(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 689 112**

(51) Int. CI.:

G06F 17/30 (2006.01)
H04L 29/08 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **15.03.2011 E 11305278 (1)**

(97) Fecha y número de publicación de la concesión europea: **25.07.2018 EP 2500832**

(54) Título: **Método y sistema para mecanismo de sincronización en sistema de reservas de múltiples servidores**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
08.11.2018

(73) Titular/es:

AMADEUS S.A.S. (100.0%)
485 Route du Pin Montard, Sophia Antipolis
06410 Biot , FR

(72) Inventor/es:

MASINI, VINCENT;
BURDESE, SAMUEL;
PAVOT, MARC;
DANIEL, JEROME y
FAUSER, DIETMAR

(74) Agente/Representante:

SUGRAÑES MOLINÉ, Pedro

DESCRIPCIÓN

Método y sistema para mecanismo de sincronización en sistema de reservas de múltiples servidores

Campo de la invención

La presente invención se refiere al campo de sistemas de reservas de viajes, más particularmente a un método y sistema para manejar reservas de viajes en múltiples servidores usando un mecanismo de sincronización de contexto distribuido.

Antecedentes de la invención

Las empresas de viajes modernas (por ejemplo, aerolíneas) normalmente emplean aplicaciones sofisticadas para manejar peticiones de reservas por clientes. Es más, y más frecuente el caso en el que se usa más de una arquitectura por todo el sistema de la empresa. En tales cases deberían tenerse en cuenta cuestiones de compatibilidad y sincronización cuando se diseña y planifica el sistema de reservas. Un ejemplo es cuando parte de la gestión de reservas se realiza en aplicaciones basadas en Internet o infraestructuras de comunicación. Otro ejemplo es cuando un sistema (no necesariamente un sistema de reservas) debe migrarse desde un sistema de ordenador central heredado (por ejemplo, TPF) a un sistema nuevo (por ejemplo, un sistema abierto). Nos referimos a este último ejemplo como un Desmantelamiento, es decir cuando una aplicación debe migrarse desde por ejemplo un ordenador central de TPF a por ejemplo un sistema abierto. Para evitar la interrupción del servicio en el sistema de reservas, se aconseja realizar esta migración de forma progresiva, en lugar de cerrar el sistema existente y cambiar al nuevo en un único movimiento, con todos los posibles problemas que podrían surgir: además de la complejidad de un gran procedimiento de activación en el momento del cambio entre el sistema viejo y nuevo, también debería considerarse la necesidad de doble mantenimiento de software en ambas plataformas, mientras el sistema nuevo está en construcción y el viejo continua evolucionando. Quizás deban desarrollarse nuevas funcionalidades y esto requiere un esfuerzo doble, mientras que, si los dos sistemas pueden trabajar juntos, todo el esfuerzo de desarrollo puede dedicarse a la nueva plataforma. Por estas razones, se prefiere una migración progresiva a una así llamada estrategia de migración de "gran explosión", sin embargo, deben considerarse algunas dificultades. En particular, cuando los servicios de reservas se distribuyen entre dos plataformas diferentes (por ejemplo, ordenador central y plataformas abiertas) requieren compartir los mismos datos contextuales de Registro de Nombres de Pasajeros (PNR) en modo lectura y escritura para realizar sus funcionalidades de actividad. Una de las cuestiones a considerar es la sincronización de datos (por ejemplo, datos de PNR) que se comparten en modo lectura y escritura a través de plataformas diferentes y a través de protocolos de comunicación (por ejemplo, ordenador central de TPF y sistemas abiertos) de modo que los sistemas pueden compartir los mismos datos de contexto de PNR actualizados. Con "contexto" queremos decir el contexto de la sesión de compra que se vincula a una sesión de usuario (final) activa. Representa toda la información funcional y técnica que se usa por el sistema para que este usuario específico realice las funcionalidades solicitadas, por ejemplo, en el sistema de reservas de viajes, el contexto de sesión de reserva (compra) que se vincula a una sesión de usuario final activo.

