de seguridad a implantar mediante mecanismos de consenso y otras obligaciones serán más estrictas que si la DLT es privada y están identificados sus componentes (lo que a su vez conlleva que sean necesarios mecanismos de identificación).

Para comenzar, se establece una clarificación de los términos empleados para definir una DLT, utilizando la taxonomía definida por BitFury Group en [7]:

- Público: Si no existen restricciones en la lectura ni en el envío de transacciones para agregar al ledger.
- Privada: Si el acceso a la información y el envío de transacciones está limitado a una lista de entidades predefinidas.
- Sin permiso (permissionless): No hay restricciones en los nodos que pueden procesar transacciones.
- Con Permiso (permissioned): El procesamiento de las transacciones es realizado por una lista nodos previamente identificados.

Según esta terminología, Bitcoin sería pública y permissionless.

Así, una permissioned DLT puede ser pública o privada. Más allá, se pueden definir varios niveles de acceso a una DLT:

- Rol de lectura de la información de la DLT con restricciones (por ejemplo, solo las transacciones que incumben al sujeto).
- Rol para leer cualquier transacción de una DLT.
- Rol para enviar transacciones a la DLT.
- Rol para validar transacciones y agregarlas al ledger.

Incluso se podrían definir modelos en transición. Por ejemplo. Una DLT sería semi-privada, en el sentido de que se podría conceder el rol de lectura / envío de transacciones a cualquier individuo que presente una identificación digital reconocida.

2.2.4. Smart contracts

El término “smart contract” hace referencia a fragmentos de código corriendo en una arquitectura DLT, en las que sus variables son los activos que se representan en la DLT y su contenido representa la lógica de negocio. No hay que confundirlos con “contratos legales”, ya que, aunque pueden representar la ejecución de un cumplimiento legal, no tiene validez jurídica completa. Ya aparecerían de forma muy limitada en la red Bitcoin.

Si todas las condiciones de un smart contract se cumplen, el smart contract auto-ejecutará la pieza de código que lleva implícita en forma de transacción. Su ejecución puede disparar una transacción de un origen a un destino, o el cambio de propiedad de un activo financiero. Su función es facilitar, verificar y cumplir la lógica de un acuerdo previo, con la

seguridad, transparencia y trazabilidad del ecosistema de una DLT. Este entorno les proporciona la capacidad de ser auto-verificables, auto-ejecutables y resistentes a la modificación. Además, este ecosistema permite disminuir los costes de transacción, evitar tener que confiar en terceros, y dotar de un grado de seguridad potencialmente superior al acuerdo entre pares fuera de una DLT.

Una característica a destacar dentro de los smart contracts es la capacidad que la DLT y el lenguaje de desarrollo le proporciona para ser "Turing Complete" o no. Con dicho término, derivado de la máquina de Alan Turing, se hace referencia a la capacidad de un lenguaje para ser completo, incluir la sintaxis necesaria y tener la potencia como para desarrollar cualquier problema (sin contar la falta de recursos en el sistema como memoria o procesador). A efectos prácticos, el lenguaje de desarrollo que la DLT provee a los smart contracts debe ser capaz de tener la habilidad de realizar bucles y condicionales, almacenar cierta información en local o acceder a información externas. Por ejemplo, se establece que el lenguaje proporcionado por Ethereum (Solidity) es Turing Complete.

Particularizando para un entorno financiero como el del problema actual, en el que la ejecución de los smart contracts se puede dar en diversos nodos, tenemos la necesidad de que la ejecución de los smart contracts sea determinista. Es decir, que la ejecución de un smart contract en cada uno de los nodos devuelva el mismo resultado, con el fin de evitar fallos en los mecanismos de consenso. Para ello, a la hora de implementar los smart contracts hay que tener en cuenta:

- Que no accedan a variables aleatorias.
- Que no accedan a valores Timestamp generados durante la ejecución.
- Que si invocan a algún API externo (por ejemplo, para obtener el cambio de una criptomoneda a otra), se asegura la no variabilidad de la información (por ejemplo, tasa de cambio en el cierre del día anterior).

2.2.5. Oracle

Es posible que, durante la ejecución de un smart contract, se requiera la obtención de un dato accediendo a una fuente externa, como podría ser la tasa de conversión entre criptomonedas. Como se ha indicado anteriormente, invocar a una URL externa no es una solución válida para una solución determinista. La solución a este problema la proporcionan los llamados "oracles". Se tratan de servicios de terceros que proporcionan un medio para interactuar entre Internet y un smart contract, ya sea a través de otro smart contract u otro medio.

Un ejemplo de estos servicios es Oraclize, cuya descripción se detalla en la referencia [18]. Se trata del servicio de oracle para smart contracts más utilizado, y se emplea en redes como Ethereum, Hyperledger o R3 Corda. La ilustración 5 muestra la funcionalidad de Oraclize.

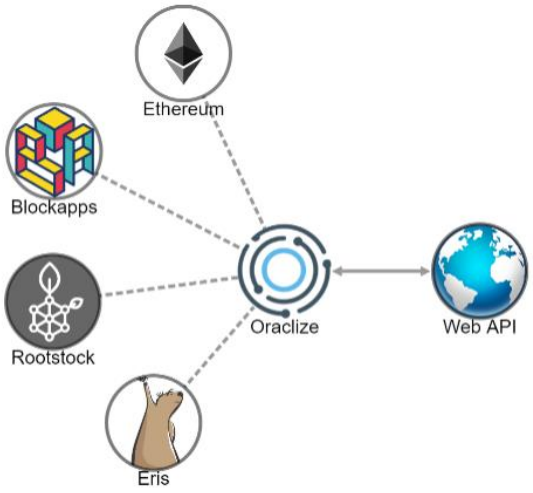

Ilustración 5 –Funcionalidad de Oraclize, tomada de [45]

En la documentación de la referencia [18] se ha obtenido el siguiente ejemplo de oracle, desarrollado en Solidity, el lenguaje de los smart contracts de Ethereum. Se trata de un método para hallar el precio del diésel en dólares. Como se puede observa, se trata de un código smart contract, en el que se define un método denominado update(), que accede a una URL externa alojada en http://www.fueleconomy.gov, para obtener el precio actual del diésel. Dicho método es invocado desde la función DieselPrice() y callback(), que actualizan el precio de la variable local.

```
    contract DieselPrice is usingOraclize {

        uint public DieselPriceUSD;

    event newOraclizeQuery(string description);
    event newDieselPrice(string price);

    function DieselPrice() {
        update(); // first check at contract creation
    }
    function __callback(bytes32 myid, string result) {
        if (msg.sender != oraclize_cbAddress()) throw;
        newDieselPrice(result);
        DieselPriceUSD = parseInt(result, 2); // let's save it as $
cents
        // do something with the USD Diesel price
    }

        function update() payable {
        newOraclizeQuery("Oraclize query was sent, standing by for
the answer..");
        oraclize_query("URL",
"xml(https://www.fueleconomy.gov/ws/rest/fuelprices).fuelPrices.dies
el");
    }
}
```

Como se puede observar en [19] existe la posibilidad de obtener este servicio en otras implementaciones DLT como Hyperledger Fabric.

Además, ofrece servicios adicionales de seguridad. Cabría destacar el apoyo en TLSNotary para firmar la respuesta proporcionada, detallado en [36], o la posibilidad de invocar a diversos oracles de distintos proveedores, para así obtener firmas múltiples y poder verificar que un quorum de n-de-m oracles alcanza un consenso en la respuesta.

2.2.6. Comunicación entre ledgers

Los distintos ledgers no son entes aislados, sino que requieren de la capacidad de comunicarse entre ellos. En el entorno de sistemas financieros, los distintos ledgers que pueden representar activos financieros o cash (criptomonedas), han de comunicarse para realizar operaciones de traspaso de valores como en el procedimiento "delivery-versus-payment". En dicho proceso, es necesario bloquear los activos líquidos para realizar el traspaso de un valor financiero.

El proyecto The Interledger tiene como objetivo proponer una solución a este problema. Es un proyecto open-source iniciado por desarrolladores de Ripple y gestionados por el grupo de estándares W3C, en su grupo "W3C Interledger Community Group", que han desarrollado el protocolo The Interledger Protocol disponible en [22]. Es tal el grado de estandarización al que quieren llegar que han desarrollado una RFC aprobada por el IETF y disponible en [23]. Está orientado a procesos de pago. Este protocolo se presenta como una alternativa al protocolo IP en la comunicación entre ledgers, situándose entre los niveles de transporte y enlace, en una nueva torre de protocolos cuyo nivel físico serían los ledgers. Se puede observar en la ilustración 6.

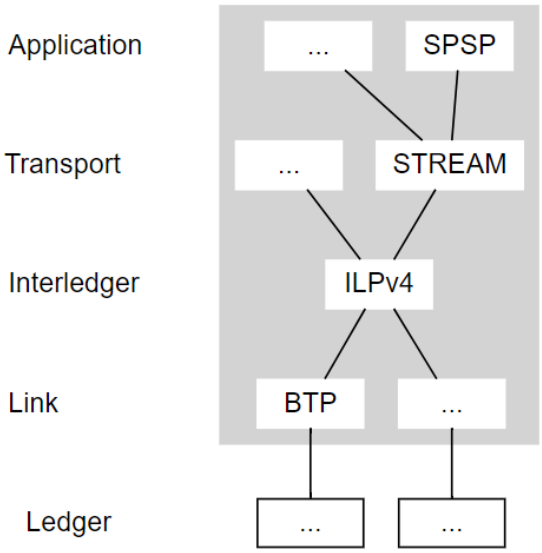

Ilustración 6 –Torre de protocolos de The Interledger Protocol, tomada de [46]

El funcionamiento básico de este protocolo es establecer una vía de comunicación entre varios ledgers, mediante la definición y creación de un módulo presente en todos ellos. La ilustración 7 muestra el modelo de operación de este protocolo.

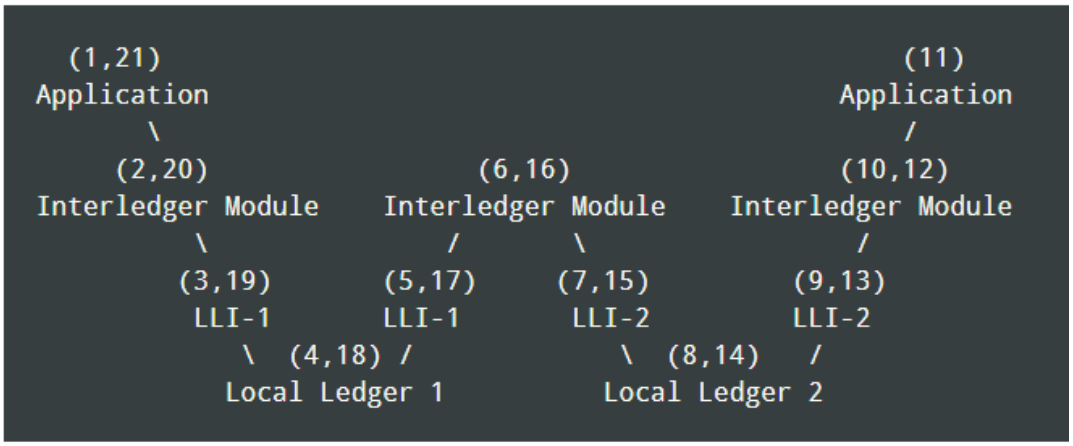

Ilustración 7 –Modelo de operación de The Interledger Protocol, tomada de [47]

Como se observa en el modelo, se quiere establecer una comunicación entre aplicaciones presentes en distintos ledgers (local ledger 1 y local ledger 2). Para ello, emplean el protocolo Interledger. Este protocolo accede a interfaces de cada uno de los ledgers (LLI-1 y LLI-2 significan local ledger interface del ledger 1 y del ledger 2, respectivamente). El procotolo pone en comunicación los interfaces de ambos ledgers.

3. Diseño de la solución DLT

A continuación se desarrollará el diseño de la solución DLT que presenta este trabajo. Como se ha indicado anteriormente, el objetivo es diseñar las características de un mecanismo DVP. A lo largo de la exposición se irán analizando los actores y las funciones de cada uno de ellos. Se tendrán en cuenta cuatro partes interesadas en el funcionamiento del intercambio de valores:

- Banco: Se encarga de gestionar las cuentas de los clientes.
- Mercado de valores: Su función es poner a disposición de los inversores valores de empresas, y gestionar su propiedad.
- Regulador: Verifica que el comportamiento de las actuaciones de los banco y del mercado de valores es correcto.
- Depositario: Se encarga de aplicar de forma efectiva el mecanismo DVP.

El mecanismo DVP se convierte en un sistema de intercambio entre:

- Una parte interesada: la organización Banco, representando a su cliente.
- Otra parte interesada: la organización mercadoDeValores, representando a una empresa que pone en el mercado valores.
- Una tercera parte de confianza: la organización depositario.
- Un organismo auditor: la organización Regulador.

En las siguientes secciones analizaremos de forma conjunta los parámetros de una arquitectura DLT estudiados en el apartado anterior, para concluir con la elección de una tecnología. Finalmente, se estudiará en detalle los componentes de la tecnología elegida y se tomarán conclusiones con respecto al diseño de una solución DVP.

3.1. Análisis de los parámetros de una DLT

En el marco teórico se analizaron las características de una solución DLT, en cuanto a consenso, métodos de participación y naturaleza de la privacidad. Como conclusión de las características, podemos inferir las propiedades que se han de exigir para cada tipo de DLT, expresadas en el cuadrante de la ilustración 8.

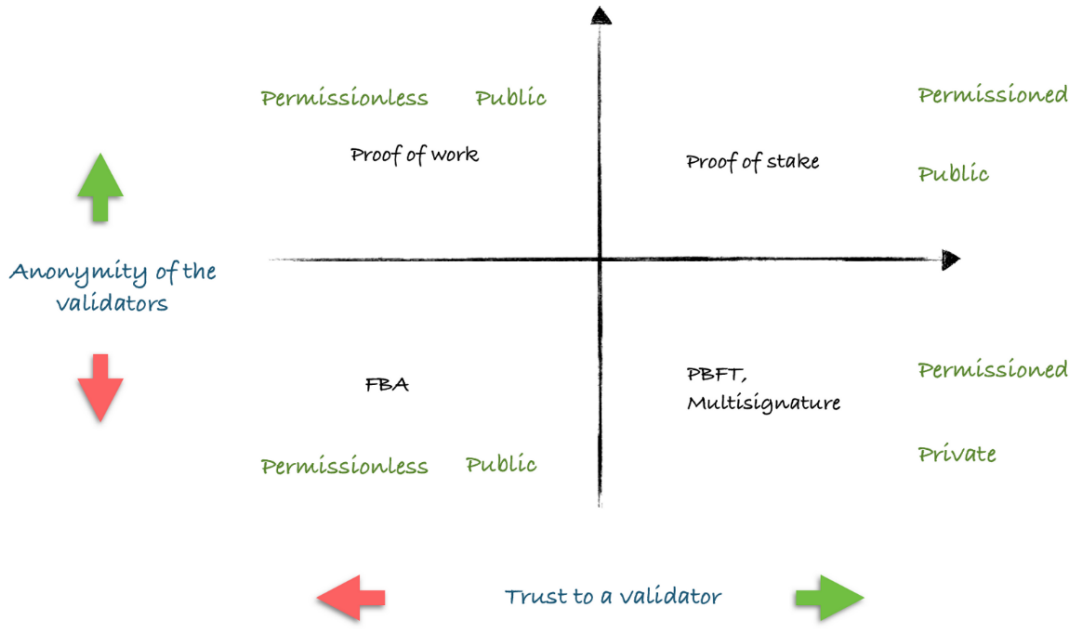

Ilustración 8 – Relación entre el anonimato y la confianza de los validadores, tomada de [10]

Analizando los cuadrantes anteriores, se puede sacar las siguientes conclusiones:

Primer cuadrante: Se sitúa arriba a la izquierda y se trata de DLT públicas y sin permiso. El anonimato de los miembros es total y no existen requisitos para formar parte de la red (solo recursos para minar). El mecanismo de consenso para este tipo de redes sería "proof-of-work". No existe confianza en los nodos que validan la información, y además no hay castigo para los nodos que intentan cometer un fraude. Con este mecanismo de consenso, la escalabilidad de la red es baja, pero la protección de la integridad de la información es muy alta. Es la configuración perfecta para redes anónimas, como Bitcoin o Ethereum, donde no se requiere ningún control de organismos supervisores ni estatales,.

Segundo cuadrante. Se sitúa arriba y a la derecha. Son DLTs públicas y con permiso. El mecanismo de consenso a emplear sería "proof-of-stake". Aumenta la confianza en los validadores, ya que se les podrían decomisar los depósitos en caso de actuar de forma fraudulenta, como hemos indicado anteriormente. La criptomoneda es inherente al sistema, no un resultado de una operación de minado como en el cuadrante anterior. Son propuestas válidas para ejecución de contratos, o sistemas financieros privados. Un ejemplo sería la blockchain de la criptomoneda Bitshares, o Ethereum tras la aplicación del hard-fork Casper.

Tercer cuadrante. Son DLTs públicas, con permiso tras una fase de autenticación pero sin una lista cerrada de usuarios, y se sitúa abajo y a la izquierda. En este caso, se tienen en cuenta los casos en los que los participantes pueden formar parte de la DLT si superan una fase de identificación, como por ejemplo una identificación digital reconocida. La confianza en los validadores es baja, ya que aunque estén identificados, no hay mecanismos de decomiso, por lo que el castigo por cometer fraude no es muy elevado. El mecanismo de consenso válido sería el acuerdo federado bizantino. El algoritmo "proof-of-work" ya no es necesario al tener la identidad de los participantes. No sería válido "proof-of-stake" tampoco ya que es posible que los participantes tengan igual participación en el sistema. ES válido para blockchains nacionales o de consorcios de empresas.

La protección de la integridad de los datos y la escalabilidad de las redes, son moderadas en estos dos cuadrantes, en comparación con el primer y cuarto cuadrante.

Cuarto cuadrante. Se tratan de DLTs con permiso, es decir, que los participantes han obtenido una licencia o forman parte de una lista reducida, y privadas. Es aplicable para organismos privados, como puede ser un banco o una infraestructura de pagos, al tratarse de una DLT privada donde la información no es pública. Las transacciones son validadas por un único nodo centralizado, por lo que son aplicables los mecanismos de consenso "Practical Byzantine Fault Tolerance". El método de firmas múltiples también sería válido, y se podrían dar varios esquemas de firma, incluyendo un quorum de un elemento (nodo validador), o requerir la firma de los interesados en las transacciones y el nodo validador.

Particularizando para el caso de una industria financiera, en la que existe una confianza mutua entre bancos y demás componentes, con acuerdos regulatorios y una legalidad que los soporta para evitar riesgos, y organismos que velan por la seguridad y el cumplimiento, la opción más correcta es una permissioned ledger. Las características de una red pública y permissionless (anonimato de los participantes, falta de un organismo central con capacidad de censura, copias públicas del ledger accesible, irreversibilidad de las transacciones), no son relevantes para el sistema financiero. En cambio, para esta industria, según se indica en [17], las normas del juego serían las siguientes:

- Necesidad de conocer de forma unívoca y fidedigna al cliente.
- Transparencia y responsabilidad frente a reguladores.
- Cumplimiento de las normativas.
- Confidencialidad de estrategias de negociación.

Lo que se puede mantener en una DLT permissioned y privada.

En cuanto al método de consenso a alcanzar entre los nodos, como ya se ha indicado anteriormente, no serían válidos los métodos "proof-of-stake" o "proof-of-work".

Cualquiera de los métodos que exigen un quorum, "Federated Signature Agreement" o firmas múltiples serían válidos, así como en el que existe un solo nodo líder "Practical Byzantine Fault Tolerance", ya que existe una confianza entre los nodos y son responsables del buen funcionamiento del sistema.

3.2. Justificación del empleo de DLT en entornos financieros de intercambio de valores

Para comenzar la justificación del empleo de una solución DLT en las transacciones de valores, es preciso introducir la motivación del interés del propio Banco Central Europeo en esta tecnología. Según la referencia [17] que analiza la posible aplicación de esta tecnología, se afirma que las medidas armonizadoras de TARGET2-Securities han conseguido grandes avances a nivel de DCVs, pero poco ha cambiado en otros niveles del mercado (custodios, casas de compensación, agentes de liquidación, repositorios de intercambio de valores). En la ilustración 9, se puede observar la diversidad y cantidad de elementos involucrados.

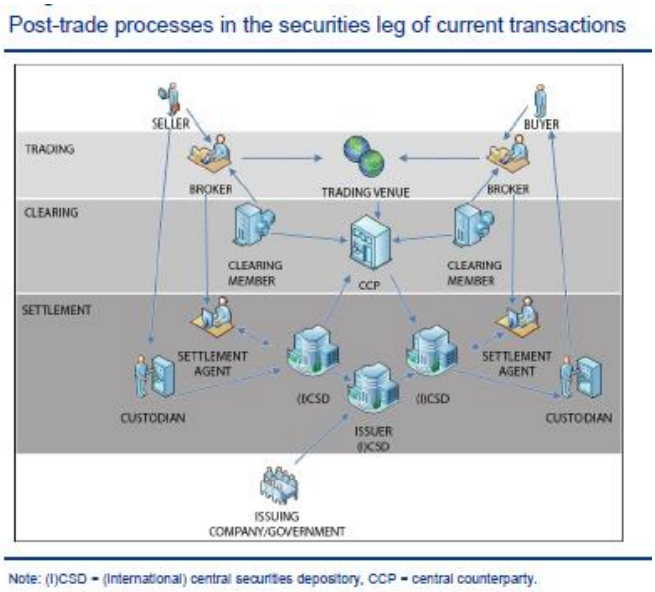

Ilustración 9 – Actores involucrados en Target2-Securities, tomada de [17]

De forma resumida, a continuación se explicarán los conceptos que aparecen en el gráfico anterior. En primer lugar, hay que indicar que cada una de las etapas horizontales representan las fases por la que pasa un valor durante una transferencia de propietario: trading (negociación), clearing (intercambio) y settlement (registro).

Los distintos actores que intervienen son seller (vendedor), broker (intermediario), buyer (comprador), clearing member (casa de intercambio), custodian (custodio del valor), issuing Company (compañía que emite un valor), CSD (depositarios centrales de valores), trading venue (plataforma de negociación), settlement agent (agente que realiza la liquidación), y ccp (central counterparty, una entidad que se encarga de finalizar los contractos y actúa de vendedor para el comprador, y de comprador para el vendedor).

Como se observa, la infraestructura actual que soporta el procedimiento del DVP requiere de intermediarios e instituciones, lo que resulta en un proceso que conlleva retrasos y gastos asociados. Un primer papel de la tecnología DLT sería el ahorro de gastos y de elementos necesarios para la transferencia de valores. En el apartado XXX analizaremos si es posible esta afirmación.

Por otro lado, y tal como se afirma en [17], tras el esfuerzo de la plataforma T2S, sigue existiendo una falta de interoperabilidad entre bases de datos propietarias, lo que contribuye al uso de registros digitales de propiedad de valores aislados. La sincronización entre registros análogos requiere de intervenciones manuales. La tecnología DLT, afirma este documento, tendría el potencial de unificar la representación de valores y demás activos financieros, y mantener trazabilidad de la ejecución y liquidación de traspasos, sin necesidad de mantener una base de datos centralizada. Frente a las bases de datos distribuidas, que requieren que cualquier actualización se envíe a la base de datos central, para su validación y replicación en las copias distribuidas, la tecnología DLT los usuarios se convierten en nodos vecinos en una base de datos compartidas, en la que pueden confiar para grabar transacciones de activos. Los usuarios de una DLT pueden enviar transacciones, y a su vez formar parte de un conjunto limitado que valida transacciones.

De lo anterior queda claro que DLT permite a los usuarios de un sistema financiero almacenar y acceder a información relacionada con activos financieros, y registrar la propiedad y la trazabilidad del intercambio de los mismos. Se mantiene inalterable la función clave de los DCVs y TARGET2-SECURITIES, que es mantener un registro único y autorizado.

Por lo tanto, se concluye que las bases de datos que contienen la información de la propiedad de los activos financieros en un CSD, y las relaciones entre ellos, podrían ser implementadas mediante tecnología DLT.

A continuación, haremos referencia al procedimiento “Delivery-versus-payment”.

En la definición de DVP se observa que existen dos paradigmas de cuentas: una de activos financieros (ya sean valores, futuros, bonos, pagarés...) y otra de activos líquidos. En una implementación en una solución DLT, se intuyen dos configuraciones distintas:

- Ambos conceptos (activos financieros y activos líquidos), conviven en un mismo ledger.
- Existen dos ledgers distintos para activos financieros y activos líquidos.

En la primera de las dos soluciones, el ledger resultante tendrá que tener la suficiente habilidad para manejar ambos conceptos. Es más, deberá estar preparado para reflejar la información de todo tipo de activos financieros, y no siempre la lógica de negocio es común para todos los tipos (por ejemplo, hay activos que requieren que se bloquee una cantidad de dinero, y otros que liberan esa cantidad de dinero para que el solicitante pueda ejercer su actividad). La ventaja de este tipo de solución es que evita tener que desarrollar un mecanismo de comunicación entre ledgers.

En la segunda solución, en cambio, dotará al sistema de más flexibilidad para crear ledgers distinto, en función del tipo de activo a representar. A su vez, permite emplear blockchains ya existentes cuya criptomoneda asociada se emplee como activo líquido. La desventaja de esta solución es tener que desarrollar un protocolo de comunicación entre ledgers.

3.3. Implementación elegida: Hyperledger Fabric

El proyecto Hyperledger es una iniciativa colaborativa mantenida por Linux Foundation, y entre cuyos miembros se encuentran empresas como Airbus, Accenture, Cisco, Fujitsu, SAP o American Express. La lista completa se puede consultar en [20]. El objetivo es crear una tecnología blockchain cross-industry.

Ha desarrollado 5 implementaciones distintas, disponibles en [21]. La más extendida y desarrollada es Hyperledger Fabric, en la cual centraremos el estudio. Se trata de su primer proyecto, y fue iniciado por IBM. Esta solución provee:

- Gestión de identidades: a través de un servicio que gestiona las identidades de los actores, autentica a todos los participantes en la red. Permite listas de control de acceso para efectuar las distintas operaciones de la red.
- Privacidad y confidencialidad: A través de canales privados de comunicación se envía información privada a los miembros de los mismos, y de forma confidencial e inaccesible a miembros de otras redes.

- Procesamiento eficiente de las transacciones: Gracias a la existencia de nodos que ejecutan transacciones y nodos que las comprometen, se permite la paralelización y la concurrencia de acciones. Un servicio de ordenación garantiza la ejecución ordenada de las transacciones.
- Lógica de negocio a través de smart contracts: Gracias a los programas conocidos como chaincodes se implementan las acciones necesarias y se gestionan la propiedad de los activos.

Como se puede observar, Hyperledger Fabric es un framework de DLT privado y permisionado. Su arquitectura se basa en un conjunto de nodos que se comunican entre otros. En la blockchain de Hyperledger ocurren básicamente tres acciones:

- Se ejecutan programas, llamados chaincodes.
- Se mantiene información sobre estados de activos del ledger.
- Se ejecutan transacciones.

Además, Hyperledger Fabric posee una Autoridad de Certificación para facilitar la autenticidad de los nodos.

3.4. Análisis de los componentes de Hyperledger Fabric

En este apartado, entraremos en detalle en cada uno de los componentes de este framework, analizando cuáles serían los detalles de diseño de cada uno de ellos para un sistema DVP e incidiendo en los aspectos de seguridad.

Durante la exposición de esta solución DLT, y por brevedad, se simplificará el término "Hyperledger Fabric" a "Hyperledger".

La versión estudiada ha sido Hyperledger Fabric 1.3.0.

3.4.1.1. Activos

Los activos, o assets en inglés, representan objetos de información, ya sean tangibles (objetos de la vida real) o intangibles (cualquier elemento de información como un contrato o un objeto de propiedad intelectual). Hyperledger permite modificar los activos empleando transacciones de chaincodes o programas.

Se representan mediante un par clave-valor, con cambios de estado registrados como transacciones en el ledger. Lo más habitual es representarlos en formato JSON, aunque también es posible representarlos en binario.

En nuestra solución, se recomendará el empleo del lenguaje JSON para claridad del contenido. En un sistema DVP, los activos a representar serán:

- Cuentas de un banco.
- Valores de una empresa.

La implementación de estas dos estructuras en el lenguaje GO serían las siguientes:

```
type CuentaCorriente struct {
        idCliente   string `json:"idCliente"`
        saldo  Double `json:"saldo"`
}

type valorEmpresa struct {
        idEmpresaPropietaria   string `json:"idEmpresaPropietaria"`
        idClientePropietario   string `json:"idClientePropietario"`
        precio  Double `json:"precio"`
        Boolean reservado `json:" reservado "`

}
```

El sentido de las propiedades de cada assets está claramente explicado con su nombre.

Además, el protocolo de intercambio DVP obliga a crear otros activos, que reflejen información derivada del proceso de canje entre dinero y un valor disponible en el mercado. Más exactamente, se trata de pares clave-valor que forman parte de los mecanismos que emplea el protocolo para garantizar la temporalidad (garantía de que el protocolo se ejecutará en tiempo finito) y la equidad. Estas estructuras se van a reflejar aquí, y en el capítulo "Diseño del mecanismo DVP e interacción entre canales" se detallará su misión. De forma breve, el activo PeticionCompra refleja una solicitud de compra de un valor por parte de un cliente, el activo BloqueoSaldo representa el bloqueo de una cantidad económica para ser entregada como contrapartida a la posesión de un valor, y BloqueoValor es la reserva de un valor tras la solicitud de inversor. Como se observa, las tres estructuras poseen un valor de control de tipo Double para representar un temporizador.

```
type PeticionCompra struct {
        inversor   string `json:"inversor"`
        empresaPropietariaValor   string `json:" empresaPropietariaValor"`
        precioPropuesto  Double `json:" precioPropuesto"`
       compraPendiente   Double `json:" compraPendiente"`
}


type BloqueoSaldo struct {
        inversor   string `json:"inversor"`
        cantidadBloqueada  Double `json:" cantidadBloqueada"`
       inicioDelBloqueo   Double `json:" inicioDelBloqueo"`
}

type BloqueoValor struct {
        idValorBloqueado   string `json:" idValorBloqueado"`
       inicioDelBloqueo   Double `json:" inicioDelBloqueo"`
}
```

3.4.1.2. Estructuras de datos

Hyperledger mantiene dos tipos de estructuras de datos: la base de datos de estado, y el ledger.

El primero de ellos hace referencia al estado del ledger, modelado como un almacenamiento de pares clave-valor. Se denomina "world state", o directamente "state". Estos pares clave-valor hace referencia a estados del ledger. Estas entradas son manipuladas son por los chaincodes mediantes operaciones de lectura y escritura. El almacenamiento es persistente, y las actualizaciones son registradas. Su propósito es facilitar la lectura de estos estados a un programa, en vez de tener que recorrer todo el registro de transacciones en el ledger. Esta base de datos es mantenida por los nodos. Un ejemplo de esta base de datos se puede ver en la ilustración 10.

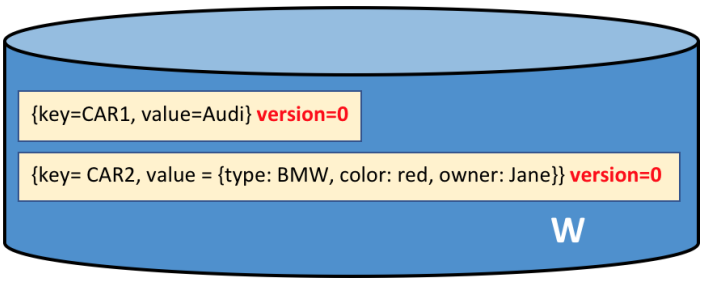

Ilustración 10 – Ejemplo de pares clave-valor en un base de datos de estado, tomada de [30]

Existen dos estados en esta base de datos. El primero hace referencia a un estado con clave="CAR1" y valor="Audi". El segundo hace referencia a un estado con clave "CAR2" y un tipo de valor expresado en sintaxis JSON: "{model:BMW, color=red, owner=Jane}".

Cada par clave-valor tiene un atributo versión, que se incrementa cada vez que se modifica el valor. Es útil a la hora de detectar cambios en los pares involucrados en una transacción. A la hora de comprometer una transacción, los pares involucrados deben tener el mismo número de versión que al comienzo de la transacción.

Hyperledger soporta dos modelos de bases de datos. LevelDB es una base de datos embebida en los nodos, y CouchDB es una base de datos externa. Ambas pueden almacenar datos binarios y JSON, aunque CouchDB proporciona un interfaz más potente para realizar búsquedas JSON, paginación de resultados e índices.

El ledger, en cambio, hace referencia a la blockchain en sí, y se trata de un registro verificable de todos los cambios realizados en la base de datos (es decir, transacciones válidas). Es construido por el Servicio de Ordenación como una cadena de bloques de transacciones ordenados. El ledger está almacenando en todos los nodos.

La forma de enlazar los bloques se implementa mediante hashes: en la cabecera de los bloques aparece un hash de las transacciones incluidas, así como una copia del hash del bloque anterior.

El primer bloque contiene el bloque génesis, con la configuración inicial del canal.

Como se ha observado, la base de datos de estado es modificable, mientras que el ledger es inmutable. El estado se refleja en una base de datos, mientras que el ledger es un archivo.

En la solución DVP de nuestro estudio, se recomienda la creación de dos ledgers distintos.

En el primero de ellos, se reflejará la información correspondiente a los clientes de un banco. Inicialmente, este ledger estará vacío, y se popularizará con cuentas de cliente según se vayan dando de alta. Los assets, por lo tanto, serán instancias del activo definido como "*CuentaCorriente".* Las distintas transacciones que aparecen en los bloques de este ledger hacen referencia a la creación de cuentas de usuario, o a las transferencias de saldo entre usuarios.

El segundo de ellos hará referencia a los valores que una empresa pone en el mercado. Se crearán assets de tipo "*valorEmpresa"* cada vez que una empresa pone a la venta cualquier tipo de valor financiero (bonos, acciones, opciones…).

Ambos ledgers deberán ser independientes.

Otro aspecto a destacar en nuestro estudio es la seguridad de la base de datos de estado. Más adelante se explicarán las funciones de cada tipo de nodo y en qué consiste una transacción, pero por el momento podemos afirmar que esta base de datos está presente en:

- Los nodos de tipo aprobador. Validan las propuestas de transacción de los clientes ejecutándolas con su réplica de la base de datos. Si la propuesta es válida, firman una respuesta y se lo envían al cliente.
- Los nodos de tipo estándar (todos los nodos), ya que reciben bloques de transacciones ordenados, los agregan al ledger local y actualizan la réplica de la base de datos de estados local.

Por lo tanto, cualquier nodo que pertenezca al canal tiene acceso a esta base de datos. Las medidas de seguridad son:

- El servicio MSP (member service provider), que se encarga de crear identidades y gestionar las peticiones de autorización. Los nodos que quieran formar parte del canal deberán tener una identidad válida.
- La lógica del chaincode. En la implementación de este programa, se pueden realizar comprobaciones necesarias para evitar que un actor malicioso modifique sin permiso esta base de datos.

Si un nodo decide comportarse de forma maliciosa y modificar de forma voluntaria su réplica de la base de datos, la red acabará dándose cuenta de su comportamiento. De facto, Hyperledger presenta una tolerancia a fallos bizantinos. Esto es debido a que a la hora de acceder datos modificados por una entidad maliciosa en una nueva propuesta de transacción, se le devolverá a la aplicación solicitante datos distintos que el resto de nodos. La política de aprobación de un chaincode detectará que la transacción no ha sido firmada por todos los nodos necesarios y en consecuencia, la red puede expulsar al nodo de la red. Al tratarse de una red permisionada, se podrán tomar las consecuencias legales con la organización.

3.4.1.3. Organizaciones

Las organizaciones, a través de la creación de nodos y canales, de la implementación e instanciación de chaincodes, y del mantenimiento y la provisión de identidades a través de un servicio de membresía son las responsables de una red Hyperledger.

En nuestra solución DVP, existirán las siguientes organizaciones:

- Una organización "Banco" será responsable del mantenimiento y la provisión de identidades del ledger que contiene los assets tipo "cuentaCorriente". Su función es gestionar el estado de las cuentas corrientes de los usuarios.

- Una organización "mercado de Valores" se encargará del ledger que almacena la información de los assets tipo "valorEmpresa". Su misión es el control de los valores de las empresas que los ponen en el mercado.

- Una organización "Depositario", que será capaz de ver la información y enviar transacciones en ambos ledger. Su función es el mecanismo DVP:

    - Responde a solicitudes de compra de valores de los inversores.
    - Verifica que los valores solicitados por los inversores se encuentran en el mercado al precio ofertado.
    - Bloquea el precio de los valores en la cuenta corriente de los clientes.
    - Establece la propiedad del valor solicitado en beneficio del inversor.
    - Traslada el precio de los valores de la cuenta corriente del cliente, a la cuenta corriente de la empresa propietaria del valor.

- Una organización "Regulador", que solo tiene capacidad de lectura en la información de ambos ledgers y audita el funcionamiento de ellos.

En la ilustración 11 se observa la relación entre las organizaciones y los canales. La relación entre un canal y un ledger es 1:1, ya que los nodos que forman parte de un canal poseen una copia distribuida del ledger de dicho canal, como más adelante veremos.

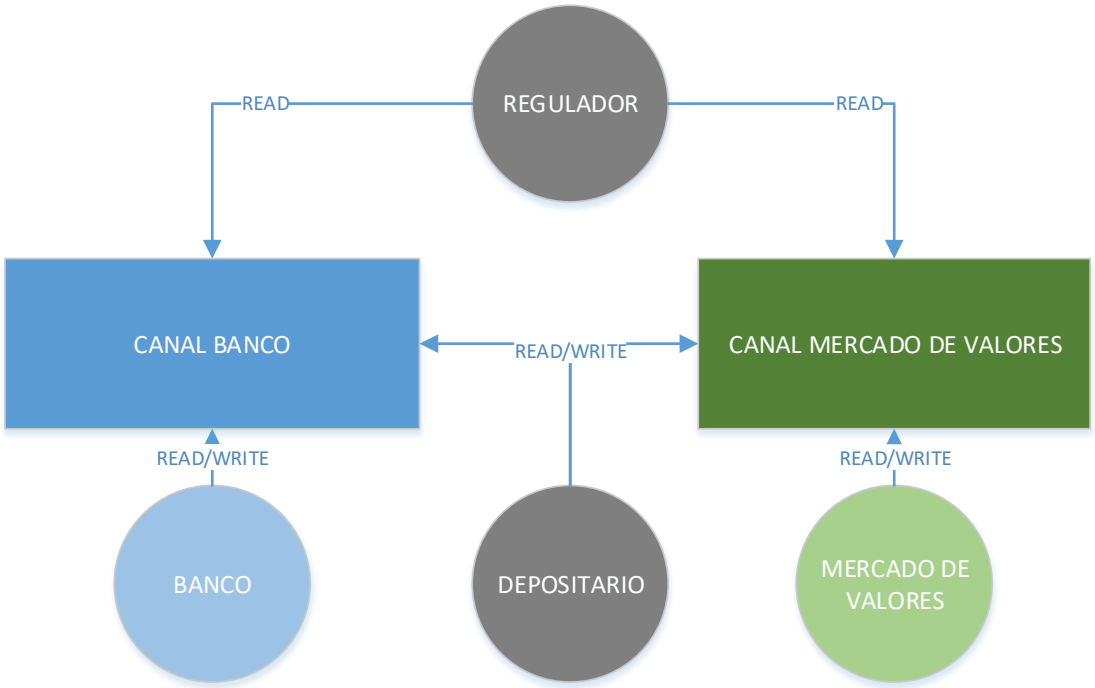

Ilustración 11 – Relación entre las organizaciones y los canales

3.4.1.4. Nodos

Se trata de entidades que forman parte de Hyperledger y se comunican entre sí. Son un elemento imprescindible ya que almacenan una instancia del ledger y varias instancias de los chaincodes instalados, es decir, la información compartida y los procesos compartidos de la plataforma. Los usuarios y las aplicaciones, por lo tanto, deben interactuar con un nodo para poder acceder a dichos recursos.

Hay que aclarar, en primer lugar, la terminología usada en este documento, ya que todas las referencias empleadas se han encontrado en inglés. Hay varios tipos de nodos, compatibles entre sí:

- Nodo-Cliente. En la documentación aparece como “client” o “client peer”. Es el nodo que envía transacciones, ya sea el inicio de una invocación a una función de un chaincode (con destino a los nodos aprobadores), o una propuesta de transacción al servicio de ordenación. Es el nodo que actúa en representación de un usuario.
- Nodo estándar. Es conocido como “commiting peer” o simplemente “peer”. Se ha traducido como nodo estándar ya que solo realiza las funciones habituales de un nodo: ejecuta (compromete) una transacción, y mantiene una copia del ledger. Reciben bloques (conjunto de actualizaciones del ledger ordenadas) del servicio de ordenación. Estos nodos pueden tener el rol de aprobadores, para el cual el término en inglés es “endorsing peer”. El concepto de aprobador está en relación a un chaincode, es decir, aprueba una invocación de una función, y es una fase previa a su compromiso.
- Nodo que ejecuta “servicio de ordenación”. Se le denomina “orderer”. Es un nodo que ejecuta este servicio de comunicación. Se encarga de proveer el canal de comunicación a nodos-clientes y nodos estándar, ofreciendo el servicio de retransmisión de mensajes que

contienen transacciones. Ofrece un servicio de entrega garantizada y ordenada de todos los mensajes a todos los nodos.

- Nodos ancla, denominados “anchor-peer”. Son necesarios para el descubrimiento de la red. Se explicará en detalle el funcionamiento de estos nodos en el apartado “canales”.

Como se ha indicado anteriormente, se puede invocar a una función de un chaincode para que ejecute una acción de lectura o de escritura. Desde el punto de vista de un nodo, la ilustración 12, resumiría la interacción entre una aplicación y un nodo.

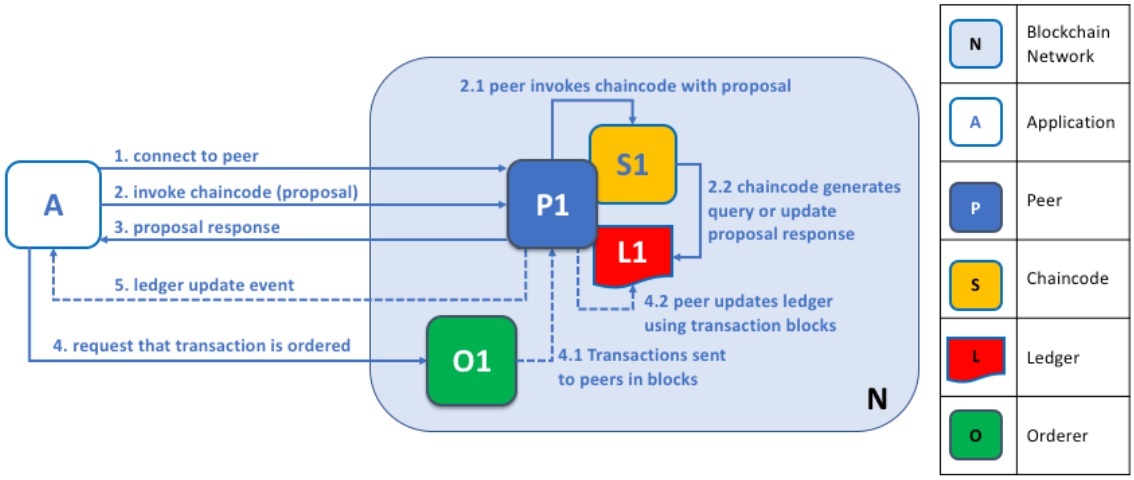

Ilustración 12 – Interacción entre una aplicación, un nodo y el servicio de ordenación, tomada de [29]

En este ejemplo, la aplicación A conecta con el nodo P1, el cual invoca al chaincode S1 para ejecutar la función correspondiente, ya sea de lectura o de escritura. El chaincode devuelve el resultado de la operación de lectura, o una “proposed ledger update”, una propuesta de actualización del ledger. Si la operación fuera de lectura, la aplicación recibe el resultado y se puede dar por concluido el proceso. Si, en cambio, fuera de escritura, la aplicación construye una transacción con la información obtenida en pasos anteriores, y se la envía al servicio de ordenación, el cual se encarga de generar bloques con transacciones y distribuirlos a todos los nodos. Cada nodo valida la transacción y actualiza el ledger. En el apartado “Transacciones” se detallará este intercambio de mensajes entre aplicaciones y nodos en Hyperledger.