El documento US2005/108298 (IYENGAR ARUN K [US] ET AL IYENGAR ARUN KWANGIL [US] ET AL) del 19 de mayo de 2005, describe un sistema de sincronización de múltiples servidores que proporciona una solución para actualizar múltiples copias del mismo objeto cuando el objeto se actualiza. Sin embargo, el contexto de la ultima actualización no incluye la ubicación de servidor en la que se hizo la actualización y deben realizarse más actualizaciones de las necesarias para mantener la consistencia.

El documento EP2259217 (ACCENTURE GLOBAL SERVICIOS GMBH [CH]) del 8 de diciembre de 2010, describe sincronización de lotes de PNR en sistemas de reservas de viajes. Sin embargo, se requiere una gran sobrecarga de recursos para implementar una solución de sincronización de este tipo.

Objeto de la invención

Un objeto de la presente invención es mitigar al menos algunos de los problemas asociados con los sistemas de la técnica anterior.

La solución de la presente invención se proporciona mediante la materia objeto de las reivindicaciones. Según un aspecto de la presente invención se proporciona un método para mecanismo de sincronización, en un método de reservas que opera en un sistema de múltiples servidores, para garantizar que se usa el registro de PNR más actualizado durante una transacción de usuario a través de al menos dos servidores del sistema de múltiples servidores, en el que una versión de contexto local del PNR se mantiene dentro de cada servidor del sistema de múltiples servidores, estando los servidores interconectados a través de un bus de sistema, incluyendo el mecanismo las etapas de: mantenimiento en un área de almacenamiento de contexto compartido, accesible por todos los servidores del sistema de múltiples servidores, de información acerca de la ultima versión actualizada de PNR; en respuesta a una petición de usuario que provoca que uno seleccionado de los servidores modifique la versión de contexto local de PNR, realización de las siguientes acciones: comprobar en el área de almacenamiento de contexto compartido qué servidor modificó por última vez el PNR; si el servidor que modificó por última vez el

PNR es diferente del servidor seleccionado, obtener la versión más actualizada de PNR; modificar la versión de contexto local de PNR para satisfacer la petición de usuario; actualizar el área de almacenamiento de contexto compartido para reflejar la ultima versión actualizada de PNR.

El método según una realización preferida de la presente invención permite la sincronización de los valores de PNR a través de un sistema de múltiples servidores (posiblemente multiplataforma) con un mecanismo eficiente y consistente. El mecanismo aborda las cuestiones de consistencia y rendimiento gracias a sus versiones y su vago comportamiento (la sincronización se produce únicamente cuando se requiere). Puede usarse como una solución durante una fase de migración desde un sistema a otro con migración progresiva de aplicaciones que comparten datos y también como una solución permanente para aplicaciones distribuidas a través de plataformas diferentes.

Según un segundo aspecto de la presente invención se proporciona un sistema que comprende uno o más componentes adaptados para realizar el método descrito anteriormente.

Según una realización adicional de la presente invención se proporciona un programa informático que comprende instrucciones para efectuar el método descrito anteriormente cuando dicho programa informático se ejecuta en un sistema informático.

Breve descripción de los dibujos

Se hará ahora referencia, a modo de ejemplo, a los dibujos adjuntos, en los que:

la Figura 1 es un diagrama del sistema de inventario según una realización de la presente invención;

la Figura 2 muestra esquemáticamente una posible estructura de un Dispositivo de Correlación de Contexto Distribuido usado en una realización preferida de la presente invención;

la Figura 3 muestra un procesamiento de peticiones en plataformas diferentes con el cambio asociado de valores en el contexto compartido con un formalismo de caso de uso;

la Figura 4 es un diagrama de un sistema informático general adaptado para soportar el método de una realización preferida de la presente invención;

las Figuras 5 a 9 (a y b) muestran los cinco servicios de Sincronización de Contexto Distribuido con una estructura de EDIFACT (figura a) y un diagrama de caso de uso (figura b), según una realización preferida de la presente invención;

la Figura 10 es un diagrama de flujo de las etapas de método de un proceso, según una realización de la presente invención.

Descripción detallada de las realizaciones

El ejemplo en el que se basa la presente descripción es la migración desde una arquitectura TPF compleja para un sistema de reservas a una arquitectura de sistema abierto. Por muchas razones, como se ha mencionado anteriormente, no es deseable hacer la migración en un único movimiento. Por lo tanto, durante un periodo de transición (que podría durar varios meses o años) el sistema de reservas se distribuye entre ordenador central, por ejemplo, sistemas TPF o MVS de International Business Corporation y plataforma abierta, por ejemplo, sistemas Unix o Linux. El sistema de reservas podría implementarse, por ejemplo, en una infraestructura de Amadeus. Sin embargo, la presente invención es aplicable a cualquier implementación de un sistema de reservas que trabaja a través de múltiples servidores con plataformas diferentes.

En el ejemplo de la migración desde ordenador central a plataforma abierta, en el ordenador central de TPF las aplicaciones de PNR comparten el mismo contexto de PNR en memoria y pueden acceder a los datos directamente a través de una API en modo lectura y/o escritura. Debido a la reingeniería de TPF y la migración de aplicaciones de PNR fuera del ordenador central de TPF, aparece el problema de compartición de contexto de PNR entre plataformas diferentes. De hecho, tenemos un concepto de un único sistema con contexto de usuario compartido a través plataformas heterogéneas que requieren aplicaciones distribuidas en plataformas diferentes para tener acceso a los mismos datos de PNR para garantizar la consistencia de acciones de funcionalidad de actividad realizadas a través de plataformas.

Un ejemplo con 2 aplicaciones, una dependiendo de la información de PNR proporcionado por la otra, se representa en el diagrama de la Figura 1. Muestra que ambos contextos de PNR locales tienen que sincronizarse para ser capaces de realizar las funcionalidades de actividad. Dos aplicaciones (App1 y App2) cooperan en un sistema de reservas: App1 (solo para hacer un ejemplo de una Aplicación de Reservas) se ejecuta en un sistema 101, mientras App2 (por ejemplo, una Aplicación de Precios) se ejecuta en un sistema 103. El sistema 101 de nuestro ejemplo es un ordenador central de TPF, mientras el sistema 103 es un sistema abierto, por ejemplo, un sistema Unix. Más

generalmente los dos sistemas 101 y 103 pueden ser cualquier sistema conocido en dos plataformas diferentes. Los dos sistemas 101 y 103 se conectan entre sí por medio de un Bus de Servicios de Empresa (ESB) 105 que también es accesible por un usuario a través de un terminal 107. La conexión entre el terminal 107 y el ESB 105 puede hacerse con cualquier disposición de red adecuada (por ejemplo, TCP/IP). También el ESB es un ejemplo de implementación, aunque podrían usarse otras estructuras conocidas en su lugar, por ejemplo, Encaminador, un Portal o un Mediador de Peticiones. Cada sistema 101 y 103 tiene acceso a un área de almacenamiento local (respectivamente 109 y 111) en la que se mantiene la información de PNR. La información de PNR local será la más actualizada para el sistema local, pero podría estar desactualizada con respecto al otro sistema. En la presente realización hemos usado un ejemplo con dos sistemas 101 y 103 trabajando en dos plataformas diferentes, pero los expertos en la materia apreciarán que otras implementaciones son posibles con varios sistemas diferentes. En el ejemplo de la Figura 1 el usuario, a través del terminal 107 y ESB 105 solicita (etapa 1) una reserva a la App1 que trabaja en una versión local de PNR 109; el PNR se actualiza (etapa 2) según la elaboración hecha por la App1. Cuando se pasa el control a la App2 (etapa 3) la aplicación trabaja (etapa 4) en la versión local del PNR 111. Antes de hacer eso es necesario verificar si el PNR local es la versión más actualizada (en el presente ejemplo no lo es), de lo contrario es necesaria una actualización. La App2 puede acceder a los sistemas externos 113 (por ejemplo, bases de datos de precios) para completar su elaboración (etapa 5).