Si la operación de escritura, debido a la política de aprobación requiriera el visto bueno de varios nodos, el sentido de la “proposed ledger update” sería que cada nodo, por sí mismo, hubiera efectuado la operación de escritura. En este caso, se requeriría el consenso de varios nodos para ejecutar la transacción.

En la arquitectura DVP del presente documento, se tendrán en cuenta las siguientes consideraciones con respecto al número de nodos:

- De forma general, el número de nodos determinará el rendimiento de la aplicación, dotando de mecanismos de tolerancia a fallo adicionales.
- En cuanto a los nodos aprobadores, en el capítulo “Políticas de instanciación y aprobación” se definen los nodos de aprobación necesarios para cada Chaincode. Para el caso que se requiera la aprobación de un nodo, la creación de N nodos significará un rendimiento de N transacciones simultáneas. Si tenemos en cuenta una tolerancia a fallos que soporte hasta la caída de la mitad de los nodos, para un rendimiento de N

transacciones simultáneas, se requieren $N + \frac{N}{2}$ nodos para cada organización. Si un chaincode requiere la aprobación de 2 nodos de una misma organización, serían necesarios $2\ x\ (N + \frac{N}{2})$ nodos en dicha organización.

- En cuanto a los nodos estándar (que reciben los bloques desde el servicio de ordenación, y actualizan su ledger y base de datos de estado), su número está determinado por las operaciones de lectura que se solicitan desde las aplicaciones (es decir, propuestas de transacciones que no requieren una actualización en el ledger). Si se requiere un rendimiento de N solicitudes de lectura, y teniendo en cuenta una tolerancia a fallos que soporte la caída de la mitad de los nodos, se requieren $N + \frac{N}{2}$ nodos para cada organización.
- Las consideraciones para el número de nodos "ancla" y nodos de ordenación se tendrán en cuenta en los apartados "Canales" y "servicio de ordenación", respectivamente.

3.4.1.5. Chaincodes

Un chaincode es un programa que representa una lógica de negocio acordada previamente por miembros de la red, y representa en Hyperledger el concepto de smart contract. La plataforma admite los lenguajes de programación Node.js, Java o Go. Se ejecutan en un contenedor seguro, aislado de los llamados "nodos aprobadores" (más adelante desarrollaremos este término). Su función básica es inicializar y gestionar el estado del ledger a través de transacciones enviadas por las aplicaciones. En otras palabras, implementan la lógica de negocio que representa la forma cómo las aplicaciones interactúan con el ledger. La ejecución de chaincodes de Hyperledger permite dotar a la lógica de negocio de:

- Trazabilidad: la información de la ejecución de los chaincodes que almacenada en el ledger.
- Transparencia: Cualquier aplicación que tenga permisos para acceder a un ledger, es decir, que estén es capaz de verificar la ejecución de chaincode anteriores.
- Confidencialidad: Solamente los nodos que pertenezcan a un determinado canal pueden acceder a la información del ledger.

Cada chaincode se implementa como un programa aislado que mantiene su propio estado en el ledger. Dicho estado pertenece a un scope que solo puede ser accedido por ese chaincode. No puede ser accedido por otro chaincode. Sin embargo, es posible que, dentro de la misma red y con los permisos de acceso necesario, un chaincode puede invocar a otro chaincode para acceder a su estado.

El ciclo de vida de un chaincode se resume en la ilustración 13.

Ilustración 13 – Ciclo de vida de un chaincode

El primer evento en la vida de un chaincode es **crear un paquete** que contendrá lo siguiente:

- El código del chaincode.
- Una política de aprobación. Define el conjunto de nodos en un canal que deben ejecutar el chaincode y aprobar los resultados de la ejecución, con el fin de considerar como válida la transacción.
- Un conjunto de firmas, para indicar quién es el creador del paquete y que permiten verificar su integridad.

El atributo firma de un paquete puede ser multivaluado, ya que puede expresar que varios actores son los dueños de un chaincode, por lo que requerirá múltiples firmas, o simplemente constar de una firma que, en su versión más sencilla, será la del nodo que emite la transacción para instalar el chaincode (se explicará más adelante).

A la hora de crear el paquete, se puede definir una política de instanciación, que expresa qué actores tienen derecho a instanciar el chaincode. Si no se provee ninguna política, por defecto el chaincode podrá ser instanciado por el rol Admin del peer del servicio MSP (se explicará más adelante este concepto).

A continuación, el siguiente evento será **instalar** el chaincode en un nodo. Es obligatorio instalar un chaincode en cada "nodo aprobador" en los que queremos que se ejecute el programa. Más adelante se desarrollarán los tipos de nodos, pero es interesante destacar un concepto de seguridad. Solo hay que instalar el programa en los nodos aprobadores necesarios, y que pertenezcan a la organización que posee el chaincode. Esto se debe a cuestiones de confidencialidad del programa. Cualquier nodo aprobador que haya instalado el chaincode, podrá acceder a su código. El resto de nodos, que no hayan instalado el programa puede validar y comprometer las transacciones al ledger.

Después, el próximo paso será **instanciar** el chaincode en un canal. Dicha acción será realizada por un actor que satisfaga la política de instanciación anteriormente indicada, y ha de poseer un permiso especial (writer) en el canal. Así, se evita que entidades deshonestas intenten instanciar programas. Como se indicó, el permiso por defecto para instanciar programas en un canal es el administrador del servicio MSP.

Después de ser correctamente instanciado, el chaincode entra en estado activo en el canal, y está listo para procesar propuestas de transacciones, según llegue al nodo aprobador. Para aclarar este concepto, se incluye un ejemplo de llamada al API de Hyperledger para instanciar un chaincode.

```
peer chaincode instantiate -C canalBanco -n GestionBanco
-c '{"Args":["usuario1","100"]}'
-P "AND ('Banco.peer','Regulador.peer')"
```

En este caso, se instancia el chaincode "programa", y se inicializa su estado con la expresión JSON {"Args":["usuario1","100"]}. La política de aprobación sería "AND (Banco.peer', 'Regulador. peer')", es decir, que requerirá la firma de estos dos nodos aprobadores para dar como válida la propuesta de transacción, uno de la organización Banco y otra de la organización Regulador.

Un paso adicional en el ciclo de vida de un chaincode es la **actualización** del mismo. Para ello, debe ser instalado en los nodos aprobadores de nuevo, pero manteniendo el mismo identificador del chaincode e incrementando el valor de la versión.

De modo explicativo, y de forma breve, se describirá la estructura de un programa chaincode.

Cada chaincode debe implementar los siguientes métodos:

- Init(), para configurar el programa.
- invoke(), para definir la lógica principal del programa.

Ambos métodos pueden aceptar un parámetro "ChaincodeStubInterface". Dicho objeto es la representación del ledger, y proporciona un cliente para interactuar con el mismo e invocar a otros chaincodes.

Un ejemplo de código de un chaincode, proporcionado en [27], se muestra a continuación. Se trata de un chaincode que crea activos (pares de clave y valor) en el ledger, programado en Go. Analizaremos cada una de las secciones por separado.

De forma general, un chaincode provee dos interfaces para interactuar con el ledger:

- Método Query(), para operaciones de lectura: se traducen como consultas a la base de datos local de estado y no conllevan transacciones.
- Método Invoke(): para operaciones de escritura: se transforman en transacciones, que necesitan ser aprobadas por los nodos aprobadores, y confirmadas por el servicio de ordenación.

En primer lugar, hay que importar los componentes necesarios. En este ejemplo, establece las dependencias "fmt" para invocar a la función Println() de depuración de errores e importa el paquete shim, que proporciona el interfaz "chaincode" que es la abstracción del programa y el chaincode stub, necesario para interactuar con el ledger. Además, importa el paquete peer protobuf, necesario para definir la estructura de los mensajes.

```
package main

import (
    "fmt"

    "github.com/Hyperledger /fabric/core/chaincode/shim"
    "github.com/Hyperledger /fabric/protos/peer"
)
```

De esta forma se definen los activos, en forma de estructura que modelizan sus propiedades.

```
type SimpleChaincode struct {
}
```

En este ejemplo, se define una estructura vacía, que interactúa con la base de datos de estado. Si la estructura tuviera propiedades, se modelizaría de la siguiente forma:

```
type SimpleAsset struct {
Model   string `json:" Model"`
owner   string `json:" owner"`

}
```

A continuación, se define el método Init(), necesario para realizar la inicialización del chaincode. El método recibe una instancia de ChaincodeStubInterface, que le proporciona los argumentos necesarios y un cliente con el que interactuar con el entorno.

Si los parámetros son correctos, este método se encarga de recibir un valor, y lo utiliza para inicializar su estado en el ledger, mediante una llamada al método ChaincodeStubInterface.PutState (). Esta función interpreta el primer argumento y lo emplea para inicializar el valor de la clave "key", del par "clave-valor" que se almacenará en la base de datos de estado.

```
func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface,
function string, args []string) ([]byte, error) {
        if len(args) != 1 {
                return nil, errors.New("Incorrect number of
arguments. Expecting 1")
        }

    err := stub.PutState("key", []byte(args[0]))
    if err != nil {
        return nil, err
    }

    return nil, nil
}}
```

Después, se define el método "Invoke" genérico, que se invoca cuando una aplicación autorizada interactúa con el chaincode. Las invocaciones se traducirán a transacciones, que posteriormente se agruparán como bloques en el ledger. El método Invoke() recibirá por parámetro el método del programa al cual se quiere llamar, junto con el objeto ChaincodeStubInterface para interactuar con el ledger, así como los argumentos. Para este ejemplo, solo se define un método, write() aparte del método init()).

```
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface,
function string, args []string) ([]byte, error) {
        fmt.Println("invoke is running " + function)

    if function == "init" {
        return t.Init(stub, "init", args)
    } else if function == "write" {
        return t.write(stub, args)
    }
    fmt.Println("invoke did not find func: " + function)

    return nil, errors.New("Received unknown function invocation: "
+ function)
}
```

A continuación, se define la función write(), invocada desde el método anterior invoke(). Dicha función crea un objeto clave-valor en la base de datos de estado, empleando el primer y segundo argumento.

```
func (t *SimpleChaincode) write(stub shim.ChaincodeStubInterface,
args []string) ([]byte, error) {
        var key, value string
        var err error
        fmt.Println("running write()")

        if len(args) != 2 {
                return nil, errors.New("Incorrect number of
arguments. Expecting 2. name of the key and value to set")
        }

        key = args[0]                                //rename for fun
        value = args[1]
        err = stub.PutState(key, []byte(value))  //write the variable
into the chaincode state
        if err != nil {
                return nil, err
        }
        return nil, nil
}
```

Después, se define la función Query(), análoga a la función Invoke() pero destinada a consultas a la base de datos de estado. Este tipo de invocaciones, como se explicará más adelante, no conlleva la ejecución de una transacción, sino solo una consulta a la base de datos local del nodo. De forma igual al método invoke, recibe como parámetro la función del chaincode que se quiere ejecutar.

```
func (t *SimpleChaincode) Query(stub shim.ChaincodeStubInterface,
function string, args []string) ([]byte, error) {
        fmt.Println("query is running " + function)
```

```
    // Handle different functions
    if function == "read" {                            //read a
variable
        return t.read(stub, args)
    }
    fmt.Println("query did not find func: " + function)

    return nil, errors.New("Received unknown function query: " +
function)
}
```

A continuación, se define la función read(), accesible a través de la función Query(). Recibe como parámetro la clave cuyo valor se quiere recuperar. Para ello, interactúa con el objeto stub, que es un representante del ledger e invoca el método "GetState()".

```
func (t *SimpleChaincode) read(stub shim.ChaincodeStubInterface,
args []string) ([]byte, error) {
       var key, jsonResp string
       var err error

    if len(args) != 1 {
        return nil, errors.New("Incorrect number of arguments.
Expecting name of the key to query")
    }

    key = args[0]
    valAsbytes, err := stub.GetState(key)
    if err != nil {
        jsonResp = "{\"Error\":\"Failed to get state for " + key +
"\"}"
        return nil, errors.New(jsonResp)
    }

    return valAsbytes, nil
}
```

Finalmente, se declara el método main(), que inicia el chaincode durante la instanciación. Llama a la función shim.Start(), que establece la comunicación entre este programa y el nodo en el que ha sido invocado.

```
func main() {
    err := shim.Start(new(SimpleChaincode))
    if err != nil {
        fmt.Printf("Error starting Simple chaincode: %s", err)
    }
}
```

Para terminar este apartado, se define la propuesta de chaincodes para esta solución DVP. Para el diseño de la misma, se tiene en cuenta maximizar la seguridad y la funcionalidad de la arquitectura. Hay que recordar que si un chaincode se instala en un nodo de una aplicación, el código del mismo está disponible para dicha organización y se permitirá ejecutar las funciones del programa. Por lo tanto, se definen varios chaincodes, agrupando las funcionalidades

necesarias para cada organización solamente en los nodos de la organización. Todos los chaincodes implementarán los métodos comunes Init() y start(), así como los métodos para invocar operaciones de lectura Query(), y operaciones que se traducirán en transacciones Invoke().

Canal Banco:

Chaincode GestionBanco: se instalará solamente en los nodos de la organización Banco.

Métodos accesibles a través de la interfaz Query():

-método obtenerSaldo(), para obtener el saldo de un cliente, a través del objeto asociado de la base de datos de estado.

Métodos accesibles a través de la interfaz Invoke():

-método crearCuenta(), que crea un nuevo activo cuentaCorriente, representado como un nuevo par-valor en la base de estados.

-método ingresarCantidad(), que ingresa una cantidad en la cuenta de un usuario, modificando las propiedades del activo cuentaCorriente asociado, interactuando con el objeto asociado.

-método extraerCantidad(), que deduce una cantidad del activo cuentaCorriente asociado a un cliente.

Chaincode AuditoriaBanco: se instalará solamente en los nodos de la organización Regulador.

Métodos accesibles a través de la interfaz Query():

-método transaccionesCliente(), para obtener el histórico de transacciones de un cliente. Para ello, es útil emplear el atributo versión que se modifica cada vez que se altera el valor de una clave en la base de datos de estado. El método GetHistoryForKey(), proporciona un interfaz para ello.

-método listadoClientes(), para obtener el conjunto de clientes de un banco.

Chaincode ServiciosDepositario: se instalará solamente en los nodos de la organización Banco. Se trata de un Chaincode elaborado por la organización Depositario, y que encapsula la funcionalidad ofrecida para los inversores. Se ofrece como un Chaincode diferenciado por seguridad, ya que los consumidores de estos servicios son distintos que los servicios que ofrece el Depositario en sus nodos.

Métodos accesibles a través de la interfaz Invoke():

-método lanzarPeticionCompraValor(), que es invocado por la organización Banco, en representación de un cliente, para solicitar la compra de un valor.

Chaincode GestionIntercambioDVP: se instalará solamente en los nodos de la organización Depositario. Ofrece la funcionalidad necesaria para que una aplicación cliente gestione el intercambio DVP. En el apartado "Diseño del mecanismo DVP e interacción entre canales" se desarrollará este intercambio.

Métodos accesibles a través de la interfaz Query():

-método obtenerPeticionPendienteCompra(), se emplea para obtener peticiones de compra de valores que han sido solicitadas por los inversores.

Métodos accesibles a través de la interfaz Invoke():

-método bloquearSaldo(), que transfiere el valor de un activo, desde la cuenta de un cliente a la cuenta del actor Depositario. De esta forma, y a partir de una solicitud de una compra de un valor, queda bloqueado el saldo de un cliente.

-método cancelarPeticionCompra(), que elimina el activo que representa la petición de compra de un valor, debido a que no tiene fondos suficientes.

-método transferirSaldo(),que transfiere una cantidad bloqueada, desde la cuenta del actor Depositario a la cuenta de la empresa que vende un valor.

-método eliminarBloqueoSaldo(), que forma parte del protocolo de finalización y elimina el objeto que hace referencia al bloqueo del saldo de un valor solicitado por el inversor.

-método deshacerBloqueoSaldo(),que devuelve el valor de un activo a la cuenta de un cliente desde la cuenta del actor Depositario, en caso de que se haya cancelado la compra de un activo. Forma parte del protocolo de cancelación.

Canal MercadoDeValores:

Chaincode GestionValores: se instalará solamente en los nodos de la organización MercadoDeValores.

Métodos accesibles a través de la interfaz Invoke():

-método ponerEnMercado (), que crea una serie de nuevo activo cuentaCorriente, representado como un nuevo par-valor en la base de estados. Dichos activos tendrá un precio

inicial y se corresponderán con un identificador de empresa. Inicialmente, no estarán asociados a ningún usuario.

Chaincode AuditoriaMercadoDeValores: se instalará solamente en los nodos de la organización Regulador.

Métodos accesibles a través de la interfaz Query():

-método listadoDeValores(), para obtener todos los activos ValorEmpresa relacionados con el identificador de una empresa.
-método listadoValoresdeCliente(), para obtener todos los activos ValorEmpresa que posee un identificador de cliente.
-método historicoValor(), para el histórico de cambios de propiedad de un valor. Para ello, es útil emplear el atributo versión que se modifica cada vez que se altera el valor de una clave en la base de datos de estado.

Chaincode GestionPropiedadValores: se instalará solamente en los nodos de la organización Depositario.

Métodos accesibles a través de la interfaz Invoke():

-método bloqueaValor(), que asigna la propiedad de un activo ValorEmpresa a un nuevo inversor.

-método eliminarBloqueoValor(),que forma parte del protocolo de finalización y elimina el objeto que hace referencia al bloqueo de un valor cuya propiedad ha solicitado un inversor.

-método deshacerBloqueoValor(), que forma parte del protocolo de cancelación, y se encarga de liberar un valor que se ha puesto a nombre de un inversor, en caso de que la operación no concluya de forma satisfactoria después de un tiempo determinado.

Más adelante, en el capítulo “Políticas de instanciación y aprobación”, se definirán las políticas de aprobación de estos nodos.

3.4.1.6. Transacciones

Una transacción es la forma en la que se ejecuta un chaincode en Hyperledger .

Existen dos tipos de transacciones:

- Transacciones de despliegue o “deploy transactions”, que se encarga de recibir un programa como parámetro e instalar el programa en un canal.

- Transacciones de invocación, o “invoke transactions”, que ejecutan una operación en el contexto de un chaincode anteriormente instalado. Este tipo de transacciones son las que interaccionan con los datos del ledger, y son las que explicaremos a continuación.

Para explicar el flujo de una transacción, partimos de una situación en la que suponemos que el chaincode se encuentra instalado en los nodos e instanciado en el canal, y que la política de aprobación del chaincode requiere que todos los nodos aprueben la transacción.

Para empezar, un cliente (que suponemos que tiene los permisos necesarios para acceder a la red) envía una petición a los nodos. Si la política, como habíamos indicado, requiere que todos los nodos aprueben la transacción, tendrá que enviar la petición a todos los nodos. El cliente, por lo tanto, emplea el API de Hyperledger para crear una propuesta (proposal) de transacción. Dicho objeto se trata de una petición para invocar a una función de un chaincode, para leer o escribir información del ledger, firmado por las credenciales del usuario que invoca la transacción. En la ilustración 14 observamos este proceso.

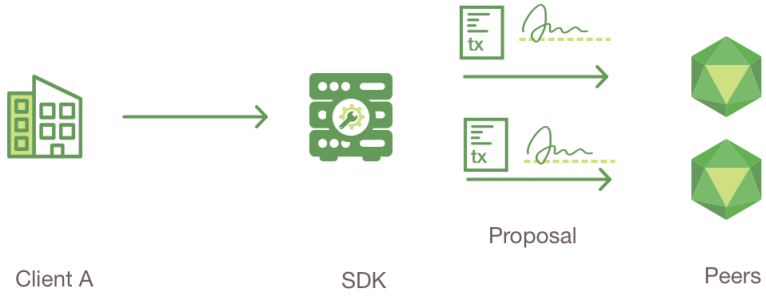

Ilustración 14 – Propuesta desde una aplicación, tomada de [37]

Los nodos aprobadores reciben esa propuesta, y verifican:

- que su estructura es correcta.
- que no ha sido enviada anteriormente, para evitar ataques de repetición.
- que la firma es correcta, empleando el servicio MSP.
- que el autor de la propuesta tiene derechos para realizar dicha operación. La política “Writer”, o de escritura es definida a la hora de crear un canal y determina que usuarios pueden enviar transacciones al canal.

Después, toman las entradas de la propuesta como parámetro de entrada de la función invocada del chaincode, y ejecutan esta función contra la base de datos de estados actual. La ejecución de dicha función produce el resultado de la transacción. No se producen cambios en el ledger por lo tanto. El resultado de la transacción, junto con la firma del nodo validador, se devuelve como una respuesta de la propuesta, como se detalla en la ilustración 15.

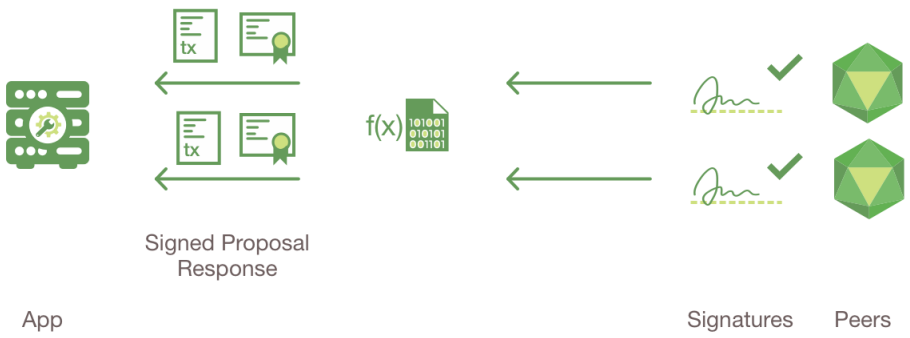

Ilustración 15 – Respuesta a la propuesta de transacción firmada por los nodos aprobadores, tomada de [37]

La aplicación verifica que las firmas de los nodos validadores son correctas. Si la operación invocada (la función de chaincode solicitada) es una operación de lectura de datos del ledger, sería suficiente con este paso (no necesitaría actualizar el ledger). Sin embargo, si requiere modificar información del ledger, es necesario enviar la transacción al Servicio de Ordenación. Antes de ello, verifica que la respuesta de la propuesta contiene las firmas requeridas (el número de nodos que la política solicitaba). Si intenta seguir el flujo de la transacción sin que todos los nodos necesarios hayan aprobado la transacción, será rechazada más adelante por la plataforma.

A continuación, la aplicación envía un mensaje al servicio de ordenación. Dicho mensaje, denominado "mensaje de transacción", contiene tanto la propuesta de transacción anterior, como la respuesta de la propuesta. Por lo tanto, en este contenido figuran los datos a modificar, las firmas de los nodos aprobadores y un identificador del canal. El servicio de ordenación recibirá estos mensajes de transacción, los organizará por canal, y creará bloques de transacción por cada canal. El orden de las transacciones dentro del bloque es clave para Hyperledger , ya que será el definitivo en todas las copias del ledger para evitar que se produzca un ledger fork: que dos copias del ledger distintas existan en el canal. Asimismo, hay que notar que todas las transacciones tienen igual valor para el servicio de ordenación e igual prioridad.

Los bloques son enviados a todos los nodos del canal como se observa en la ilustración 16. Todas sus transacciones serán validadas para asegurar:

- Que se cumple la política de aprobación.
- Que los datos a leer o a escribir no se han modificado desde que se inició la transacción.

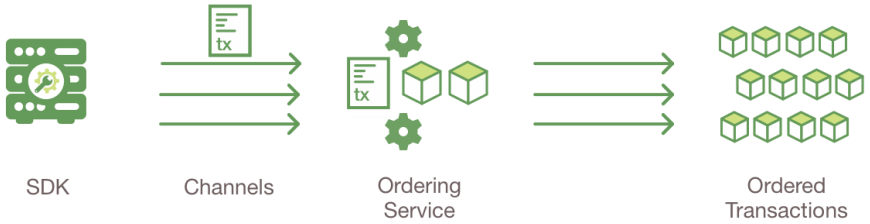

Ilustración 16 – Bloques enviados por el servicio de ordenación a los nodos, tomada de [37]

Finalmente, cada nodo agregar el bloque a la cadena de bloques del canal, y para cada transacción válida se escriben los datos pendientes de modificar en la base de datos actual. Para terminar se envía un mensaje de confirmación a la aplicación.

En la ilustración 17, obtenida de se muestra el flujo completo de una transacción.

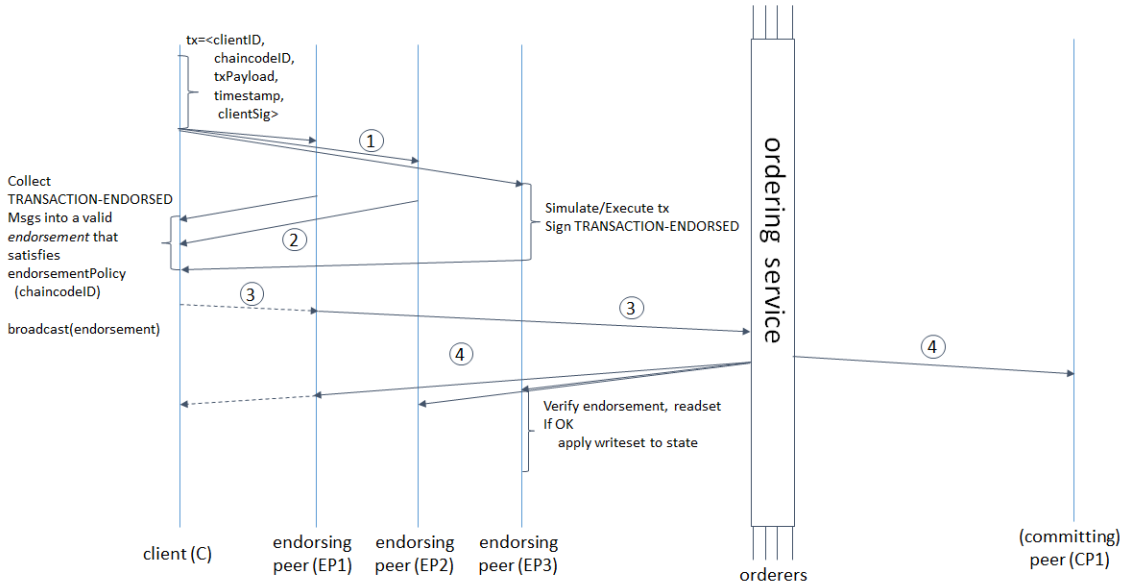

Ilustración 17 – Flujo completo de una transacción, tomada de [28]

A modo de resumen, se detallan los actores involucrados y las acciones en cada una de las 3 fases que implican una actualización de los datos del ledger:

Primera Fase: Propuesta.

Interacción: entre aplicación y nodos aprobadores:

- La aplicación envía una propuesta de transacción para ser aprobada.
- Cada nodo ejecuta el chaincode solicitado y genera una respuesta de propuesta, firmada con su clave privada.
- Cuando la aplicación recibe el número de respuestas solicitadas por la política de aprobación del chaincode, se da por terminada la fase.

Hay que tener en cuenta que los nodos pueden tener de forma puntual la copia del ledger en diversos estados, y por lo tanto, el contenido de la respuesta de propuesta, será distinto. La aplicación puede volver a enviar una propuesta al nodo para que vuelva a ejecutar la transacción. Si la aplicación reúne respuestas inconsistentes de nodos, será rechazada por la red más adelante.

Segunda fase: Creación de bloques.

Interacción: entre aplicación y el servicio de ordenación.

- La aplicación envía un mensaje de transacción, con las respuestas de propuestas firmadas por los nodos, al servicio de ordenación.
- El servicio de ordenación crea un bloque con varias transacciones ordenándolas, y lo envía a todos los nodos del canal.

Tercera fase: Validación.

Interacción: Actividad en los nodos

- Se distribuyen los bloques a todos los nodos.
- En los nodos se valida que tenga el número de aprobadores especificado en la política de aprobación del chaincode, y que el estado del ledger (a través de la base de datos del ledger) es compatible con el estado del ledger cuando la propuesta de transacción fue generada.
- Si todo es correcto, el bloque se agrega al final del ledger.

Esta fase del análisis de Hyperledger no conlleva una conclusión para la arquitectura DVP, pero es completamente necesaria para el entendimiento de su funcionalidad y la comprensión del resto de apartados.

3.4.1.7. Servicio MSP

El servicio MSP (Membership Service Provider) se encarga de gestionar el control de acceso a un canal. Define qué es una identidad y cómo se validan y autentican estas identidades. Una red Hyperledger admite varios MSPs, para poder de estar forma permitir que varias organizaciones interactúen entre sí, sin obligar a confiar en la validación de las demás.

Este servicio se encarga de mapear identidades a organizaciones. Además, determina el rol que va a tener un nodo dentro de una organización, y por lo tanto, cómo consigue el acceso a los recursos de la red.

Recordemos que los clientes del servicio MSP serán los nodos que requieran generar firmas, así como todos los miembros que se autentiquen en un canal (nodos aprobadores, nodos de ordenación y clientes).

En la implementación por defecto de un MSP, la autenticación y firma se basa en certificados x.509, y requiere parámetros como los siguientes:

- Lista de certificados raíces y de autoridades de certificación subordinadas de confianza.
- Lista de certificados de entidad final, emitidos por una autoridad de confianza, que representan a los administradores del MSP. Su función es modificar la configuración del MSP, como por ejemplo, agregar certificados a la lista de confianza.
- Listas de revocación de certificados.

Para que un certificado sea una identidad válida en una implementación por defecto de un MSP, ha de ser emitido por un certificado incluido en la lista de confianza, y no ha de estar incluido en ninguna lista de revocación.

A la hora de crear un canal, se definen las equivalencias entre ACs, organizaciones y servicios MSP en la política del canal.

Este servicio MSP se instanciará en un nodo determinado de la red Hyperledger. Como nota de seguridad, hay que destacar que solo admite certificados basados en clave ECDSA (Elliptic Curve Digital Signature Algorithm) y no RSA.

Para crear las identidades de los miembros de una red, se puede emplear openssl o la herramienta cryptogen, disponible en el API de Hyperledger. Por otra parte, la red permite instanciar una autoridad de certificación conocida como "Hyperledger Fabric CA". Está compuesta por una arquitectura cliente-servidor y una base de datos, y una vez instanciada, provee un interfaz de línea de comandos.

Es interesante destacar la clasificación de identidades en una implementación estándar de un MSP. En un archivo de configuración llamado config.yaml se pueden definir qué identidades serán clientes (es decir, podrán enviar transacciones o hacer consultas a los nodos) o nodos (para aprobar o ejecutar transacciones). El siguiente ejemplo establece que las identidades emitidas por la autoridad de certificación cuyo certificado está en la ruta "cacerts/cacert.pem" y cuyo Organizational Unit (atributo OU del Domain Name del certificado) contenga "client" serán autenticadas como clientes, y si alberga la cadena "peer", como nodos:

```
NodeOUs:

  Enable: true

  ClientOUIdentifier:
    Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "client"

  PeerOUIdentifier:
    Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "peer"
```

Según Hyperledger, existe 4 tipos de roles:

- Admin, si una identidad posee un certificado que están dentro de la ruta AdminCerts de la configuración del MSP.
- Client o Peer, si el atributo OrganizationalUnit del certificado poseen los valores indicados en el fichero anterior, y es emitido por una autoridad de certificación de confianza del MSP.
- Member, cualquier certificado emitido por una autoridad de certificación de confianza del MSP.

En la arquitectura de Hyperledger, existen varios tipos de MSP, en función del ámbito al que se aplican. A continuación, se describirán desde un nivel más bajo a un nivel más alto. Los llamados MSPs locales, establecen los permisos para autenticar el origen de los mensajes de los nodos de la misma, fuera del contexto del canal, y para fijar los permisos de un componente particular (por ejemplo, para instalar chaincode en un nodo). Así, el MSP local de un cliente le permite autenticarse en una transacción como miembro del canal, o como el poseedor de un determinado rol en el sistema (por ejemplo, el rol de admin). Todos los nodos y los clientes han de tener definido un MSP local.

Otro tipo de MSP es el MSP de canal, que establece los permisos para operar sobre el canal. Así, para poder enviar transacciones al servicio de ordenación, es necesario el permiso de escritura sobre el canal. Igual sucede con el envío de propuestas de transacción a los nodos

aprobadores, el cual no es posible sin dicho permiso. El permiso de lectura sobre el canal es necesario, por ejemplo, para que un nodo se suscriba a recibir nuevos bloques de transacciones. Éste MSP se encarga de establecer el permiso para instanciar chaincodes en un canal. Los MSPs de canal definen derechos administrativos y de participación a nivel del canal. Cada organización que participa en un canal tiene que tener un MSP de canal definido. Estos MSPs tienen visibilidad a nivel de canal, y todos los nodos y clientes pueden observar estos MSPS y autenticarse contra ellos.

El rol de admin en un nodo no tiene por qué tener el rol admin en el canal. Es decir, un administrador de un nodo, que puede instalar chaincodes en el nodo, no tiene que ser el mismo que agregue nodos a un canal.

Por último, existe el MSP de red, que define quiénes son los miembros de la red (a través de la definición de los MSP de los organizaciones participantes) y qué miembros pueden realizar tareas administrativas (crear un canal).

Para un mejor entendimiento de los ámbitos de los MSP, podemos tomar el siguiente ejemplo de la ilustración 18.

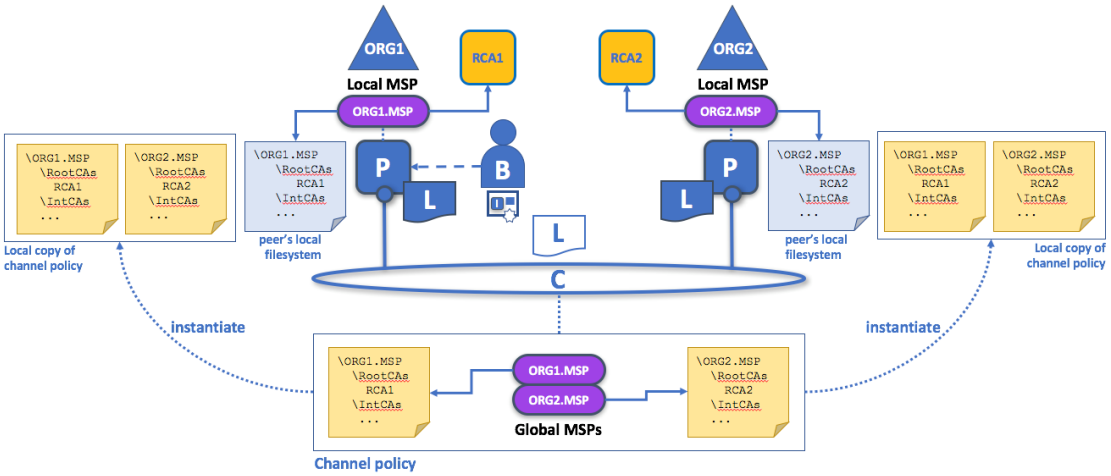

Ilustración 18 – Ámbito de un MSP, tomada de [34]

En este ejemplo, vemos que existen MSPs de canal (definidos como Global MSPs) llamados ORG1.MSP y ORG2.MSP, con su propia configuración. También existe MSPs locales, denominados también ORG1.MSP y ORG2.MSP, pero que son independientes de los anteriores, y que tienen cuyo Cas de confianza RCA1 y RCA2. Si el administrador B conecta con el peer P de ORG1 y desea instalar un chaincode en el nodo, éste comprobará su identidad contra el MSP local ORG1.MSP. Sin embargo, si desea instanciar dicho chaincode en el canal, el nodo tendrá que comprobar los permisos contra el MSP global ORG1.MSP. Esto es posible ya que los nodos tienen una copia sincronizada de la política del canal.

El estudio de las características de los MSPs de la arquitectura DVP de este documento se reflejará en el siguiente apartado, ya que los MSPs se han de declarar en la creación del canal.

3.4.1.8. Canales

Se trata de un mecanismo que refleja una subred entre miembros de una red Hyperledger. Se define por los miembros (organizaciones que poseen nodos), un ledger compartido, chaincodes y el servicio de ordenación. En el canal se ejecutan las transacciones, las cuales son creadas por miembros del canal autenticados y autorizados a través del servicio MSP. Este servicio además provee de identidad a los nodos dentro de un canal. Los componentes de un canal, por lo tanto, se ponen de acuerdo en colaborar para compartir y mantener copias idénticas del ledger asociado al canal.

Un canal es iniciado a través de un bloque génesis, que contiene la configuración del canal: políticas, organizaciones y "anchor peers" (nodos ancla) de cada organización. Este bloque se le proporciona a cada nodo que ingresa en el canal.

El encargado de crear un canal es el nodo de ordenación (el nodo en el que se ejecuta el servicio de ordenación). De forma previa, este nodo posee el bloque génesis. Al iniciar un sistema, y dentro del bloque génesis de un canal, se incluye los parámetros de verificación de todos los MSPs que intervienen, incluyendo identificador del MSP y todos los certificados necesarios. El nodo de ordenación, al tener este bloque génesis, es capaz de validar peticiones de creación de canal. Como se observa, la configuración correcta del bloque génesis es crucial a la hora de establecer la seguridad de un canal.

En nuestra plataforma DVP se crearán dos canales. Se toma esta decisión ya que manejan información que debe ser tratada de forma independiente por los distintos actores interesados. Dichos canales serían los siguientes:

- Canal Banco. Contiene el ledger que refleja las transacciones que modifican el estado de los activos Cuentas corrientes, así como su base de datos de estado asociada.
- Canal Mercado de valores. Contiene el ledger y la base de datos de estado asociada a los activos "valor de una empresa".

La ilustración 19 representa la relación entre canales, ledgers y activos.

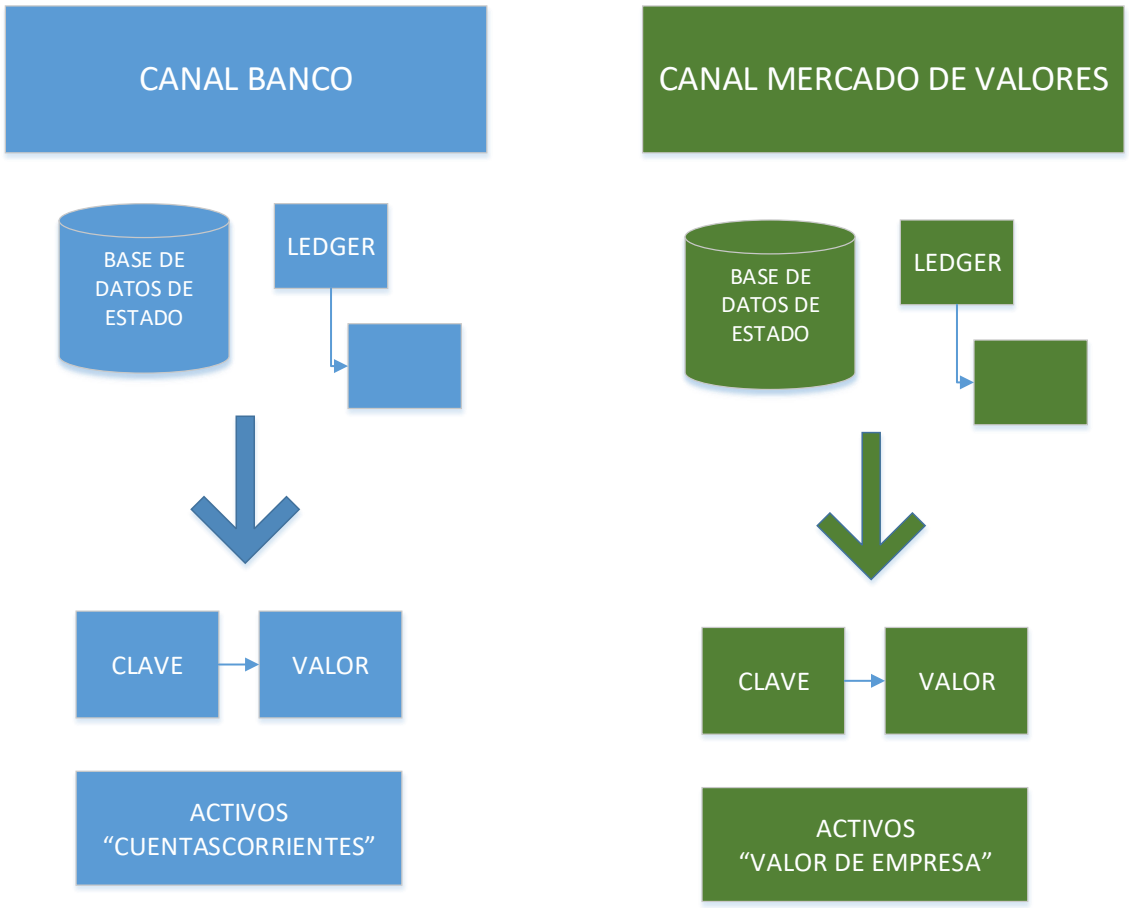

Ilustración 19 – Relación entre canales, ledgers y activos

Es importante explicar el funcionamiento de un tipo especial, los nodos ancla dentro de un canal. Como ser verá más adelante, la carga necesaria para el mantenimiento de las transacciones en Hyperledger se divide en dos ejecuciones: por un lado nodos que aprueban y comprometen transacciones, y por otros nodos que implementan el servicio de ordenación. Es necesario un protocolo de comunicación fiable, seguro y escalable entre los componentes y que asegure integridad y consistencia de los datos. Dicho protocolo se conoce con el nombre de "gossip". La función principal de este protocolo es distribuir los mensajes entre los nodos, aplicándole una firma digital a cada uno de ellos, con el objetivo de aportar integridad y detectar participantes bizantinos (es decir, que intentan enviar información modificada). Además se encarga de:

- descubrir, identificar y actualizar nuevos nodos y detectar si algún nodo no responde.
- diseminar una copia del ledger en un canal, para que los nodos tengan la información sincronizada.

El mecanismo de goosip consiste en que los nodos reemiten a otros nodos los mensajes que les llegan de forma continua, para permitir que la información del ledger y de la base de datos de estado está sincronizada. Los nuevos bloques creados por el servicio de información un nodo líder se encarga de transmitirlos. La elección de dicho nodo líder es elegida de forma estática por el administrador del canal, o de forma dinámica por los nodos.

A la hora de definir un mecanismo de elección de líder para nuestra solución DVP, elegimos la opción de asignación dinámica, por la tolerancia a fallos que representa. El funcionamiento de este mecanismo es el siguiente: el nodo líder envía un mensaje (llamado heartbeat) al resto de nodos para comunicar que se encuentra activo. Si durante un periodo de tiempo (configurable) no llegan los heartbeats, el resto de nodos inician una ronda de elección de líder. A modo de ejemplo, se indican las variables necesarias de los nodos para este funcionamiento, que se establecen en el archivo de configuración de cada uno de ellos:

```
peer:
    gossip:
        useLeaderElection: true
        orgLeader: false

    election:
        leaderAliveThreshold: 10s
```

Estas variables activan el mecanismo dinámico de elección de líder, y parametrizan el valor de espera por un heartbeat del líder a diez segundos.