En el método y sistema según una realización preferida de la presente invención, se replica una copia en memoria local del contexto de PNR en cada plataforma; las actualizaciones se realizan localmente y la sincronización se produce cuando otra plataforma necesita acceder a los datos de contexto actualizados. Esto es lo que llamamos sincronización de contexto distribuido. La complejidad del mecanismo de sincronización es determinar si una copia local está desactualizada o no y determinar dónde se ubican los datos de PNR más actualizados para cogerlos. Este mecanismo trabaja en todos los tipos de consultas de usuarios cualquiera que sea el protocolo de comunicación y no depende de las características técnicas de plataforma tal como representación de datos (por ejemplo, fila de datos pequeña o grande).

El presente enfoque a la sincronización de contexto de PNR responde a todos estos requisitos de una forma optimizada ya que la sincronización se realiza únicamente cuando se necesita y proporciona únicamente las actualizaciones a hacer en la copia de contexto local. Un elemento clave de la presente invención es un mecanismo para garantizar que el valor más actualizado de un parámetro compartido se usa en cualquier momento durante el proceso. En el método y sistema según una realización preferida de la presente invención se usa un dispositivo de correlación de contexto compartido distribuido. Como un ejemplo en el sistema de reservas de Amadeus descrito esto se llama DCX (Dispositivo de Correlación de Contexto Distribuido). DCX transmite información adicional encima de cada mensaje que viene desde la misma sesión de usuario, en todos los tipos de protocolos de comunicación para representar la distribución de los contextos aplicativos en las diferentes plataformas y aplicaciones.

Esta entidad de DCX se crea y almacena en el ESB y se transmite en todos los mensajes dentro de la infraestructura de Amadeus en el encabezamiento de sesión. La Figura 2 muestra un ejemplo de la estructura de DCX 200 según una realización preferida de la presente invención. Contiene referencias a contextos en las diferentes plataformas, que significa que no contiene los datos de contexto en sí. Se formatea en XML y compone de 3 partes como se muestra en la Figura 2: una reservada para información de contexto de ESB usada para encaminamiento y otros casos de uso (201), otra parte se dedica a seguridad y autenticación de usuario (203) y finalmente la tercera parte es la parte aplicativa (205) en la que la aplicación puede añadir sus preferencias de contexto e indicadores de estado asociados a las mismas. Es en la parte aplicativa que el proceso de Sincronización de Contexto almacena la información relacionada con los contextos de PNR distribuidos y es la base del mecanismo sin la que no funcionaría.

El DCX ofrece dos otras características requeridas para el mecanismo de sincronización que son la afinidad y la compartición de contexto entre diferentes protocolos de comunicación. La afinidad se requiere para dirigirse a exactamente el mismo servidor de aplicación cada vez que se invoca el mismo servicio y se necesita ya que los contextos de PNR son locales a los servidores de aplicación. Preferentemente, la información relacionada con la afinidad se comprende en claves, que pueden denominarse como "Claves de Afinidad", estando dichas claves comprendidas en el DCX. La compartición de información de contexto a través de protocolos se requiere para garantizar que un usuario que invoca los servicios de PNR en diferentes protocolos seguirá trabajando en el exacto mismo contexto de PNR.