Continuamos con el funcionamiento de los nodos ancla de una organización. Dichos nodos son usados por el protocolo goosip para asegurar que las distintas organizaciones de una red tienen conocimiento unas de otras. Los nodos conocen la topología de la red a través de consultas a los nodos ancla de las distintas organizaciones. En nuestro ejemplo, teniendo en cuenta las organizaciones Banco, Regulador y Depositario, suponemos que la organización depositario contiene el nodo ancla “ancla.depositario”. Cuando el nodo nodo1.banco contacta con ancla.depositario, le informará del nodo2.banco. Si más adelante el nodo nodo1.regulador comunica con ancla.depositario, éste le informará de nodo2.banco.

Por su necesidad, y para nuestra solución DVP, estableceremos dos nodos ancla por cada organización, por motivos de redundancia y disponibilidad. Es decir, las 4 organizaciones crearán dos nodos ancla en cada uno de los canales en los que intervengan. La forma de definir estos nodos ancla será a través del archivo de configuración del canal configtx.yaml (más exactamente, de la transacción de creación de canal). Se adjunta un ejemplo para mayor claridad, creando los dos nodos ancla de la organización “depositario” en uno de los dos canales:

```
Organizations:

    - &depositario
        Name: depositarioMSP
        # ID to load the MSP definition as
        ID: depositarioMSP

        MSPDir: crypto-config/peerOrganizations/depositario.mistic-
uoc.com/msp

        AnchorPeers:
```

```
            - Host: peer0. depositario. mistic-uoc.com
              Port: 7051
            - Host: peer1. depositario. mistic-uoc.com
              Port: 7051
```

Siguiendo con el análisis de seguridad de la arquitectura DVP, un punto importante es la configuración del control de acceso en un canal. Hyperledger emplea listas de control de acceso (ACL, a partir de ahora) para gestionar el permiso en un canal, asociando políticas a recursos, como puede ser un chaincode. Las políticas especifican reglas a evaluar sobre un conjunto de identidades. Por defecto, Hyperledger proporciona una serie de políticas.

Las políticas se especifican en el archivo "configtx.yaml", es decir, el archivo de configuración del canal, e incluyen políticas de modificación de información y de control de acceso. Las políticas de aprobación de una transacción se definen a la hora de instanciar un chaincode, como se verá más adelante.

Las políticas definidas pueden ser de dos tipos:

- Signature: se refieren a que las identidades especificadas deben crear una firma para que la política se apruebe:

```
Policies:
     PolíticaEjemplo:
          Type: Signature
          Rule: "Banco.Peer OR  Regulador.Peer"
```

En este ejemplo, se cumplirá la política "PoliticaEjemplo" con la firma de una identidad que tenga el rol "peer" de la organización Banco o de la organización Regulador.

- ImplicitMeta: Se trata de políticas más complejas, que pueden agregar un conjunto de políticas de tipo "Signature" o referirse a roles en el canal. Un ejemplo sería:

```
Policies:
  PolíticaEjemplo2:
    Type: ImplicitMeta
    Rule: "MAJORITY Admins"
```

Siendo los roles referenciados en el propio canal los que poseen el rol "Admins".

En la relación acciones y políticas, se establecen cláusulas del tipo Recurso:Identidad. En el siguiente ejemplo se establece el permiso para invocar chaincodes en un nodo. Dicho recurso se denomina "peer/propose", y en el ejemplo se define que solo puedan invocarlo un administrador de la organización "Banco":

```
Policies: &ApplicationDefaultPolicies
    PoliticaChaincodeGestionBanco:
        Type: Signature
```

```
        Rule: "OR(Banco.admin')"
peer/Propose: /Channel/Application/PoliticaChaincodeGestionBanco
```

Una vez explicado el concepto de ACLs en Hyperledger, hay que retomar el concepto de tipos de MSPs explicado en el capítulo anterior. El MSP de canal define permisos sobre distintos recursos y acciones que las identidades pueden ejecutar sobre el canal. Dichos permisos están establecidos en un fichero especial denominado "configtx.yaml", y presenta la siguiente estructura:

```
1-/Channel/{Readers,Writers,Admins}
2-/Channel/Orderer/{Readers,Writers,Admins }
3-/Channel/Applications/{Readers,Writers,Admins}
4-/Channel/Applications/OrgName/{Readers,Writers,Admins}
```

Es decir:

- 1-Políticas de permisos de lectura, escritura y administración específicos sobre el canal.
- 2- Políticas de permisos de lectura, escritura y administración específicos sobre el nodo de ordenación.
- 3- Políticas de permisos de lectura, escritura y administración específicos sobre los chaincodes.
- 4-Políticas de permisos de lectura, escritura y administración específicos sobre cada uno de las organizaciones.

A la hora de establecer la relación entre organizaciones, canales y MSPs en nuestra solución DVP, se tendrán en cuenta las siguientes decisiones:

- Cada organización tendrá su propio MSP y jerarquía de autoridades de certificación, por privacidad y para que cada una pueda gestionar la propiedad de los nodos.
- En el canal Banco se registrarán los MSP de las organizaciones Banco, Regulador y Depositario.
- En el canal Mercado de Valores se registrarán los MSP de las organizaciones Mercado de Valores, Regulador y Depositario.

El esquema, por lo tanto, se configurará como se muestra en la ilustración 19.

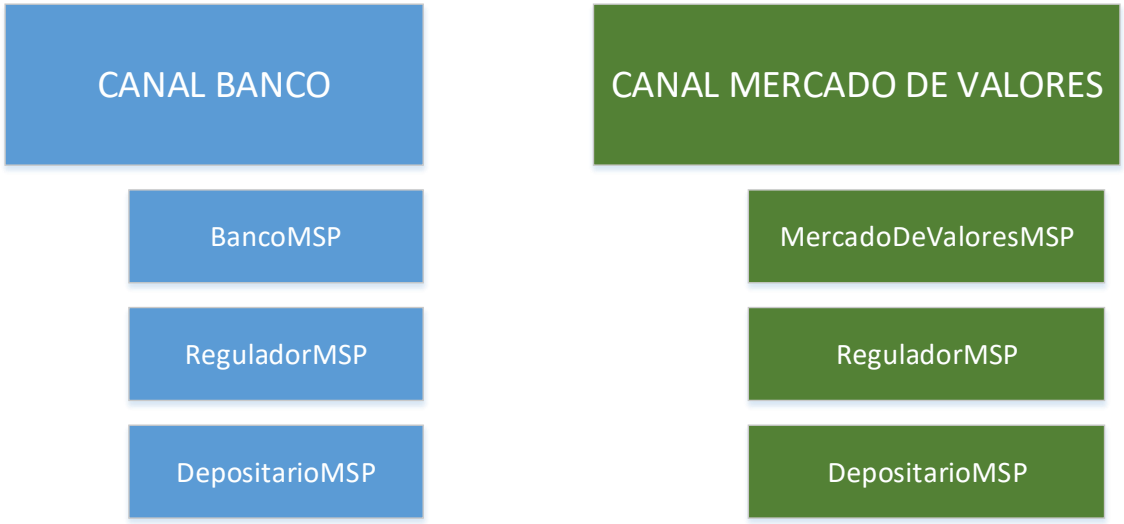

Ilustración 20 – Relación entre canales y MSPs

Para crear los parámetros criptográficos de cada MSP, se empleará la herramienta cryptogen, ofrecida por la suite de Hyperledger. Utiliza un fichero de configuración llamada crypto-config.yaml, que contiene la topología de la red y permite generar los certificados y claves para cada organización. Crea una autoridad de certificación para cada organización, la cual emitirá los certificados para los nodos estándar y los nodos de ordenación de dicha organización. Tiene una variable llamada Count, que especifica el número de nodos (variable Template:Count) y el número de clientes (variable Users:Count) para los que esta herramienta creará las claves y certificados necesarios. Para un mejor entendimiento de este proceso, se adjunta el fichero crypto-config.yaml.

En primer lugar, se muestran los parámetros para la autoridad de certificación y el certificado del nodo de ordenación. Después, se indican los detalles para cada autoridad de certificación de cada organización, y el número de nodos a los que hay que crear una clave y un certificado. Por defecto, esta herramienta creará la configuración de todos los MSPs del sistema: locales a nodos, clientes, administradores, y de canal para cada organización. Para cada MSP creará claves, certificados, certificados TLS, y almacén de ACs de confianza.

```
OrdererOrgs:
  - Name: Orderer
    Domain: mistic-uoc.com
    CA:
        Country: ES
        Province: Barcelona
        Locality: Barcelona
    Specs:
      - Hostname: orderer

PeerOrgs:
  - Name: banco
    Domain: banco.mistic-uoc.com
    EnableNodeOUs: true
    CA:
```

```
      Country: ES
      Province: Barcelona
      Locality: Barcelona
    Template:
      Count: 1
    Users:
      Count: 1
  - Name: depositario
    Domain: depositario.mistic-uoc.com
    EnableNodeOUs: true
    CA:
        Country: ES
        Province: Barcelona
        Locality: Barcelona
    Template:
      Count: 1
    Users:
      Count: 1
   - Name: regulador
    Domain: regulador.mistic-uoc.com
    EnableNodeOUs: true
    CA:
      Country: ES
      Province: Barcelona
      Locality: Barcelona
    Template:
      Count: 1
    Users:
      Count: 1
```

En cuanto a las reglas de control de acceso para nuestra solución DVP, se definirán las siguientes políticas:

CANAL BANCO

Organización Banco:

-Permiso de administración, de tipo Signature, para cualquier usuario que se autentique con el certificado Admin del bancoMSP. Es necesario para la administración de los nodos de la organización, por ejemplo, para instalar chaincodes en los nodos de esta organización.

-Permiso de Escritura, de tipo Signature, para cualquier usuario que se autentique con un certificado tipo Peer o tipo Client del bancoMSP (en el capítulo anterior habíamos definido que los tipos Peers o Client se establecían mediante la OU del certificado).

-Permiso de lectura, de tipo Signature, para cualquier usuario que se autentique con un certificado tipo Member del bancoMSP (se había definido como tipo Member cualquier certificado emitido por la AC de confianza de bancoMSP).

Organización Regulador:

-Permiso de administración, de tipo Signature, para cualquier usuario que se autentique con el certificado Admin del ReguladorMSP. Es necesario para la administración de los nodos de la organización.

-Permiso de Escritura, de tipo Signature, para cualquier usuario que se autentique con un certificado tipo Peer o tipo Client del ReguladorMSP.

-Permiso de lectura, de tipo Signature, para cualquier usuario que se autentique con un certificado tipo Member del ReguladorMSP.

Organización Depositario:

-Permiso de administración, de tipo Signature, para cualquier usuario que se autentique con el certificado Admin del DepositarioMSP. Es necesario para la administración de los nodos de la organización.

-Permiso de Escritura, de tipo Signature, para cualquier usuario que se autentique con un certificado tipo Peer o tipo Client del DepositarioMSP.

-Permiso de lectura, de tipo Signature, para cualquier usuario que se autentique con un certificado tipo Member del DepositarioMSP.

Permisos del nodo de ordenación (orderer):

-Permiso de administración, de tipo Signature, para cualquier usuario que se autentique con el certificado Admin del bancoMSP. Se trata de la entidad que va a mantener este canal, y por lo tanto, es responsable de su servicio de ordenación.

-Permiso de Escritura, de tipo Signature, para cualquier usuario que se autentique con un certificado tipo Peer o tipo Client del bancoMSP.

-Permiso de lectura, de tipo Signature, para cualquier usuario que se autentique con un certificado tipo Member del bancoMSP.

Permisos sobre las aplicaciones:

-Permiso de administración, de tipo Signature, para cualquier usuario que se autentique con el certificado Admin del bancoMSP. Se trata de la entidad que va a mantener este canal, y por lo tanto, es responsable de la administración de las aplicaciones, por ejemplo, de su actualización.

-Permiso de Escritura, de tipo “ImplicitMeta”, para cualquier identidad que posea el permiso de escritura en otro tipo de permiso (es decir, for ANY of /Channel/Application/*/Writers). De esta forma, establecemos los permisos necesarios para las organizaciones de forma global

-Permiso de lectura, de tipo “ImplicitMeta”, para cualquier identidad que posea el permiso de lectura en otro tipo de permiso. (es decir, for ANY of /Channel/Application/*/Readers). De esta forma, establecemos los permisos necesarios para las organizaciones de forma global

Permisos sobre el canal:

-Permiso de administración, de tipo Signature, para cualquier usuario que se autentique con el certificado Admin del bancoMSP. Se trata de la entidad que va a mantener este canal. Así mismo, las identidades que se autentiquen con el certificado Admin de ReguladorMSP o DepositarioMSP tendrán permiso de administración, ya que van a requerir instanciar chaincodes (como se indicará en el apartado "políticas de instanciación y aprobación").

-Permiso de Escritura. Es necesario para poder enviar transacciones de actualización al ledger, por lo que solo se podrán enviar para las organizaciones Depositario o Banco, por lo que se será de tipo Signature y limitada para los certificados de tipo client o peer de los MSP BancoMSP o DepositarioMSP.

-Permiso de lectura, de tipo "ImplicitMeta", para cualquier identidad que posea el permiso de lectura en otro tipo de permiso. (es decir, for ANY of /Channel/Application/*/Readers). De esta forma, establecemos los permisos necesarios para las organizaciones de forma global

Para el canal "Mercado de valores" se establecerán de forma análoga teniendo en cuenta la sustitución de la organización Banco por la organización "Mercado de Valores".

Finalmente, en la ilustración 21 se muestra la imagen global, teniendo en cuenta los distintos tipos de MSP que intervienen en cada componente de la solución DVP, las Autoridades de Certificación asociadas y los canales y organizaciones. Se ha indicado que el actor que regula la red es la organización Regulador. La forma de implementar esto se observará en el capítulo "Consorcio de organizaciones".

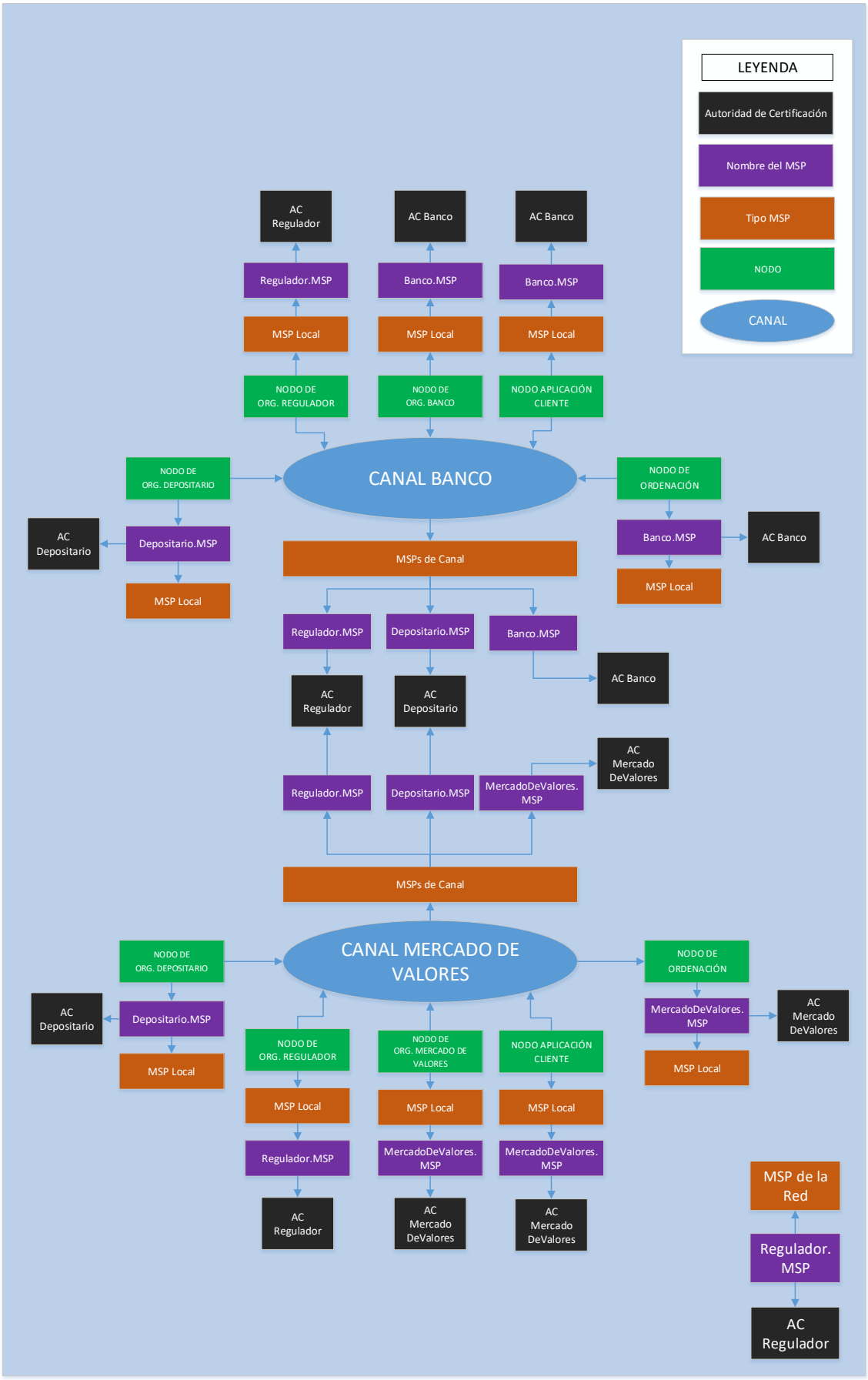

Ilustración 21 – Relación entre todos los actores involucrados

3.4.1.9. Servicio de ordenación

Este servicio, ejecutado en un tipo especial de nodo (nodo de ordenación), no procesa transacciones, ni ejecuta chaincodes o mantiene una copia del ledger. Solo acepta las transacciones aprobadas, las ordena en bloques y envía dichos bloques a todos los nodos estándar (sean aprobadores o no).

En los anteriores apartados ya se ha descrito la funcionalidad de este servicio a la hora de intervenir en el ciclo de una transacción.

Este servicio puede ser configurado de dos formas, mediante módulos conectables. Dichos módulos se conocen por el mecanismo que emplean. A través de un parámetro en la configuración del canal:

```
# Available types are "solo" and "kafka"
OrdererType: kafka
```

Podemos definir el tipo de ordenación. Se explican los dos valores:

- SOLO. Se trata de un método para una red en el que solo existe un servicio de ordenación en un único nodo. No es un método válido para redes de producción, ya que no es tolerante a fallos. Solo se emplea en entornos de prueba.
- Kafka. Emplea el manejador de mensajes desarrollado por la Apache Software Foundation llamado "Apache Kafka". En primer lugar, explicaremos este sistema. Su modelo es el de consumidor-productor: unos actores se suscriben a un tema determinado (denominado topic), y otros actores producen mensajes relacionados con tema. Los productores de mensajes mantienen particiones por cada tema. Los consumidores mantienen un proceso por cada tema de los que se han suscrito. Kafka garantiza que todos los mensajes dentro de una partición están ordenados, y se garantiza dicho orden hasta el final. Esta ordenación es clave para Hyperledger, más adelante veremos porqué. Los mensajes se mantienen una vez que los consumidores los han leído, por lo que se permite volver a procesarlos en caso de error. Al existir un proceso por tema, existe un control sobre los mensajes leídos.

En la ilustración 22 vemos una estructura básica de Kafka:

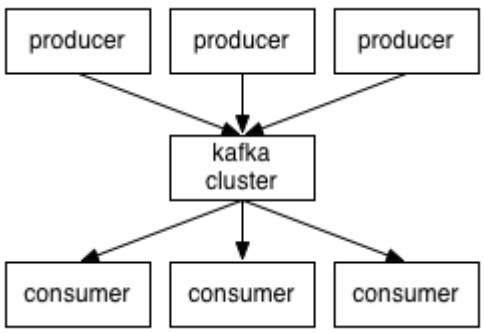

Ilustración 22 – Estructura básica de Kakfa, obtenida de [31]

Su arquitectura es distribuida, replicada y particionada, por lo que es escalable y tolerante a fallos. El hecho de que existan particiones replicadas entre los productores es el responsable de que sea tolerante a fallos, en caso de caída de un productor. Existe un productor líder, que se encarga de gestionar los mensajes de una partición. Si dicho productor cae, uno de los productores que mantienen la partición replicada asumirá el control.

Un servicio denominado Zookeeper se encarga de indicar a qué productor tienen que dirigirse los consumidores para consumir los mensajes; es decir, informará qué productor es el líder para un tema. Sin entrar en detalle de la implementación de este servicio, hay que indicar está proporcionado por servidores Zookeeper, los cuales son tolerante a fallos.

Retomando Hyperledger, explicaremos cuál es la función de Kafka en la red. Para cada canal, existe una partición. El servicio de ordenación (ejecutado desde el nodo de ordenación) retransmite las propuestas de transacción de los clientes en un determinado canal, a la partición correspondiente del canal. Kafka se encarga de ordenar los mensajes en una partición, por lo que cada canal mantiene una lista ordenada de transacciones. En una red donde existe varios nodos de ordenación, la lista ordenada de transacciones es común para todos los nodos de ordenación gracias a Kafka. Los nodos obtienen esta lista ordenada de transacciones para componer los bloques. En caso de caída de un nodo de ordenación, otro nodo de ordenación puede enviar los bloques.

En la referencia [33] se pueden conocer a bajo detalle las características de la funcionalidad de Kafka en Hyperledger Fabric.

En el roadmap de Hyperledger queda la inclusión de SBFT, como se puede comprobar en [35] "In-process with peer (system chaincode) Pluggable consensus with PBFT". Son las siglas de "Simplified Byzantine Fault Tolerance", sistema tolerante a fallos bizantino simplificado. Se trata de la implementación del mecanismo de consenso identificado en el marco teórico, que es tolerante a fallos y es capaz de llegar a un consenso incluso con la presencia de nodos maliciosos. En la referencia [32] se encuentra un estudio sobre un servicio de ordenación tolerante a fallos bizantinos para Hyperledger Fabric.

Sin embargo, y como hemos descrito anteriormente, Hyperledger tiene una tolerancia a fallo bizantino parcial y de facto. Si un nodo decide modificar su réplica de la base de datos de estado, en la siguiente propuesta de transacción que quiera acceder a sus datos, la red detectará que la respuesta de transacción no es idéntica que el resto de nodos y puede ser expulsado. Para reforzar esta medida de seguridad es recomendable que en nuestra solución se implemente una política de aprobación que incluya todos los nodos, ya que una propuesta de transacción que acceda a datos modificados en un nodo no logrará el consenso, y se podrá detectar al actor malicioso.

Para el diseño de una solución DVP, el mecanismo SOLO no es viable, ya que no proporciona un método de tolerancia a fallos, y cualquier caída del nodo de ordenación provocará un fallo grave en la plataforma.

Hyperledger proporciona imágenes para servidores Kafka y Zookeeper. La documentación oficial recomienda tener 4 servidores Kafka en cada canal para proveer tolerancia a fallo. El número de servidores Zookeeper ha de ser impar, ya que así evitar un escenario de "Split-brain": una situación en la que dos o más partes del sistema hacen cambios

de forma independiente e inconsistente. Siendo un número impar, se llegaría a un quorum sobre el productor al que tienen que suscribirse los nodos para recibir mensajes.

Como conclusión, para nuestra solución DVP, se definiría la siguiente configuración del servicio de ordenación. En primer lugar, se indica la configuración del tipo de consenso en el canal (Kafka), y de los nodos de ordenación en modo tolerancia a fallos. Después, se observa la configuración de 4 nodos de ordenación, y de 4 productores (brokers) de mensajes Kafka. A continuación se indica la configuración de los nodos Kafka. No se pretende entrar en detalles de la configuración de la plataforma. Se indica solamente la configuración del primer broker (kafka0) para observar la dependencia de los nodos zookeeper. Se obvia la configuración de los brokers Kafka1, Kafka2 y kakfa3. Justo después se indica la configuración del primer nodo de ordenación (orderer0), para observar la dependencia con los brokers kafka. En los volúmenes de este nodo, se observa que monta la ruta local donde se almacenan el MSP local de este nodo, y los certificados TLS. En la prueba de concepto se detallarán el significado del montaje de volúmenes. Se obvia la configuración de los nodos orderer1, orderer2 y orderer3.

```
Orderer:
    OrdererType: kafka
    Addresses:
        - orderer0.mistic-uoc.com:7050
        - orderer1.mistic-uoc.com:7050
        - orderer2.mistic-uoc.com:7050
        - orderer3.mistic-uoc.com:7050
……
    Kafka:
        Brokers:
            - kafka0.mistic-uoc.com:9092
            - kafka1.mistic-uoc.com:9092
            - kafka2.mistic-uoc.com:9092
            - kafka3.mistic-uoc.com:9092
…..
kafka0.mistic-uoc.com:
        extends:
            file: docker-compose-base.yml
            service: kafka
        container_name: kafka0.mistic-uoc.com
        environment:
            - KAFKA_BROKER_ID=0
            - KAFKA_ZOOKEEPER_CONNECT=zookeeper0.mistic-
uoc.com:2181,zookeeper1.mistic-uoc.com:2181,zookeeper2.mistic-
uoc.com:2181
            - KAFKA_MESSAGE_MAX_BYTES=${KAFKA_MESSAGE_MAX_BYTES}
            -
KAFKA_REPLICA_FETCH_MAX_BYTES=${KAFKA_REPLICA_FETCH_MAX_BYTES}
            -
KAFKA_REPLICA_FETCH_RESPONSE_MAX_BYTES=${KAFKA_REPLICA_FETCH_RESPONS
E_MAX_BYTES}
        depends_on:
            - zookeeper0.mistic-uoc.com
            - zookeeper1.mistic-uoc.com
```

```
            - zookeeper2.mistic-uoc.com
        networks:
          behave:
             aliases:
               - ${CORE_PEER_NETWORKID}
…..
orderer0.mistic-uoc.com:
        extends:
            file: docker-compose-base.yml
            service: orderer
        container_name: orderer0.mistic-uoc.com
        environment:
            - ORDERER_HOST=orderer0.mistic-uoc.com
            - CONFIGTX_ORDERER_ORDERERTYPE=kafka
            - CONFIGTX_ORDERER_KAFKA_BROKERS=[kafka0.mistic-
uoc.com:9092,kafka1.mistic-uoc.com:9092,kafka2.mistic-
uoc.com:9092,kafka3.mistic-uoc.com:9092]
            - ORDERER_KAFKA_RETRY_SHORTINTERVAL=1s
            - ORDERER_KAFKA_RETRY_SHORTTOTAL=30s
            - ORDERER_KAFKA_VERBOSE=true
            - ORDERER_GENERAL_GENESISPROFILE=SampleInsecureKafka
            - ORDERER_ABSOLUTEMAXBYTES=${ORDERER_ABSOLUTEMAXBYTES}
            - ORDERER_PREFERREDMAXBYTES=${ORDERER_PREFERREDMAXBYTES}
        volumes:
            -
../configs/${CORE_PEER_NETWORKID}/ordererOrganizations/mistic-
uoc.com/orderers/orderer0.mistic-uoc.com/msp:/var/Hyperledger /msp
            -
../configs/${CORE_PEER_NETWORKID}/ordererOrganizations/mistic-
ouc.com/orderers/orderer0.mistic-uoc.com/tls:/var/Hyperledger /tls
        depends_on:
            - kafka0.mistic-uoc.com
            - kafka1.mistic-uoc.com
            - kafka2.mistic-uoc.com
            - kafka3.mistic-uoc.com
        networks:
          behave:
             aliases:
               - ${CORE_PEER_NETWORKID}
        ports:
          - 7050:7050
…….
```

A continuación se indica la configuración de los servidores Zookeeper. Se indica solo la configuración del nodo zookeeper0, obviando los nodos zookeper1 y zookeper2. Como se observa, se establece un clúster de nodos.

```
zookeeper0.mistic-uoc.com:
        extends:
```

```
            file: docker-compose-base.yml
            service: zookeeper
        container_name: zookeeper0.mistic-uoc.com
        environment:
            - ZOO_MY_ID=1
            - ZOO_SERVERS=server.1=zookeeper0.mistic-
uoc.com:2888:3888 server.2=zookeeper1.mistic-uoc.com:2888:3888
server.3=zookeeper2.mistic-uoc.com:2888:3888
        networks:
          behave:
             aliases:
               - ${CORE_PEER_NETWORKID}
```

3.4.1.10. Consenso

El consenso en Hyperledger se define como el mecanismo por el que una red de nodos proporciona una ordenación garantizada de transacciones y valida bloques de transacciones.

Se materializa a través del flujo completo de una transacción, ya que todos los nodos han alcanzado un acuerdo en la ejecución y en el contenido de las transacciones, y se establece el orden de las transacciones en un proceso mediado por el servicio de ordenación.

Se puede afirmar que el consenso en Hyperledger se consigue en 4 fases:

- Aprobación: El cliente envía a los nodos aprobadores una propuesta de transacción, y a través de la política de aprobación del chaincode, se consiguen que los nodos firmen las respuestas de la propuesta de transacción.
- Ordenación: En el nodo de ordenación, se llega a un acuerdo para ordenar las transacciones aprobadas. Dicho orden es en el que se comprometerán en el ledger.
- Validación: Los nodos reciben las transacciones en forma de bloque de ítems ordenados, y se valida que los resultados sean correctos (se vuelve a comprobar el cumplimiento de la política de aprobación).
- Compromiso. Los nodos, después de la validación, se aseguran que versión que tienen de la base de datos de estado es correcta (no ha habido cambios desde que se inició la transacción), actualizan la base de datos de estado local y agregan el bloque recibido al ledger.

Junto con esta serie de comprobaciones, existen verificaciones de identidad y autorización entre todos los actores en el ciclo de vida de la transacción, a través de firmas y listas de control de acceso, como ya se ha explicado en apartados anteriores. Así mismo, se ha desarrollado en el capítulo de “Servicio de ordenación”, las configuraciones posibles que admite Hyperledger para alcanzar el consenso, al estar tan íntimamente ligados estos dos conceptos.

3.4.1.11. Consorcio de organizaciones

Hyperledger permite la creación de consorcios de organizaciones. El sentido es agrupar organizaciones que tienen necesidad de negociar entre sí, y de esta forma establecer las reglas que gobiernan la red.

El consorcio se define en los archivos de configuración del canal. Quedaría definido de la siguiente forma:

- Canal Banco:

```
Consortiums:
          BancoConsortium:
              Organizations:
                  - *banco
                  - *regulador
                  - *depositario
 canalBanco:
     Consortium:
               BancoConsortium:
                      ChannelCreationPolicy:
                              Admins:
                              Type: Signature
                              Rule: "OR('ReguladorMSP.admin')"
     Application:
         <<: *ApplicationDefaults
         Organizations:
             - *banco
             - *regulador
             - *depositario
```

- Canal Mercado de Valores:

```
Consortiums:
          MercadodevaloresConsortium:
              Organizations:
                  - *mercadodevalores
                  - *regulador
                  - *depositario
 CanalMercadoDeValores:
          Consortium:
                MercadodevaloresConsortium:
                      ChannelCreationPolicy:
                              Admins:
                              Type: Signature
                              Rule: "OR('ReguladorMSP.admin')"
     Application:
         <<: *ApplicationDefaults
         Organizations:
             - * mercadodevalores
             - *regulador
```

```
                - *depositario
```

Como se había indicado en apartados anteriores, la organización "Regulador" es el encargado de establecer el comportamiento de la red. Para ello, y a través de la política "ChannelCreationPolicy", se establece que solamente los administradores del MSP correspondiente a esta organización son los encargados de poder crear los canales. Mediante la herramienta "configtxgen", se creará una propuesta de transacción para los nodos de transacción, que contiene las políticas de control de acceso definidas en apartados anteriores, y la información del consorcio indicada en este apartado, y se firmará por un administrador de la organización "Regulador".

3.4.1.12. Políticas de instanciación y aprobación

Como se ha definido anteriormente, cada chaincode tiene una política asignadas unas políticas de instanciación y de aprobación. La política de instanciación establece qué identidades puede instanciar un chaincode en un canal. El concepto de instanciación se ha definido durante el apartado "Chaincodes". La política de aprobación, en cambio, define los nodos de un canal que tienen que aprobar los resultados de la ejecución de un programa, para que la transacción se considere válida. Dicho conjunto de nodos, también se comprueba en el número de firmas necesarias para que una propuesta de transacción se comprometa.

La política de instanciación se define durante la creación del paquete que contiene el chaincode. A modo de ejemplo, esta sería la sentencia que crear el paquete para el chaincode "GestionBanco", y cuya política de instanciación estaría permitida para la identidad admin del MSP "BancoMSP

```
peer chaincode package -n GestionBanco -p chaincode/GestionBanco -v
0 -s -S -i "AND('BancoMSP.admin')" GestionBanco.out
```

A modo explicativo, dada la importancia de este concepto en Hyperledger, se muestra un ejemplo de creación de política. La siguiente llamada al API de Hyperledger instancia el programa "GestionBanco" en el canal "canalBanco" con la política "es necesaria la firma de un nodo de la organización Banco y otro de la organización Regulador".

```
peer chaincode instantiate -C canalBanco -n GestionBanco -P
"AND('Banco.peer', Regulador.peer')"
```

Como se ha indicado anteriormente, el servicio MSP es el encargado de dotar de identificadores a los nodos en el canal, así como de roles, conocidos en Hyperledger como "principals". Estos valores pueden ser "admin", "member", "client" o "peer", para definir los distintos tipos de nodos en un canal.

Existe un segundo tipo de políticas de aprobación referidas a los pares clave-valor de la base de datos de estado, para establecer un grado mayor de granularidad. En este caso hay que invocar al SDK de Hyperledger para definirlas. En el mismo se define un objeto KeyEndorsementPolicy, que encapsula esta función, y métodos AddOrgs(), DeleteOrgs() para interactuar con la lista de miembros necesarios para aprobar un cambio de un par clave-valor.

Para el diseño de la arquitectura DVP de este documento, solo se tendrá en cuenta las políticas de aprobación relativas a los chaincodes. En las siguientes tablas se definirán los responsables de las políticas de instanciación y aprobación necesarias para cada uno de los chaincodes definidos en apartados anteriores.

Nombre del Chaincode	**GestionBanco**
Regla para la política de instanciación	AND('BancoMSP.admin')
Descripción de la política de instanciación	
El chaincode GestionBanco encapsula la funcionalidad para interactuar con los activos CuentaCorriente desde un punto de vista de la organización Banco, por lo que es el que debe instanciar e iniciar este programa.	
Regla para la política de aprobación	AND('BancoMSP.peer', 'BancoMSP.peer', 'ReguladorMSP.peer')
Descripción de la política de aprobación	
Se ha definido esta regla de aprobación para modelar que la organización Regulador de su confirmación cualquier operación que se ejecute sobre una transacción. Además, se requiere la aprobación de dos nodos de la organización Banco para evitar la actividad maliciosa de uno de los nodos.	

Nombre del Chaincode	**GestionIntercambioDVP**
Regla para la política de instanciación	AND(DepositarioMSP.admin')
Descripción de la política de instanciación	
El chaincode GestionIntercambioDVP encapsula la funcionalidad para interactuar con los activos CuentaCorriente para ejecutar las acciones de la organización Depositario, con el objetivo de bloquear y transferir cantidades asociadas a los valores. Por este motivo, esta organización será la encargada de instanciar este chaincode.	
Regla para la política de aprobación	AND('DepositarioMSP.peer', 'BancoMSP.peer', 'ReguladorMSP.peer')
Descripción de la política de aprobación	
Se ha definido esta regla de aprobación para modelar que las operaciones de transferencia de líquido orientadas a soportar la propiedad de un valor (bloqueo y transmisión de líquido) requieran la aprobación de las tres organizaciones implicadas en este canal: Depositario, Banco y Regulador, y ninguna de ellas se comporte de forma fraudulenta.	

Nombre del Chaincode	**ServiciosDepositario**
Regla para la política de instanciación	AND(DepositarioMSP.admin')
Descripción de la política de instanciación	
El chaincode ServiciosDepositario se encarga de ofrecer servicios a los inversores por parte del Depositario, a la hora de solicitar valores. Por este motivo, esta organización será la encargada de instanciar este chaincode.	
Regla para la política de aprobación	AND('DepositarioMSP.peer', 'BancoMSP.peer', 'ReguladorMSP.peer')
Descripción de la política de aprobación	

Se ha definido esta regla de aprobación para permitir que las organizaciones implicadas en este canal: Depositario, Banco y Regulador aprueben las transacciones que implican una acción de petición de adquisición de un valor por parte de un inversor a un precio determinado.

Nombre del Chaincode	**AuditoriaBanco**
Regla para la política de instanciación	AND('ReguladorMSP.admin')
Descripción de la política de instanciación	
El chaincode AuditoriaBanco encapsula la funcionalidad que debe poseer la organización Regulador para auditar las acciones del banco, por lo que esta organización será la que tiene el permiso de instanciar este chaincode.	
Regla para la política de aprobación	AND('ReguladorMSP.peer', 'ReguladorMSP.peer')
Descripción de la política de aprobación	
Los métodos de este chaincode solo de tipo lectura (invocados desde la función Query ()) por lo que las invocaciones a los mismos no se traducirán a transacciones. Se ha definido esta regla de aprobación para que en las lecturas se reciban las respuestas de dos nodos**.** Se ha definido esta regla de aprobación para que sea necesaria la consulta a dos nodos de la organización Regulador, para evitar comportamientos maliciosos de los mismos o bases de datos de estado desactualizadas.	

Nombre del Chaincode	**GestionValores**
Regla para la política de instanciación	AND('MercadodeValoresMSP.admin')
Descripción de la política de instanciación	
El chaincode GestionValores encapsula la funcionalidad para interactuar con los activos ValoresdeEmpresa desde un punto de vista de la organización MercadoDeValores, por lo que es el que debe instanciar e iniciar este programa.	
Regla para la política de aprobación	AND('MercadodeValoresMSP.peer', 'MercadodeValoresMSP.peer', 'ReguladorMSP.peer')
Descripción de la política de aprobación	
Se ha definido esta regla de aprobación para modelar que la organización Regulador de su confirmación cualquier operación que se ejecute sobre una transacción. Además, se requiere la aprobación de dos nodos de la organización MercadoDeValores para evitar la actividad maliciosa de uno de los nodos.	

Nombre del Chaincode	**AuditoriaMercadoDeValores**
Regla para la política de instanciación	AND('ReguladorMSP.admin')
Descripción de la política de instanciación	
El chaincode AuditoriaMercadoDeValores encapsula la funcionalidad que debe poseer la organización Regulador para auditar las acciones que se desarrollan en el mercado de valores, por lo que esta organización será la que tiene el permiso de instanciar este chaincode.	
Regla para la política de aprobación	AND('ReguladorMSP.peer', 'ReguladorMSP.peer')
Descripción de la política de aprobación	

Los métodos de este chaincode solo de tipo lectura (invocados desde la función Query()) por lo que las invocaciones a los mismos no se traducirán a transacciones. Se ha definido esta regla de aprobación para que en las lecturas se reciban las respuestas de dos nodos. Se ha definido esta regla de aprobación para que sea necesaria la consulta a dos nodos de la organización Regulador, para evitar comportamientos maliciosos de los mismos o bases de datos de estado desactualizadas.

Nombre del Chaincode	**GestionPropiedadValores**
Regla para la política de instanciación	AND(DepositarioMSP.admin')
Descripción de la política de instanciación	
El chaincode GestionPropiedadValores encapsula la funcionalidad para interactuar con los activos ValorDeEmpresa para ejecutar las acciones de la organización Depositario, con el objetivo de reservar y asignar la propiedad de un valor emitido por una empresa. Por este motivo, esta organización será la encargada de instanciar este chaincode.	
Regla para la política de aprobación	AND('DepositarioMSP.peer', 'MercadodeValoresMSP.peer', 'ReguladorMSP.peer')
Descripción de la política de aprobación	
Se ha definido esta regla de aprobación para modelar que las operaciones de gestión de propiedad de un valor (reserva y asignación) requieran la aprobación de las tres organizaciones implicadas en este canal: Depositario, MercadoDeValores y Regulador, y ninguna de ellas se comporte de forma fraudulenta.	

3.4.1.13. Mecanismos de baja de identidades

En este apartado se detallan los métodos para comunicar a la arquitectura DVP la baja de identidades.

Habría que distinguir dos casos, la revocación de una identidad concreta, y la revocación de una autoridad de certificación.

En caso de revocación de una autoridad de certificación, hay que reconfigurar el MSP para no admitir más certificados emitidos por esta autoridad. Existirán dos formas de hacerlo:

- Eliminar esta autoridad de certificación de la lista de certificados de confianza.
- Solicitar a la autoridad raíz de certificación una lista de certificados revocados actualizada, con este certificado incluido.

En el caso de revocar una identidad, se solicitará a la autoridad de certificación una copia de la CRL donde figure el certificado revocado, y se incluirá la CRL en el apartado correspondiente de cada MSP relacionado.

4. Diseño del Mecanismo DVP e interacción entre canales

En este capítulo describiremos el procedimiento de intercambio entre las partes interesadas que se produce en el sistema DVP. Recordemos de nuevo los actores:

En primer lugar, nos encontramos con una primera parte interesada, la organización Banco, representando a su cliente. Este actor ofrece en el intercambio una cantidad de dinero pactada, y maneja una información que no tiene que ser pública (el universo de los datos de los saldos y propietarios de las cuentas) salvo a las partes interesadas. Dichas partes pueden tener un interés en conocer cierta información ya que persiguen un fin legítimo, como es el caso de la Empresa que ofrece un valor a cambio de ese dinero, y está representada por la organización Depositario, o por razones de auditoría, como es el caso de la organización Regulador.

La segunda parte interesada es la organización mercadoDeValores, representando a una empresa que pone en el mercado valores, la cual ofrece la propiedad de un valor a un precio pactado. Sucede lo mismo que en el caso anterior, ya que toda la información no tiene porqué ser pública, salvo a las partes que persiguen un fin legítimo o por razones de auditoría. La información de los valores puestos en el mercado no tiene que ser totalmente pública ya que se ha de preservar la privacidad de los dueños de los valores o el precio de venta pactado, y la empresa ha de tener el derecho de hacer público o no el número de valores en el mercado o el precio de salida.

En este intercambio, es necesario la intervención de una tercera parte de confianza (TTP a partir de ahora, por sus siglas en inglés Trusted Third Party), ya que se manejan dos universos de datos (los activos cuentas corrientes y valores de una empresa), que no son públicos y no son accesibles a los actores que manejan los otros datos. El TTP actúa en representación de cada una de las partes involucradas en el intercambio, y asume sus riesgos. Esta responsabilidad cae en la organización Depositario, el cual tiene acceso a los dos canales de información: puede transferir y retener dinero en las cuentas corrientes, y reservar o poner en propiedad un valor. Al existir el TTP, no es necesario establecer una relación de confianza directa entre el inversor y la empresa de valores. Este actor vendría a representar en la vida real el papel de una cámara de compensación o “Clearing house”, o en la jerarquía de Target2Securities, al papel del DCV.

Para establecer las reglas del juego en la arquitectura DVP es necesaria también la intervención de un organismo auditor, papel que recae en la organización Regulador. Este rol representaría al Banco de España o al Banco Central Europeo. Tendría un papel auditor en los dos sistemas de información y velaría por el buen funcionamiento de la red y la neutralidad de los organismos.

Pese a la existencia de este organismo regulador, el diseño de la ejecución del mecanismo de DVP debe proporcionar por sí mismo la equidad entre las dos partes interesadas y que el organismo regulador actuar lo menos posible. Dicha propiedad garantiza que ninguna de las partes se encuentra en una situación de desventaja frente a la otra.

Básicamente, el protocolo a desarrollar es un doble sistema de comprobación y bloqueo, en dos universos de información distintos: uno de efectivo en el canal Banco, y otro de la propiedad de un valor en el canal MercadoDevalores. El sistema DVP impone que sea una entrega-contra-pago, es decir, que la propiedad del valor se modifique justo después del pago.

Antes de empezar a explicar el diseño del mecanismo DVP, hay que resaltar un hecho importante. La información, como se ha indicado, reside en canales distintos. Una organización puede tener nodos en dos canales, e incluso es posible que un nodo de tipo ancla pertenezca a dos canales, pero no es posible que datos vinculados al ledger de un canal pasen al otro canal. Esta separación de ledgers, por el canal al que pertenecen, está soportada por la configuración de los chaincodes, la identidad de los nodos y el protocolo goosip explicado anteriormente. Dicho protocolo se encarga de difundir la información sobre las transacciones, el estado del ledger y los miembros del canal, y está restringido a los nodos que han demostrado que poseen una identidad que pertenece al canal. Este mecanismo de aislamiento de nodos y datos del ledger, mediantes canales, permite ofrecer privacidad y confidencialidad a los miembros de la red. Por esta razón, la interacción entre ambos ledgers ha de realizarse desde una aplicación externa, no desde un programa chaincode. Dicha aplicación, pertenece a la organización Regulador y tendría identidades para autenticarse contra ambos canales (sería un nodo de tipo "client", como se ha comentado anteriormente). A modo de ejemplo, se describe la autenticación de una aplicación externa contra el sistema Hyperledger. Suponemos que las credenciales de administración se encuentran cargadas en el contexto previo, que se recogen en la variable "admin_user". En este ejemplo, se emplea la Autoridad de Certificación Fabric, que ofrece Hyperledger de ejemplo, y se encuentra escuchando en el puerto 7054. El objeto "fabric_ca_client" es un cliente contra esta autoridad de certificación. En primer lugar se registra dicho usuario contra la autoridad de certificado, obteniendo una clave secreta (variable secret), y después se utiliza esta clave para enrolar este usuario con el MSP asociado (DepositarioMSP).

```
fabric_ca_client = new Fabric_CA_Client('http://localhost:7054', null , '',
crypto_suite);
admin_user  =fabric_client.getUserContext('admin', true);

//registramos el usuario con el Servidor de CA Fabric

    return fabric_ca_client.register({enrollmentID: 'aplicacionDepositario',
affiliation: 'Depositario.aplicacion',role: 'client'}, admin_user);
}).then((secret) => {

//Enrolamos este usuario contra la autoridad de certificación

    console.log('Successfully registered 'aplicacionDepositario '- secret:'+
secret);

    return fabric_ca_client.enroll({enrollmentID: 'aplicacionDepositario ',
enrollmentSecret: secret});
}).then((enrollment) => {
  console.log('Successfully enrolled member user "'aplicacionDepositario'"
');
  return fabric_client.createUser(
     {username: 'aplicacionDepositario',
     mspid: 'DepositarioMSP',
     cryptoContent: { privateKeyPEM: enrollment.key.toBytes(),
signedCertPEM: enrollment.certificate }
     });
}).then((user) => {
     member_user = user;
```

```
    return fabric_client.setUserContext(member_user);
}).then(()=>{
    console.log('aplicacionDepositario was successfully registered and
enrolled and is ready to interact with the fabric network');
```

A partir de esta identidad, la aplicación podría enviar una propuesta de transacción y si está aprobada, enviar la transacción al servicio de ordenación. A continuación se muestra un código de ejemplo con estos pasos, resaltando únicamente las líneas de código que resultan explicativas para la comprensión del lector, se han obviado muchas líneas de comprobaciones adicionales y otras configuraciones (por ejemplo, la respuesta a una transacción no se obtiene directamente, sino a través de una subscripción a eventos producidos en el canal). Suponemos que la identidad del usuario se obtiene del contexto, como se quedó en el cuadro anterior, y la configuración del canal ya ha sido cargada.

```
tx_id = fabric_client.newTransactionID();

var request = {
                chaincodeId: 'GestionDepositario',
                fcn: 'bloquearSaldoUsuario',
                args: ['Usuario1','100'],
                chainId: 'CanalBanco',
                txId: tx_id
        };

results = channel.sendTransactionProposal(request);
  var proposalResponses = results[0];
        var proposal = results[1];
        if (proposalResponses && proposalResponses[0].response &&
                proposalResponses[0].response.status === 200) {
                        console.log('Transaction proposal was
good');
var request = {
                        proposalResponses: proposalResponses,
                        proposal: proposal
                };
Results = channel.sendTransaction(request);

if(results && results[1] && results[1].event_status === 'VALID')
                console.log('Successfully committed the change to
the ledger by the peer');
```

Si en cambio, solo se ejecutase una operación de lectura contra el ledger, se emplearía un código cuyas líneas principales serían las siguientes:

```
const request = {
                chaincodeId: 'GestionDepositario',
```

```
            fcn: 'obtenerPeticionesCompra',
            args: ['']
    };

    query_responses = channel.queryByChaincode(request);
    console.log("Response is ", query_responses[0].toString());
```

Por lo tanto, a esta aplicación externa se le asignaría un usuario y una clave secreta, registrados en el DepositarioMSP en ambos canales, y por lo tanto tendría permiso para ejecutar los chaincodes de ambos canales.

Una vez que hemos resuelto la interacción de estos canales mediante una aplicación externa, comenzamos a describir el diseño del protocolo que implementa el mecanismo DVP. Durante la ejecución de los chaincodes, hay que recordad, que tal, y como se explicaba en el marco teórico, nunca se pueden ejecutar código no determinista. Esto es debido a que si se acceden a variables del tipo Time.now() o random(), pueden dar resultados distintos a la hora de evaluar la ejecución de los chaincodes en los nodos, y provocar un fallo en la comprobación de la respuesta de los nodos aprobadores. Por lo tanto, se deciden implementar temporizadores para que una de las partes no se quede indefinidamente esperando a la otra y permitir un protocolo de cancelación, deben de tomar una referencia obtenida en una ejecución centralizada, y nunca desde el código del chaincode.

El protocolo a continuación descrito tiene la complicación de que están interviniendo varias organizaciones a dos fuentes de información aisladas. El punto de intercambio de información entre las organizaciones será la base de datos de estado del ledger. Comencemos a describirlo.

Todo comienza con una operación de compra desde un nodo de la organización Banco, que actúa en representación de un cliente. Se invoca a la función lanzarPeticionCompraValor() del chaincode ServiciosDepositario, con los argumentos usuario, empresa, y precioPropuesto. Como es una operación tipo invoke(), ha de generar una propuesta de transacción, recibir las respuestas de los nodos aprobadores y enviar la transacción. Para este chaincode se solicita la aprobación de las 3 organizaciones implicadas en este canal: Banco, Regulador y Depositario. El resultado de esta función es crear un activo tipo PeticionCompra en la forma de un par clave-valor que sería un objeto JSON con la información usuario, empresa, precioPropuesto y una valor numérico de control “compraPendiente” inicializada a “-1”. Para agilizar la explicación de este protocolo, en las operaciones de tipo invoke que aparecen a continuación, no se volverá a repetir los pasos que conllevan (envío de propuesta, espera de aprobaciones, envío de transacción), y nos limitaremos a mencionar la función del chaincode que se invoca.

Desde la aplicación externa de la organización Depositario se invocará a las funciones del chaincode ServiciosDepositario tipo invoke bloquearPeticionPendienteCompra (Timestamp NOW) y tipo query obtenerPeticionPendienteCompra(). La primera de ellas bloqueará la primera operación pendiente de compra, empleando el valor de control “compraPendiente” estableciéndola al valor Timestamp NOW. De esta forma, indicamos que esta petición de compra ha sido procesada y se inicia un contador para que esta petición no se posponga indefinidamente. El valor Timestamp NOW ha sido creado desde un punto centralizado, como

se explicaba anteriormente. Con la segunda operación se obtienen los datos de la petición de compra de valor.

La aplicación externa ya tiene los datos de una petición de compra de un valor. En este instante, tiene que realizar dos comprobaciones: que el usuario tiene saldo suficiente, y que en el mercado de valores se ha puesto en venta un valor con el precio fijado por el usuario inversor. Ambas comprobaciones conllevan una operación de bloqueo, tanto del valor como del fondo del usuario. Para garantizar la equidad de ambas partes, el protocolo ha de tener en cuenta temporizadores para no beneficiar a ninguna de las partes y poder cancelar los bloqueos. Como se trata de un mecanismo "Delivery versus payment", en el que por definición la entrega se hace frente a un pago, se realizará en primer lugar la comprobación y reserva del saldo del usuario. En primer lugar, se invocará a la función bloquearSaldo() del chaincode ServiciosDepositario, con los parámetros usuario, precioFijado y Timestamp NOW. Será una operación de tipo Invoke, que devolverá un código de error en caso de que no se haya podido bloquear el saldo del usuario. La comprobación y el bloqueo se harán en la misma función, de forma atómica, para que no intervengan otras operaciones de extracción de saldo y después de confirmar la existencia de saldo, no se pueda extraer el dinero. Si esto se produjera, se invocaría a la función cancelarPeticionCompra()del chaincode ServiciosDepositario, que elimina la estructura PeticionCompra asociada y se finalizaría la ejecución de la aplicación, comenzando de nuevo desde el principio. Analicemos qué conlleva el hecho de bloquear el saldo del usuario. En primer lugar, se hace una transferencia de saldo, de la cuenta del usuario a la cuenta del depositario, modificando los dos pares claves-valor correspondientes, a los activos tipo cuentaCorriente asociados a los dos actores. En segundo lugar, se crea un activo de tipo BloqueoSaldo, siendo un objeto JSON con la información del usuario, la cantidad bloqueada y parámetro TimeStamp Now, para así proteger a la parte implicada Usuario en caso de que no se finalice el intercambio.

Si se ha producido el bloqueo del saldo correctamente, la aplicación interactúa con el canal MercadoDeValores, realizando la segunda operación de comprobación y reserva. En este caso, invoca a la función tipo invoke bloqueaValor() del Chaincode GestionValores con los parámetros usuario, empresa, precioPropuesto y Timestamp NOW. Si existiera algún problema durante la reserva del valor, (porque no existiera el valor o el precio del valor fuera superior al precio propuesto), la función devolvería un mensaje de error. No sería necesario tratar este mensaje de error, ya que los activos BloqueoSaldo y PeticionCompra del canal canalBanco poseen un timestamp a modo de temporizador que, en caso de no ejecutarse el procedimiento de finalización del intercambio, permitiría ejecutar un procedimiento de cancelación automático. En caso de que fuera posible el bloqueo del valor, se modificaría el par clave-valor que representa el valor asociado. De forma análoga al caso anterior, se ha propuesto que la comprobación y el bloqueo del valor se hagan de forma atómica. También es necesario, durante el bloqueo del valor, crear un par clave-valor representando al activo BloqueoValor, que sería un objeto JSON identificando el IDdelValor y el valor TimeStamp Now. Este valor timestamp protegería a la parte interesada MercadoDeValores, que representa los intereses de una empresa. Así, actuaría a modo de temporizador, para que en caso de que no se ejecute el protocolo de finalización, pueda actuar de forma automática un protocolo de cancelación.

En este último paso del flujo del mecanismo DVP, se produciría la transferencia de la cantidad propuesta del valor del activo, desde la cuenta corriente del Depositario a la cuenta corriente de la empresa, modificando el par clave-valor correspondiente en el canal canalBanco. Para ello, la aplicación externa invocará a la función transferirSaldo() del chaincode ServiciosDepositario con los parámetros empresa, cantidad y usuario, de tipo invoke.

Finalmente, se ejecutaría el protocolo de finalización del mecanismo DVP, en el que la aplicación invocaría a las funciones eliminarBloqueoValor del chaincode GestionMercadoDeValores y eliminarBloqueoSaldo del chaincode ServiciosDepositario, de forma secuencial. Dichos métodos eliminarían las estructuras BloqueoValor y BloqueoSaldo, que contienen temporizadores para deshacer las reservas del valor y del efectivo, respectivamente, en caso de error del procedimiento.

De forma periódica, la aplicación ejecutaría el protocolo de cancelación automático. Explicaremos en qué consiste. Es necesario, que se revisen los temporizadores que contienen las estructuras PeticionCompra, BloqueoValor y BloqueoSaldo, para que, si no se ha ejecutado el protocolo de finalización, y el temporizador ha superado un tiempo determinado, se invalide la petición de compra, se libere el valor, o se restaure el saldo al usuario. Para ello, se ejecutarían las funciones de deshacerBloqueoSaldo del chaincode ServiciosDepositario, y deshacerBloqueoValor del chaincode GestionMercadoDeValores, con el parámetro Timestamp NOW para comprobar si han superado un valor máximo de espera.

A continuación, en la ilustración 23, se refleja en un diagrama de flujo el protocolo anteriormente descrito.

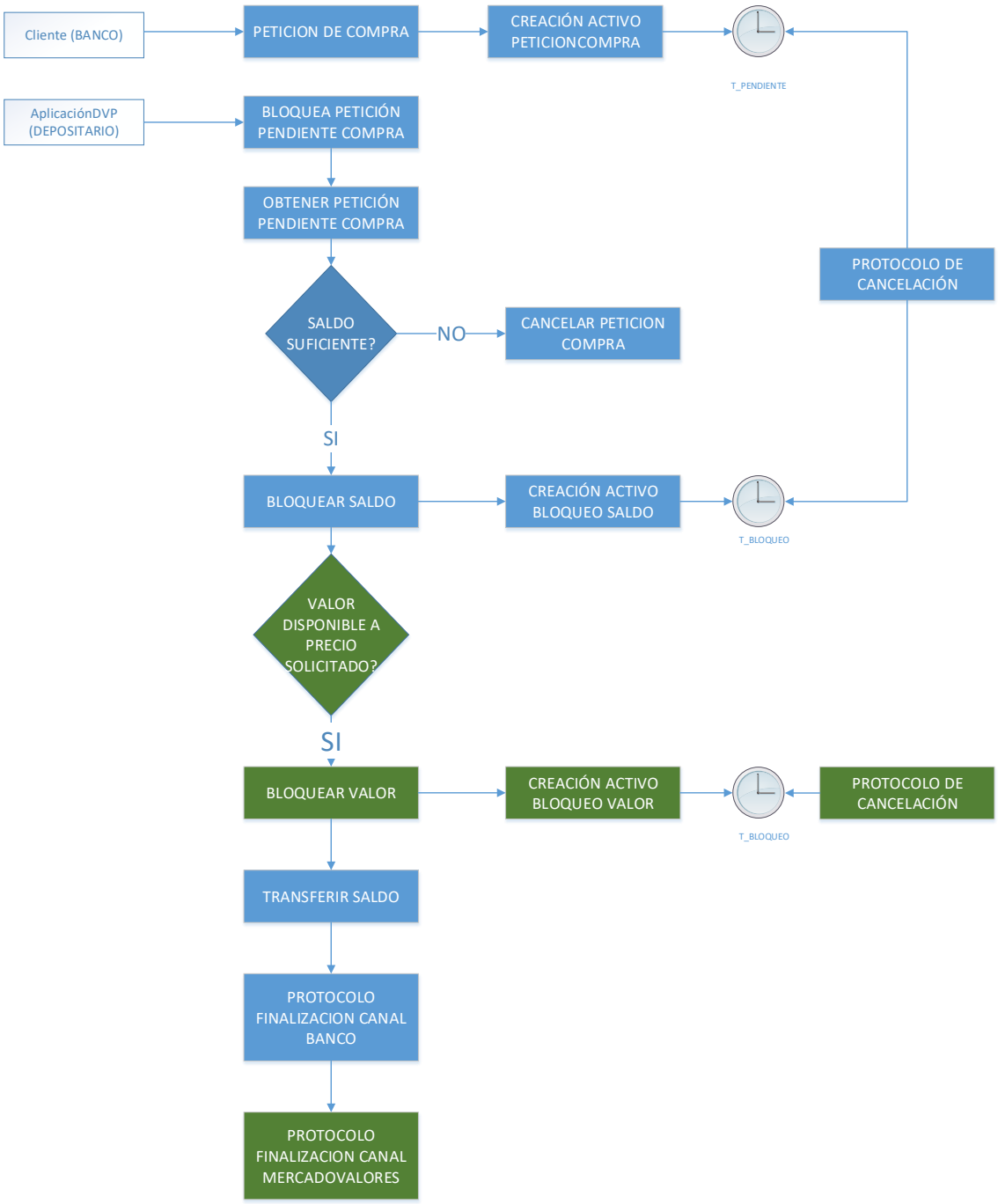

Ilustración 23 – Diagrama de flujo mecanismo DVP

En la ilustración 24 podemos comprobar las llamadas a las funciones de los chaincodes, realizadas de forma previa al inicio del protocolo DVP.

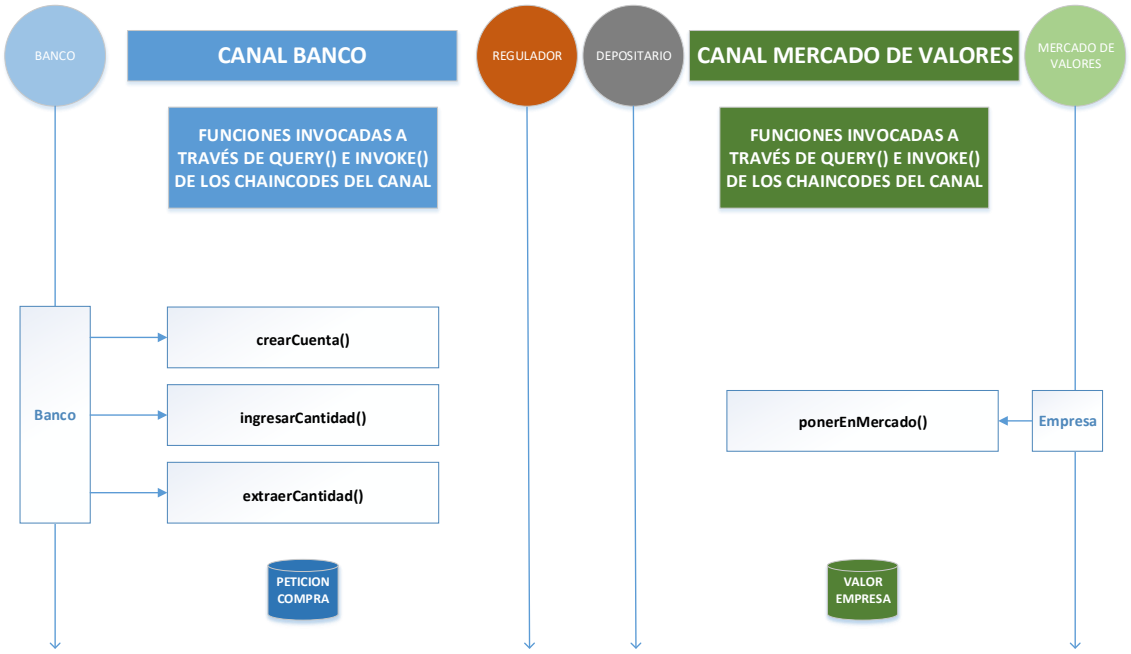

Ilustración 24 – Llamadas a funciones de chaincodes de forma previa al DVP

En la ilustración 25, se observan las llamadas a funciones de los chaincodes disponibles en los canales, durante la ejecución del intercambio DVP.

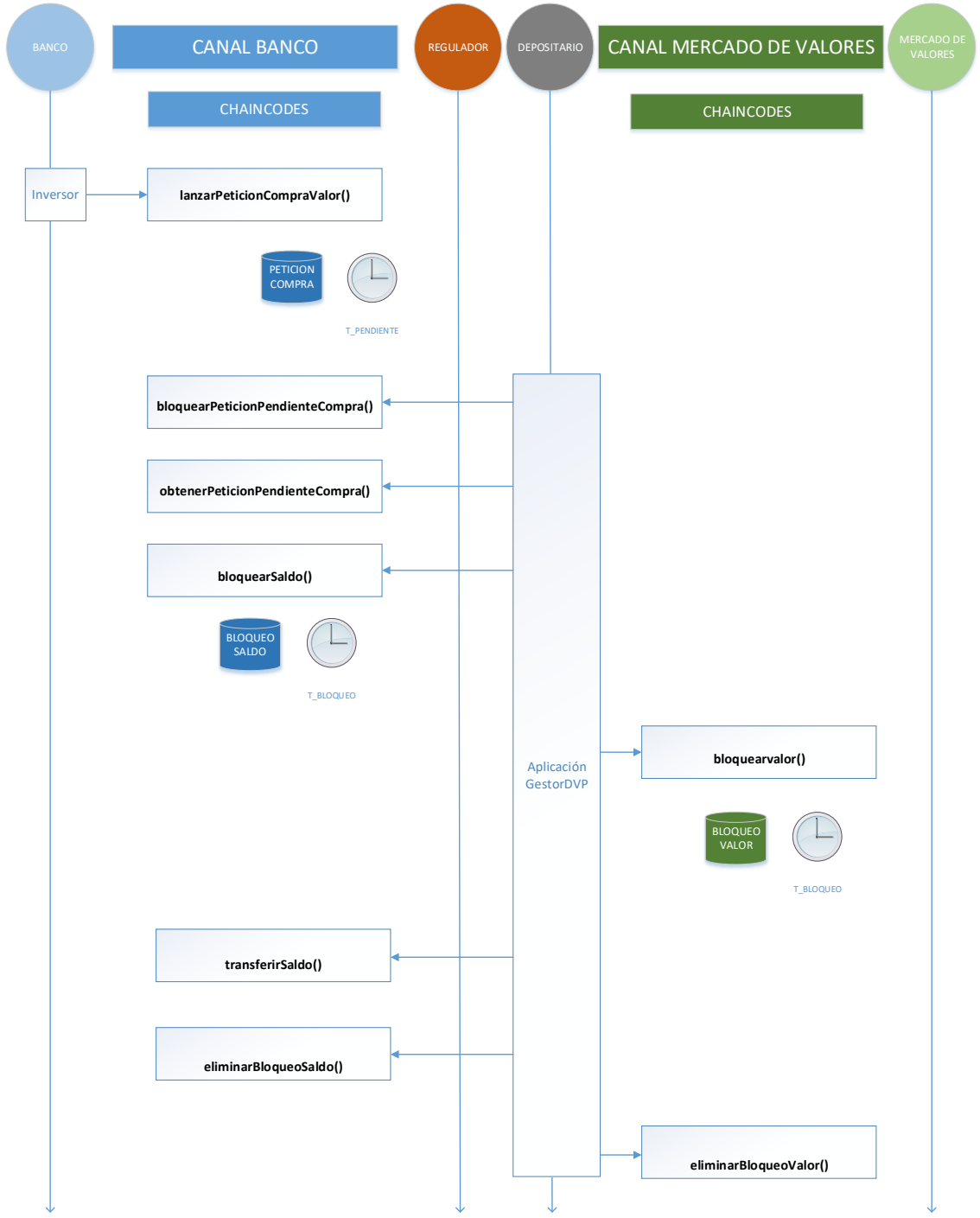

Ilustración 25 – Llamadas a funciones de chaincodes durante la ejecución del DVP

Por completar las invocaciones a las funciones de los chaincodes, se muestran en la ilustración 26 las realizadas desde los nodos de la organización Regulador, que pueden ser producidas en todo momento.

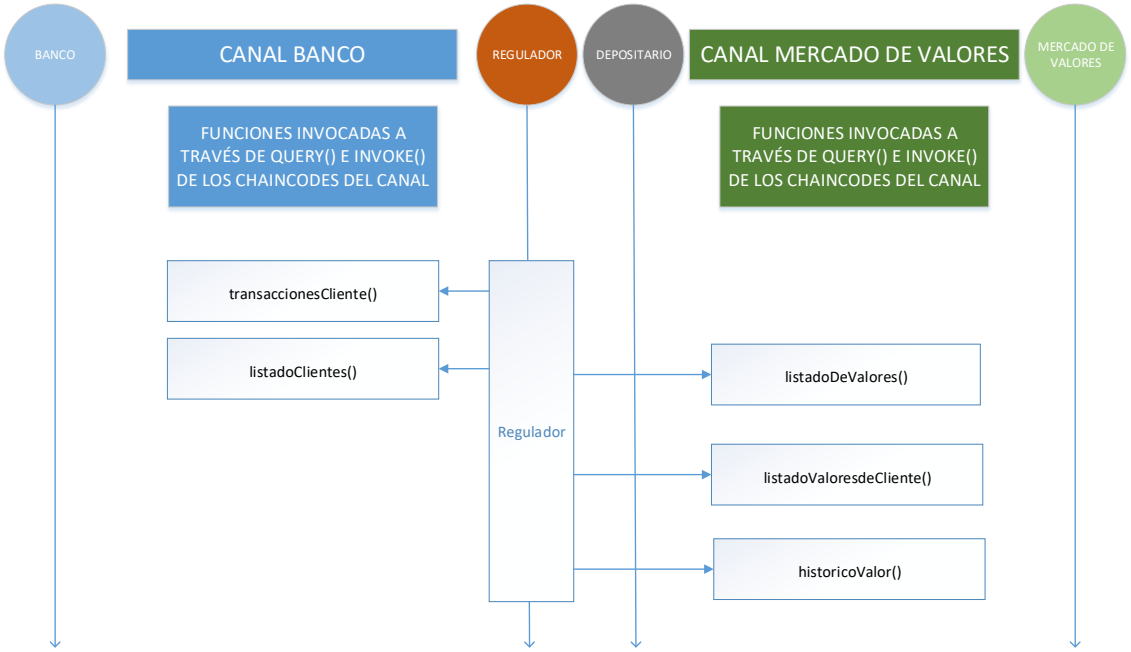

Ilustración 26 – Llamadas a funciones de chaincodes en todo momento por el organismo Regulador

Estudiaremos ahora las propiedades de este protocolo de intercambio. En primer lugar, la equidad. Analicemos las posibles situaciones que pueden darse:

- Los mecanismos de bloqueo y reserva de efectivo y de valores funcionan correctamente y se ejecuta el protocolo de finalización. Ambas partes obtienen lo que requerían.
- El usuario no dispone de fondos suficientes. En este caso, el algoritmo se detiene en la fase de bloqueo y reserva de los fondos, y la empresa no asigna el valor solicitado.
- No existe ningún valor en el mercado, o no está disponibles al precio propuesto. En este caso, el temporizador indicado en la estructura BloqueoSaldo devolverá al usuario el dinero bloqueado.
- La transferencia de efectivo entre la cuenta del depositario y la cuenta de la empresa dueña del valor no llega a realizarse. En este caso, los temporizadores existentes en las estructuras BloqueoSaldo y BloqueoValor permitirán que, pasado un tiempo marcado, se deshagan estos bloqueos, el saldo vuelva al usuario y el valor quede de nuevo en el mercado.

En segundo lugar, analizaremos la propiedad de que la TTP (la organización Depositario) sea verificable o no, es decir, si las partes intervinientes pueden demostrar un mal comportamiento de la TTP si no sigue los pasos previstos. En este caso, la red Hyperledger, gracias a que refleja las transacciones en la cadena de bloques de los ledgers, registra cada uno de los pasos efectuados en el protocolo. Si realizara un ataque, el ledger sería inmutable e inviolable, y al estar compartido por varias organizaciones sería detectable un uso fraudulento de las capacidades de la TTP. El primer ataque que puede realizar es una alianza con los usuarios. Es decir, que nunca llegara a ingresar los fondos en la cuenta de la empresa, y asignara los valores a los inversores. En este caso, la organización mercadoDeValores puede presentar la información del ledger del canal canalMercadoDeValores, que demuestra que ha puesto la propiedad de un valor a nombre de un inversor, y presentar una evidencia para demostrar a posteriori que los inversores o la TTP hicieron trampas. El otro ataque que puede hacer es que

se alíe con la organización MercadoDeValores, y nunca llegue a asignar valores a inversores. En este caso, los usuarios pueden demostrar a posteriori que han efectuado una transferencia de dinero con la información presente la transacción asociada en el ledger.

A continuación, analizaremos si se cumple la propiedad de temporalidad. Dicha característica se cumple si ambas partes tienen la garantía de que el protocolo se cumplirá en un tiempo finito sin riesgo de perder la equidad. Como se ha detallado, existen temporizadores en los bloqueos de saldo y de propiedad de valor, que después de un tiempo marcado, deshacen el bloqueo de saldo y la asignación del valor.

También se puede afirmar que el protocolo es asíncrono, ya que no requiere de la sincronización de los relojes de las partes involucradas. Las marcas de tiempo establecidas en los timestamp que son la base de los temporizadores se asignan de forma centralizada.

Terminamos este análisis con la propiedad de confidencialidad frente a la TTP. En este caso, ambas partes implicadas no son confidenciales frente a la organización depositario, ya que requiere de toda la información para acceder a las cuentas y las propiedades de los valores en el mercado. Esta propiedad se podría lugar mediante el uso de pseudónimos para identificar a inversores y empresas. Sin embargo, dada las características de la solución DVP, no es factible que las partes interesadas manejen información cifrada entre inversor y empresa, ya que la organización depositario es una parte implicada, y tiene que acceder a la información manejada para operar en ambos canales.

5. Prueba de concepto.

Para completar el análisis del diseño de una aplicación práctica del framework Hyperledger Fabric, se va a desarrollar en este capítulo una prueba de concepto de esta plataforma. El objetivo será implementar una red similar a la analizada, sin tener en cuenta, por la extensión de su contenido, el desarrollo de los chaincodes necesarios ni la interacción con una red externa.

5.1. Entorno de la prueba de concepto

La presente prueba se ha realizado sobre una máquina virtual de "VMWare Workstation Player 15", en el que el equipo anfitrión se encuentra instalado Windows Home 10, y el equipo huésped, una imagen creada a partir de los instalables de Ubuntu 18.04. Más exactamente, la imagen se ha instalado a partir de la versión denominada como "ubuntu-18.04.1-desktop-amd64.iso". Después de la instalación, se ha realizado una actualización de los paquetes con el comando:

```
sudo apt-get update && sudo apt-get upgrade –y
```

5.1.1. Instalación de docker

Antes de continuar con la instalación de la prueba de concepto, es necesario explicar una serie de conceptos sobre el entorno donde se ejecuta Hyperledger, es decir, Docker. Se trata de una plataforma de contenedores ligeros y portables, donde una aplicación puede ejecutarse sin tener en cuenta el sistema operativo en el que Docker está instalado. De esta forma, los desarrolladores de aplicaciones no tienen que dedicar esfuerzos en adaptar su código a la máquina host, ya que el contenedor el proporcionará el entorno necesario. Además, es mucho más ligero que una máquina virtual, ya que en esta última hay que instalar un sistema operativo, mientras que Docker utiliza el sistema en el que se ha instalado. Con los años, ha tomado cada vez más popularidad en los desarrolladores, hecho que podemos observar en la ilustración 27, obtenida a partir de Google Trends:

Ilustración 27 - Evolución del término Docker en las búsquedas de Google

Tal y como se afirma en la página web del proyecto Hyperledger, se encuentran trabajando en el desarrollo de un instalable para cada sistema. Mientras tanto, se ofrece dicha plataforma mediante imágenes para los contenedores de Docker.

A continuación comentaremos la instalación de Docker.

En primer lugar, se decidió instalar Docker directamente sobre el equipo Windows 10 Home, pero debido a los constantes errores encontrados y a lo difícil que resulta avanzar en la prueba de concepto, se decidió a realizar una instalación sobre Ubuntu. Este hecho facilitó sobremanera el avance en la prueba de concepto.

A lo largo de esta prueba de concepto, se explicarán en lo posible los pasos realizados para la configuración de la misma. El objetivo de ser tan minuciosos es que dicha prueba de concepto sea lo más reproducible posible por parte del lector, y de esta forma, evitar tener que adjuntar un fichero pesado al actual documento.

Se indican a continuación los pasos desarrollados para instalar Docker:

- Como forma preventiva, se desinstalan otras versiones de docker:

```
sudo apt-get remove docker docker-engine docker.io
```

- El repositorio de los paquetes de Docker se encuentra accesible a través de https, por lo que es necesario instalar otros paquetes para que apt pueda acceder a dicho repositorio seguro.

```
sudo apt-get install apt-transport-https ca-certificates
curl software-properties-common
```

- Para que apt pueda establecer una relación de confianza con la clave PGP del repositorio docker, hay que agregar su clave al repositorio de apt:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
```

- Se verifica que la clave se ha instalado correctamente:

```
sudo apt-key list
/etc/apt/trusted.gpg
--------------------
pub   rsa4096 2017-02-22 [SCEA]
      9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid           [ unknown] Docker Release (CE deb)
<docker@docker.com>
sub   rsa4096 2017-02-22 [S]
```

- Se instala el repositorio estable de Docker, y se actualiza la base de datos de paquetes con los del repositorio.

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"
sudo apt update
```

- Finalmente, se instala Docker, y se comprueba la versión instalada del servicio:

```
sudo apt install docker-ce
sudo systemctl status docker
docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled;
vendor preset: enabled)
   Active: active (running) since Sat 2018-12-22 15:48:39 PST; 6
days ago
     Docs: https://docs.docker.com
 Main PID: 861 (dockerd)
    Tasks: 21
   CGroup: /system.slice/docker.service
           └─861 /usr/bin/dockerd -H unix://
```

- Finalmente, se ejecuta el programa de prueba para comprobar que la instalación ha sido satisfactoria.

```
pani@ubuntu:~$ docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working
correctly.
```

5.1.2. Instalación de dependencias de Hyperledger

Tal y como informa en la página de instalación de Hyperledger, descrita en [38], se requiere la descarga de paquetes adicionales. Se listan a continuación para permitir la reproducción por parte del lector de la prueba de concepto, y dejar constancia de las versiones instaladas.

- Instalación de GO desde [39]. Para ello, se ha descargado la versión "go1.11.4.linux-amd64.tar.gz", y se ha descomprimido dicho archivo en la ruta "$HOME/go". La configuración del software además requiere las siguientes variables:

```
export GOPATH=/home/pani/workspace (en una ruta externa a la
instalación de GO)
export GOROOT=/home/pani/go
```

- Se comprueban que las versiones del software siguiente que vienen instaladas con esta versión de Ubuntu son las requeridas por Hyperledger. Se indican las versiones del intérprete Python, del framework de Javascript Node.js, y de npm, el gestor de paquetes de Node.js.

```
pani@ubuntu:~$ python -V
Python 2.7.15rc1
pani@ubuntu:~$ node -v
v11.6.0
pani@ubuntu:~$ npm -version
6.5.0-next.0
```

5.1.3. Instalación y configuración de Hyperledger

En primer lugar, se instalan los binarios de Hyperledger, a partir de la ruta que se indica en [40].

```
curl -sSL http://bit.ly/2ysbOFE | bash -s 1.3.0
```

El siguiente paso es instalar las herramientas necesarias para la ejecución de Hyperledger, además de incluir en Docker la posibilidad de instalar las imágenes proporcionadas por este Framework. Se detallan a continuación para reflejar el versionado:

```
===> List out hyperledger docker images
```

hyperledger/fabric-zookeeper	0.4.14
hyperledger/fabric-kafka	0.4.14
hyperledger/fabric-couchdb	0.4.14
hyperledger/fabric-javaenv	1.3.0
hyperledger/fabric-ca	1.3.0
hyperledger/fabric-tools	1.3.0
hyperledger/fabric-ccenv	1.3.0
hyperledger/fabric-orderer	1.3.0
hyperledger/fabric-peer	1.3.0
hyperledger/fabric-baseos	amd64-0.4.13

A continuación se describen brevemente dichas imágenes:

- Fabric-ca: Se trata de una autoridad de certificación que proporciona hyperledger.
- fabric-zookeeper: Es el contenedor para los servidores Zookeeper, que se encarga de informar qué servidor Kafka tienen que suscribirse los nodos para recibir el mensaje de un canal.

- Fabric-kakfa: Es la imagen para los servidores kakfa, que implementan la lógica para poseer varios nodos ejecutando el servicio de ordenación.
- Fabric-couchdb: Se trata de la base de datos de estado alternativa a LevelDB, la que se encuentra embebida en los nodos.
- fabric-javaenv: Proporciona una máquina virtual para ejecutar chaincodes en Java.
- fabric-tools: Contiene una imagen para acceder a las herramientas proporcionadas por el framework Hyperledger.
- fabric-ccenv: Se trata de la aplicación necesaria para la instanciación, instalación y ejecución de chaincodes.
- fabric-orderer: Es el contenedor para los nodos del servicio de ordenación.
- fabric-peer: Es la imagen de los nodos estándar, incluyendo los aprobadores.
- fabric-baseos: sirve como base para la creación de todo tipo de imágenes (nodos, nodos de ordenación, nodos zookeeper, Kafka…)

5.2. Configuración de la prueba de concepto

A continuación se va a describir los pasos seguidos para la elaboración de la prueba de concepto. En la misma, se va a desarrollar una red Hypeledger con un canal, que sería la simulación del canal "canalBanco" indicado durante este documento. Por otro lado, se crearán las 3 organizaciones involucradas en dicho canal. Estas organizaciones tendrán 4 nodos estándar y 2 nodos ancla cada una. Posteriomente, se incorporarán los nodos de las organizaciones al canal.

Como parte de la prueba de concepto, y desde el contenedor creado a partir de la imagen de Hyperledger "fabric-tools", se instalará un chaincode básico en los primeros nodos de cada organización y se instanciará en el canal con dos pares clave-valor. Finalmente, se ejecutará una llamada a una función tipo query y otra tipo invoke del chaincode.

Durante la exposición de la prueba de concepto nos limitaremos a indicar los comandos y no entrar en conceptos teóricos, ya que se han explicado en apartados anteriores. Se ha tomado como referencia la propuesta de ejemplo desarrollada en la referencia [41].

5.2.1. Creación de la criptografía necesaria

A partir de la configuración del fichero "crypto-config.yaml" propuesto en el Anexo, se crean las claves y certificados correspondientes a los MSPs de las 3 organizaciones.

```
../bin/cryptogen generate --config=./crypto-config.yaml

banco.mistic-uoc.com
depositario.mistic-uoc.com
regulador.mistic-uoc.com
```

Toda la criptografía creada, y la relación de MSPs se adjuntan en el Anexo "Salida cryptogen".

5.2.2. Creación del bloque génesis y la información sobre el canal

Con la ejecución de los siguientes comandos se crea el bloque génesis del canal, así como se generar las transacciones de actualizaciones sobre la información de los nodos ancla de las 3 organizaciones. La base para la elaboración de los perfiles que emplea la herramienta configtxgen, y que indican la estructura de la red se muestran en el anexo "Perfiles para cryptogen en el archivo configtx.yaml":

```
$GOPATH/bin/configtxgen -profile BancoOrdererGenesis -channelID
syschain  -outputBlock ./channel-artifacts/genesis.block

$GOPATH/bin/configtxgen -profile bancochannel -outputCreateChannelTx
./channel-artifacts/bancochannel.tx -channelID "bancochannel"

$GOPATH/bin/configtxgen -profile bancochannel -
outputAnchorPeersUpdate ./channel-artifacts/bancoMSPanchors.tx -
channelID "bancochannel" -asOrg bancoMSP

$GOPATH/bin/configtxgen -profile bancochannel -
outputAnchorPeersUpdate ./channel-artifacts/depositarioMSPanchors.tx
-channelID "bancochannel" -asOrg depositarioMSP

$GOPATH/bin/configtxgen -profile bancochannel -
outputAnchorPeersUpdate ./channel-artifacts/reguladorMSPanchors.tx -
channelID "bancochannel" -asOrg reguladorMSP
```

Se muestra de forma parcial la salida de dichos comandos para una mejor comprensión. Más específicamente, se muestra la salida de la creación del bloque génesis, y de la actualización para los nodos ancla de la organización Banco.

```
##Salida a la creación del bloque génesis para el canal bancoChannel

2018-12-30 18:09:58.534 PST [common.tools.configtxgen] main -> INFO
001 Loading configuration
2018-12-30 18:09:58.597 PST [common.tools.configtxgen.localconfig]
completeInitialization -> INFO 002 orderer type: solo
2018-12-30 18:09:58.599 PST [common.tools.configtxgen.localconfig]
Load -> INFO 003 Loaded configuration:
/home/pani/sistemaDVP/configtx.yaml
2018-12-30 18:09:58.659 PST [common.tools.configtxgen.localconfig]
completeInitialization -> INFO 004 orderer type: solo
```

```
2018-12-30 18:09:58.660 PST [common.tools.configtxgen.localconfig]
LoadTopLevel -> INFO 005 Loaded configuration:
/home/pani/sistemaDVP/configtx.yaml
2018-12-30 18:09:58.697 PST [common.tools.configtxgen] doOutputBlock
-> INFO 006 Generating genesis block
2018-12-30 18:09:58.699 PST [common.tools.configtxgen] doOutputBlock
-> INFO 007 Writing genesis block


###ANCHOR PEERS PARA BANCOMSP

2018-12-30 18:09:59.055 PST [common.tools.configtxgen] main -> INFO
001 Loading configuration
2018-12-30 18:09:59.101 PST [common.tools.configtxgen.localconfig]
Load -> INFO 002 Loaded configuration:
/home/pani/sistemaDVP/configtx.yaml
2018-12-30 18:09:59.140 PST [common.tools.configtxgen.localconfig]
completeInitialization -> INFO 003 orderer type: solo
2018-12-30 18:09:59.140 PST [common.tools.configtxgen.localconfig]
LoadTopLevel -> INFO 004 Loaded configuration:
/home/pani/sistemaDVP/configtx.yaml
2018-12-30 18:09:59.140 PST [common.tools.configtxgen]
doOutputAnchorPeersUpdate -> INFO 005 Generating anchor peer update
2018-12-30 18:09:59.141 PST [common.tools.configtxgen]
doOutputAnchorPeersUpdate -> INFO 006 Writing anchor peer update
```

5.2.3. Generación de la plantilla de servicios para la red Hyperledger

A continuación, se define el archivo “docker-compose-template.yaml”, que contiene la estructura de los servicios que la red Hyperledger ha de cargar al iniciar. Su contenido se muestra en el anexo “Plantillas de los servcios de la red Hyperledger”, y detalla:

- Las tres autoridades de certificación creadas para cada una de las organizaciones.
- El nodo que ejecutará el servicio de ordenación. Para esta prueba de concepto, se definirá un modelo de consenso tipo “SOLO”.
- Los 4 nodos creados para cada una de las organizaciones.
- El container “fabric-tools”, con las herramientas del framework Hyperledger.

En cada uno de los elementos de configuración de los distintos contenedores a crear, se detalla, entre otras características, la imagen de Hyperledger que instanciarán, las variables de entorno, los puertos en los que escucharán y los volúmenes creados.

Sobre la plantilla de configuración indicada en el anexo, solo queda sustituir las variables CA1_PRIVATE_KEY, CA2_PRIVATE_KEY y CA3_PRIVATE_KEY, por las claves aleatorias generadas en el apartado “Creación de la criptografía necesaria”, de la siguiente manera:

```
cd crypto-config/peerOrganizations/banco.mistic-uoc.com/ca/
    PRIV_KEY=$(ls *_sk)
```

```
    cd $CURRENT_DIR
    sed $OPTS "s/CA1_PRIVATE_KEY/${PRIV_KEY}/g" docker-compose.yaml
```

5.2.4. Descarga de las imágenes Hyperledger para los containers Docker

De forma previa a la creación de los distintos containers, es necesario descargar del repositorio de Hyperledger las distintas imágenes necesarias. Se han descargado las imágenes siguientes:

- Fabric-peer: imagen para cada uno de los nodos.
- Fabric-orderer: imagen para el servicio de ordenación.
- Fabric-tools: imagen para las herramientas del framework de Hyperledger.
- Fabric-ccenv: imagen para el contenedor de chaincodes.
- Fabric-ca: imagen para las autoridades de certificación.