La duración de la vida útil del contexto se controla mediante las conversaciones establecidas entre el ESB y el sistema abierto (o el ordenador central). El DCX ofrece una vista global de la actividad de usuario significando que si el usuario trabaja a través de una conversación específica (por ejemplo, conversación de EDIFACT), las otras conversaciones de protocolo se mantendrán para garantizar consistencia a través de protocolos. Cuando el usuario se desconecta desde el ESB (mediante un cierre específico de conversación o mediante una expiración por inactividad), las conversaciones a los sistemas abiertos y ordenador central también se cerrarán y desencadenará la limpieza de los contextos. Una descripción de DCX está también disponible en solicitudes pendientes por "METHOD AND SYSTEM FOR PROVIDING A SESSION INVOLVING A PLURALITY OF SOFTWARE APPLICATIONS" y "METHOD AND SYSTEM FOR PROVIDING A SESSION IN A HETEROGENEOUS ENVIRONMENT" presentadas por el mismo solicitante y que tienen la misma fecha de prioridad de la presente invención.

En los ejemplos de la descripción de la presente invención las conexiones entre servidores se realizan por medio de un ESB, sin embargo, los expertos en la materia apreciarán que podría usarse, en su lugar, cualquier otro medio de encaminamiento del estado de la técnica, capaz de encaminar una transacción a un servidor de aplicación apropiado, por ejemplo, un encaminador, un Portal o un Mediador de Peticiones.

El mecanismo de sincronización de contexto distribuido usa la parte aplicativa de la entidad de contexto compartido (DCX) para almacenar información acerca de los estados de contexto locales en las plataformas diferentes, también denominadas como máquinas o servidores de aplicación. Cada plataforma necesita referenciar su contexto local y actualizar el estado en cada transacción que implica un cambio en el contexto. Los datos relacionados con los contextos se estructuran de la siguiente forma:

```
<Application = SBR>
      <Platform, Context Version, Context Key, Context State>
</Application>
```

Todas las plataformas implicadas en la sincronización de contexto de PNR distribuida tendrán sus claves de contexto almacenadas en el DCX en el mismo tipo de aplicación (en este punto llamada "SBR").

El campo "Platform" corresponde al acrónimo de tres letras comúnmente usado para diseñar un sistema tal como TPF o sistema abierto RES.

El campo "Context Version" corresponde a la versión del contexto presente en la plataforma asociada, esto es un número que se aumenta cada vez que se modifica el contexto.

El campo "Context Key", también se denomina como Clave de Contexto Aplicativa, que corresponde al identificativo único que permite la recuperación del contexto en la plataforma asociada.

El campo "Context State" corresponde al estado del contexto en la plataforma asociada. El estado de contexto puede representar el hecho de que el contexto está activo, inactivo, corrompido y si fue el último contexto actualizado.

Un ejemplo de claves de aplicación en DCX, después de 1 caso de uso procesado en TPF y a continuación 1 caso de uso procesado en RES OBE (EXTREMO POSTERIOR ABIERTO) podría ser:

```
<Application = SBR>
     <TPF, 1, Key1, ACT>
     <RES, 1, Key2, ACT/Last>
</Application>
```

Las versiones de los contextos locales es la clave para implementar un mecanismo de sincronización vago (se realiza únicamente cuando se requiere) como se verá en el siguiente capítulo acerca del algoritmo. El indicador de "Último Actualizador" es también la clave para determinar qué plataforma tiene el último contexto, de modo que la sincronización se hace con esta plataforma.

Además del estado actual de los contextos de PNR distribuidos que se transmiten dentro de la entidad de DCX en todos los mensajes, cada plataforma tiene un estado de sincronización local almacenado asociado a su contexto. Este estado de sincronización local representa el estado de los contextos de PNR distribuidos en las otras plataformas en el momento de la última sincronización realizada. Permite determinar si un contexto local está desactualizado en comparación con las otras plataformas, que es la condición desencadenante para la sincronización. De hecho, si se han hecho varias actualizaciones sucesivas en un contexto preciso, la sincronización no se realizará cada vez, ya que el contexto local estará actualizado en comparación con otros contextos de plataforma.

La estructura de datos descrita anteriormente es una de las alternativas posibles, sin embargo pueden realizarse otras implementaciones para garantizar la consistencia de las diferentes instancias del mismo parámetro compartido. El requisito de tal estructura de datos es que información acerca de la ubicación de la versión más actualizada de PNR puede compartirse entre todas las aplicaciones a través de todas las plataformas usadas por el sistema.

Como se muestra en las Figuras 3, los flujos globales de modificaciones hechas en las claves aplicativas en el contexto compartido (DCX en el ejemplo actual) para la parte de sincronización se explican tomando el ejemplo de 2 plataformas llamadas Sistema1 y Sistema2 (por ejemplo, una ejecutándose en un ordenador central y una en Sistema Abierto). En el diagrama de la Figura 3 se describe el procesamiento de casos de uso en plataformas diferentes con el cambio asociado de valores en el contexto compartido. El estado actual transmitido en el contexto compartido y los estados de sincronización local de las diferentes plataformas se representan en el diagrama para mostrar la diferencia entre los mismos.

En una primera etapa (etapa 1 en la Figura 3), el propio usuario iniciará sesión en el sistema a través del ESB; en esta fase se creará un contexto compartido vacío para esta sesión y almacenará en el ESB (etapa 2).

A continuación, el usuario empieza a trabajar, enviando una consulta de actividad que se manejará por un servicio ubicado en el Sistema1 (etapa 3). El ESB transmitirá automáticamente el contexto compartido con la consulta de usuario al Sistema1. Ya que el Sistema1 no encuentra información de sincronización ni localmente ni en el contexto compartido, procesa directamente la consulta de usuario recibida (etapa 4). En caso de que un contexto de reserva se ha creado mediante el procesamiento de la consulta, los datos de sincronización locales se actualizan con el nuevo estado que es que el Sistema1 tiene un contexto de reserva en la versión 1 y fue el último actualizador para la misma (etapa 5). La respuesta de actividad se envía de vuelta al usuario, junto con el contexto compartido actualizado con datos de sincronización; el contexto compartido se almacena en el ESB antes de reenviar la respuesta al usuario de modo que puede usarse en las siguientes consultas realizadas por el usuario (etapa 6). El usuario envía después otra consulta de actividad que se manejará por un servicio ubicado en el Sistema2 (etapa 7). El ESB transmitirá automáticamente el contexto compartido con la consulta de usuario al Sistema2. En el Sistema2 no hay estado de sincronización local almacenado y una comparación con los datos de sincronización presentes en el contexto compartido recibido con la consulta muestra que se requiere sincronización con el Sistema1 ya que mantiene un nuevo contexto de reservas (etapa 8). Así que el Sistema2 preguntará al Sistema1 su contexto de reservas (etapa 9) para almacenar el mismo localmente y permitir el procesamiento de la consulta de usuario (etapa 10). Junto con el almacenamiento del contexto de reserva, el estado de sincronización local se inicializa con las diferentes claves aplicativas que representan la situación. A continuación, tiene lugar el procesamiento de la consulta (etapa 11); una vez que se completa, en el caso de que el contexto de reservas se ha actualizado, la sincronización local se actualiza para representar ese hecho de que el Sistema2 es ahora el último actualizador de la información de reserva y que su contexto ahora tiene la versión 1 (etapa 12). La respuesta de actividad se envía de vuelta al usuario, junto con el contexto compartido actualizado con datos de sincronización; el contexto compartido se almacena en el ESB antes de reenviar la respuesta al usuario de modo que puede usarse en las siguientes consultas realizadas por el usuario (etapa 13).

Puede continuar con la misma secuencia de acción con posteriores consultas del usuario, dirigiéndose o bien al Sistema1 o bien Sistema2 indiferentemente. Dependiendo de la comparación del estado de sincronización local y el contexto compartido, el sistema determinará si se requiere sincronización o no.