Para ello, se ejecutan los siguientes comandos:

```
docker pull hyperledger/fabric-peer:1.3.0
docker tag hyperledger/fabric-peer: 1.3.0 hyperledger/fabric-peer

docker pull hyperledger/fabric-orderer:1.3.0
docker tag hyperledger/fabric-orderer: 1.3.0 hyperledger/fabric-
orderer

docker pull hyperledger/fabric-ccenv:1.3.0
docker tag hyperledger/fabric-ccenv: 1.3.0 hyperledger/fabric-ccenv

docker pull hyperledger/fabric-ccenv:1.3.0
docker tag hyperledger/fabric-ccenv: 1.3.0 hyperledger/fabric-ccenv

docker pull hyperledger/fabric-ca:1.3.0
docker tag hyperledger/fabric-ca: 1.3.0 hyperledger/fabric-ca
```

Los hashes obtenidos para las imágenes descargadas han sido los siguientes:

```
1.3.0: Pulling from hyperledger/fabric-peer
Digest:
sha256:6756c7c48234ae6b0a8822a378681017cf6dbfeadfbf1f8a528ee6c4db343621
Status: Image is up to date for hyperledger/fabric-peer:1.3.0
==> FABRIC IMAGE: orderer

1.3.0: Pulling from hyperledger/fabric-orderer
Digest:
sha256:6ee1abcfd84031765d67544e5d6b4f3af08c3f064312c65715587d392fe7c3eb
Status: Image is up to date for hyperledger/fabric-orderer:1.3.0
==> FABRIC IMAGE: ccenv

1.3.0: Pulling from hyperledger/fabric-ccenv
```

```
Digest:
sha256:05fce5513fcae3110ac041469ed9e0e4c9661f44782f52ef5d8930eb416c2197
Status: Image is up to date for hyperledger/fabric-ccenv:1.3.0
==> FABRIC IMAGE: tools

1.3.0: Pulling from hyperledger/fabric-tools
Digest:
sha256:058cff3b378c1f3ebe35d56deb7bf33171bf19b327d91b452991509b8e9c7870
Status: Image is up to date for hyperledger/fabric-tools:1.3.0
==> FABRIC IMAGE: ca

1.3.0: Pulling from hyperledger/fabric-ca
Digest:
sha256:83abc367c5273a12d59ef9777637eb6c1abf04a5d00d66a0bffc55c40075850e
Status: Image is up to date for hyperledger/fabric-ca:1.3.0
```

5.2.5. Creacíon de la red Hyperledger

El siguiente paso será la creación de la red, y para ello, ejecutamos el siguiente comando, tomando como parámetro la plantilla definida anteriormente. Además, requiere los ficheros “docker-compose-base.yaml” y “peer-base.yaml”, cuyo contenido se adjunta en el capitulo correspondiente a los anexos. Dichos ficheros completan el fichero del perfil de la red, y definen las variables de entorno y comando a ejecutar cuando se inicia un nodo, respectivamente.

```
docker-compose -f docker-compose-template.yaml up -d
```

La salida de dicho comando, así como el listado de contenedores docker inicializados, es la siguiente:

```
Creating peer0.depositario.mistic-uoc.com ... done
Creating orderer.mistic-uoc.com           ... done
Creating peer1.depositario.mistic-uoc.com ... done
Creating peer1.banco.mistic-uoc.com       ... done
Creating peer1.regulador.mistic-uoc.com   ... done
Creating peer2.regulador.mistic-uoc.com   ... done
Creating peer2.depositario.mistic-uoc.com ... done
Creating peer0.banco.mistic-uoc.com       ... done
Creating ca_peerDepositario               ... done
Creating peer3.banco.mistic-uoc.com       ... done
Creating ca_peerRegulador                 ... done
Creating ca_peerBanco                     ... done
Creating peer3.regulador.mistic-uoc.com   ... done
Creating peer2.banco.mistic-uoc.com       ... done
Creating peer3.depositario.mistic-uoc.com ... done
Creating peer0.regulador.mistic-uoc.com   ... done
Creating cli
                            ... done
pani@ubuntu:~/sistemaDVP$ docker ps -a
```

```
CONTAINER ID        IMAGE                          COMMAND
CREATED              STATUS              PORTS
NAMES
62439ca179ac        hyperledger/fabric-tools     "/bin/bash"              31
seconds ago       Up 27 seconds                                        cli
2ef0d4eb2c3b        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 33 seconds       0.0.0.0:9006-9008->9006-9008/tcp
peer0.regulador.mistic-uoc.com
30ef5cb60b1c        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 33 seconds       0.0.0.0:9030-9032->9030-9032/tcp
peer3.depositario.mistic-uoc.com
06e01ab21b6f        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 32 seconds       0.0.0.0:9033-9035->9033-9035/tcp
peer3.regulador.mistic-uoc.com
ee37b341c271        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 31 seconds       0.0.0.0:9018-9020->9018-9020/tcp
peer2.banco.mistic-uoc.com
f5553ab9d78f        hyperledger/fabric-ca        "sh -c 'fabric-ca-se…"
About a minute ago   Up 50 seconds       0.0.0.0:7054->7054/tcp
ca_peerBanco
675df1960eb3        hyperledger/fabric-ca        "sh -c 'fabric-ca-se…"
About a minute ago   Up 47 seconds       0.0.0.0:9054->7054/tcp
ca_peerRegulador
cda7375f74c3        hyperledger/fabric-ca        "sh -c 'fabric-ca-se…"
About a minute ago   Up 49 seconds       0.0.0.0:8054->7054/tcp
ca_peerDepositario
63565cb2a057        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 32 seconds       0.0.0.0:9027-9029->9027-9029/tcp
peer3.banco.mistic-uoc.com
811ac4437cf7        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 31 seconds       0.0.0.0:9024-9026->9024-9026/tcp
peer2.regulador.mistic-uoc.com
43b950af7e39        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 33 seconds       0.0.0.0:9021-9023->9021-9023/tcp
peer2.depositario.mistic-uoc.com
ed70c3f3ef78        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 40 seconds       0.0.0.0:9015-9017->9015-9017/tcp
peer1.regulador.mistic-uoc.com
b02686214a06        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 34 seconds       0.0.0.0:9000-9002->9000-9002/tcp
peer0.banco.mistic-uoc.com
f47d73f4020d        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 36 seconds       0.0.0.0:9012-9014->9012-9014/tcp
peer1.depositario.mistic-uoc.com
69adf5857528        hyperledger/fabric-orderer   "orderer"
About a minute ago   Up 52 seconds       0.0.0.0:7050->7050/tcp
orderer.mistic-uoc.com
4b8e715d07ab        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 46 seconds       0.0.0.0:9009-9011->9009-9011/tcp
peer1.banco.mistic-uoc.com
4a8984280d11        hyperledger/fabric-peer      "peer node start"
About a minute ago   Up 49 seconds       0.0.0.0:9003-9005->9003-9005/tcp
peer0.depositario.mistic-uoc.com
```

5.2.6. Creación del canal

A partir de ahora, nos conectaremos al contenedor "fabric-tools", que posee el identificador "cli" para realizar las siguientes acciones. La forma de conectarnos al mismo es la siguiente:

```
docker exec -it cli /bin/bash
```

Para crear el canal, se ejecuta el siguiente comando:

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
CORE_PEER_LOCALMSPID="bancoMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-uoc.com/tls/ca.crt
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/users/Admin@banco.mistic-uoc.com/msp
CORE_PEER_ADDRESS=peer0.banco.mistic-uoc.com:7051
CHANNEL_NAME=bancochannel
CORE_PEER_TLS_ENABLED=true
ORDERER_SYSCHAN_ID=syschain
peer channel create -o orderer.mistic-uoc.com:7050 -c $CHANNEL_NAME
-f ./channel-artifacts/bancochannel.tx --tls $CORE_PEER_TLS_ENABLED
--cafile $ORDERER_CA
```

La salida de este commando se muestra en el anexo "Salida comando creación de canal".

5.2.7. Incorporar organizaciones al canal

En este apartado se incorporarán los nodos de las organizaciones al canal. Los comandos necesarios serían los siguientes, en los que solo se muestran los comandos para el nodo peer0 de cada organización por brevedad.

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
CORE_PEER_LOCALMSPID="bancoMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-uoc.com/tls/ca.crt
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/users/Admin@banco.mistic-uoc.com/msp
CORE_PEER_ADDRESS=peer0.banco.mistic-uoc.com:7051
CHANNEL_NAME=bancochannel
CORE_PEER_TLS_ENABLED=true
```

```
peer channel join -b $CHANNEL_NAME.block
```

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
CORE_PEER_LOCALMSPID="depositarioMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/tls/ca.crt
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/msp
CORE_PEER_ADDRESS=peer0.depositario.mistic-uoc.com:7051
CHANNEL_NAME=bancochannel
CORE_PEER_TLS_ENABLED=true

peer channel join -b $CHANNEL_NAME.block
```

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
CORE_PEER_LOCALMSPID="reguladorMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/tls/ca.crt
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/msp
CORE_PEER_ADDRESS=peer0.regulador.mistic-uoc.com:7051
CHANNEL_NAME=bancochannel
CORE_PEER_TLS_ENABLED=true

peer channel join -b $CHANNEL_NAME.block
```

La salida de los tres comandos se muestra en el anexo “Salida incorporación nodos al canal”.

5.2.8. Instalación de un chaincode básico en los nodos de las organizaciones

Como parte de la prueba de concepto, se ha realizado un chaincode básico que incorpora y modifica pares clave-valor en la base de datos de estado del ledger. Se ha tomado como modelo el chaincode implementado en la referencia [41].

El código de dicho chaincode se incorpora en el anexo “Chaincode crearCuentas”. El funcionamiento de dicho ejemplo es el siguiente:

- En el método init, y a la hora de instanciar el programa, crear dos pares clave-valor:

```
`Inversor1","100","Inversor2","200"
```

Dichos valores representan la información de dos inversores y su saldo.

- En el método query devuelve el saldo del inversor “Inversor1”.
- En el método Invoke, sustrae una cantidad que se pasa por párametro, de un inversor a otro.

Para instalar dicho programa en el nodo peer0.banco.mistic-uoc.com, se ejecutaría el siguiente comando:

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/orderer
Organizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
CORE_PEER_LOCALMSPID="depositarioMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/pe
er/crypto/peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/tls/ca.crt
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/c
rypto/peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/msp
CORE_PEER_ADDRESS=peer0.depositario.mistic-uoc.com:7051
CHANNEL_NAME=bancochannel
CORE_PEER_TLS_ENABLED=true
peer chaincode install -n crearCuentas -v 1.0 -p
github.com/hyperledger/fabric/examples/chaincode/go/chaincode_crearCuentas
```

La salida de dicho comando se muestra en el anexo “Instalación del chaincode en el nodo peer0.banco.mistic-uoc.com”.

5.2.9. Instanciación del chaincode en el canal

Para que un chaincode pueda ser invocado, ha de ser instanciado en el canal correspondiente, con unos valores de inicialización. El comando para realizar esta acción sería el siguiente:

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
CORE_PEER_LOCALMSPID="bancoMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-uoc.com/tls/ca.crt
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/users/Admin@banco.mistic-uoc.com/msp
CORE_PEER_ADDRESS=peer0.banco.mistic-uoc.com:7051
CHANNEL_NAME=bancochannel
CORE_PEER_TLS_ENABLED=true
peer chaincode instantiate -o orderer.mistic-uoc.com:7050 --tls
$CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C $CHANNEL_NAME -n
crearCuentas -v 1.0 -c
'{"Args":["init","Inversor1","100","Inversor2","200"]}'
```

Como se ha comentado anteriormente, crea dos pares clave-valor en la base de datos, representado dos instancias de un active. La salida de dicho comando se muestra en el anexo “instanciación del programa”.

5.2.10. Ejecución de una función tipo Query

El siguiente paso será invocar la función query del chaincode, para comprobar el saldo del inversor “inversorA”. Dicha acción se realizaría con el siguiente comando:

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
CORE_PEER_LOCALMSPID="bancoMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-uoc.com/tls/ca.crt
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/users/Admin@banco.mistic-uoc.com/msp
CORE_PEER_ADDRESS=peer0.banco.mistic-uoc.com:7051
CHANNEL_NAME=bancochannel
CORE_PEER_TLS_ENABLED=true

peer chaincode query -C $CHANNEL_NAME -n crearCuentas -c
'{"Args":["query","Inversor1"]}'
```

El resultado de dicha llamada se muestra a continuación de forma resumida, y en el anexo “Salida ejecución query” de forma completa.

```
InversorA: 100
```

5.2.11. Ejecución de una función tipo Invoke

Como último paso, se ejecutará la función invoke del chaincode ejemplo. En dicho código, se resta una cantidad de dinero del saldo de un inversor, y se transfiere a otro. La ejecución de dicha función invoke sería la siguiente:

```
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
CORE_PEER_LOCALMSPID="bancoMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-uoc.com/tls/ca.crt
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/users/Admin@banco.mistic-uoc.com/msp
CORE_PEER_ADDRESS=peer0.banco.mistic-uoc.com:7051
```

```
CHANNEL_NAME=bancochannel
CORE_PEER_TLS_ENABLED=true

peer chaincode invoke -o orderer.mistic-uoc.com:7050  --tls
$CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C $CHANNEL_NAME -n
crearCuentas -c '{"Args":["invoke","Inversor1","Inversor2","10"]}'
```

La salida de la invocación a la función invoke se muestra de forma resumida en la siguiente línea, y de forma completa en el anexo “salida ejecución invoke”.

```
INFO 04f Chaincode invoke successful. result: status:200
```

Para comprobar que se ha realizado correctamente, se vuelve a invocar la función query:

```
peer chaincode query -C $CHANNEL_NAME -n crearCuentas -c
'{"Args":["query","Inversor1"]}'
```

Y se obtiene un saldo de 90.

6. Conclusiones.

Mediante el presente Trabajo Fin de Máster se ha podido realizar una profunda investigación sobre las características comunes que nos pueden ofrecer las distintas soluciones DLT, basadas en blockchain o no, en cuanto a privacidad, métodos de participación, mecanismos de consenso o comunicación con elementos externos.

Se ha podido partir y analizar una solución concreta, como es el mecanismo DVP, para, tras analizar todos los componentes de una implementación concreta como es Hyperledger Fabric, realizar un diseño concreto de una red. En dicho diseño se han tenido en cuenta conclusiones de seguridad, redundancia y tolerancia a fallos.

Finalmente, se ha realizado una prueba de concepto para reflejar la implementación práctica de una red Hyperledger con las conclusiones de la solución.

Como resultado de los trabajos indicados, se ha puesto de manifiesto la complejidad teórica que presenta este tipo de soluciones. Se basan en un nuevo paradigma de la información, frente a los modelos tradicionales de almacenamiento como son las base de datos centralizadas. Pero, por otro lado, una vez se han adquirido los conocimientos necesarios, es más sencillo implementar y pensar en modelos de aplicación en cualquier ámbito de la información.

También se concluye con la importancia en el mantenimiento de las características de seguridad que requieren estas soluciones, en cuanto a integridad, disponibilidad y autenticidad de la información, y cómo las proporciona la implementación Hyperledger Fabric.

Por otro lado, se ha mostrado que Hyperledger Fabric, debido al rápido consenso alcanzado por los nodos de aprobación y ordenación, es una solución a las operaciones que requieran ser tratadas con rapidez, en comparación con otras redes de consenso “proof-of-work” tipo Bitcoin.

La implementación de mecanismos de intercambio de información mediante chaincodes en una arquitectura distribuida cambia el paradigma de programación. Tenemos que tener en cuenta que se accede a unas funciones que se encuentran disponibles en un canal compartido, con la limitación de seguridad que impone que solo pueden ser invocadas desde los nodos de las organizaciones en los que se ha instalado la aplicación, y la necesidad de ser aprobadas por los nodos. Además, los activos forman parte de una base de datos distribuidas.

En cuanto al framework Hyperledger Fabric se puede destacar que se encuentra en un gran nivel de madurez, si bien se hace imprescindible la creación del módulo de consenso “Simplified Byzantine Fault Tolerance” en los nodos de ordenación.

7. Líneas de trabajo futuras.

A partir del presente trabajo fin de máster, se sugieren las siguientes propuestas de trabajo.

- Modificar el mecanismo DVP para que se cumpla la propiedad de confidencialidad frente al TTP.
- Implementar de forma completa, mediante chaincodes, todo el mecanismo DVP comentado.
- Interactuar con una plataforma de pruebas de alguna criptomoneda comercial (denominadas testnets).
- Evaluar la posibilidad de incorporar la comunicación con otra solución DLT empleando The Interledger Protocol.
- Emplear sistemas de identificación Open Authorization (OAuth) para la autenticación de los nodos.

8. Referencias

[1] Estratégicas tecnologías en el 2018 para Gartner

https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2018/

[2] Nodos de blockchain

https://www.blockchain.com/es/pools

[3] Desarrollo del concepto del problema de generales bizantinos

https://www.microsoft.com/en-us/research/uploads/prod/2016/12/The-Byzantine-Generals-Problem.pdf

[4] Problema de generales bizantinos en bitcoin

https://web.ua.es/en/recsi2014/documentos/papers/bitcoins-y-el-problema-de-los-generales-bizantinos.pdf

[5] Consumo de energía en la red Bitcoin

https://digiconomist.net/bitcoin-energy-consumption

[6] Hardfork Casper de Ethereum

https://blockonomi.com/ethereum-casper/

[7] Blockchain privadas versus públicas

https://bitfury.com/content/downloads/public-vs-private-pt1-1.pdf

[8] Protocolo de consenso stellar

https://www.stellar.org/blog/stellar-consensus-protocol-proof-code/

[9] Modelo de un sistema tolerante a fallos bizantinos práctico

http://pmg.csail.mit.edu/papers/osdi99.pdf

[10] Relación entre la confianza y el anonimato de los validadores

https://medium.com/@pavelkravchenko/ok-i-need-a-blockchain-but-which-one-ca75c1e2100

[11] Funciones de Iberclear

https://www.bde.es/bde/es/areas/sispago/Sistemas_de_comp/vigilancia-de-lo/Iberclear.html

[12] Infografía del Banco Central Europeo sobre tecnologías DLTs

https://www.ecb.europa.eu/pub/annual/special-features/2016/html/index.en.html

[13] Grupo de trabajo del Banco Central Europeo sobre DLT

https://www.ecb.europa.eu/ecb/access_to_documents/document/dialogue/distributed_ledger_technologies_task_force_(dlt-tf)/html/index.en.html

[14] "Análisis de los beneficios de Target2-Securities", elaborado por PWC

http://www.clearstream.com/blob/6220/fea603b397e51f16a0256b31fda02ad2/migrated-9b3hc6580nsgden-t2s-pwc-paper-pdf-data.pdf

[15] "Métodos de pago, valores y derivados, y el role del Eurosistema", elaborado por el Banco Central Europeo.

https://www.ecb.europa.eu/pub/pdf/other/paymentsystem201009en.pdf

[16] "Stella. Sistemas de pago, mecanismos de ahorro de liquidez en una solución DLT".

https://www.ecb.europa.eu/pub/pdf/other/ecb.stella_project_report_september_2017.pdf

[17] Actores de Target2-Securities

https://www.ecb.europa.eu/pub/pdf/scpops/ecbop172.en.pdf

[18] Servicios de Oraclize

http://www.oraclize.it/

[19] Oraclize sobre Hyperledger Fabric

https://blog.oraclize.it/launching-the-oraclize-service-on-Hyperledger -fabric-c336c2d7d9b1

[20] Miembros de Hyperledger

https://www.Hyperledger .org/members

[21] Proyectos de Hyperledger

https://www.Hyperledger .org/projects

[22] Protocolo "The Interledger Protocol"

https://interledger.org/rfcs/0003-interledger-protocol/

[23] RFC 791 sobre "The internet Protocol"

https://tools.ietf.org/html/rfc791

[24] "Qué es Target2-Securities" elaborado por el Banco Central Europeo

https://www.ecb.europa.eu/paym/t2s/stakeholders/csd/html/index.en.html

[25] Enlaces de Iberclear con otros DCVs

http://www.iberclear.es/esp/Servicios/Enlaces-con-otros-DCV

[26] Miembros del grupo de trabajo "DLT-TF"

https://www.ecb.europa.eu/ecb/access_to_documents/document/pa_document/shared/data/ecb.dr.gro2016DLTTF_composition.en.pdf?979d599cb9c7b45c20074b0ab6cdb8f3

[27] Ejemplo de chaincode.

https://github.com/IBM-Blockchain-Archive/learn-chaincode

[28] Flujo completo de una transacción.

https://Hyperledger -fabric.readthedocs.io/en/release-1.3/arch-deep-dive.html#swimlane

[29] Interacción entre una aplicación y un nodo

https://Hyperledger -fabric.readthedocs.io/en/release-1.3/peers/peers.html

[30] Representación de una base de datos de estado

https://Hyperledger -fabric.readthedocs.io/en/release-1.3/ledger/ledger.html

[31] Esquema de Kafka

https://codeburst.io/the-abcs-of-kafka-in-Hyperledger -fabric-81e6dc18da56

[32] A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform

http://www.di.fc.ul.pt/~bessani/publications/dsn18-hlfsmart.pdf

[33] A Kafka-based Ordering Service for Fabric

https://docs.google.com/document/d/19JihmW-8blTzN99lAubOfseLUZqdrB6sBR0HsRgCAnY/

[34] Ámbitos de los MSP

https://hyperledger-fabric.readthedocs.io/en/release-1.3/membership/membership.html

[35] “Release notes de Hyperledger Fabric 1.3”, página 207.

https://media.readthedocs.org/pdf/Hyperledger -fabric/stable/Hyperledger -fabric.pdf

[36] Explicación del soporte TLSNotary en Oraclize

https://blog.oraclize.it/understanding-oracles-99055c9c9f7b

[37] Ciclo de vida de una transacción

https://hyperledger-fabric.readthedocs.io/en/release-1.3/txflow.html

[38] Instalación de pre-requisitos de Hyperledger

https://hyperledger-fabric.readthedocs.io/en/release-1.3/prereqs.html

[39] Ruta de descarga de GO

https://golang.org/dl/

[40] Instalación de Hyperledger

https://hyperledger-fabric.readthedocs.io/en/release-1.3/install.html

[41] Ejemplo de configuración de una red Hyperledger Fabric

https://github.com/mtnieto/hf-2orgs

[42] NEM, blockchain de activos inteligentes.

https://hardwareate.com/nem-blockchain-activos-inteligentes

[43] Problema de los generales bizantinos.

http://www.blockchainservices.es/formacion/el-problema-de-los-generales-bizantinos/

[44] Cadena de bloques de Bitcoin.

http://libroblockchain.com/consenso/

[45] Funcionalidad de Oraclize.

http://www.oraclize.it/

[46] Torre de protocolos de The Interledger Protocol

https://interledger.org/rfcs/0003-interledger-protocol/

[47] Modelo de operación de The Interledger Protocol

https://interledger.org/rfcs/0003-interledger-protocol/#model-of-operation

9. Anexos

En el siguiente anexo se detallarán los ficheros relenvantes y mencionados durante la prueba de concepto.

9.1. Fichero crypto-config.yaml

```
OrdererOrgs:
  - Name: Orderer
    Domain: mistic-uoc.com
    CA:
        Country: ES
        Province: Barcelona
        Locality: Barcelona
    Specs:
      - Hostname: orderer


PeerOrgs:
  - Name: banco
    Domain: banco.mistic-uoc.com
    EnableNodeOUs: true
    CA:
      Country: ES
      Province: Barcelona
      Locality: Barcelona
    Template:
      Count: 4
    Users:
      Count: 1

  - Name: depositario
    Domain: depositario.mistic-uoc.com
    EnableNodeOUs: true
    CA:
        Country: ES
        Province: Barcelona
        Locality: Barcelona
    Template:
      Count: 4
    Users:
      Count: 1

  - Name: regulador
    Domain: regulador.mistic-uoc.com
    EnableNodeOUs: true
    CA:
        Country: ES
```

```
        Province: Barcelona
        Locality: Barcelona
    Template:
      Count: 4
    Users:
      Count: 1
```

9.2. Salida commando cryptogen

```
./peerOrganizations
./peerOrganizations/depositario.mistic-uoc.com
./peerOrganizations/depositario.mistic-uoc.com/peers
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/msp
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/msp/signcerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-
uoc.com/msp/signcerts/peer3.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/msp/keystore
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-
uoc.com/msp/keystore/ead4804cc26da3680adcc4cc692f3f7f33ff71e219e9abb
030a20d4b4a2b9ee6_sk
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/msp/config.yaml
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/msp/cacerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/msp/admincerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-
uoc.com/msp/admincerts/Admin@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/tls
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/tls/server.key
```

```
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/tls/ca.crt
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-uoc.com/tls/server.crt
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/msp
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/msp/signcerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-
uoc.com/msp/signcerts/peer1.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/msp/keystore
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-
uoc.com/msp/keystore/39e8449a778559e89025c45d784e249c0fddda3df8123b6
ac1c7de8669b33451_sk
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/msp/config.yaml
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/msp/cacerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/msp/admincerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-
uoc.com/msp/admincerts/Admin@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/tls
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/tls/server.key
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/tls/ca.crt
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-uoc.com/tls/server.crt
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/msp
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/msp/signcerts
```

```
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-
uoc.com/msp/signcerts/peer2.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/msp/keystore
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-
uoc.com/msp/keystore/ed002c791ce62cee6b4a3b8d757e9bba91145b0a9b123e2
aa4d92bf31bb38648_sk
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/msp/config.yaml
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/msp/cacerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/msp/admincerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-
uoc.com/msp/admincerts/Admin@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/tls
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/tls/server.key
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/tls/ca.crt
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-uoc.com/tls/server.crt
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/msp
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/msp/signcerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-
uoc.com/msp/signcerts/peer0.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/msp/keystore
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-
uoc.com/msp/keystore/86045685f4486c22d470451b9671b855b0eb15487230071
8b40ef9012308f902_sk
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/msp/config.yaml
```

```
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/msp/cacerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/msp/admincerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-
uoc.com/msp/admincerts/Admin@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/tls
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/tls/server.key
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/tls/ca.crt
./peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-uoc.com/tls/server.crt
./peerOrganizations/depositario.mistic-uoc.com/users
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/msp
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/msp/signcerts
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-
uoc.com/msp/signcerts/User1@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/msp/keystore
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-
uoc.com/msp/keystore/7e55001d99a4e2e088664ce3c844e06516260626472fce6
d25bd6fd0a572e3f7_sk
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/msp/cacerts
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/msp/admincerts
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-
uoc.com/msp/admincerts/User1@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/msp/tlscacerts
```

```
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/tls
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/tls/client.crt
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/tls/ca.crt
./peerOrganizations/depositario.mistic-
uoc.com/users/User1@depositario.mistic-uoc.com/tls/client.key
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/msp
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/msp/signcerts
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/signcerts/Admin@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/msp/keystore
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/keystore/97b74cfb535ee951720a89d2daf64c3301d0a184363eef3
13f308e287be45c91_sk
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/msp/cacerts
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/msp/admincerts
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/admincerts/Admin@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/tls
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/tls/client.crt
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/tls/ca.crt
./peerOrganizations/depositario.mistic-
uoc.com/users/Admin@depositario.mistic-uoc.com/tls/client.key
./peerOrganizations/depositario.mistic-uoc.com/tlsca
```

```
./peerOrganizations/depositario.mistic-
uoc.com/tlsca/aac04cc03b2c2090e673724a17985751ce8ae40eea6ed7f6cea69f
44e91ebe59_sk
./peerOrganizations/depositario.mistic-
uoc.com/tlsca/tlsca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-uoc.com/msp
./peerOrganizations/depositario.mistic-uoc.com/msp/config.yaml
./peerOrganizations/depositario.mistic-uoc.com/msp/cacerts
./peerOrganizations/depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-uoc.com/msp/admincerts
./peerOrganizations/depositario.mistic-
uoc.com/msp/admincerts/Admin@depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-uoc.com/ca
./peerOrganizations/depositario.mistic-
uoc.com/ca/ca.depositario.mistic-uoc.com-cert.pem
./peerOrganizations/depositario.mistic-
uoc.com/ca/1461b36ac52f427a46f1fcf57aa99b9adfaecd474e85820e645866926
04e7c3b_sk
./peerOrganizations/regulador.mistic-uoc.com
./peerOrganizations/regulador.mistic-uoc.com/peers
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/msp
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/msp/signcerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-
uoc.com/msp/signcerts/peer1.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/msp/keystore
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-
uoc.com/msp/keystore/7dbeb767467cdf54cb2b4106782df0009a48f2d76e59958
638a8ffe0acbf03c4_sk
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/msp/config.yaml
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/msp/cacerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-
uoc.com/msp/cacerts/ca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/msp/admincerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-
uoc.com/msp/admincerts/Admin@regulador.mistic-uoc.com-cert.pem
```

```
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-
uoc.com/msp/tlscacerts/tlsca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/tls
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/tls/server.key
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/tls/ca.crt
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-uoc.com/tls/server.crt
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/msp
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/msp/signcerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-
uoc.com/msp/signcerts/peer2.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/msp/keystore
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-
uoc.com/msp/keystore/fb4052cbb8d7ad36da129d8890b601c96a21c2624409c90
87727e940584de81d_sk
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/msp/config.yaml
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/msp/cacerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-
uoc.com/msp/cacerts/ca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/msp/admincerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-
uoc.com/msp/admincerts/Admin@regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-
uoc.com/msp/tlscacerts/tlsca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/tls
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/tls/server.key
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/tls/ca.crt
```

```
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-uoc.com/tls/server.crt
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/msp
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/msp/signcerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-
uoc.com/msp/signcerts/peer0.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/msp/keystore
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-
uoc.com/msp/keystore/24818f64e2bbdd346460f775cba12c68722ba2735e89770
5492ac8bf5cc49144_sk
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/msp/config.yaml
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/msp/cacerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-
uoc.com/msp/cacerts/ca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/msp/admincerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-
uoc.com/msp/admincerts/Admin@regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-
uoc.com/msp/tlscacerts/tlsca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/tls
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/tls/server.key
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/tls/ca.crt
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-uoc.com/tls/server.crt
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/msp
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/msp/signcerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-
uoc.com/msp/signcerts/peer3.regulador.mistic-uoc.com-cert.pem
```

```
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/msp/keystore
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-
uoc.com/msp/keystore/824a30aa96a9205c61111a28fb7368f2ef0b8a3d82f51ca
296ba16ca41e1a14a_sk
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/msp/config.yaml
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/msp/cacerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-
uoc.com/msp/cacerts/ca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/msp/admincerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-
uoc.com/msp/admincerts/Admin@regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-
uoc.com/msp/tlscacerts/tlsca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/tls
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/tls/server.key
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/tls/ca.crt
./peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-uoc.com/tls/server.crt
./peerOrganizations/regulador.mistic-uoc.com/users
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/msp
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/msp/signcerts
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-
uoc.com/msp/signcerts/User1@regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/msp/keystore
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-
uoc.com/msp/keystore/7b799c4808c5a48312ffaaf14286a7a209356bc88b61586
a24722e39ba3e98d3_sk
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/msp/cacerts
```

```
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-
uoc.com/msp/cacerts/ca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/msp/admincerts
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-
uoc.com/msp/admincerts/User1@regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-
uoc.com/msp/tlscacerts/tlsca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/tls
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/tls/client.crt
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/tls/ca.crt
./peerOrganizations/regulador.mistic-
uoc.com/users/User1@regulador.mistic-uoc.com/tls/client.key
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/msp
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/msp/signcerts
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/signcerts/Admin@regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/msp/keystore
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/keystore/07f9b6486c443122d3f3ea2f0039bcfa0a6f08adde97a22
8e9110e99125d223d_sk
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/msp/cacerts
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/cacerts/ca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/msp/admincerts
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/admincerts/Admin@regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/tlscacerts/tlsca.regulador.mistic-uoc.com-cert.pem
```

```
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/tls
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/tls/client.crt
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/tls/ca.crt
./peerOrganizations/regulador.mistic-
uoc.com/users/Admin@regulador.mistic-uoc.com/tls/client.key
./peerOrganizations/regulador.mistic-uoc.com/tlsca
./peerOrganizations/regulador.mistic-
uoc.com/tlsca/8f06e9a2bb57483d7b1ecce59ad67e74d2dd115b35c7652bf9d486
9377f64e63_sk
./peerOrganizations/regulador.mistic-
uoc.com/tlsca/tlsca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-uoc.com/msp
./peerOrganizations/regulador.mistic-uoc.com/msp/config.yaml
./peerOrganizations/regulador.mistic-uoc.com/msp/cacerts
./peerOrganizations/regulador.mistic-
uoc.com/msp/cacerts/ca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-uoc.com/msp/admincerts
./peerOrganizations/regulador.mistic-
uoc.com/msp/admincerts/Admin@regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/regulador.mistic-
uoc.com/msp/tlscacerts/tlsca.regulador.mistic-uoc.com-cert.pem
./peerOrganizations/regulador.mistic-uoc.com/ca
./peerOrganizations/regulador.mistic-uoc.com/ca/ca.regulador.mistic-
uoc.com-cert.pem
./peerOrganizations/regulador.mistic-
uoc.com/ca/451d2987eeda648e1e04cccad85c701bd57a78fc3cbfa9fb7b228ed8c
821a9fe_sk
./peerOrganizations/banco.mistic-uoc.com
./peerOrganizations/banco.mistic-uoc.com/peers
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/signcerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/signcerts/peer3.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/keystore
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/keystore/c76856f763a1b07e1d13e169676635c68dd5a857790031d
23f4390a60d2d677e_sk
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/config.yaml
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/cacerts
```

```
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/admincerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/tlscacerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/tls
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/tls/server.key
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/tls/ca.crt
./peerOrganizations/banco.mistic-uoc.com/peers/peer3.banco.mistic-
uoc.com/tls/server.crt
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/signcerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/signcerts/peer2.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/keystore
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/keystore/8a7d70a8b70f24e1d680af3f8634dd4b99997c2694569cf
c69796a3bace5f3bb_sk
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/config.yaml
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/cacerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/admincerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/tlscacerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/tls
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/tls/server.key
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/tls/ca.crt
```

```
./peerOrganizations/banco.mistic-uoc.com/peers/peer2.banco.mistic-
uoc.com/tls/server.crt
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/signcerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/signcerts/peer1.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/keystore
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/keystore/249f0d428cce24a9330f65b19718ea6cb5697d5507a9c3c
761f64e7bda41ed4d_sk
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/config.yaml
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/cacerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/admincerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/tlscacerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/tls
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/tls/server.key
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/tls/ca.crt
./peerOrganizations/banco.mistic-uoc.com/peers/peer1.banco.mistic-
uoc.com/tls/server.crt
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/signcerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/signcerts/peer0.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/keystore
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/keystore/130dcd63365ea951a15c84e73aec38c20c17c8132ad6b0d
04ea67f316017248f_sk
```

```
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/config.yaml
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/cacerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/admincerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/tlscacerts
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/tls
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/tls/server.key
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/tls/ca.crt
./peerOrganizations/banco.mistic-uoc.com/peers/peer0.banco.mistic-
uoc.com/tls/server.crt
./peerOrganizations/banco.mistic-uoc.com/users
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts/Admin@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore/d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9e
d81d15b879b71a2dc_sk
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/tls
```

```
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/tls/client.crt
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/tls/ca.crt
./peerOrganizations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/tls/client.key
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/signcerts
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/signcerts/User1@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/keystore
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/keystore/b1298c8bdd98f3ba0c2cacd8e4939adbabad9e45303e332
a9733daca0df5c29d_sk
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/cacerts
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/admincerts
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/admincerts/User1@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/tlscacerts
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/tls
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/tls/client.crt
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/tls/ca.crt
./peerOrganizations/banco.mistic-uoc.com/users/User1@banco.mistic-
uoc.com/tls/client.key
./peerOrganizations/banco.mistic-uoc.com/tlsca
./peerOrganizations/banco.mistic-
uoc.com/tlsca/00997f7e480d7cbd08b346c6fd93cea4fda53638c7d6b56a77747a
3e1495bef6_sk
./peerOrganizations/banco.mistic-uoc.com/tlsca/tlsca.banco.mistic-
uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/msp
./peerOrganizations/banco.mistic-uoc.com/msp/config.yaml
./peerOrganizations/banco.mistic-uoc.com/msp/cacerts
./peerOrganizations/banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/msp/admincerts
```

```
./peerOrganizations/banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/msp/tlscacerts
./peerOrganizations/banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
./peerOrganizations/banco.mistic-uoc.com/ca
./peerOrganizations/banco.mistic-uoc.com/ca/ca.banco.mistic-uoc.com-
cert.pem
./peerOrganizations/banco.mistic-
uoc.com/ca/7b596ca78a20ccd961cbe25eecbe7347582318ba284cfb50a1ffae136
4eecc8f_sk
./ordererOrganizations
./ordererOrganizations/mistic-uoc.com
./ordererOrganizations/mistic-uoc.com/orderers
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/signcerts
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/signcerts/orderer.mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/keystore
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/keystore/872525ba8a3676e14c5fe35347716f8d7db555c3b8fc496
bbbdbfcb0061cf01f_sk
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/cacerts
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/cacerts/ca.mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/admincerts
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/admincerts/Admin@mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/tls
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/tls/server.key
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/tls/ca.crt
./ordererOrganizations/mistic-uoc.com/orderers/orderer.mistic-
uoc.com/tls/server.crt
./ordererOrganizations/mistic-uoc.com/users
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-uoc.com
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-uoc.com/msp
```

```
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/signcerts
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/signcerts/Admin@mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/keystore
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/keystore/200b5d730dc643776efd71133b6290a9ed75f8257a6040e
1cbd963f2a801c156_sk
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/cacerts
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/cacerts/ca.mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/admincerts
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/admincerts/Admin@mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/tlscacerts
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/msp/tlscacerts/tlsca.mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-uoc.com/tls
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/tls/client.crt
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/tls/ca.crt
./ordererOrganizations/mistic-uoc.com/users/Admin@mistic-
uoc.com/tls/client.key
./ordererOrganizations/mistic-uoc.com/tlsca
./ordererOrganizations/mistic-
uoc.com/tlsca/b5a99357c02a8d0d51afa0672e72757aa7ba8ff74389d00ae42bc5
90b12505ad_sk
./ordererOrganizations/mistic-uoc.com/tlsca/tlsca.mistic-uoc.com-
cert.pem
./ordererOrganizations/mistic-uoc.com/msp
./ordererOrganizations/mistic-uoc.com/msp/cacerts
./ordererOrganizations/mistic-uoc.com/msp/cacerts/ca.mistic-uoc.com-
cert.pem
./ordererOrganizations/mistic-uoc.com/msp/admincerts
./ordererOrganizations/mistic-uoc.com/msp/admincerts/Admin@mistic-
uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/msp/tlscacerts
./ordererOrganizations/mistic-uoc.com/msp/tlscacerts/tlsca.mistic-
uoc.com-cert.pem
./ordererOrganizations/mistic-uoc.com/ca
./ordererOrganizations/mistic-uoc.com/ca/ca.mistic-uoc.com-cert.pem
./ordererOrganizations/mistic-
uoc.com/ca/7243771025874dffecd405ea982287cf4ffd7cf5c40450874c62550ba
5312f86_sk
```

9.3. Perfiles para cryptogen en el archivo configtx.yaml

```
---
Organizations:

    - &OrdererOrg
        Name: OrdererOrg

        ID: OrdererMSP

        MSPDir: crypto-config/ordererOrganizations/mistic-
uoc.com/msp

        Policies:
            Readers:
                Type: Signature
                Rule: "OR('OrdererMSP.member')"
            Writers:
                Type: Signature
                Rule: "OR('OrdererMSP.member')"
            Admins:
                Type: Signature
                Rule: "OR('OrdererMSP.admin')"

    - &banco
        Name: bancoMSP

        ID: bancoMSP

        MSPDir: crypto-config/peerOrganizations/banco.mistic-
uoc.com/msp

        Policies:
            Readers:
                Type: Signature
                Rule: "OR('bancoMSP.admin', 'bancoMSP.peer',
'bancoMSP.client')"
            Writers:
                Type: Signature
                Rule: "OR('bancoMSP.admin', 'bancoMSP.client')"
            Admins:
                Type: Signature
                Rule: "OR('bancoMSP.admin')"

        AnchorPeers:
            - Host: peer0.banco.mistic-uoc.com
              Port: 7051
            - Host: peer1.banco.mistic-uoc.com
              Port: 7051
```

```
    - &depositario
        Name: depositarioMSP

        ID: depositarioMSP

        MSPDir: crypto-config/peerOrganizations/depositario.mistic-
uoc.com/msp

        Policies:
            Readers:
                Type: Signature
                Rule: "OR('depositarioMSP.admin',
'depositarioMSP.peer', 'depositarioMSP.client')"
            Writers:
                Type: Signature
                Rule: "OR('depositarioMSP.admin',
'depositarioMSP.client')"
            Admins:
                Type: Signature
                Rule: "OR('depositarioMSP.admin')"

        AnchorPeers:
            - Host: peer0.depositario.mistic-uoc.com
              Port: 7051
            - Host: peer1.depositario.mistic-uoc.com
              Port: 7051

    - &regulador
        Name: reguladorMSP

        ID: reguladorMSP

        MSPDir: crypto-config/peerOrganizations/regulador.mistic-
uoc.com/msp

        Policies:
            Readers:
                Type: Signature
                Rule: "OR('reguladorMSP.admin', 'reguladorMSP.peer',
'reguladorMSP.client')"
            Writers:
                Type: Signature
                Rule: "OR('reguladorMSP.admin',
'reguladorMSP.client')"
            Admins:
                Type: Signature
                Rule: "OR('reguladorMSP.admin')"

        AnchorPeers:
            - Host: peer0.regulador.mistic-uoc.com
```

```
              Port: 7051
            - Host: peer1.regulador.mistic-uoc.com
              Port: 7051

Capabilities:
    Global: &ChannelCapabilities
        V1_1: true

    Orderer: &OrdererCapabilities
        V1_1: true

    Application: &ApplicationCapabilities
        V1_2: true

Application: &ApplicationDefaults

    Organizations:

    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"

    Capabilities:
        <<: *ApplicationCapabilities

Orderer: &OrdererDefaults

    OrdererType: solo

    Addresses:
        - orderer.mistic-uoc.com:7050

    BatchTimeout: 2s

    BatchSize:

        MaxMessageCount: 10

        AbsoluteMaxBytes: 98 MB

        PreferredMaxBytes: 512 KB

    Kafka:
        Brokers:
```

```
            - kafka0:9092
            - kafka1:9092
            - kafka2:9092
            - kafka3:9092

    Organizations:

    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
        BlockValidation:
            Type: ImplicitMeta
            Rule: "ANY Writers"

    Capabilities:
        <<: *OrdererCapabilities

Channel: &ChannelDefaults
    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"


    Capabilities:
        <<: *ChannelCapabilities

Profiles:

    BancoOrdererGenesis:
        <<: *ChannelDefaults
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
        Consortiums:
            BancoConsortium:
                Organizations:
```

```
                    - *banco
                    - *depositario
                    - *regulador
    bancochannel:
        Consortium: BancoConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *banco
                - *depositario
                - *regulador
```

9.4. Plantilla de los servicios de la red Hyperledger

```
version: '2'

services:
  caBanco:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-banco
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-
server-config/ca.banco.mistic-uoc.com-cert.pem
      - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-
server-config/CA1_PRIVATE_KEY
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile
/etc/hyperledger/fabric-ca-server-config/ca.banco.mistic-uoc.com-
cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-
config/CA1_PRIVATE_KEY -b admin:adminpw -d'
    volumes:
      - ./crypto-config/peerOrganizations/banco.mistic-
uoc.com/ca/:/etc/hyperledger/fabric-ca-server-config
    container_name: ca_peerBanco

  caDepositario:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-depositario
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-
server-config/ca.depositario.mistic-uoc.com-cert.pem
      - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-
server-config/CA2_PRIVATE_KEY
    ports:
```

```
      - "8054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile
/etc/hyperledger/fabric-ca-server-config/ca.depositario.mistic-
uoc.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-
config/CA2_PRIVATE_KEY -b admin:adminpw -d'
    volumes:
      - ./crypto-config/peerOrganizations/depositario.mistic-
uoc.com/ca/:/etc/hyperledger/fabric-ca-server-config
    container_name: ca_peerDepositario

  caRegulador:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-regulador
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-
server-config/ca.regulador.mistic-uoc.com-cert.pem
      - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-
server-config/CA3_PRIVATE_KEY
    ports:
      - "9054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile
/etc/hyperledger/fabric-ca-server-config/ca.regulador.mistic-
uoc.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-
config/CA3_PRIVATE_KEY -b admin:adminpw -d'
    volumes:
      - ./crypto-config/peerOrganizations/regulador.mistic-
uoc.com/ca/:/etc/hyperledger/fabric-ca-server-config
    container_name: ca_peerRegulador

  orderer.mistic-uoc.com:
    extends:
      file:   base/docker-compose-base.yaml
      service: orderer.mistic-uoc.com
    container_name: orderer.mistic-uoc.com

  peer0.banco.mistic-uoc.com:
    container_name: peer0.banco.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.banco.mistic-uoc.com

  peer0.depositario.mistic-uoc.com:
    container_name: peer0.depositario.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.depositario.mistic-uoc.com

  peer0.regulador.mistic-uoc.com:
    container_name: peer0.regulador.mistic-uoc.com
```

```
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.regulador.mistic-uoc.com

  peer1.banco.mistic-uoc.com:
    container_name: peer1.banco.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer1.banco.mistic-uoc.com

  peer1.depositario.mistic-uoc.com:
    container_name: peer1.depositario.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer1.depositario.mistic-uoc.com

  peer1.regulador.mistic-uoc.com:
    container_name: peer1.regulador.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer1.regulador.mistic-uoc.com

  peer2.banco.mistic-uoc.com:
    container_name: peer2.banco.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer2.banco.mistic-uoc.com

  peer2.depositario.mistic-uoc.com:
    container_name: peer2.depositario.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer2.depositario.mistic-uoc.com

  peer2.regulador.mistic-uoc.com:
    container_name: peer2.regulador.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer2.regulador.mistic-uoc.com

  peer3.banco.mistic-uoc.com:
    container_name: peer3.banco.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer3.banco.mistic-uoc.com

  peer3.depositario.mistic-uoc.com:
    container_name: peer3.depositario.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer3.depositario.mistic-uoc.com
```

```
  peer3.regulador.mistic-uoc.com:
    container_name: peer3.regulador.mistic-uoc.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer3.regulador.mistic-uoc.com

  cli:
    container_name: cli
    image: hyperledger/fabric-tools
    tty: true
    environment:
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_LOGGING_LEVEL=DEBUG
      - CORE_PEER_ID=cli
      - CORE_PEER_ADDRESS=peer0.banco.mistic-uoc.com:7051
      - CORE_PEER_LOCALMSPID=bancoMSP
      - CORE_PEER_TLS_ENABLED=true
      -
CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-uoc.com/tls/server.crt
      -
CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric
/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-uoc.com/tls/server.key
      -
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-uoc.com/tls/ca.crt
      -
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/banco.mistic-
uoc.com/users/Admin@banco.mistic-uoc.com/msp
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    #command: /bin/bash -c './scripts/script.sh ${CHANNEL_NAME};
sleep $TIMEOUT'
    volumes:
        - /var/run/:/host/var/run/
        -
./chaincodes/:/opt/gopath/src/github.com/hyperledger/fabric/examples
/chaincode/go
        - ./crypto-
config/:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
        -
./scripts/:/opt/gopath/src/github.com/hyperledger/fabric/peer/script
s/
        - ./channel-
artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel
-artifacts
```

```
    depends_on:
      - orderer.mistic-uoc.com
      - peer0.banco.mistic-uoc.com
      - peer0.depositario.mistic-uoc.com
      - peer0.regulador.mistic-uoc.com
      - peer1.banco.mistic-uoc.com
      - peer1.depositario.mistic-uoc.com
      - peer1.regulador.mistic-uoc.com
      - peer2.banco.mistic-uoc.com
      - peer2.depositario.mistic-uoc.com
      - peer2.regulador.mistic-uoc.com
      - peer3.banco.mistic-uoc.com
      - peer3.depositario.mistic-uoc.com
      - peer3.regulador.mistic-uoc.com
```

9.5. Fichero docker-compose-base.yaml

```
version: '2'

services:

  orderer.mistic-uoc.com:
    container_name: orderer.mistic-uoc.com
    image: hyperledger/fabric-orderer
    environment:
      - ORDERER_GENERAL_LOGLEVEL=debug
      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
      - ORDERER_GENERAL_GENESISMETHOD=file
      -
ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis
.block
      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
      - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp
      # enabled TLS
      - ORDERER_GENERAL_TLS_ENABLED=true
      -
ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.k
ey
      -
ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.
crt
      -
ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric
    command: orderer
    volumes:
```

```
    - ../channel-
artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesis.blo
ck
    - ../crypto-config/ordererOrganizations/mistic-
uoc.com/orderers/orderer.mistic-
uoc.com/msp:/var/hyperledger/orderer/msp
    - ../crypto-config/ordererOrganizations/mistic-
uoc.com/orderers/orderer.mistic-
uoc.com/tls/:/var/hyperledger/orderer/tls
    ports:
      - 7050:7050

  peer0.banco.mistic-uoc.com:
    container_name: peer0.banco.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer0.banco.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer0.banco.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.banco.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=bancoMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/banco.mistic-
uoc.com/peers/peer0.banco.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9000:9000
      - 9001:9001
      - 9002:9002

  peer0.depositario.mistic-uoc.com:
    container_name: peer0.depositario.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer0.depositario.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer0.depositario.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.depositario.mistic-
uoc.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.depositario.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=depositarioMSP
    volumes:
        - /var/run/:/host/var/run/
```

```
        - ../crypto-config/peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/depositario.mistic-
uoc.com/peers/peer0.depositario.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9003:9003
      - 9004:9004
      - 9005:9005

  peer0.regulador.mistic-uoc.com:
    container_name: peer0.regulador.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer0.regulador.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer0.regulador.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.regulador.mistic-
uoc.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.regulador.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=reguladorMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/regulador.mistic-
uoc.com/peers/peer0.regulador.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9006:9006
      - 9007:9007
      - 9008:9008

  peer1.banco.mistic-uoc.com:
    container_name: peer1.banco.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer1.banco.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer1.banco.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.banco.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=bancoMSP
    volumes:
        - /var/run/:/host/var/run/
```

```
        - ../crypto-config/peerOrganizations/banco.mistic-
uoc.com/peers/peer1.banco.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/banco.mistic-
uoc.com/peers/peer1.banco.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9009:9009
      - 9010:9010
      - 9011:9011

  peer1.depositario.mistic-uoc.com:
    container_name: peer1.depositario.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer1.depositario.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer1.depositario.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.depositario.mistic-
uoc.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.depositario.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=depositarioMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/depositario.mistic-
uoc.com/peers/peer1.depositario.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9012:9012
      - 9013:9013
      - 9014:9014

  peer1.regulador.mistic-uoc.com:
    container_name: peer1.regulador.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer1.regulador.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer1.regulador.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.regulador.mistic-
uoc.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.regulador.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=reguladorMSP
    volumes:
```

```
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/regulador.mistic-
uoc.com/peers/peer1.regulador.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9015:9015
      - 9016:9016
      - 9017:9017

  peer2.banco.mistic-uoc.com:
    container_name: peer2.banco.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer2.banco.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer2.banco.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer2.banco.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=bancoMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/banco.mistic-
uoc.com/peers/peer2.banco.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/banco.mistic-
uoc.com/peers/peer2.banco.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9018:9018
      - 9019:9019
      - 9020:9020

  peer2.depositario.mistic-uoc.com:
    container_name: peer2.depositario.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer2.depositario.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer2.depositario.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer2.depositario.mistic-
uoc.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer2.depositario.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=depositarioMSP
    volumes:
        - /var/run/:/host/var/run/
```

```
        - ../crypto-config/peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/depositario.mistic-
uoc.com/peers/peer2.depositario.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9021:9021
      - 9022:9022
      - 9023:9023



  peer2.regulador.mistic-uoc.com:
    container_name: peer2.regulador.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer2.regulador.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer2.regulador.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer2.regulador.mistic-
uoc.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer2.regulador.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=reguladorMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/regulador.mistic-
uoc.com/peers/peer2.regulador.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9024:9024
      - 9025:9025
      - 9026:9026



  peer3.banco.mistic-uoc.com:
    container_name: peer3.banco.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer3.banco.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer3.banco.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer3.banco.mistic-
uoc.com:7051
```

```
      - CORE_PEER_LOCALMSPID=bancoMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/banco.mistic-
uoc.com/peers/peer3.banco.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/banco.mistic-
uoc.com/peers/peer3.banco.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9027:9027
      - 9028:9028
      - 9029:9029

  peer3.depositario.mistic-uoc.com:
    container_name: peer3.depositario.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer3.depositario.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer3.depositario.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer3.depositario.mistic-
uoc.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer3.depositario.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=depositarioMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/depositario.mistic-
uoc.com/peers/peer3.depositario.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9030:9030
      - 9031:9031
      - 9032:9032

  peer3.regulador.mistic-uoc.com:
    container_name: peer3.regulador.mistic-uoc.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer3.regulador.mistic-uoc.com
      - CORE_PEER_ADDRESS=peer3.regulador.mistic-uoc.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer3.regulador.mistic-
uoc.com:7051
```

```
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer3.regulador.mistic-
uoc.com:7051
      - CORE_PEER_LOCALMSPID=reguladorMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-
uoc.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/regulador.mistic-
uoc.com/peers/peer3.regulador.mistic-
uoc.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 9033:9033
      - 9034:9034
      - 9035:9035
```

9.6. Fichero peer-base.yaml

```
version: '2'
services:
  peer-base:
    image: hyperledger/fabric-peer
    environment:
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=sistemadvp_default
      - CORE_LOGGING_LEVEL=DEBUG
      - CORE_PEER_TLS_ENABLED=true
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - CORE_PEER_PROFILE_ENABLED=true
      - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
      - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
      - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: peer node start
```

9.7. Salida comando creación de canal

```
2018-12-31 02:47:19.509 UTC [viperutil] getKeysRecursively -> DEBU
001 Found map[string]interface{} value for peer.BCCSP
2018-12-31 02:47:19.509 UTC [viperutil] getKeysRecursively -> DEBU
002 Found map[string]interface{} value for peer.BCCSP.SW
2018-12-31 02:47:19.526 UTC [viperutil] getKeysRecursively -> DEBU
003 Found map[string]interface{} value for
peer.BCCSP.SW.FileKeyStore
2018-12-31 02:47:19.555 UTC [viperutil] unmarshalJSON -> DEBU 004
Unmarshal JSON: value cannot be unmarshalled: unexpected end of JSON
input
```

```
2018-12-31 02:47:19.555 UTC [viperutil] getKeysRecursively -> DEBU
005 Found real value for peer.BCCSP.SW.FileKeyStore.KeyStore setting
to string
2018-12-31 02:47:19.555 UTC [viperutil] unmarshalJSON -> DEBU 006
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 02:47:19.555 UTC [viperutil] getKeysRecursively -> DEBU
007 Found real value for peer.BCCSP.SW.Hash setting to string SHA2
2018-12-31 02:47:19.556 UTC [viperutil] unmarshalJSON -> DEBU 008
Unmarshal JSON: value is not a string: 256
2018-12-31 02:47:19.556 UTC [viperutil] getKeysRecursively -> DEBU
009 Found real value for peer.BCCSP.SW.Security setting to int 256
2018-12-31 02:47:19.560 UTC [viperutil] getKeysRecursively -> DEBU
00a Found map[string]interface{} value for peer.BCCSP.PKCS11
2018-12-31 02:47:19.586 UTC [viperutil] unmarshalJSON -> DEBU 00b
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:47:19.596 UTC [viperutil] getKeysRecursively -> DEBU
00c Found real value for peer.BCCSP.PKCS11.Pin setting to <nil>
<nil>
2018-12-31 02:47:19.602 UTC [viperutil] unmarshalJSON -> DEBU 00d
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:47:19.602 UTC [viperutil] getKeysRecursively -> DEBU
00e Found real value for peer.BCCSP.PKCS11.Hash setting to <nil>
<nil>
2018-12-31 02:47:19.603 UTC [viperutil] unmarshalJSON -> DEBU 00f
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:47:19.603 UTC [viperutil] getKeysRecursively -> DEBU
010 Found real value for peer.BCCSP.PKCS11.Security setting to <nil>
<nil>
2018-12-31 02:47:19.603 UTC [viperutil] getKeysRecursively -> DEBU
011 Found map[string]interface{} value for
peer.BCCSP.PKCS11.FileKeyStore
2018-12-31 02:47:19.604 UTC [viperutil] unmarshalJSON -> DEBU 012
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:47:19.605 UTC [viperutil] getKeysRecursively -> DEBU
013 Found real value for peer.BCCSP.PKCS11.FileKeyStore.KeyStore
setting to <nil> <nil>
2018-12-31 02:47:19.605 UTC [viperutil] unmarshalJSON -> DEBU 014
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:47:19.605 UTC [viperutil] getKeysRecursively -> DEBU
015 Found real value for peer.BCCSP.PKCS11.Library setting to <nil>
<nil>
2018-12-31 02:47:19.606 UTC [viperutil] unmarshalJSON -> DEBU 016
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:47:19.609 UTC [viperutil] getKeysRecursively -> DEBU
017 Found real value for peer.BCCSP.PKCS11.Label setting to <nil>
<nil>
2018-12-31 02:47:19.613 UTC [viperutil] unmarshalJSON -> DEBU 018
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
```

```
2018-12-31 02:47:19.631 UTC [viperutil] getKeysRecursively -> DEBU
019 Found real value for peer.BCCSP.Default setting to string SW
2018-12-31 02:47:19.635 UTC [viperutil] EnhancedExactUnmarshalKey ->
DEBU 01a map[peer.BCCSP:map[SW:map[Hash:SHA2 Security:256
FileKeyStore:map[KeyStore:]] PKCS11:map[Label:<nil> Pin:<nil>
Hash:<nil> Security:<nil> FileKeyStore:map[KeyStore:<nil>]
Library:<nil>] Default:SW]]
2018-12-31 02:47:19.636 UTC [bccsp_sw] openKeyStore -> DEBU 01b
KeyStore opened at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore]...done
2018-12-31 02:47:19.652 UTC [bccsp] initBCCSP -> DEBU 01c Initialize
BCCSP [SW]
2018-12-31 02:47:19.652 UTC [msp] getPemMaterialFromDir -> DEBU 01d
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts
2018-12-31 02:47:19.652 UTC [msp] getPemMaterialFromDir -> DEBU 01e
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 02:47:19.654 UTC [msp] getPemMaterialFromDir -> DEBU 01f
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts
2018-12-31 02:47:19.654 UTC [msp] getPemMaterialFromDir -> DEBU 020
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
2018-12-31 02:47:19.655 UTC [msp] getPemMaterialFromDir -> DEBU 021
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts
2018-12-31 02:47:19.657 UTC [msp] getPemMaterialFromDir -> DEBU 022
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 02:47:19.658 UTC [msp] getPemMaterialFromDir -> DEBU 023
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts
2018-12-31 02:47:19.659 UTC [msp] getMspConfig -> DEBU 024
Intermediate certs folder not found at
```

```
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts: no such file or directory]
2018-12-31 02:47:19.659 UTC [msp] getPemMaterialFromDir -> DEBU 025
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts
2018-12-31 02:47:19.660 UTC [msp] getPemMaterialFromDir -> DEBU 026
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
2018-12-31 02:47:19.664 UTC [msp] getPemMaterialFromDir -> DEBU 027
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts
2018-12-31 02:47:19.664 UTC [msp] getMspConfig -> DEBU 028 TLS
intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts: no such file or directory]
2018-12-31 02:47:19.664 UTC [msp] getPemMaterialFromDir -> DEBU 029
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls
2018-12-31 02:47:19.664 UTC [msp] getMspConfig -> DEBU 02a crls
folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls: no such file or directory]
2018-12-31 02:47:19.664 UTC [msp] getMspConfig -> DEBU 02b MSP
configuration file not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml]: [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml: no such file or directory]
```

```
2018-12-31 02:47:19.664 UTC [msp] newBccspMsp -> DEBU 02c Creating
BCCSP-based MSP instance
2018-12-31 02:47:19.664 UTC [msp] New -> DEBU 02d Creating Cache-MSP
instance
2018-12-31 02:47:19.664 UTC [msp] loadLocaMSP -> DEBU 02e Created
new local MSP
2018-12-31 02:47:19.665 UTC [msp] Setup -> DEBU 02f Setting up MSP
instance bancoMSP
2018-12-31 02:47:19.669 UTC [msp/identity] newIdentity -> DEBU 030
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICSTCCAe+gAwIBAgIQbgg64bHbeBkccIDXmNIWWDAKBggqhkjOPQQDAjB2MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExHTAbBgNVBAoTFGJhbmNvLm1pc3RpYy11b2MuY29tMSAwHgYDVQQDExdjYS5i
YW5jby5taXN0aWMtdW9jLmNvbTAeFw0xODEyMzEwMTQ5MDBaFw0yODEyMjgwMTQ5
MDBaMHYxCzAJBgNVBAYTAkVTMRIwEAYDVQQIEwlCYXJjZWxvbmExEjAQBgNVBAcT
CUJhcmNlbG9uYTEdMBsGA1UEChMUYmFuY28ubWlzdGljLXVvYy5jb20xIDAeBgNV
BAMTF2NhLmJhbmNvLm1pc3RpYy11b2MuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0D
AQcDQgAExRW9JaiK41T4fy9dxUwGc+zLYcmFN4oAmQL8kDGMVHEonIvJ6sPL1+qE
5iNzvFdEQebaslvxXtzxCiys0PWrc6NfMF0wDgYDVR0PAQH/BAQDAgGmMA8GA1Ud
JQQIMAYGBFUdJQAwDwYDVR0TAQH/BAUwAwEB/zApBgNVHQ4EIgQge1lsp4ogzNlh
y+Je7L5zR1gjGLooTPtQof+uE2TuzI8wCgYIKoZIzj0EAwIDSAAwRQIhAL15n5S/
AG5/Wt6+ZwuLuIRdxCausLypFX8cfrxIPSiuAiAPHY1pQ/788218DnCcw8CcmKoS
RL+l0ZS7OL2DUOkFrA==
-----END CERTIFICATE-----
2018-12-31 02:47:19.669 UTC [msp/identity] newIdentity -> DEBU 031
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 02:47:19.769 UTC [msp/identity] newIdentity -> DEBU 032
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
```

```
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 02:47:19.781 UTC [bccsp_sw] loadPrivateKey -> DEBU 033
Loading private key
[d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9ed81d15b879b71a2dc]
at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore/d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9e
d81d15b879b71a2dc_sk]...
2018-12-31 02:47:19.783 UTC [msp/identity] newIdentity -> DEBU 034
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 02:47:19.783 UTC [msp] setupSigningIdentity -> DEBU 035
Signing identity expires at 2028-12-28 01:49:00 +0000 UTC
2018-12-31 02:47:19.784 UTC [msp] Validate -> DEBU 036 MSP bancoMSP
validating identity
2018-12-31 02:47:19.786 UTC [msp] GetDefaultSigningIdentity -> DEBU
037 Obtaining default signing identity
2018-12-31 02:47:19.789 UTC [grpc] DialContext -> DEBU 038 parsed
scheme: ""
2018-12-31 02:47:19.791 UTC [grpc] DialContext -> DEBU 039 scheme ""
not registered, fallback to default scheme
2018-12-31 02:47:19.791 UTC [grpc] watcher -> DEBU 03a
ccResolverWrapper: sending new addresses to cc: [{orderer.mistic-
uoc.com:7050 0  <nil>}]
2018-12-31 02:47:19.793 UTC [grpc] switchBalancer -> DEBU 03b
ClientConn switching balancer to "pick_first"
2018-12-31 02:47:19.799 UTC [grpc] HandleSubConnStateChange -> DEBU
03c pickfirstBalancer: HandleSubConnStateChange: 0xc4203b4220,
CONNECTING
2018-12-31 02:47:19.881 UTC [grpc] HandleSubConnStateChange -> DEBU
03d pickfirstBalancer: HandleSubConnStateChange: 0xc4203b4220, READY
2018-12-31 02:47:19.882 UTC [channelCmd] InitCmdFactory -> INFO 03e
Endorser and orderer connections initialized
2018-12-31 02:47:19.882 UTC [msp] GetDefaultSigningIdentity -> DEBU
03f Obtaining default signing identity
```

```
2018-12-31 02:47:19.883 UTC [msp] GetDefaultSigningIdentity -> DEBU
040 Obtaining default signing identity
2018-12-31 02:47:19.883 UTC [msp/identity] Sign -> DEBU 041 Sign:
plaintext:
0ABB060A0862616E636F4D535012AE06...0F42616E636F436F6E736F727469756D
2018-12-31 02:47:19.897 UTC [msp/identity] Sign -> DEBU 042 Sign:
digest:
59630650361B9704D255F31BA077F18016FB934F253051D5F6768F70C3357628
2018-12-31 02:47:19.898 UTC [msp] GetDefaultSigningIdentity -> DEBU
043 Obtaining default signing identity
2018-12-31 02:47:19.903 UTC [msp] GetDefaultSigningIdentity -> DEBU
044 Obtaining default signing identity
2018-12-31 02:47:19.906 UTC [msp/identity] Sign -> DEBU 045 Sign:
plaintext:
0AF5060A1808021A0608B786A6E10522...35596665057AFF178440DB615EFC1173
2018-12-31 02:47:19.906 UTC [msp/identity] Sign -> DEBU 046 Sign:
digest:
F29C1D5AA8724163C5AD984D382AAB38FA7A1A4532996D2232078C5DF9714559
2018-12-31 02:47:19.909 UTC [grpc] DialContext -> DEBU 047 parsed
scheme: ""
2018-12-31 02:47:19.909 UTC [grpc] DialContext -> DEBU 048 scheme ""
not registered, fallback to default scheme
2018-12-31 02:47:19.909 UTC [grpc] watcher -> DEBU 049
ccResolverWrapper: sending new addresses to cc: [{orderer.mistic-
uoc.com:7050 0  <nil>}]
2018-12-31 02:47:19.910 UTC [grpc] switchBalancer -> DEBU 04a
ClientConn switching balancer to "pick_first"
2018-12-31 02:47:19.910 UTC [grpc] HandleSubConnStateChange -> DEBU
04b pickfirstBalancer: HandleSubConnStateChange: 0xc4201ef350,
CONNECTING
2018-12-31 02:47:20.040 UTC [grpc] HandleSubConnStateChange -> DEBU
04c pickfirstBalancer: HandleSubConnStateChange: 0xc4201ef350, READY
2018-12-31 02:47:20.553 UTC [msp] GetDefaultSigningIdentity -> DEBU
04d Obtaining default signing identity
2018-12-31 02:47:20.554 UTC [msp] GetDefaultSigningIdentity -> DEBU
04e Obtaining default signing identity
2018-12-31 02:47:20.554 UTC [msp/identity] Sign -> DEBU 04f Sign:
plaintext:
0AF5060A1808051A0608B886A6E10522...70443B1EA65012080A021A0012021A00
2018-12-31 02:47:20.554 UTC [msp/identity] Sign -> DEBU 050 Sign:
digest:
5C5FD0FBC1F20A38343F57351339D59AE721A98BEDEAC159E690B73198DA0856
2018-12-31 02:47:20.788 UTC [cli/common] readBlock -> INFO 051
Received block: 0
```

9.8. Salida comando incorporación nodos al canal

Nodo peer0.depositario.mistic-uoc.com

```
2018-12-31 02:54:47.799 UTC [viperutil] getKeysRecursively -> DEBU
001 Found map[string]interface{} value for peer.BCCSP
2018-12-31 02:54:47.800 UTC [viperutil] getKeysRecursively -> DEBU
002 Found map[string]interface{} value for peer.BCCSP.SW
2018-12-31 02:54:47.800 UTC [viperutil] unmarshalJSON -> DEBU 003
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 02:54:47.800 UTC [viperutil] getKeysRecursively -> DEBU
004 Found real value for peer.BCCSP.SW.Hash setting to string SHA2
2018-12-31 02:54:47.800 UTC [viperutil] unmarshalJSON -> DEBU 005
Unmarshal JSON: value is not a string: 256
2018-12-31 02:54:47.800 UTC [viperutil] getKeysRecursively -> DEBU
006 Found real value for peer.BCCSP.SW.Security setting to int 256
2018-12-31 02:54:47.801 UTC [viperutil] getKeysRecursively -> DEBU
007 Found map[string]interface{} value for
peer.BCCSP.SW.FileKeyStore
2018-12-31 02:54:47.801 UTC [viperutil] unmarshalJSON -> DEBU 008
Unmarshal JSON: value cannot be unmarshalled: unexpected end of JSON
input
2018-12-31 02:54:47.801 UTC [viperutil] getKeysRecursively -> DEBU
009 Found real value for peer.BCCSP.SW.FileKeyStore.KeyStore setting
to string
2018-12-31 02:54:47.801 UTC [viperutil] getKeysRecursively -> DEBU
00a Found map[string]interface{} value for peer.BCCSP.PKCS11
2018-12-31 02:54:47.802 UTC [viperutil] unmarshalJSON -> DEBU 00b
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:54:47.806 UTC [viperutil] getKeysRecursively -> DEBU
00c Found real value for peer.BCCSP.PKCS11.Security setting to <nil>
<nil>
2018-12-31 02:54:47.806 UTC [viperutil] getKeysRecursively -> DEBU
00d Found map[string]interface{} value for
peer.BCCSP.PKCS11.FileKeyStore
2018-12-31 02:54:47.808 UTC [viperutil] unmarshalJSON -> DEBU 00e
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:54:47.809 UTC [viperutil] getKeysRecursively -> DEBU
00f Found real value for peer.BCCSP.PKCS11.FileKeyStore.KeyStore
setting to <nil> <nil>
2018-12-31 02:54:47.821 UTC [viperutil] unmarshalJSON -> DEBU 010
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:54:47.822 UTC [viperutil] getKeysRecursively -> DEBU
011 Found real value for peer.BCCSP.PKCS11.Library setting to <nil>
<nil>
2018-12-31 02:54:47.822 UTC [viperutil] unmarshalJSON -> DEBU 012
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:54:47.823 UTC [viperutil] getKeysRecursively -> DEBU
013 Found real value for peer.BCCSP.PKCS11.Label setting to <nil>
<nil>
2018-12-31 02:54:47.824 UTC [viperutil] unmarshalJSON -> DEBU 014
Unmarshal JSON: value is not a string: <nil>
```

```
2018-12-31 02:54:47.825 UTC [viperutil] getKeysRecursively -> DEBU
015 Found real value for peer.BCCSP.PKCS11.Pin setting to <nil>
<nil>
2018-12-31 02:54:47.828 UTC [viperutil] unmarshalJSON -> DEBU 016
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:54:47.829 UTC [viperutil] getKeysRecursively -> DEBU
017 Found real value for peer.BCCSP.PKCS11.Hash setting to <nil>
<nil>
2018-12-31 02:54:47.833 UTC [viperutil] unmarshalJSON -> DEBU 018
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 02:54:47.833 UTC [viperutil] getKeysRecursively -> DEBU
019 Found real value for peer.BCCSP.Default setting to string SW
2018-12-31 02:54:47.833 UTC [viperutil] EnhancedExactUnmarshalKey ->
DEBU 01a map[peer.BCCSP:map[SW:map[Security:256
FileKeyStore:map[KeyStore:] Hash:SHA2] PKCS11:map[Security:<nil>
FileKeyStore:map[KeyStore:<nil>] Library:<nil> Label:<nil> Pin:<nil>
Hash:<nil>] Default:SW]]
2018-12-31 02:54:47.842 UTC [bccsp_sw] openKeyStore -> DEBU 01b
KeyStore opened at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/keystore]...done
2018-12-31 02:54:47.843 UTC [bccsp] initBCCSP -> DEBU 01c Initialize
BCCSP [SW]
2018-12-31 02:54:47.843 UTC [msp] getPemMaterialFromDir -> DEBU 01d
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/signcerts
2018-12-31 02:54:47.844 UTC [msp] getPemMaterialFromDir -> DEBU 01e
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/signcerts/Admin@depositario.mistic-uoc.com-cert.pem
2018-12-31 02:54:47.846 UTC [msp] getPemMaterialFromDir -> DEBU 01f
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/cacerts
2018-12-31 02:54:47.847 UTC [msp] getPemMaterialFromDir -> DEBU 020
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
2018-12-31 02:54:47.848 UTC [msp] getPemMaterialFromDir -> DEBU 021
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/admincerts
```

```
2018-12-31 02:54:47.850 UTC [msp] getPemMaterialFromDir -> DEBU 022
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/admincerts/Admin@depositario.mistic-uoc.com-cert.pem
2018-12-31 02:54:47.853 UTC [msp] getPemMaterialFromDir -> DEBU 023
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/intermediatecerts
2018-12-31 02:54:47.853 UTC [msp] getMspConfig -> DEBU 024
Intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/intermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/intermediatecerts: no such file or directory]
2018-12-31 02:54:47.853 UTC [msp] getPemMaterialFromDir -> DEBU 025
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlscacerts
2018-12-31 02:54:47.855 UTC [msp] getPemMaterialFromDir -> DEBU 026
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
2018-12-31 02:54:47.856 UTC [msp] getPemMaterialFromDir -> DEBU 027
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlsintermediatecerts
2018-12-31 02:54:47.856 UTC [msp] getMspConfig -> DEBU 028 TLS
intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlsintermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlsintermediatecerts: no such file or directory]
2018-12-31 02:54:47.856 UTC [msp] getPemMaterialFromDir -> DEBU 029
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/crls
2018-12-31 02:54:47.856 UTC [msp] getMspConfig -> DEBU 02a crls
folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/crls]. Skipping. [stat
```

```
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/crls: no such file or directory]
2018-12-31 02:54:47.856 UTC [msp] getMspConfig -> DEBU 02b MSP
configuration file not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/config.yaml]: [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/config.yaml: no such file or directory]
2018-12-31 02:54:47.857 UTC [msp] newBccspMsp -> DEBU 02c Creating
BCCSP-based MSP instance
2018-12-31 02:54:47.857 UTC [msp] New -> DEBU 02d Creating Cache-MSP
instance
2018-12-31 02:54:47.857 UTC [msp] loadLocaMSP -> DEBU 02e Created
new local MSP
2018-12-31 02:54:47.857 UTC [msp] Setup -> DEBU 02f Setting up MSP
instance depositarioMSP
2018-12-31 02:54:47.857 UTC [msp/identity] newIdentity -> DEBU 030
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICYzCCAgmgAwIBAgIQXpEJkWIgnatKS2s1J3uZJTAKBggqhkjOPQQDAjCBgjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5taXN0aWMtdW9jLmNvbTEmMCQGA1UE
AxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAw
WhcNMjgxMjI4MDE0OTAwWjCBgjELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNl
bG9uYTESMBAGA1UEBxMJQmFyY2Vsb25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5t
aXN0aWMtdW9jLmNvbTEmMCQGA1UEAxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVv
Yy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAASUPiGnIGeYmW+0y7pGYOKY
J7lUggQugMcMbdZnxniKq3pS4apfiy2LOrF7S2lGX+GiRW95SWX+qK79GtG2YMK0
o18wXTAOBgNVHQ8BAf8EBAMCAaYwDwYDVR0lBAgwBgYEVR0lADAPBgNVHRMBAf8E
BTADAQH/MCkGA1UdDgQiBCAUYbNqxS9Cekbx/PV6qZua367NR06Fgg5kWGaSYE58
OzAKBggqhkjOPQQDAgNIADBFAiEAzPiMqxPtY3czfRHwXBmpMnHDDWNYBe97bVKN
6S1/+jECIGMm84lhL8KsxYnw/ii8QbjoCCrWqrnQRrGU1PY4+Rrk
-----END CERTIFICATE-----
2018-12-31 02:54:47.857 UTC [msp/identity] newIdentity -> DEBU 031
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICPzCCAeWgAwIBAgIQFI/S+qjbp/dGulzZpg7vCzAKBggqhkjOPQQDAjCBgjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5taXN0aWMtdW9jLmNvbTEmMCQGA1UE
AxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAw
WhcNMjgxMjI4MDE0OTAwWjBxMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vs
b25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEpMCcGA1UE
AwwgQWRtaW5AZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIB
BggqhkjOPQMBBwNCAATyWuGH0QW+hqbr4QBiq2HwdUVdh3895sbfuIj0B+K+XGNI
8RFBsyUjpizDAJyTIYtcK8DR9iRl+tBAYRyFvuqNo00wSzAOBgNVHQ8BAf8EBAMC
B4AwDAYDVR0TAQH/BAIwADArBgNVHSMEJDAigCAUYbNqxS9Cekbx/PV6qZua367N
R06Fgg5kWGaSYE58OzAKBggqhkjOPQQDAgNIADBFAiEA/2wUkPalmn91Tewk7lkj
ZvRADOTD5PsJfqsiHoC2tN8CIA/lIO/Hs5bp3yEuHj7chshHEm5/h10bHRYmaY70
GBXw
-----END CERTIFICATE-----
```

```
2018-12-31 02:54:47.928 UTC [msp/identity] newIdentity -> DEBU 032
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICPzCCAeWgAwIBAgIQFI/S+qjbp/dGulzZpg7vCzAKBggqhkjOPQQDAjCBgjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5taXN0aWMtdW9jLmNvbTEmMCQGA1UE
AxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAw
WhcNMjgxMjI4MDE0OTAwWjBxMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vs
b25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEpMCcGA1UE
AwwgQWRtaW5AZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIB
BggqhkjOPQMBBwNCAATyWuGH0QW+hqbr4QBiq2HwdUVdh3895sbfuIj0B+K+XGNI
8RFBsyUjpizDAJyTIYtcK8DR9iRl+tBAYRyFvuqNo00wSzAOBgNVHQ8BAf8EBAMC
B4AwDAYDVR0TAQH/BAIwADArBgNVHSMEJDAigCAUYbNqxS9Cekbx/PV6qZua367N
R06Fgg5kWGaSYE58OzAKBggqhkjOPQQDAgNIADBFAiEA/2wUkPalmn91Tewk7lkj
ZvRADOTD5PsJfqsiHoC2tN8CIA/lIO/Hs5bp3yEuHj7chshHEm5/h10bHRYmaY70
GBXw
-----END CERTIFICATE-----
2018-12-31 02:54:47.930 UTC [bccsp_sw] loadPrivateKey -> DEBU 033
Loading private key
[97b74cfb535ee951720a89d2daf64c3301d0a184363eef313f308e287be45c91]
at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/keystore/97b74cfb535ee951720a89d2daf64c3301d0a184363eef3
13f308e287be45c91_sk]...
2018-12-31 02:54:47.940 UTC [msp/identity] newIdentity -> DEBU 034
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICPzCCAeWgAwIBAgIQFI/S+qjbp/dGulzZpg7vCzAKBggqhkjOPQQDAjCBgjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5taXN0aWMtdW9jLmNvbTEmMCQGA1UE
AxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAw
WhcNMjgxMjI4MDE0OTAwWjBxMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vs
b25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEpMCcGA1UE
AwwgQWRtaW5AZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIB
BggqhkjOPQMBBwNCAATyWuGH0QW+hqbr4QBiq2HwdUVdh3895sbfuIj0B+K+XGNI
8RFBsyUjpizDAJyTIYtcK8DR9iRl+tBAYRyFvuqNo00wSzAOBgNVHQ8BAf8EBAMC
B4AwDAYDVR0TAQH/BAIwADArBgNVHSMEJDAigCAUYbNqxS9Cekbx/PV6qZua367N
R06Fgg5kWGaSYE58OzAKBggqhkjOPQQDAgNIADBFAiEA/2wUkPalmn91Tewk7lkj
ZvRADOTD5PsJfqsiHoC2tN8CIA/lIO/Hs5bp3yEuHj7chshHEm5/h10bHRYmaY70
GBXw
-----END CERTIFICATE-----
2018-12-31 02:54:47.942 UTC [msp] setupSigningIdentity -> DEBU 035
Signing identity expires at 2028-12-28 01:49:00 +0000 UTC
2018-12-31 02:54:47.943 UTC [msp] Validate -> DEBU 036 MSP
depositarioMSP validating identity
2018-12-31 02:54:47.948 UTC [msp] GetDefaultSigningIdentity -> DEBU
037 Obtaining default signing identity
2018-12-31 02:54:47.956 UTC [grpc] DialContext -> DEBU 038 parsed
scheme: ""
2018-12-31 02:54:47.957 UTC [grpc] DialContext -> DEBU 039 scheme ""
not registered, fallback to default scheme
```

```
2018-12-31 02:54:47.958 UTC [grpc] watcher -> DEBU 03a
ccResolverWrapper: sending new addresses to cc:
[{peer0.depositario.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 02:54:47.958 UTC [grpc] switchBalancer -> DEBU 03b
ClientConn switching balancer to "pick_first"
2018-12-31 02:54:47.958 UTC [grpc] HandleSubConnStateChange -> DEBU
03c pickfirstBalancer: HandleSubConnStateChange: 0xc420253b20,
CONNECTING
2018-12-31 02:54:47.986 UTC [grpc] HandleSubConnStateChange -> DEBU
03d pickfirstBalancer: HandleSubConnStateChange: 0xc420253b20, READY
2018-12-31 02:54:47.986 UTC [channelCmd] InitCmdFactory -> INFO 03e
Endorser and orderer connections initialized
2018-12-31 02:54:47.987 UTC [msp/identity] Sign -> DEBU 03f Sign:
plaintext:
0AD8070A5C08011A0C08F789A6E10510...B65E8C14FA951A080A000A000A000A00
2018-12-31 02:54:47.987 UTC [msp/identity] Sign -> DEBU 040 Sign:
digest:
011BB61336C20D0AA3C51F544126FD31377DC8B6AA2BA477263BF0D0512C8342
2018-12-31 02:54:48.440 UTC [channelCmd] executeJoin -> INFO 041
Successfully submitted proposal to join channel
```

Nodo peer0.banco.mistic-uoc.com

```
2018-12-31 02:57:05.071 UTC [viperutil] getKeysRecursively -> DEBU
001 Found map[string]interface{} value for peer.BCCSP
2018-12-31 02:57:05.071 UTC [viperutil] unmarshalJSON -> DEBU 002
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 02:57:05.071 UTC [viperutil] getKeysRecursively -> DEBU
003 Found real value for peer.BCCSP.Default setting to string SW
2018-12-31 02:57:05.071 UTC [viperutil] getKeysRecursively -> DEBU
004 Found map[string]interface{} value for peer.BCCSP.SW
2018-12-31 02:57:05.074 UTC [viperutil] unmarshalJSON -> DEBU 005
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 02:57:05.074 UTC [viperutil] getKeysRecursively -> DEBU
006 Found real value for peer.BCCSP.SW.Hash setting to string SHA2
2018-12-31 02:57:05.075 UTC [viperutil] unmarshalJSON -> DEBU 007
Unmarshal JSON: value is not a string: 256
2018-12-31 02:57:05.075 UTC [viperutil] getKeysRecursively -> DEBU
008 Found real value for peer.BCCSP.SW.Security setting to int 256
2018-12-31 02:57:05.076 UTC [viperutil] getKeysRecursively -> DEBU
009 Found map[string]interface{} value for
peer.BCCSP.SW.FileKeyStore
2018-12-31 02:57:05.076 UTC [viperutil] unmarshalJSON -> DEBU 00a
Unmarshal JSON: value cannot be unmarshalled: unexpected end of JSON
input
2018-12-31 02:57:05.076 UTC [viperutil] getKeysRecursively -> DEBU
00b Found real value for peer.BCCSP.SW.FileKeyStore.KeyStore setting
to string
```

```
2018-12-31 02:57:05.076 UTC [viperutil] getKeysRecursively -> DEBU
00c Found map[string]interface{} value for peer.BCCSP.PKCS11
2018-12-31 02:57:05.077 UTC [viperutil] unmarshalJSON -> DEBU 00d
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:57:05.077 UTC [viperutil] getKeysRecursively -> DEBU
00e Found real value for peer.BCCSP.PKCS11.Pin setting to <nil>
<nil>
2018-12-31 02:57:05.083 UTC [viperutil] unmarshalJSON -> DEBU 00f
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:57:05.085 UTC [viperutil] getKeysRecursively -> DEBU
010 Found real value for peer.BCCSP.PKCS11.Hash setting to <nil>
<nil>
2018-12-31 02:57:05.086 UTC [viperutil] unmarshalJSON -> DEBU 011
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:57:05.087 UTC [viperutil] getKeysRecursively -> DEBU
012 Found real value for peer.BCCSP.PKCS11.Security setting to <nil>
<nil>
2018-12-31 02:57:05.088 UTC [viperutil] getKeysRecursively -> DEBU
013 Found map[string]interface{} value for
peer.BCCSP.PKCS11.FileKeyStore
2018-12-31 02:57:05.088 UTC [viperutil] unmarshalJSON -> DEBU 014
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:57:05.089 UTC [viperutil] getKeysRecursively -> DEBU
015 Found real value for peer.BCCSP.PKCS11.FileKeyStore.KeyStore
setting to <nil> <nil>
2018-12-31 02:57:05.090 UTC [viperutil] unmarshalJSON -> DEBU 016
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:57:05.090 UTC [viperutil] getKeysRecursively -> DEBU
017 Found real value for peer.BCCSP.PKCS11.Library setting to <nil>
<nil>
2018-12-31 02:57:05.090 UTC [viperutil] unmarshalJSON -> DEBU 018
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:57:05.091 UTC [viperutil] getKeysRecursively -> DEBU
019 Found real value for peer.BCCSP.PKCS11.Label setting to <nil>
<nil>
2018-12-31 02:57:05.091 UTC [viperutil] EnhancedExactUnmarshalKey ->
DEBU 01a map[peer.BCCSP:map[Default:SW SW:map[Hash:SHA2 Security:256
FileKeyStore:map[KeyStore:]] PKCS11:map[Pin:<nil> Hash:<nil>
Security:<nil> FileKeyStore:map[KeyStore:<nil>] Library:<nil>
Label:<nil>]]]
2018-12-31 02:57:05.091 UTC [bccsp_sw] openKeyStore -> DEBU 01b
KeyStore opened at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore]...done
2018-12-31 02:57:05.099 UTC [bccsp] initBCCSP -> DEBU 01c Initialize
BCCSP [SW]
2018-12-31 02:57:05.099 UTC [msp] getPemMaterialFromDir -> DEBU 01d
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
```

```
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts
2018-12-31 02:57:05.099 UTC [msp] getPemMaterialFromDir -> DEBU 01e
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 02:57:05.099 UTC [msp] getPemMaterialFromDir -> DEBU 01f
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts
2018-12-31 02:57:05.099 UTC [msp] getPemMaterialFromDir -> DEBU 020
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
2018-12-31 02:57:05.100 UTC [msp] getPemMaterialFromDir -> DEBU 021
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts
2018-12-31 02:57:05.100 UTC [msp] getPemMaterialFromDir -> DEBU 022
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 02:57:05.101 UTC [msp] getPemMaterialFromDir -> DEBU 023
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts
2018-12-31 02:57:05.101 UTC [msp] getMspConfig -> DEBU 024
Intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts: no such file or directory]
2018-12-31 02:57:05.101 UTC [msp] getPemMaterialFromDir -> DEBU 025
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts
2018-12-31 02:57:05.101 UTC [msp] getPemMaterialFromDir -> DEBU 026
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
```

```
2018-12-31 02:57:05.101 UTC [msp] getPemMaterialFromDir -> DEBU 027
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts
2018-12-31 02:57:05.101 UTC [msp] getMspConfig -> DEBU 028 TLS
intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts: no such file or directory]
2018-12-31 02:57:05.101 UTC [msp] getPemMaterialFromDir -> DEBU 029
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls
2018-12-31 02:57:05.101 UTC [msp] getMspConfig -> DEBU 02a crls
folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls: no such file or directory]
2018-12-31 02:57:05.101 UTC [msp] getMspConfig -> DEBU 02b MSP
configuration file not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml]: [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml: no such file or directory]
2018-12-31 02:57:05.107 UTC [msp] newBccspMsp -> DEBU 02c Creating
BCCSP-based MSP instance
2018-12-31 02:57:05.107 UTC [msp] New -> DEBU 02d Creating Cache-MSP
instance
2018-12-31 02:57:05.107 UTC [msp] loadLocaMSP -> DEBU 02e Created
new local MSP
2018-12-31 02:57:05.113 UTC [msp] Setup -> DEBU 02f Setting up MSP
instance bancoMSP
2018-12-31 02:57:05.120 UTC [msp/identity] newIdentity -> DEBU 030
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICSTCCAe+gAwIBAgIQbgg64bHbeBkccIDXmNIWWDAKBggqhkjOPQQDAjB2MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExHTAbBgNVBAoTFGJhbmNvLm1pc3RpYy11b2MuY29tMSAwHgYDVQQDExdjYS5i
YW5jby5taXN0aWMtdW9jLmNvbTAeFw0xODEyMzEwMTQ5MDBaFw0yODEyMjgwMTQ5
MDBaMHYxCzAJBgNVBAYTAkVTMRIwEAYDVQQIEwlCYXJjZWxvbmExEjAQBgNVBAcT
CUJhcmNlbG9uYTEdMBsGA1UEChMUYmFuY28ubWlzdGljLXVvYy5jb20xIDAeBgNV
BAMTF2NhLmJhbmNvLm1pc3RpYy11b2MuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0D
```

```
AQcDQgAExRW9JaiK41T4fy9dxUwGc+zLYcmFN4oAmQL8kDGMVHEonIvJ6sPL1+qE
5iNzvFdEQebaslvxXtzxCiys0PWrc6NfMF0wDgYDVR0PAQH/BAQDAgGmMA8GA1Ud
JQQIMAYGBFUdJQAwDwYDVR0TAQH/BAUwAwEB/zApBgNVHQ4EIgQge1lsp4ogzNlh
y+Je7L5zR1gjGLooTPtQof+uE2TuzI8wCgYIKoZIzj0EAwIDSAAwRQIhAL15n5S/
AG5/Wt6+ZwuLuIRdxCausLypFX8cfrxIPSiuAiAPHY1pQ/788218DnCcw8CcmKoS
RL+l0ZS7OL2DUOkFrA==
-----END CERTIFICATE-----
2018-12-31 02:57:05.121 UTC [msp/identity] newIdentity -> DEBU 031
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 02:57:05.306 UTC [msp/identity] newIdentity -> DEBU 032
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 02:57:05.308 UTC [bccsp_sw] loadPrivateKey -> DEBU 033
Loading private key
[d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9ed81d15b879b71a2dc]
at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore/d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9e
d81d15b879b71a2dc_sk]...
2018-12-31 02:57:05.310 UTC [msp/identity] newIdentity -> DEBU 034
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
```

```
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 02:57:05.312 UTC [msp] setupSigningIdentity -> DEBU 035
Signing identity expires at 2028-12-28 01:49:00 +0000 UTC
2018-12-31 02:57:05.313 UTC [msp] Validate -> DEBU 036 MSP bancoMSP
validating identity
2018-12-31 02:57:05.315 UTC [msp] GetDefaultSigningIdentity -> DEBU
037 Obtaining default signing identity
2018-12-31 02:57:05.317 UTC [grpc] DialContext -> DEBU 038 parsed
scheme: ""
2018-12-31 02:57:05.317 UTC [grpc] DialContext -> DEBU 039 scheme ""
not registered, fallback to default scheme
2018-12-31 02:57:05.317 UTC [grpc] watcher -> DEBU 03a
ccResolverWrapper: sending new addresses to cc:
[{peer0.banco.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 02:57:05.317 UTC [grpc] switchBalancer -> DEBU 03b
ClientConn switching balancer to "pick_first"
2018-12-31 02:57:05.317 UTC [grpc] HandleSubConnStateChange -> DEBU
03c pickfirstBalancer: HandleSubConnStateChange: 0xc42048df90,
CONNECTING
2018-12-31 02:57:05.338 UTC [grpc] HandleSubConnStateChange -> DEBU
03d pickfirstBalancer: HandleSubConnStateChange: 0xc42048df90, READY
2018-12-31 02:57:05.338 UTC [channelCmd] InitCmdFactory -> INFO 03e
Endorser and orderer connections initialized
2018-12-31 02:57:05.340 UTC [msp/identity] Sign -> DEBU 03f Sign:
plaintext:
0AB9070A5C08011A0C08818BA6E10510...B65E8C14FA951A080A000A000A000A00
2018-12-31 02:57:05.341 UTC [msp/identity] Sign -> DEBU 040 Sign:
digest:
3EBA4AFD01F75E1D037985EBE28940D5CAEAE0FCD6838A8F27845CF6ED961AD2
2018-12-31 02:57:05.791 UTC [channelCmd] executeJoin -> INFO 041
Successfully submitted proposal to join channel
```

Nodo peer0.regulador.mistic-uoc.com

```
2018-12-31 02:59:07.466 UTC [viperutil] getKeysRecursively -> DEBU
001 Found map[string]interface{} value for peer.BCCSP
2018-12-31 02:59:07.468 UTC [viperutil] getKeysRecursively -> DEBU
002 Found map[string]interface{} value for peer.BCCSP.PKCS11
2018-12-31 02:59:07.471 UTC [viperutil] unmarshalJSON -> DEBU 003
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:59:07.474 UTC [viperutil] getKeysRecursively -> DEBU
004 Found real value for peer.BCCSP.PKCS11.Label setting to <nil>
<nil>
```

```
2018-12-31 02:59:07.474 UTC [viperutil] unmarshalJSON -> DEBU 005
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:59:07.474 UTC [viperutil] getKeysRecursively -> DEBU
006 Found real value for peer.BCCSP.PKCS11.Pin setting to <nil>
<nil>
2018-12-31 02:59:07.475 UTC [viperutil] unmarshalJSON -> DEBU 007
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:59:07.475 UTC [viperutil] getKeysRecursively -> DEBU
008 Found real value for peer.BCCSP.PKCS11.Hash setting to <nil>
<nil>
2018-12-31 02:59:07.476 UTC [viperutil] unmarshalJSON -> DEBU 009
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:59:07.476 UTC [viperutil] getKeysRecursively -> DEBU
00a Found real value for peer.BCCSP.PKCS11.Security setting to <nil>
<nil>
2018-12-31 02:59:07.477 UTC [viperutil] getKeysRecursively -> DEBU
00b Found map[string]interface{} value for
peer.BCCSP.PKCS11.FileKeyStore
2018-12-31 02:59:07.477 UTC [viperutil] unmarshalJSON -> DEBU 00c
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:59:07.478 UTC [viperutil] getKeysRecursively -> DEBU
00d Found real value for peer.BCCSP.PKCS11.FileKeyStore.KeyStore
setting to <nil> <nil>
2018-12-31 02:59:07.478 UTC [viperutil] unmarshalJSON -> DEBU 00e
Unmarshal JSON: value is not a string: <nil>
2018-12-31 02:59:07.479 UTC [viperutil] getKeysRecursively -> DEBU
00f Found real value for peer.BCCSP.PKCS11.Library setting to <nil>
<nil>
2018-12-31 02:59:07.479 UTC [viperutil] unmarshalJSON -> DEBU 010
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 02:59:07.479 UTC [viperutil] getKeysRecursively -> DEBU
011 Found real value for peer.BCCSP.Default setting to string SW
2018-12-31 02:59:07.481 UTC [viperutil] getKeysRecursively -> DEBU
012 Found map[string]interface{} value for peer.BCCSP.SW
2018-12-31 02:59:07.481 UTC [viperutil] unmarshalJSON -> DEBU 013
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 02:59:07.481 UTC [viperutil] getKeysRecursively -> DEBU
014 Found real value for peer.BCCSP.SW.Hash setting to string SHA2
2018-12-31 02:59:07.482 UTC [viperutil] unmarshalJSON -> DEBU 015
Unmarshal JSON: value is not a string: 256
2018-12-31 02:59:07.482 UTC [viperutil] getKeysRecursively -> DEBU
016 Found real value for peer.BCCSP.SW.Security setting to int 256
2018-12-31 02:59:07.484 UTC [viperutil] getKeysRecursively -> DEBU
017 Found map[string]interface{} value for
peer.BCCSP.SW.FileKeyStore
2018-12-31 02:59:07.484 UTC [viperutil] unmarshalJSON -> DEBU 018
Unmarshal JSON: value cannot be unmarshalled: unexpected end of JSON
input
```

```
2018-12-31 02:59:07.484 UTC [viperutil] getKeysRecursively -> DEBU
019 Found real value for peer.BCCSP.SW.FileKeyStore.KeyStore setting
to string
2018-12-31 02:59:07.484 UTC [viperutil] EnhancedExactUnmarshalKey ->
DEBU 01a map[peer.BCCSP:map[SW:map[Hash:SHA2 Security:256
FileKeyStore:map[KeyStore:]] PKCS11:map[Label:<nil> Pin:<nil>
Hash:<nil> Security:<nil> FileKeyStore:map[KeyStore:<nil>]
Library:<nil>] Default:SW]]
2018-12-31 02:59:07.485 UTC [bccsp_sw] openKeyStore -> DEBU 01b
KeyStore opened at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/keystore]...done
2018-12-31 02:59:07.485 UTC [bccsp] initBCCSP -> DEBU 01c Initialize
BCCSP [SW]
2018-12-31 02:59:07.485 UTC [msp] getPemMaterialFromDir -> DEBU 01d
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/signcerts
2018-12-31 02:59:07.486 UTC [msp] getPemMaterialFromDir -> DEBU 01e
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/signcerts/Admin@regulador.mistic-uoc.com-cert.pem
2018-12-31 02:59:07.494 UTC [msp] getPemMaterialFromDir -> DEBU 01f
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/cacerts
2018-12-31 02:59:07.494 UTC [msp] getPemMaterialFromDir -> DEBU 020
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/cacerts/ca.regulador.mistic-uoc.com-cert.pem
2018-12-31 02:59:07.496 UTC [msp] getPemMaterialFromDir -> DEBU 021
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/admincerts
2018-12-31 02:59:07.496 UTC [msp] getPemMaterialFromDir -> DEBU 022
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/admincerts/Admin@regulador.mistic-uoc.com-cert.pem
2018-12-31 02:59:07.497 UTC [msp] getPemMaterialFromDir -> DEBU 023
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/intermediatecerts
```

```
2018-12-31 02:59:07.497 UTC [msp] getMspConfig -> DEBU 024
Intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/intermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/intermediatecerts: no such file or directory]
2018-12-31 02:59:07.497 UTC [msp] getPemMaterialFromDir -> DEBU 025
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/tlscacerts
2018-12-31 02:59:07.498 UTC [msp] getPemMaterialFromDir -> DEBU 026
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/tlscacerts/tlsca.regulador.mistic-uoc.com-cert.pem
2018-12-31 02:59:07.499 UTC [msp] getPemMaterialFromDir -> DEBU 027
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/tlsintermediatecerts
2018-12-31 02:59:07.499 UTC [msp] getMspConfig -> DEBU 028 TLS
intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/tlsintermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/tlsintermediatecerts: no such file or directory]
2018-12-31 02:59:07.499 UTC [msp] getPemMaterialFromDir -> DEBU 029
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/crls
2018-12-31 02:59:07.499 UTC [msp] getMspConfig -> DEBU 02a crls
folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/crls]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/crls: no such file or directory]
2018-12-31 02:59:07.499 UTC [msp] getMspConfig -> DEBU 02b MSP
configuration file not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/config.yaml]: [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
```

```
zations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/config.yaml: no such file or directory]
2018-12-31 02:59:07.499 UTC [msp] newBccspMsp -> DEBU 02c Creating
BCCSP-based MSP instance
2018-12-31 02:59:07.500 UTC [msp] New -> DEBU 02d Creating Cache-MSP
instance
2018-12-31 02:59:07.500 UTC [msp] loadLocaMSP -> DEBU 02e Created
new local MSP
2018-12-31 02:59:07.501 UTC [msp] Setup -> DEBU 02f Setting up MSP
instance reguladorMSP
2018-12-31 02:59:07.501 UTC [msp/identity] newIdentity -> DEBU 030
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICWTCCAf+gAwIBAgIQA5p0/kdlQKxFYRo2CIQtgDAKBggqhkjOPQQDAjB+MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExITAfBgNVBAoTGHJlZ3VsYWRvci5taXN0aWMtdW9jLmNvbTEkMCIGA1UEAxMb
Y2EucmVndWxhZG9yLm1pc3RpYy11b2MuY29tMB4XDTE4MTIzMTAxNDkwMFoXDTI4
MTIyODAxNDkwMFowfjELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTES
MBAGA1UEBxMJQmFyY2Vsb25hMSEwHwYDVQQKExhyZWd1bGFkb3IubWlzdGljLXVv
Yy5jb20xJDAiBgNVBAMTG2NhLnJlZ3VsYWRvci5taXN0aWMtdW9jLmNvbTBZMBMG
ByqGSM49AgEGCCqGSM49AwEHA0IABKlg7gQyAVCSByPKiZ6Dm6/b0bgZYr6oz3+z
JrRkFumS0Yo59HaqZsZtDD2eKLZUTqP/pIHCFaLT37pJw2s2CHCjXzBdMA4GA1Ud
DwEB/wQEAwIBpjAPBgNVHSUECDAGBgRVHSUAMA8GA1UdEwEB/wQFMAMBAf8wKQYD
VR0OBCIEIEUdKYfu2mSOHgTMythccBvVenj8PL+p+3sijtjIIan+MAoGCCqGSM49
BAMCA0gAMEUCIQC3Kq5fWBAcK2VMjWTVZf597hqD48EU52PWKvWez+iPBAIgSO4h
Uic0OBd+pni44T2g8o8fzsZtyskEnyTNhk5Uckc=
-----END CERTIFICATE-----
2018-12-31 02:59:07.501 UTC [msp/identity] newIdentity -> DEBU 031
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICODCCAd6gAwIBAgIQTHkQg9F248Qzc/wOaywXtzAKBggqhkjOPQQDAjB+MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExITAfBgNVBAoTGHJlZ3VsYWRvci5taXN0aWMtdW9jLmNvbTEkMCIGA1UEAxMb
Y2EucmVndWxhZG9yLm1pc3RpYy11b2MuY29tMB4XDTE4MTIzMTAxNDkwMFoXDTI4
MTIyODAxNDkwMFowbzELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTES
MBAGA1UEBxMJQmFyY2Vsb25hMQ8wDQYDVQQLEwZjbGllbnQxJzAlBgNVBAMMHkFk
bWluQHJlZ3VsYWRvci5taXN0aWMtdW9jLmNvbTBZMBMGByqGSM49AgEGCCqGSM49
AwEHA0IABK4/MspNX9/j1I10K3aEwkrI8vszE/mfSUu01jzcFopq9401OOuIPOPl
7EsvTQZ6EEMIJURegBCZaLyZovFjBEGjTTBLMA4GA1UdDwEB/wQEAwIHgDAMBgNV
HRMBAf8EAjAAMCsGA1UdIwQkMCKAIEUdKYfu2mSOHgTMythccBvVenj8PL+p+3si
jtjIIan+MAoGCCqGSM49BAMCA0gAMEUCIQD5sO8pocd3zn3e/pFqmSLePkCk5kXp
vwvgHn5GrGq+vwIgHhO9wWoSXirIV+1lLa3IbWT0D0xTl5W7+UYxYf4A19A=
-----END CERTIFICATE-----
2018-12-31 02:59:07.568 UTC [msp/identity] newIdentity -> DEBU 032
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICODCCAd6gAwIBAgIQTHkQg9F248Qzc/wOaywXtzAKBggqhkjOPQQDAjB+MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExITAfBgNVBAoTGHJlZ3VsYWRvci5taXN0aWMtdW9jLmNvbTEkMCIGA1UEAxMb
Y2EucmVndWxhZG9yLm1pc3RpYy11b2MuY29tMB4XDTE4MTIzMTAxNDkwMFoXDTI4
MTIyODAxNDkwMFowbzELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTES
MBAGA1UEBxMJQmFyY2Vsb25hMQ8wDQYDVQQLEwZjbGllbnQxJzAlBgNVBAMMHkFk
bWluQHJlZ3VsYWRvci5taXN0aWMtdW9jLmNvbTBZMBMGByqGSM49AgEGCCqGSM49
AwEHA0IABK4/MspNX9/j1I10K3aEwkrI8vszE/mfSUu01jzcFopq9401OOuIPOPl
```

```
7EsvTQZ6EEMIJURegBCZaLyZovFjBEGjTTBLMA4GA1UdDwEB/wQEAwIHgDAMBgNV
HRMBAf8EAjAAMCsGA1UdIwQkMCKAIEUdKYfu2mSOHgTMythccBvVenj8PL+p+3si
jtjIIan+MAoGCCqGSM49BAMCA0gAMEUCIQD5sO8pocd3zn3e/pFqmSLePkCk5kXp
vwvgHn5GrGq+vwIgHhO9wWoSXirIV+1lLa3IbWT0D0xTl5W7+UYxYf4A19A=
-----END CERTIFICATE-----
2018-12-31 02:59:07.569 UTC [bccsp_sw] loadPrivateKey -> DEBU 033
Loading private key
[07f9b6486c443122d3f3ea2f0039bcfa0a6f08adde97a228e9110e99125d223d]
at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/regulador.mistic-uoc.com/users/Admin@regulador.mistic-
uoc.com/msp/keystore/07f9b6486c443122d3f3ea2f0039bcfa0a6f08adde97a22
8e9110e99125d223d_sk]...
2018-12-31 02:59:07.581 UTC [msp/identity] newIdentity -> DEBU 034
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICODCCAd6gAwIBAgIQTHkQg9F248Qzc/wOaywXtzAKBggqhkjOPQQDAjB+MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExITAfBgNVBAoTGHJlZ3VsYWRvci5taXN0aWMtdW9jLmNvbTEkMCIGA1UEAxMb
Y2EucmVndWxhZG9yLm1pc3RpYy11b2MuY29tMB4XDTE4MTIzMTAxNDkwMFoXDTI4
MTIyODAxNDkwMFowbzELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTES
MBAGA1UEBxMJQmFyY2Vsb25hMQ8wDQYDVQQLEwZjbGllbnQxJzAlBgNVBAMMHkFk
bWluQHJlZ3VsYWRvci5taXN0aWMtdW9jLmNvbTBZMBMGByqGSM49AgEGCCqGSM49
AwEHA0IABK4/MspNX9/j1I10K3aEwkrI8vszE/mfSUu01jzcFopq9401OOuIPOPl
7EsvTQZ6EEMIJURegBCZaLyZovFjBEGjTTBLMA4GA1UdDwEB/wQEAwIHgDAMBgNV
HRMBAf8EAjAAMCsGA1UdIwQkMCKAIEUdKYfu2mSOHgTMythccBvVenj8PL+p+3si
jtjIIan+MAoGCCqGSM49BAMCA0gAMEUCIQD5sO8pocd3zn3e/pFqmSLePkCk5kXp
vwvgHn5GrGq+vwIgHhO9wWoSXirIV+1lLa3IbWT0D0xTl5W7+UYxYf4A19A=
-----END CERTIFICATE-----
2018-12-31 02:59:07.582 UTC [msp] setupSigningIdentity -> DEBU 035
Signing identity expires at 2028-12-28 01:49:00 +0000 UTC
2018-12-31 02:59:07.582 UTC [msp] Validate -> DEBU 036 MSP
reguladorMSP validating identity
2018-12-31 02:59:07.584 UTC [msp] GetDefaultSigningIdentity -> DEBU
037 Obtaining default signing identity
2018-12-31 02:59:07.588 UTC [grpc] DialContext -> DEBU 038 parsed
scheme: ""
2018-12-31 02:59:07.588 UTC [grpc] DialContext -> DEBU 039 scheme ""
not registered, fallback to default scheme
2018-12-31 02:59:07.588 UTC [grpc] watcher -> DEBU 03a
ccResolverWrapper: sending new addresses to cc:
[{peer0.regulador.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 02:59:07.588 UTC [grpc] switchBalancer -> DEBU 03b
ClientConn switching balancer to "pick_first"
2018-12-31 02:59:07.593 UTC [grpc] HandleSubConnStateChange -> DEBU
03c pickfirstBalancer: HandleSubConnStateChange: 0xc42036c400,
CONNECTING
2018-12-31 02:59:07.614 UTC [grpc] HandleSubConnStateChange -> DEBU
03d pickfirstBalancer: HandleSubConnStateChange: 0xc42036c400, READY
2018-12-31 02:59:07.615 UTC [channelCmd] InitCmdFactory -> INFO 03e
Endorser and orderer connections initialized
```

```
2018-12-31 02:59:07.618 UTC [msp/identity] Sign -> DEBU 03f Sign:
plaintext:
0ACD070A5C08011A0C08FB8BA6E10510...B65E8C14FA951A080A000A000A000A00
2018-12-31 02:59:07.618 UTC [msp/identity] Sign -> DEBU 040 Sign:
digest:
A10924511155BBA1F7540F87F61F11E564555B861BD8365C449A920AD64F1A32
2018-12-31 02:59:08.233 UTC [channelCmd] executeJoin -> INFO 041
Successfully submitted proposal to join channel
```

9.9. Chaincode crearCuentas

```
package main


import (
        "fmt"
        "strconv"

        "github.com/hyperledger/fabric/core/chaincode/shim"
        pb "github.com/hyperledger/fabric/protos/peer"
)

type SimpleChaincode struct {
}

func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response
{
        fmt.Println("ex02 Init")
        _, args := stub.GetFunctionAndParameters()
        var A, B string    // Entities
        var Aval, Bval int // Asset holdings
        var err error

        if len(args) != 4 {
                return shim.Error("Incorrect number of arguments. Expecting
4")
        }

        // Initialize the chaincode
        A = args[0]
        Aval, err = strconv.Atoi(args[1])
        if err != nil {
                return shim.Error("Expecting integer value for asset
holding")
        }
        B = args[2]
        Bval, err = strconv.Atoi(args[3])
        if err != nil {
                return shim.Error("Expecting integer value for asset
holding")
        }
        fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)
```

```
        // Write the state to the ledger
        err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
        if err != nil {
                return shim.Error(err.Error())
        }

        err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
        if err != nil {
                return shim.Error(err.Error())
        }

        return shim.Success(nil)
}

func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface)
pb.Response {
        fmt.Println("ex02 Invoke")
        function, args := stub.GetFunctionAndParameters()
        if function == "invoke" {
                // Make payment of X units from A to B
                return t.invoke(stub, args)
        } else if function == "delete" {
                // Deletes an entity from its state
                return t.delete(stub, args)
        } else if function == "query" {
                // the old "Query" is now implemtned in invoke
                return t.query(stub, args)
        }

        return shim.Error("Invalid invoke function name. Expecting
\"invoke\" \"delete\" \"query\"")
}

func (t *SimpleChaincode) invoke(stub shim.ChaincodeStubInterface, args
[]string) pb.Response {
        var A, B string    // Entities
        var Aval, Bval int // Asset holdings
        var X int          // Transaction value
        var err error

        if len(args) != 3 {
                return shim.Error("Incorrect number of arguments. Expecting
3")
        }

        A = args[0]
        B = args[1]

        // Get the state from the ledger
        // TODO: will be nice to have a GetAllState call to ledger
        Avalbytes, err := stub.GetState(A)
        if err != nil {
                return shim.Error("Failed to get state")
        }
        if Avalbytes == nil {
                return shim.Error("Entity not found")
        }
```

```
	Aval, _ = strconv.Atoi(string(Avalbytes))

	Bvalbytes, err := stub.GetState(B)
	if err != nil {
		return shim.Error("Failed to get state")
	}
	if Bvalbytes == nil {
		return shim.Error("Entity not found")
	}
	Bval, _ = strconv.Atoi(string(Bvalbytes))

	// Perform the execution
	X, err = strconv.Atoi(args[2])
	if err != nil {
		return shim.Error("Invalid transaction amount, expecting a
integer value")
	}
	Aval = Aval - X
	Bval = Bval + X
	fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)

	// Write the state back to the ledger
	err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
	if err != nil {
		return shim.Error(err.Error())
	}

	err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
	if err != nil {
		return shim.Error(err.Error())
	}

	return shim.Success(nil)
}

// Deletes an entity from state
func (t *SimpleChaincode) delete(stub shim.ChaincodeStubInterface, args
[]string) pb.Response {
	if len(args) != 1 {
		return shim.Error("Incorrect number of arguments. Expecting
1")
	}

	A := args[0]

	// Delete the key from the state in ledger
	err := stub.DelState(A)
	if err != nil {
		return shim.Error("Failed to delete state")
	}

	return shim.Success(nil)
}

// query callback representing the query of a chaincode
func (t *SimpleChaincode) query(stub shim.ChaincodeStubInterface, args
[]string) pb.Response {
```

```
        var A string // Entities
        var err error

        if len(args) != 1 {
                return shim.Error("Incorrect number of arguments. Expecting
name of the person to query")
        }

        A = args[0]

        // Get the state from the ledger
        Avalbytes, err := stub.GetState(A)
        if err != nil {
                jsonResp := "{\"Error\":\"Failed to get state for " + A +
"\"}"
                return shim.Error(jsonResp)
        }

        if Avalbytes == nil {
                jsonResp := "{\"Error\":\"Nil amount for " + A + "\"}"
                return shim.Error(jsonResp)
        }

        jsonResp := "{\"Name\":\"" + A + "\",\"Amount\":\"" +
string(Avalbytes) + "\"}"
        fmt.Printf("Query Response:%s\n", jsonResp)
        return shim.Success(Avalbytes)
}

func main() {
        err := shim.Start(new(SimpleChaincode))
        if err != nil {
                fmt.Printf("Error starting Simple chaincode: %s", err)
        }
}
```

9.10. Instalación del chaincode en el nodo peer0.banco.mistic-uoc.com

```
2018-12-31 03:14:40.084 UTC [viperutil] getKeysRecursively -> DEBU
001 Found map[string]interface{} value for peer.BCCSP
2018-12-31 03:14:40.085 UTC [viperutil] unmarshalJSON -> DEBU 002
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 03:14:40.085 UTC [viperutil] getKeysRecursively -> DEBU
003 Found real value for peer.BCCSP.Default setting to string SW
2018-12-31 03:14:40.085 UTC [viperutil] getKeysRecursively -> DEBU
004 Found map[string]interface{} value for peer.BCCSP.SW
2018-12-31 03:14:40.086 UTC [viperutil] unmarshalJSON -> DEBU 005
Unmarshal JSON: value is not a string: 256
2018-12-31 03:14:40.086 UTC [viperutil] getKeysRecursively -> DEBU
006 Found real value for peer.BCCSP.SW.Security setting to int 256
```

```
2018-12-31 03:14:40.086 UTC [viperutil] getKeysRecursively -> DEBU
007 Found map[string]interface{} value for
peer.BCCSP.SW.FileKeyStore
2018-12-31 03:14:40.086 UTC [viperutil] unmarshalJSON -> DEBU 008
Unmarshal JSON: value cannot be unmarshalled: unexpected end of JSON
input
2018-12-31 03:14:40.086 UTC [viperutil] getKeysRecursively -> DEBU
009 Found real value for peer.BCCSP.SW.FileKeyStore.KeyStore setting
to string
2018-12-31 03:14:40.086 UTC [viperutil] unmarshalJSON -> DEBU 00a
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 03:14:40.086 UTC [viperutil] getKeysRecursively -> DEBU
00b Found real value for peer.BCCSP.SW.Hash setting to string SHA2
2018-12-31 03:14:40.087 UTC [viperutil] getKeysRecursively -> DEBU
00c Found map[string]interface{} value for peer.BCCSP.PKCS11
2018-12-31 03:14:40.087 UTC [viperutil] unmarshalJSON -> DEBU 00d
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:14:40.087 UTC [viperutil] getKeysRecursively -> DEBU
00e Found real value for peer.BCCSP.PKCS11.Pin setting to <nil>
<nil>
2018-12-31 03:14:40.091 UTC [viperutil] unmarshalJSON -> DEBU 00f
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:14:40.095 UTC [viperutil] getKeysRecursively -> DEBU
010 Found real value for peer.BCCSP.PKCS11.Hash setting to <nil>
<nil>
2018-12-31 03:14:40.097 UTC [viperutil] unmarshalJSON -> DEBU 011
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:14:40.098 UTC [viperutil] getKeysRecursively -> DEBU
012 Found real value for peer.BCCSP.PKCS11.Security setting to <nil>
<nil>
2018-12-31 03:14:40.098 UTC [viperutil] getKeysRecursively -> DEBU
013 Found map[string]interface{} value for
peer.BCCSP.PKCS11.FileKeyStore
2018-12-31 03:14:40.101 UTC [viperutil] unmarshalJSON -> DEBU 014
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:14:40.102 UTC [viperutil] getKeysRecursively -> DEBU
015 Found real value for peer.BCCSP.PKCS11.FileKeyStore.KeyStore
setting to <nil> <nil>
2018-12-31 03:14:40.102 UTC [viperutil] unmarshalJSON -> DEBU 016
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:14:40.102 UTC [viperutil] getKeysRecursively -> DEBU
017 Found real value for peer.BCCSP.PKCS11.Library setting to <nil>
<nil>
2018-12-31 03:14:40.103 UTC [viperutil] unmarshalJSON -> DEBU 018
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:14:40.103 UTC [viperutil] getKeysRecursively -> DEBU
019 Found real value for peer.BCCSP.PKCS11.Label setting to <nil>
<nil>
2018-12-31 03:14:40.103 UTC [viperutil] EnhancedExactUnmarshalKey ->
DEBU 01a map[peer.BCCSP:map[PKCS11:map[Label:<nil> Pin:<nil>
```

```
Hash:<nil> Security:<nil> FileKeyStore:map[KeyStore:<nil>]
Library:<nil>] Default:SW SW:map[Hash:SHA2 Security:256
FileKeyStore:map[KeyStore:]]]]
2018-12-31 03:14:40.105 UTC [bccsp_sw] openKeyStore -> DEBU 01b
KeyStore opened at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/keystore]...done
2018-12-31 03:14:40.106 UTC [bccsp] initBCCSP -> DEBU 01c Initialize
BCCSP [SW]
2018-12-31 03:14:40.106 UTC [msp] getPemMaterialFromDir -> DEBU 01d
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/signcerts
2018-12-31 03:14:40.107 UTC [msp] getPemMaterialFromDir -> DEBU 01e
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/signcerts/Admin@depositario.mistic-uoc.com-cert.pem
2018-12-31 03:14:40.107 UTC [msp] getPemMaterialFromDir -> DEBU 01f
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/cacerts
2018-12-31 03:14:40.107 UTC [msp] getPemMaterialFromDir -> DEBU 020
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/cacerts/ca.depositario.mistic-uoc.com-cert.pem
2018-12-31 03:14:40.107 UTC [msp] getPemMaterialFromDir -> DEBU 021
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/admincerts
2018-12-31 03:14:40.107 UTC [msp] getPemMaterialFromDir -> DEBU 022
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/admincerts/Admin@depositario.mistic-uoc.com-cert.pem
2018-12-31 03:14:40.107 UTC [msp] getPemMaterialFromDir -> DEBU 023
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/intermediatecerts
2018-12-31 03:14:40.107 UTC [msp] getMspConfig -> DEBU 024
Intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/intermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
```

```
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/intermediatecerts: no such file or directory]
2018-12-31 03:14:40.108 UTC [msp] getPemMaterialFromDir -> DEBU 025
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlscacerts
2018-12-31 03:14:40.108 UTC [msp] getPemMaterialFromDir -> DEBU 026
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlscacerts/tlsca.depositario.mistic-uoc.com-cert.pem
2018-12-31 03:14:40.108 UTC [msp] getPemMaterialFromDir -> DEBU 027
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlsintermediatecerts
2018-12-31 03:14:40.108 UTC [msp] getMspConfig -> DEBU 028 TLS
intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlsintermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/tlsintermediatecerts: no such file or directory]
2018-12-31 03:14:40.108 UTC [msp] getPemMaterialFromDir -> DEBU 029
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/crls
2018-12-31 03:14:40.108 UTC [msp] getMspConfig -> DEBU 02a crls
folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/crls]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/crls: no such file or directory]
2018-12-31 03:14:40.108 UTC [msp] getMspConfig -> DEBU 02b MSP
configuration file not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/config.yaml]: [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/config.yaml: no such file or directory]
2018-12-31 03:14:40.109 UTC [msp] newBccspMsp -> DEBU 02c Creating
BCCSP-based MSP instance
2018-12-31 03:14:40.109 UTC [msp] New -> DEBU 02d Creating Cache-MSP
instance
```

```
2018-12-31 03:14:40.109 UTC [msp] loadLocaMSP -> DEBU 02e Created
new local MSP
2018-12-31 03:14:40.109 UTC [msp] Setup -> DEBU 02f Setting up MSP
instance depositarioMSP
2018-12-31 03:14:40.111 UTC [msp/identity] newIdentity -> DEBU 030
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICYzCCAgmgAwIBAgIQXpEJkWIgnatKS2s1J3uZJTAKBggqhkjOPQQDAjCBgjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5taXN0aWMtdW9jLmNvbTEmMCQGA1UE
AxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAw
WhcNMjgxMjI4MDE0OTAwWjCBgjELMAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNl
bG9uYTESMBAGA1UEBxMJQmFyY2Vsb25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5t
aXN0aWMtdW9jLmNvbTEmMCQGA1UEAxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVv
Yy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAASUPiGnIGeYmW+0y7pGYOKY
J7lUggQugMcMbdZnxniKq3pS4apfiy2LOrF7S2lGX+GiRW95SWX+qK79GtG2YMK0
o18wXTAOBgNVHQ8BAf8EBAMCAaYwDwYDVR0lBAgwBgYEVR0lADAPBgNVHRMBAf8E
BTADAQH/MCkGA1UdDgQiBCAUYbNqxS9Cekbx/PV6qZua367NR06Fgg5kWGaSYE58
OzAKBggqhkjOPQQDAgNIADBFAiEAzPiMqxPtY3czfRHwXBmpMnHDDWNYBe97bVKN
6S1/+jECIGMm84lhL8KsxYnw/ii8QbjoCCrWqrnQRrGU1PY4+Rrk
-----END CERTIFICATE-----
2018-12-31 03:14:40.113 UTC [msp/identity] newIdentity -> DEBU 031
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICPzCCAeWgAwIBAgIQFI/S+qjbp/dGulzZpg7vCzAKBggqhkjOPQQDAjCBgjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5taXN0aWMtdW9jLmNvbTEmMCQGA1UE
AxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAw
WhcNMjgxMjI4MDE0OTAwWjBxMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vs
b25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEpMCcGA1UE
AwwgQWRtaW5AZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIB
BggqhkjOPQMBBwNCAATyWuGH0QW+hqbr4QBiq2HwdUVdh3895sbfuIj0B+K+XGNI
8RFBsyUjpizDAJyTIYtcK8DR9iRl+tBAYRyFvuqNo00wSzAOBgNVHQ8BAf8EBAMC
B4AwDAYDVR0TAQH/BAIwADArBgNVHSMEJDAigCAUYbNqxS9Cekbx/PV6qZua367N
R06Fgg5kWGaSYE58OzAKBggqhkjOPQQDAgNIADBFAiEA/2wUkPalmn91Tewk7lkj
ZvRADOTD5PsJfqsiHoC2tN8CIA/lIO/Hs5bp3yEuHj7chshHEm5/h10bHRYmaY70
GBXw
-----END CERTIFICATE-----
2018-12-31 03:14:40.230 UTC [msp/identity] newIdentity -> DEBU 032
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICPzCCAeWgAwIBAgIQFI/S+qjbp/dGulzZpg7vCzAKBggqhkjOPQQDAjCBgjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5taXN0aWMtdW9jLmNvbTEmMCQGA1UE
AxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAw
WhcNMjgxMjI4MDE0OTAwWjBxMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vs
b25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEpMCcGA1UE
AwwgQWRtaW5AZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIB
BggqhkjOPQMBBwNCAATyWuGH0QW+hqbr4QBiq2HwdUVdh3895sbfuIj0B+K+XGNI
8RFBsyUjpizDAJyTIYtcK8DR9iRl+tBAYRyFvuqNo00wSzAOBgNVHQ8BAf8EBAMC
B4AwDAYDVR0TAQH/BAIwADArBgNVHSMEJDAigCAUYbNqxS9Cekbx/PV6qZua367N
R06Fgg5kWGaSYE58OzAKBggqhkjOPQQDAgNIADBFAiEA/2wUkPalmn91Tewk7lkj
ZvRADOTD5PsJfqsiHoC2tN8CIA/lIO/Hs5bp3yEuHj7chshHEm5/h10bHRYmaY70
GBXw
```

```
-----END CERTIFICATE-----
2018-12-31 03:14:40.234 UTC [bccsp_sw] loadPrivateKey -> DEBU 033
Loading private key
[97b74cfb535ee951720a89d2daf64c3301d0a184363eef313f308e287be45c91]
at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/depositario.mistic-uoc.com/users/Admin@depositario.mistic-
uoc.com/msp/keystore/97b74cfb535ee951720a89d2daf64c3301d0a184363eef3
13f308e287be45c91_sk]...
2018-12-31 03:14:40.237 UTC [msp/identity] newIdentity -> DEBU 034
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICPzCCAeWgAwIBAgIQFI/S+qjbp/dGulzZpg7vCzAKBggqhkjOPQQDAjCBgjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMSMwIQYDVQQKExpkZXBvc2l0YXJpby5taXN0aWMtdW9jLmNvbTEmMCQGA1UE
AxMdY2EuZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAw
WhcNMjgxMjI4MDE0OTAwWjBxMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vs
b25hMRIwEAYDVQQHEwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEpMCcGA1UE
AwwgQWRtaW5AZGVwb3NpdGFyaW8ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIB
BggqhkjOPQMBBwNCAATyWuGH0QW+hqbr4QBiq2HwdUVdh3895sbfuIj0B+K+XGNI
8RFBsyUjpizDAJyTIYtcK8DR9iRl+tBAYRyFvuqNo00wSzAOBgNVHQ8BAf8EBAMC
B4AwDAYDVR0TAQH/BAIwADArBgNVHSMEJDAigCAUYbNqxS9Cekbx/PV6qZua367N
R06Fgg5kWGaSYE58OzAKBggqhkjOPQQDAgNIADBFAiEA/2wUkPalmn91Tewk7lkj
ZvRADOTD5PsJfqsiHoC2tN8CIA/lIO/Hs5bp3yEuHj7chshHEm5/h10bHRYmaY70
GBXw
-----END CERTIFICATE-----
2018-12-31 03:14:40.238 UTC [msp] setupSigningIdentity -> DEBU 035
Signing identity expires at 2028-12-28 01:49:00 +0000 UTC
2018-12-31 03:14:40.239 UTC [msp] Validate -> DEBU 036 MSP
depositarioMSP validating identity
2018-12-31 03:14:40.250 UTC [grpc] DialContext -> DEBU 037 parsed
scheme: ""
2018-12-31 03:14:40.250 UTC [grpc] DialContext -> DEBU 038 scheme ""
not registered, fallback to default scheme
2018-12-31 03:14:40.255 UTC [grpc] watcher -> DEBU 039
ccResolverWrapper: sending new addresses to cc:
[{peer0.depositario.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 03:14:40.255 UTC [grpc] switchBalancer -> DEBU 03a
ClientConn switching balancer to "pick_first"
2018-12-31 03:14:40.255 UTC [grpc] HandleSubConnStateChange -> DEBU
03b pickfirstBalancer: HandleSubConnStateChange: 0xc420434db0,
CONNECTING
2018-12-31 03:14:40.277 UTC [grpc] HandleSubConnStateChange -> DEBU
03c pickfirstBalancer: HandleSubConnStateChange: 0xc420434db0, READY
2018-12-31 03:14:40.282 UTC [grpc] DialContext -> DEBU 03d parsed
scheme: ""
2018-12-31 03:14:40.282 UTC [grpc] DialContext -> DEBU 03e scheme ""
not registered, fallback to default scheme
2018-12-31 03:14:40.282 UTC [grpc] watcher -> DEBU 03f
ccResolverWrapper: sending new addresses to cc:
[{peer0.depositario.mistic-uoc.com:7051 0  <nil>}]
```

```
2018-12-31 03:14:40.282 UTC [grpc] switchBalancer -> DEBU 040
ClientConn switching balancer to "pick_first"
2018-12-31 03:14:40.282 UTC [grpc] HandleSubConnStateChange -> DEBU
041 pickfirstBalancer: HandleSubConnStateChange: 0xc4205dede0,
CONNECTING
2018-12-31 03:14:40.302 UTC [grpc] HandleSubConnStateChange -> DEBU
042 pickfirstBalancer: HandleSubConnStateChange: 0xc4205dede0, READY
2018-12-31 03:14:40.309 UTC [msp] GetDefaultSigningIdentity -> DEBU
043 Obtaining default signing identity
2018-12-31 03:14:40.310 UTC [chaincodeCmd] checkChaincodeCmdParams -
> INFO 044 Using default escc
2018-12-31 03:14:40.310 UTC [chaincodeCmd] checkChaincodeCmdParams -
> INFO 045 Using default vscc
2018-12-31 03:14:41.240 UTC [golang-platform] getCodeFromFS -> DEBU
046 getCodeFromFS
github.com/hyperledger/fabric/examples/chaincode/go/chaincode_crearC
uentas
2018-12-31 03:14:42.824 UTC [golang-platform] func1 -> DEBU 047
Discarding GOROOT package fmt
2018-12-31 03:14:42.824 UTC [golang-platform] func1 -> DEBU 048
Discarding provided package
github.com/hyperledger/fabric/core/chaincode/shim
2018-12-31 03:14:42.824 UTC [golang-platform] func1 -> DEBU 049
Discarding provided package
github.com/hyperledger/fabric/protos/peer
2018-12-31 03:14:42.824 UTC [golang-platform] func1 -> DEBU 04a
Discarding GOROOT package strconv
2018-12-31 03:14:42.824 UTC [golang-platform] GetDeploymentPayload -
> DEBU 04b done
2018-12-31 03:14:42.825 UTC [container] WriteFileToPackage -> DEBU
04c Writing file to tarball:
src/github.com/hyperledger/fabric/examples/chaincode/go/chaincode_cr
earCuentas/chaincode_crearCuentas.go
2018-12-31 03:14:42.831 UTC [msp/identity] Sign -> DEBU 04d Sign:
plaintext:
0AD8070A5C08031A0C08A293A6E10510...1F8CFF040000FFFFE8A0C26F001C0000
2018-12-31 03:14:42.831 UTC [msp/identity] Sign -> DEBU 04e Sign:
digest:
0C157450055329A877F1023EA0327CBE672A518EA59B1A69A246E9A6C3D513E1
2018-12-31 03:14:42.842 UTC [chaincodeCmd] install -> INFO 04f
Installed remotely response:<status:200 payload:"OK" >
```

9.11. Instanciación del programa

```
2018-12-31 03:18:50.293 UTC [viperutil] getKeysRecursively -> DEBU
001 Found map[string]interface{} value for peer.BCCSP
2018-12-31 03:18:50.296 UTC [viperutil] getKeysRecursively -> DEBU
002 Found map[string]interface{} value for peer.BCCSP.PKCS11
```

```
2018-12-31 03:18:50.298 UTC [viperutil] unmarshalJSON -> DEBU 003
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:18:50.299 UTC [viperutil] getKeysRecursively -> DEBU
004 Found real value for peer.BCCSP.PKCS11.Label setting to <nil>
<nil>
2018-12-31 03:18:50.302 UTC [viperutil] unmarshalJSON -> DEBU 005
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:18:50.303 UTC [viperutil] getKeysRecursively -> DEBU
006 Found real value for peer.BCCSP.PKCS11.Pin setting to <nil>
<nil>
2018-12-31 03:18:50.306 UTC [viperutil] unmarshalJSON -> DEBU 007
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:18:50.310 UTC [viperutil] getKeysRecursively -> DEBU
008 Found real value for peer.BCCSP.PKCS11.Hash setting to <nil>
<nil>
2018-12-31 03:18:50.311 UTC [viperutil] unmarshalJSON -> DEBU 009
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:18:50.313 UTC [viperutil] getKeysRecursively -> DEBU
00a Found real value for peer.BCCSP.PKCS11.Security setting to <nil>
<nil>
2018-12-31 03:18:50.314 UTC [viperutil] getKeysRecursively -> DEBU
00b Found map[string]interface{} value for
peer.BCCSP.PKCS11.FileKeyStore
2018-12-31 03:18:50.314 UTC [viperutil] unmarshalJSON -> DEBU 00c
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:18:50.315 UTC [viperutil] getKeysRecursively -> DEBU
00d Found real value for peer.BCCSP.PKCS11.FileKeyStore.KeyStore
setting to <nil> <nil>
2018-12-31 03:18:50.315 UTC [viperutil] unmarshalJSON -> DEBU 00e
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:18:50.315 UTC [viperutil] getKeysRecursively -> DEBU
00f Found real value for peer.BCCSP.PKCS11.Library setting to <nil>
<nil>
2018-12-31 03:18:50.316 UTC [viperutil] unmarshalJSON -> DEBU 010
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 03:18:50.316 UTC [viperutil] getKeysRecursively -> DEBU
011 Found real value for peer.BCCSP.Default setting to string SW
2018-12-31 03:18:50.316 UTC [viperutil] getKeysRecursively -> DEBU
012 Found map[string]interface{} value for peer.BCCSP.SW
2018-12-31 03:18:50.316 UTC [viperutil] unmarshalJSON -> DEBU 013
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 03:18:50.316 UTC [viperutil] getKeysRecursively -> DEBU
014 Found real value for peer.BCCSP.SW.Hash setting to string SHA2
2018-12-31 03:18:50.317 UTC [viperutil] unmarshalJSON -> DEBU 015
Unmarshal JSON: value is not a string: 256
2018-12-31 03:18:50.317 UTC [viperutil] getKeysRecursively -> DEBU
016 Found real value for peer.BCCSP.SW.Security setting to int 256
```

```
2018-12-31 03:18:50.326 UTC [viperutil] getKeysRecursively -> DEBU
017 Found map[string]interface{} value for
peer.BCCSP.SW.FileKeyStore
2018-12-31 03:18:50.326 UTC [viperutil] unmarshalJSON -> DEBU 018
Unmarshal JSON: value cannot be unmarshalled: unexpected end of JSON
input
2018-12-31 03:18:50.326 UTC [viperutil] getKeysRecursively -> DEBU
019 Found real value for peer.BCCSP.SW.FileKeyStore.KeyStore setting
to string
2018-12-31 03:18:50.326 UTC [viperutil] EnhancedExactUnmarshalKey ->
DEBU 01a map[peer.BCCSP:map[PKCS11:map[Security:<nil>
FileKeyStore:map[KeyStore:<nil>] Library:<nil> Label:<nil> Pin:<nil>
Hash:<nil>] Default:SW SW:map[FileKeyStore:map[KeyStore:] Hash:SHA2
Security:256]]]
2018-12-31 03:18:50.326 UTC [bccsp_sw] openKeyStore -> DEBU 01b
KeyStore opened at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore]...done
2018-12-31 03:18:50.326 UTC [bccsp] initBCCSP -> DEBU 01c Initialize
BCCSP [SW]
2018-12-31 03:18:50.326 UTC [msp] getPemMaterialFromDir -> DEBU 01d
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts
2018-12-31 03:18:50.326 UTC [msp] getPemMaterialFromDir -> DEBU 01e
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 03:18:50.326 UTC [msp] getPemMaterialFromDir -> DEBU 01f
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts
2018-12-31 03:18:50.327 UTC [msp] getPemMaterialFromDir -> DEBU 020
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
2018-12-31 03:18:50.327 UTC [msp] getPemMaterialFromDir -> DEBU 021
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts
2018-12-31 03:18:50.327 UTC [msp] getPemMaterialFromDir -> DEBU 022
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
```

```
2018-12-31 03:18:50.327 UTC [msp] getPemMaterialFromDir -> DEBU 023
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts
2018-12-31 03:18:50.327 UTC [msp] getMspConfig -> DEBU 024
Intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts: no such file or directory]
2018-12-31 03:18:50.327 UTC [msp] getPemMaterialFromDir -> DEBU 025
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts
2018-12-31 03:18:50.327 UTC [msp] getPemMaterialFromDir -> DEBU 026
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
2018-12-31 03:18:50.327 UTC [msp] getPemMaterialFromDir -> DEBU 027
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts
2018-12-31 03:18:50.327 UTC [msp] getMspConfig -> DEBU 028 TLS
intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts: no such file or directory]
2018-12-31 03:18:50.327 UTC [msp] getPemMaterialFromDir -> DEBU 029
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls
2018-12-31 03:18:50.327 UTC [msp] getMspConfig -> DEBU 02a crls
folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls: no such file or directory]
2018-12-31 03:18:50.327 UTC [msp] getMspConfig -> DEBU 02b MSP
configuration file not found at
```

```
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml]: [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml: no such file or directory]
2018-12-31 03:18:50.327 UTC [msp] newBccspMsp -> DEBU 02c Creating
BCCSP-based MSP instance
2018-12-31 03:18:50.327 UTC [msp] New -> DEBU 02d Creating Cache-MSP
instance
2018-12-31 03:18:50.327 UTC [msp] loadLocaMSP -> DEBU 02e Created
new local MSP
2018-12-31 03:18:50.327 UTC [msp] Setup -> DEBU 02f Setting up MSP
instance bancoMSP
2018-12-31 03:18:50.328 UTC [msp/identity] newIdentity -> DEBU 030
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICSTCCAe+gAwIBAgIQbgg64bHbeBkccIDXmNIWWDAKBggqhkjOPQQDAjB2MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExHTAbBgNVBAoTFGJhbmNvLm1pc3RpYy11b2MuY29tMSAwHgYDVQQDExdjYS5i
YW5jby5taXN0aWMtdW9jLmNvbTAeFw0xODEyMzEwMTQ5MDBaFw0yODEyMjgwMTQ5
MDBaMHYxCzAJBgNVBAYTAkVTMRIwEAYDVQQIEwlCYXJjZWxvbmExEjAQBgNVBAcT
CUJhcmNlbG9uYTEdMBsGA1UEChMUYmFuY28ubWlzdGljLXVvYy5jb20xIDAeBgNV
BAMTF2NhLmJhbmNvLm1pc3RpYy11b2MuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0D
AQcDQgAExRW9JaiK41T4fy9dxUwGc+zLYcmFN4oAmQL8kDGMVHEonIvJ6sPL1+qE
5iNzvFdEQebaslvxXtzxCiys0PWrc6NfMF0wDgYDVR0PAQH/BAQDAgGmMA8GA1Ud
JQQIMAYGBFUdJQAwDwYDVR0TAQH/BAUwAwEB/zApBgNVHQ4EIgQge1lsp4ogzNlh
y+Je7L5zR1gjGLooTPtQof+uE2TuzI8wCgYIKoZIzj0EAwIDSAAwRQIhAL15n5S/
AG5/Wt6+ZwuLuIRdxCausLypFX8cfrxIPSiuAiAPHY1pQ/788218DnCcw8CcmKoS
RL+l0ZS7OL2DUOkFrA==
-----END CERTIFICATE-----
2018-12-31 03:18:50.328 UTC [msp/identity] newIdentity -> DEBU 031
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 03:18:50.455 UTC [msp/identity] newIdentity -> DEBU 032
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
```

```
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 03:18:50.455 UTC [bccsp_sw] loadPrivateKey -> DEBU 033
Loading private key
[d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9ed81d15b879b71a2dc]
at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore/d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9e
d81d15b879b71a2dc_sk]...
2018-12-31 03:18:50.456 UTC [msp/identity] newIdentity -> DEBU 034
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 03:18:50.461 UTC [msp] setupSigningIdentity -> DEBU 035
Signing identity expires at 2028-12-28 01:49:00 +0000 UTC
2018-12-31 03:18:50.466 UTC [msp] Validate -> DEBU 036 MSP bancoMSP
validating identity
2018-12-31 03:18:50.481 UTC [grpc] DialContext -> DEBU 037 parsed
scheme: ""
2018-12-31 03:18:50.481 UTC [grpc] DialContext -> DEBU 038 scheme ""
not registered, fallback to default scheme
2018-12-31 03:18:50.481 UTC [grpc] watcher -> DEBU 039
ccResolverWrapper: sending new addresses to cc:
[{peer0.banco.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 03:18:50.481 UTC [grpc] switchBalancer -> DEBU 03a
ClientConn switching balancer to "pick_first"
2018-12-31 03:18:50.481 UTC [grpc] HandleSubConnStateChange -> DEBU
03b pickfirstBalancer: HandleSubConnStateChange: 0xc4200eeb40,
CONNECTING
2018-12-31 03:18:50.496 UTC [grpc] HandleSubConnStateChange -> DEBU
03c pickfirstBalancer: HandleSubConnStateChange: 0xc4200eeb40, READY
```

```
2018-12-31 03:18:50.498 UTC [grpc] DialContext -> DEBU 03d parsed
scheme: ""
2018-12-31 03:18:50.498 UTC [grpc] DialContext -> DEBU 03e scheme ""
not registered, fallback to default scheme
2018-12-31 03:18:50.498 UTC [grpc] watcher -> DEBU 03f
ccResolverWrapper: sending new addresses to cc:
[{peer0.banco.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 03:18:50.498 UTC [grpc] switchBalancer -> DEBU 040
ClientConn switching balancer to "pick_first"
2018-12-31 03:18:50.498 UTC [grpc] HandleSubConnStateChange -> DEBU
041 pickfirstBalancer: HandleSubConnStateChange: 0xc420470a10,
CONNECTING
2018-12-31 03:18:50.508 UTC [grpc] HandleSubConnStateChange -> DEBU
042 pickfirstBalancer: HandleSubConnStateChange: 0xc420470a10, READY
2018-12-31 03:18:50.511 UTC [msp] GetDefaultSigningIdentity -> DEBU
043 Obtaining default signing identity
2018-12-31 03:18:50.513 UTC [grpc] DialContext -> DEBU 044 parsed
scheme: ""
2018-12-31 03:18:50.514 UTC [grpc] DialContext -> DEBU 045 scheme ""
not registered, fallback to default scheme
2018-12-31 03:18:50.516 UTC [grpc] watcher -> DEBU 046
ccResolverWrapper: sending new addresses to cc: [{orderer.mistic-
uoc.com:7050 0  <nil>}]
2018-12-31 03:18:50.517 UTC [grpc] switchBalancer -> DEBU 047
ClientConn switching balancer to "pick_first"
2018-12-31 03:18:50.517 UTC [grpc] HandleSubConnStateChange -> DEBU
048 pickfirstBalancer: HandleSubConnStateChange: 0xc4204a8fe0,
CONNECTING
2018-12-31 03:18:50.532 UTC [grpc] HandleSubConnStateChange -> DEBU
049 pickfirstBalancer: HandleSubConnStateChange: 0xc4204a8fe0, READY
2018-12-31 03:18:50.533 UTC [chaincodeCmd] checkChaincodeCmdParams -
> INFO 04a Using default escc
2018-12-31 03:18:50.533 UTC [chaincodeCmd] checkChaincodeCmdParams -
> INFO 04b Using default vscc
2018-12-31 03:18:50.542 UTC [msp/identity] Sign -> DEBU 04c Sign:
plaintext:
0AC7070A6A08031A0C089A95A6E10510...30300A000A04657363630A0476736363
2018-12-31 03:18:50.543 UTC [msp/identity] Sign -> DEBU 04d Sign:
digest:
B5A58D6182DFE7C9ECF02F47ECD7326C072DC525A49EA3949CBA3953D9EEA963
2018-12-31 03:18:55.978 UTC [msp/identity] Sign -> DEBU 04e Sign:
plaintext:
0AC7070A6A08031A0C089A95A6E10510...10EF8F7EFA05F22C690BDC5B48A8333A
2018-12-31 03:18:55.978 UTC [msp/identity] Sign -> DEBU 04f Sign:
digest:
BCC40C5C688B81102127BF9E8EDF7FDB17409930429DEEA4BB2BD35FB913E94F
==================== Chaincode Instantiation on PEER on channel
'bancochannel' is successful ====================
```

9.12. Salida ejecución query

```
2018-12-31 03:25:25.205 UTC [viperutil] getKeysRecursively -> DEBU
001 Found map[string]interface{} value for peer.BCCSP
2018-12-31 03:25:25.205 UTC [viperutil] unmarshalJSON -> DEBU 002
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 03:25:25.205 UTC [viperutil] getKeysRecursively -> DEBU
003 Found real value for peer.BCCSP.Default setting to string SW
2018-12-31 03:25:25.206 UTC [viperutil] getKeysRecursively -> DEBU
004 Found map[string]interface{} value for peer.BCCSP.SW
2018-12-31 03:25:25.206 UTC [viperutil] unmarshalJSON -> DEBU 005
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 03:25:25.206 UTC [viperutil] getKeysRecursively -> DEBU
006 Found real value for peer.BCCSP.SW.Hash setting to string SHA2
2018-12-31 03:25:25.207 UTC [viperutil] unmarshalJSON -> DEBU 007
Unmarshal JSON: value is not a string: 256
2018-12-31 03:25:25.207 UTC [viperutil] getKeysRecursively -> DEBU
008 Found real value for peer.BCCSP.SW.Security setting to int 256
2018-12-31 03:25:25.207 UTC [viperutil] getKeysRecursively -> DEBU
009 Found map[string]interface{} value for
peer.BCCSP.SW.FileKeyStore
2018-12-31 03:25:25.207 UTC [viperutil] unmarshalJSON -> DEBU 00a
Unmarshal JSON: value cannot be unmarshalled: unexpected end of JSON
input
2018-12-31 03:25:25.207 UTC [viperutil] getKeysRecursively -> DEBU
00b Found real value for peer.BCCSP.SW.FileKeyStore.KeyStore setting
to string
2018-12-31 03:25:25.207 UTC [viperutil] getKeysRecursively -> DEBU
00c Found map[string]interface{} value for peer.BCCSP.PKCS11
2018-12-31 03:25:25.208 UTC [viperutil] unmarshalJSON -> DEBU 00d
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:25:25.208 UTC [viperutil] getKeysRecursively -> DEBU
00e Found real value for peer.BCCSP.PKCS11.Label setting to <nil>
<nil>
2018-12-31 03:25:25.208 UTC [viperutil] unmarshalJSON -> DEBU 00f
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:25:25.209 UTC [viperutil] getKeysRecursively -> DEBU
010 Found real value for peer.BCCSP.PKCS11.Pin setting to <nil>
<nil>
2018-12-31 03:25:25.213 UTC [viperutil] unmarshalJSON -> DEBU 011
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:25:25.215 UTC [viperutil] getKeysRecursively -> DEBU
012 Found real value for peer.BCCSP.PKCS11.Hash setting to <nil>
<nil>
2018-12-31 03:25:25.216 UTC [viperutil] unmarshalJSON -> DEBU 013
Unmarshal JSON: value is not a string: <nil>
```

```
2018-12-31 03:25:25.217 UTC [viperutil] getKeysRecursively -> DEBU
014 Found real value for peer.BCCSP.PKCS11.Security setting to <nil>
<nil>
2018-12-31 03:25:25.217 UTC [viperutil] getKeysRecursively -> DEBU
015 Found map[string]interface{} value for
peer.BCCSP.PKCS11.FileKeyStore
2018-12-31 03:25:25.218 UTC [viperutil] unmarshalJSON -> DEBU 016
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:25:25.218 UTC [viperutil] getKeysRecursively -> DEBU
017 Found real value for peer.BCCSP.PKCS11.FileKeyStore.KeyStore
setting to <nil> <nil>
2018-12-31 03:25:25.219 UTC [viperutil] unmarshalJSON -> DEBU 018
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:25:25.220 UTC [viperutil] getKeysRecursively -> DEBU
019 Found real value for peer.BCCSP.PKCS11.Library setting to <nil>
<nil>
2018-12-31 03:25:25.220 UTC [viperutil] EnhancedExactUnmarshalKey ->
DEBU 01a map[peer.BCCSP:map[Default:SW SW:map[Hash:SHA2 Security:256
FileKeyStore:map[KeyStore:]] PKCS11:map[Pin:<nil> Hash:<nil>
Security:<nil> FileKeyStore:map[KeyStore:<nil>] Library:<nil>
Label:<nil>]]]
2018-12-31 03:25:25.222 UTC [bccsp_sw] openKeyStore -> DEBU 01b
KeyStore opened at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore]...done
2018-12-31 03:25:25.223 UTC [bccsp] initBCCSP -> DEBU 01c Initialize
BCCSP [SW]
2018-12-31 03:25:25.223 UTC [msp] getPemMaterialFromDir -> DEBU 01d
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts
2018-12-31 03:25:25.223 UTC [msp] getPemMaterialFromDir -> DEBU 01e
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 03:25:25.223 UTC [msp] getPemMaterialFromDir -> DEBU 01f
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts
2018-12-31 03:25:25.224 UTC [msp] getPemMaterialFromDir -> DEBU 020
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
2018-12-31 03:25:25.226 UTC [msp] getPemMaterialFromDir -> DEBU 021
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
```

```
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts
2018-12-31 03:25:25.227 UTC [msp] getPemMaterialFromDir -> DEBU 022
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 03:25:25.227 UTC [msp] getPemMaterialFromDir -> DEBU 023
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts
2018-12-31 03:25:25.228 UTC [msp] getMspConfig -> DEBU 024
Intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts: no such file or directory]
2018-12-31 03:25:25.228 UTC [msp] getPemMaterialFromDir -> DEBU 025
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts
2018-12-31 03:25:25.229 UTC [msp] getPemMaterialFromDir -> DEBU 026
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
2018-12-31 03:25:25.229 UTC [msp] getPemMaterialFromDir -> DEBU 027
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts
2018-12-31 03:25:25.229 UTC [msp] getMspConfig -> DEBU 028 TLS
intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts: no such file or directory]
2018-12-31 03:25:25.229 UTC [msp] getPemMaterialFromDir -> DEBU 029
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls
2018-12-31 03:25:25.229 UTC [msp] getMspConfig -> DEBU 02a crls
folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
```

```
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls: no such file or directory]
2018-12-31 03:25:25.229 UTC [msp] getMspConfig -> DEBU 02b MSP
configuration file not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml]: [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml: no such file or directory]
2018-12-31 03:25:25.229 UTC [msp] newBccspMsp -> DEBU 02c Creating
BCCSP-based MSP instance
2018-12-31 03:25:25.230 UTC [msp] New -> DEBU 02d Creating Cache-MSP
instance
2018-12-31 03:25:25.230 UTC [msp] loadLocaMSP -> DEBU 02e Created
new local MSP
2018-12-31 03:25:25.230 UTC [msp] Setup -> DEBU 02f Setting up MSP
instance bancoMSP
2018-12-31 03:25:25.234 UTC [msp/identity] newIdentity -> DEBU 030
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICSTCCAe+gAwIBAgIQbgg64bHbeBkccIDXmNIWWDAKBggqhkjOPQQDAjB2MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExHTAbBgNVBAoTFGJhbmNvLm1pc3RpYy11b2MuY29tMSAwHgYDVQQDExdjYS5i
YW5jby5taXN0aWMtdW9jLmNvbTAeFw0xODEyMzEwMTQ5MDBaFw0yODEyMjgwMTQ5
MDBaMHYxCzAJBgNVBAYTAkVTMRIwEAYDVQQIEwlCYXJjZWxvbmExEjAQBgNVBAcT
CUJhcmNlbG9uYTEdMBsGA1UEChMUYmFuY28ubWlzdGljLXVvYy5jb20xIDAeBgNV
BAMTF2NhLmJhbmNvLm1pc3RpYy11b2MuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0D
AQcDQgAExRW9JaiK41T4fy9dxUwGc+zLYcmFN4oAmQL8kDGMVHEonIvJ6sPL1+qE
5iNzvFdEQebaslvxXtzxCiys0PWrc6NfMF0wDgYDVR0PAQH/BAQDAgGmMA8GA1Ud
JQQIMAYGBFUdJQAwDwYDVR0TAQH/BAUwAwEB/zApBgNVHQ4EIgQge1lsp4ogzNlh
y+Je7L5zR1gjGLooTPtQof+uE2TuzI8wCgYIKoZIzj0EAwIDSAAwRQIhAL15n5S/
AG5/Wt6+ZwuLuIRdxCausLypFX8cfrxIPSiuAiAPHY1pQ/788218DnCcw8CcmKoS
RL+l0ZS7OL2DUOkFrA==
-----END CERTIFICATE-----
2018-12-31 03:25:25.234 UTC [msp/identity] newIdentity -> DEBU 031
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
```

```
-----END CERTIFICATE-----
2018-12-31 03:25:25.359 UTC [msp/identity] newIdentity -> DEBU 032
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 03:25:25.360 UTC [bccsp_sw] loadPrivateKey -> DEBU 033
Loading private key
[d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9ed81d15b879b71a2dc]
at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore/d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9e
d81d15b879b71a2dc_sk]...
2018-12-31 03:25:25.360 UTC [msp/identity] newIdentity -> DEBU 034
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 03:25:25.361 UTC [msp] setupSigningIdentity -> DEBU 035
Signing identity expires at 2028-12-28 01:49:00 +0000 UTC
2018-12-31 03:25:25.361 UTC [msp] Validate -> DEBU 036 MSP bancoMSP
validating identity
2018-12-31 03:25:25.367 UTC [grpc] DialContext -> DEBU 037 parsed
scheme: ""
2018-12-31 03:25:25.368 UTC [grpc] DialContext -> DEBU 038 scheme ""
not registered, fallback to default scheme
2018-12-31 03:25:25.368 UTC [grpc] watcher -> DEBU 039
ccResolverWrapper: sending new addresses to cc:
[{peer0.banco.mistic-uoc.com:7051 0  <nil>}]
```

```
2018-12-31 03:25:25.368 UTC [grpc] switchBalancer -> DEBU 03a
ClientConn switching balancer to "pick_first"
2018-12-31 03:25:25.369 UTC [grpc] HandleSubConnStateChange -> DEBU
03b pickfirstBalancer: HandleSubConnStateChange: 0xc420323590,
CONNECTING
2018-12-31 03:25:25.413 UTC [grpc] HandleSubConnStateChange -> DEBU
03c pickfirstBalancer: HandleSubConnStateChange: 0xc420323590, READY
2018-12-31 03:25:25.425 UTC [grpc] DialContext -> DEBU 03d parsed
scheme: ""
2018-12-31 03:25:25.425 UTC [grpc] DialContext -> DEBU 03e scheme ""
not registered, fallback to default scheme
2018-12-31 03:25:25.425 UTC [grpc] watcher -> DEBU 03f
ccResolverWrapper: sending new addresses to cc:
[{peer0.banco.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 03:25:25.425 UTC [grpc] switchBalancer -> DEBU 040
ClientConn switching balancer to "pick_first"
2018-12-31 03:25:25.425 UTC [grpc] HandleSubConnStateChange -> DEBU
041 pickfirstBalancer: HandleSubConnStateChange: 0xc42034cf60,
CONNECTING
2018-12-31 03:25:25.462 UTC [grpc] HandleSubConnStateChange -> DEBU
042 pickfirstBalancer: HandleSubConnStateChange: 0xc42034cf60, READY
2018-12-31 03:25:25.472 UTC [msp] GetDefaultSigningIdentity -> DEBU
043 Obtaining default signing identity
2018-12-31 03:25:25.473 UTC [msp/identity] Sign -> DEBU 044 Sign:
plaintext:
0ACF070A7208031A0C08A598A6E10510...71756572790A09496E766572736F7231
2018-12-31 03:25:25.473 UTC [msp/identity] Sign -> DEBU 045 Sign:
digest:
128A9A337BC386F615DC3A78C8DA52E3B119BCD3E0C21F26A5DCE978B9407155
100
===================== Chaincode Instantiation on PEER on channel
'bancochannel' is successful =====================
```

9.13. Salida ejecución invoke

```
2018-12-31 03:27:27.625 UTC [viperutil] getKeysRecursively -> DEBU
001 Found map[string]interface{} value for peer.BCCSP
2018-12-31 03:27:27.626 UTC [viperutil] unmarshalJSON -> DEBU 002
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 03:27:27.626 UTC [viperutil] getKeysRecursively -> DEBU
003 Found real value for peer.BCCSP.Default setting to string SW
2018-12-31 03:27:27.626 UTC [viperutil] getKeysRecursively -> DEBU
004 Found map[string]interface{} value for peer.BCCSP.SW
2018-12-31 03:27:27.626 UTC [viperutil] unmarshalJSON -> DEBU 005
Unmarshal JSON: value cannot be unmarshalled: invalid character 'S'
looking for beginning of value
2018-12-31 03:27:27.626 UTC [viperutil] getKeysRecursively -> DEBU
006 Found real value for peer.BCCSP.SW.Hash setting to string SHA2
```

```
2018-12-31 03:27:27.626 UTC [viperutil] unmarshalJSON -> DEBU 007
Unmarshal JSON: value is not a string: 256
2018-12-31 03:27:27.626 UTC [viperutil] getKeysRecursively -> DEBU
008 Found real value for peer.BCCSP.SW.Security setting to int 256
2018-12-31 03:27:27.626 UTC [viperutil] getKeysRecursively -> DEBU
009 Found map[string]interface{} value for
peer.BCCSP.SW.FileKeyStore
2018-12-31 03:27:27.626 UTC [viperutil] unmarshalJSON -> DEBU 00a
Unmarshal JSON: value cannot be unmarshalled: unexpected end of JSON
input
2018-12-31 03:27:27.626 UTC [viperutil] getKeysRecursively -> DEBU
00b Found real value for peer.BCCSP.SW.FileKeyStore.KeyStore setting
to string
2018-12-31 03:27:27.626 UTC [viperutil] getKeysRecursively -> DEBU
00c Found map[string]interface{} value for peer.BCCSP.PKCS11
2018-12-31 03:27:27.627 UTC [viperutil] unmarshalJSON -> DEBU 00d
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:27:27.627 UTC [viperutil] getKeysRecursively -> DEBU
00e Found real value for peer.BCCSP.PKCS11.Label setting to <nil>
<nil>
2018-12-31 03:27:27.627 UTC [viperutil] unmarshalJSON -> DEBU 00f
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:27:27.629 UTC [viperutil] getKeysRecursively -> DEBU
010 Found real value for peer.BCCSP.PKCS11.Pin setting to <nil>
<nil>
2018-12-31 03:27:27.630 UTC [viperutil] unmarshalJSON -> DEBU 011
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:27:27.631 UTC [viperutil] getKeysRecursively -> DEBU
012 Found real value for peer.BCCSP.PKCS11.Hash setting to <nil>
<nil>
2018-12-31 03:27:27.631 UTC [viperutil] unmarshalJSON -> DEBU 013
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:27:27.631 UTC [viperutil] getKeysRecursively -> DEBU
014 Found real value for peer.BCCSP.PKCS11.Security setting to <nil>
<nil>
2018-12-31 03:27:27.632 UTC [viperutil] getKeysRecursively -> DEBU
015 Found map[string]interface{} value for
peer.BCCSP.PKCS11.FileKeyStore
2018-12-31 03:27:27.632 UTC [viperutil] unmarshalJSON -> DEBU 016
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:27:27.633 UTC [viperutil] getKeysRecursively -> DEBU
017 Found real value for peer.BCCSP.PKCS11.FileKeyStore.KeyStore
setting to <nil> <nil>
2018-12-31 03:27:27.633 UTC [viperutil] unmarshalJSON -> DEBU 018
Unmarshal JSON: value is not a string: <nil>
2018-12-31 03:27:27.633 UTC [viperutil] getKeysRecursively -> DEBU
019 Found real value for peer.BCCSP.PKCS11.Library setting to <nil>
<nil>
2018-12-31 03:27:27.633 UTC [viperutil] EnhancedExactUnmarshalKey ->
DEBU 01a
map[peer.BCCSP:map[PKCS11:map[FileKeyStore:map[KeyStore:<nil>]
```

```
Library:<nil> Label:<nil> Pin:<nil> Hash:<nil> Security:<nil>]
Default:SW SW:map[Hash:SHA2 Security:256
FileKeyStore:map[KeyStore:]]]]
2018-12-31 03:27:27.633 UTC [bccsp_sw] openKeyStore -> DEBU 01b
KeyStore opened at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore]...done
2018-12-31 03:27:27.633 UTC [bccsp] initBCCSP -> DEBU 01c Initialize
BCCSP [SW]
2018-12-31 03:27:27.633 UTC [msp] getPemMaterialFromDir -> DEBU 01d
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts
2018-12-31 03:27:27.633 UTC [msp] getPemMaterialFromDir -> DEBU 01e
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/signcerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 03:27:27.633 UTC [msp] getPemMaterialFromDir -> DEBU 01f
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts
2018-12-31 03:27:27.633 UTC [msp] getPemMaterialFromDir -> DEBU 020
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/cacerts/ca.banco.mistic-uoc.com-cert.pem
2018-12-31 03:27:27.633 UTC [msp] getPemMaterialFromDir -> DEBU 021
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts
2018-12-31 03:27:27.634 UTC [msp] getPemMaterialFromDir -> DEBU 022
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/admincerts/Admin@banco.mistic-uoc.com-cert.pem
2018-12-31 03:27:27.634 UTC [msp] getPemMaterialFromDir -> DEBU 023
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts
2018-12-31 03:27:27.634 UTC [msp] getMspConfig -> DEBU 024
Intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
```

```
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/intermediatecerts: no such file or directory]
2018-12-31 03:27:27.634 UTC [msp] getPemMaterialFromDir -> DEBU 025
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts
2018-12-31 03:27:27.634 UTC [msp] getPemMaterialFromDir -> DEBU 026
Inspecting file
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlscacerts/tlsca.banco.mistic-uoc.com-cert.pem
2018-12-31 03:27:27.634 UTC [msp] getPemMaterialFromDir -> DEBU 027
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts
2018-12-31 03:27:27.634 UTC [msp] getMspConfig -> DEBU 028 TLS
intermediate certs folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/tlsintermediatecerts: no such file or directory]
2018-12-31 03:27:27.634 UTC [msp] getPemMaterialFromDir -> DEBU 029
Reading directory
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls
2018-12-31 03:27:27.634 UTC [msp] getMspConfig -> DEBU 02a crls
folder not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls]. Skipping. [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/crls: no such file or directory]
2018-12-31 03:27:27.634 UTC [msp] getMspConfig -> DEBU 02b MSP
configuration file not found at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml]: [stat
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgani
zations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/config.yaml: no such file or directory]
2018-12-31 03:27:27.634 UTC [msp] newBccspMsp -> DEBU 02c Creating
BCCSP-based MSP instance
2018-12-31 03:27:27.634 UTC [msp] New -> DEBU 02d Creating Cache-MSP
instance
```

```
2018-12-31 03:27:27.634 UTC [msp] loadLocaMSP -> DEBU 02e Created
new local MSP
2018-12-31 03:27:27.635 UTC [msp] Setup -> DEBU 02f Setting up MSP
instance bancoMSP
2018-12-31 03:27:27.636 UTC [msp/identity] newIdentity -> DEBU 030
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICSTCCAe+gAwIBAgIQbgg64bHbeBkccIDXmNIWWDAKBggqhkjOPQQDAjB2MQsw
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv
bmExHTAbBgNVBAoTFGJhbmNvLm1pc3RpYy11b2MuY29tMSAwHgYDVQQDExdjYS5i
YW5jby5taXN0aWMtdW9jLmNvbTAeFw0xODEyMzEwMTQ5MDBaFw0yODEyMjgwMTQ5
MDBaMHYxCzAJBgNVBAYTAkVTMRIwEAYDVQQIEwlCYXJjZWxvbmExEjAQBgNVBAcT
CUJhcmNlbG9uYTEdMBsGA1UEChMUYmFuY28ubWlzdGljLXVvYy5jb20xIDAeBgNV
BAMTF2NhLmJhbmNvLm1pc3RpYy11b2MuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0D
AQcDQgAExRW9JaiK41T4fy9dxUwGc+zLYcmFN4oAmQL8kDGMVHEonIvJ6sPL1+qE
5iNzvFdEQebaslvxXtzxCiys0PWrc6NfMF0wDgYDVR0PAQH/BAQDAgGmMA8GA1Ud
JQQIMAYGBFUdJQAwDwYDVR0TAQH/BAUwAwEB/zApBgNVHQ4EIgQge1lsp4ogzNlh
y+Je7L5zR1gjGLooTPtQof+uE2TuzI8wCgYIKoZIzj0EAwIDSAAwRQIhAL15n5S/
AG5/Wt6+ZwuLuIRdxCausLypFX8cfrxIPSiuAiAPHY1pQ/788218DnCcw8CcmKoS
RL+l0ZS7OL2DUOkFrA==
-----END CERTIFICATE-----
2018-12-31 03:27:27.637 UTC [msp/identity] newIdentity -> DEBU 031
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 03:27:27.673 UTC [msp/identity] newIdentity -> DEBU 032
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
```

```
2018-12-31 03:27:27.673 UTC [bccsp_sw] loadPrivateKey -> DEBU 033
Loading private key
[d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9ed81d15b879b71a2dc]
at
[/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/banco.mistic-uoc.com/users/Admin@banco.mistic-
uoc.com/msp/keystore/d49f7484ae6186c0f989c422d11c0c844927919e8fc0c9e
d81d15b879b71a2dc_sk]...
2018-12-31 03:27:27.673 UTC [msp/identity] newIdentity -> DEBU 034
Creating identity instance for cert -----BEGIN CERTIFICATE-----
MIICLDCCAdOgAwIBAgIRAOi4XWXkQ2RI5EaY0Ih7uoswCgYIKoZIzj0EAwIwdjEL
MAkGA1UEBhMCRVMxEjAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vs
b25hMR0wGwYDVQQKExRiYW5jby5taXN0aWMtdW9jLmNvbTEgMB4GA1UEAxMXY2Eu
YmFuY28ubWlzdGljLXVvYy5jb20wHhcNMTgxMjMxMDE0OTAwWhcNMjgxMjI4MDE0
OTAwWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQH
EwlCYXJjZWxvbmExDzANBgNVBAsTBmNsaWVudDEjMCEGA1UEAwwaQWRtaW5AYmFu
Y28ubWlzdGljLXVvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAS0shE8
I0d8p0tLrDuLwK4lugiDtgT+R1NgnfEaOvXquazDJkZZyQmKtUB7WgdSlZG70O9c
+elD1Gd6scF1drbLo00wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADAr
BgNVHSMEJDAigCB7WWyniiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggq
hkjOPQQDAgNHADBEAiACMoNLYGu4MhYUSuKyWYAB49xFpCr/bswu93OylQwBBgIg
Ez0e+JTYD765OJ+MQ4nZDRbeRafLgXpyqYD2BCkLAr8=
-----END CERTIFICATE-----
2018-12-31 03:27:27.674 UTC [msp] setupSigningIdentity -> DEBU 035
Signing identity expires at 2028-12-28 01:49:00 +0000 UTC
2018-12-31 03:27:27.674 UTC [msp] Validate -> DEBU 036 MSP bancoMSP
validating identity
2018-12-31 03:27:27.675 UTC [grpc] DialContext -> DEBU 037 parsed
scheme: ""
2018-12-31 03:27:27.675 UTC [grpc] DialContext -> DEBU 038 scheme ""
not registered, fallback to default scheme
2018-12-31 03:27:27.675 UTC [grpc] watcher -> DEBU 039
ccResolverWrapper: sending new addresses to cc:
[{peer0.banco.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 03:27:27.675 UTC [grpc] switchBalancer -> DEBU 03a
ClientConn switching balancer to "pick_first"
2018-12-31 03:27:27.675 UTC [grpc] HandleSubConnStateChange -> DEBU
03b pickfirstBalancer: HandleSubConnStateChange: 0xc420482a50,
CONNECTING
2018-12-31 03:27:27.685 UTC [grpc] HandleSubConnStateChange -> DEBU
03c pickfirstBalancer: HandleSubConnStateChange: 0xc420482a50, READY
2018-12-31 03:27:27.687 UTC [grpc] DialContext -> DEBU 03d parsed
scheme: ""
2018-12-31 03:27:27.687 UTC [grpc] DialContext -> DEBU 03e scheme ""
not registered, fallback to default scheme
2018-12-31 03:27:27.687 UTC [grpc] watcher -> DEBU 03f
ccResolverWrapper: sending new addresses to cc:
[{peer0.banco.mistic-uoc.com:7051 0  <nil>}]
2018-12-31 03:27:27.687 UTC [grpc] switchBalancer -> DEBU 040
ClientConn switching balancer to "pick_first"
```

```
2018-12-31 03:27:27.687 UTC [grpc] HandleSubConnStateChange -> DEBU
041 pickfirstBalancer: HandleSubConnStateChange: 0xc420274a90,
CONNECTING
2018-12-31 03:27:27.695 UTC [grpc] HandleSubConnStateChange -> DEBU
042 pickfirstBalancer: HandleSubConnStateChange: 0xc420274a90, READY
2018-12-31 03:27:27.696 UTC [msp] GetDefaultSigningIdentity -> DEBU
043 Obtaining default signing identity
2018-12-31 03:27:27.697 UTC [grpc] DialContext -> DEBU 044 parsed
scheme: ""
2018-12-31 03:27:27.697 UTC [grpc] DialContext -> DEBU 045 scheme ""
not registered, fallback to default scheme
2018-12-31 03:27:27.697 UTC [grpc] watcher -> DEBU 046
ccResolverWrapper: sending new addresses to cc: [{orderer.mistic-
uoc.com:7050 0  <nil>}]
2018-12-31 03:27:27.697 UTC [grpc] switchBalancer -> DEBU 047
ClientConn switching balancer to "pick_first"
2018-12-31 03:27:27.697 UTC [grpc] HandleSubConnStateChange -> DEBU
048 pickfirstBalancer: HandleSubConnStateChange: 0xc420337070,
CONNECTING
2018-12-31 03:27:27.706 UTC [grpc] HandleSubConnStateChange -> DEBU
049 pickfirstBalancer: HandleSubConnStateChange: 0xc420337070, READY
2018-12-31 03:27:27.709 UTC [msp/identity] Sign -> DEBU 04a Sign:
plaintext:
0ACF070A7208031A0C089F99A6E10510...310A09496E766572736F72320A023130
2018-12-31 03:27:27.709 UTC [msp/identity] Sign -> DEBU 04b Sign:
digest:
21A7781AE00F71B86356A0E12D8F9124A2FAACA38CCA568557F75AC562C1475A
2018-12-31 03:27:27.750 UTC [msp/identity] Sign -> DEBU 04c Sign:
plaintext:
0ACF070A7208031A0C089F99A6E10510...54B4AC798AC887B63DA0F171A47E2EA4
2018-12-31 03:27:27.750 UTC [msp/identity] Sign -> DEBU 04d Sign:
digest:
C606DFB4566E4A2A0FF7F9731BEE91D201369E5A4C4F4DADC6EB26CA4DC21EC7
2018-12-31 03:27:27.775 UTC [chaincodeCmd] chaincodeInvokeOrQuery ->
DEBU 04e ESCC invoke result: version:1 response:<status:200 >
payload:"\n
\247\267\345\257V8\310\223\\\357A\002\336]\341\323\373=`\327\025\354
o)2E\021\232\304<\200\340\022\221\001\nu\022U\n\014crearCuentas\022E
\n\017\n\tInversor1\022\002\010\001\n\017\n\tInversor2\022\002\010\0
01\032\017\n\tInversor1\032\00290\032\020\n\tInversor2\032\003210\02
2\034\n\004lscc\022\024\n\022\n\014crearCuentas\022\002\010\001\032\
003\010\310\001\"\023\022\014crearCuentas\032\0031.0"
endorsement:<endorser:"\n\010bancoMSP\022\252\006-----BEGIN
CERTIFICATE-----
\nMIICKjCCAdCgAwIBAgIQeA93enZwgWsfdHbWq5K0GjAKBggqhkjOPQQDAjB2MQsw\n
CQYDVQQGEwJFUzESMBAGA1UECBMJQmFyY2Vsb25hMRIwEAYDVQQHEwlCYXJjZWxv\nbm
ExHTAbBgNVBAoTFGJhbmNvLm1pc3RpYy11b2MuY29tMSAwHgYDVQQDExdjYS5i\nYW5j
by5taXN0aWMtdW9jLmNvbTAeFw0xODEyMzEwMTQ5MDBaFw0yODEyMjgwMTQ5\nMDBaMG
kxCzAJBgNVBAYTAkVTMRIwEAYDVQQIEwlCYXJjZWxvbmExEjAQBgNVBAcT\nCUJhcmNl
bG9uYTENMAsGA1UECxMEcGVlcjEjMCEGA1UEAxMacGVlcjAuYmFuY28u\nbWlzdGljLX
VvYy5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAStxDQwTwxT\nuKwdoht+T4QE
```

```
LyTv9ePgZ2dDNMS4qxhLg8Z2eNbNtPDqUBQbJ1CNWpSuc7+UCAZT\n7dGE0WNsJeJ/o0
0wSzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0TAQH/BAIwADArBgNV\nHSMEJDAigCB7WWyn
iiDM2WHL4l7svnNHWCMYuihM+1Ch/64TZO7MjzAKBggqhkjO\nPQQDAgNIADBFAiEA9D
v2pgepTN3P9B4LdO9DA6WzIPN1JRe9HSQUiti4C8YCIDX1\n6tuIsQlPHWe6vZQsCcvW
vaNvtp0wHKAUMM6YeML7\n-----END CERTIFICATE-----\n"
signature:"0E\002!\000\343aX\rwr\326\373:\337\3139\310\210\271E\344\
030\036\267\371;\322\223<\024\323@\002\343\217\355\002
1\272\024oH\316z\326+\017\273s\n\316\233~T\264\254y\212\310\207\266=
\240\361q\244~.\244" >
2018-12-31 03:27:27.776 UTC [chaincodeCmd] chaincodeInvokeOrQuery ->
INFO 04f Chaincode invoke successful. result: status:200
===================== Chaincode Instantiation on PEER on channel
'bancochannel' is successful =====================
```