En la plataforma de Sistema1, por ejemplo, cuando se llama a un servicio que usa el contexto de reservas (por ejemplo, PNR) en modo lectura o escritura, se aplicará el siguiente algoritmo para determinar si se requiere o no la sincronización con otro sistema:

0- Recibir la petición
1- Conseguir Claves de Aplicación en contexto compartido actual recibido en la petición de usuario (por ejemplo, Sistema1 v1 / Sistema2 v2 Última)
2- Conseguir estado de sincronización previo almacenado localmente (por ejemplo, Sistema1 v1 / Sistema2 v1 Última)
3- Ejecutar comprobación de consistencia:

3a- ¿Es la versión de Sistema1 actual la misma que la previa almacenada localmente? ¿Es la clave la misma? (OK si SÍ)
3b- ¿Es la versión de Sistema2 actual mayor o igual que la previa almacenada localmente? (OK si SÍ)

4- Ejecutar comprobación de sincronización: ¿es la versión de Sistema2 actual mayor que la almacenada localmente y está presente bandera de "último actualizador"? (Se necesita sincronización si SÍ)
5- Si se necesita sincronización

5a- Ejecutar la sincronización (CONSEGUIR el contexto de la plataforma del Sistema2)
5b- Si satisfactoria, Sistema1 actualiza el estado de sincronización previo almacenado localmente con valores: Sistema1 v1 / Sistema2 v2 (significa que ahora en el Sistema1 el contexto está sincronizado con el Sistema2 v2)

6- Procesar consulta de actividad en el Sistema1
7- Actualizar la versión de Sistema1 si modificaciones hechas en el contexto de reservas (PNR)

7a- Actualizar la versión del Sistema1 en contexto compartido actual (Sistema1 v2 Última / Sistema2 v2). Este contexto compartido se enviará de vuelta al ESB para mantener el mismo para siguientes consultas de usuario
7b- Actualizar la versión del Sistema1 en estado de sincronización previo almacenado localmente con valores: Sistema1 v2 / Sistema2 v2
8- Contestar

Se puede ver en este algoritmo que se necesitan comparar 2 conjuntos de claves de contexto para ser capaces de determinar qué tiene que hacerse desde un punto de vista de sincronización. La bandera de "último actualizador" se requiere cuando más de 2 plataformas tienen contextos de PNR, para determinar la que tiene la última versión del contexto. Para 2 plataformas, el mecanismo podría funcionar sin la misma.

Una ventaja adicional de este mecanismo es que se pueden detectar errores de transmisión de contexto compartido en la siguiente fase de sincronización. De hecho, gracias a las comprobaciones de consistencia y la información de sincronización localmente almacenada, es posible determinar si se ha producido un error. El hecho de que la versión de la plataforma local no coincide o las claves de contexto son diferentes, muestran una corrupción del contenido de contexto compartido y esto es probable que sea la consecuencia de un error. El manejo del error puede ser bastante complejo debido a comunicaciones e intercambios colaterales, por lo que no se detallará en este documento, pero los expertos en la materia apreciarán que pueden usarse varios métodos de estado de la técnica para implementar esta tarea.

Con referencia a la Figura 4 un ordenador genérico del sistema (por ejemplo, cualquier ordenador, servidor de reservas, ordenador central de TPF, servidor de Sistema Abierto, subsistema de gestión de base de datos, encaminador, servidor de red) se indica con 450. El ordenador 450 se forma por varias unidades que se conectan en paralelo a un bus de sistema 453. En detalle, uno o más microprocesadores 456 controlan la operación del ordenador 450; una RAM 459 se usa directamente como una memoria de trabajo por los microprocesadores 456 y una ROM 462 almacena código básico para una rutina de arranque del ordenador 450. Unidades periféricas se agrupan alrededor de un bus local 465 (por medio de respectivas interfaces). Particularmente, una memoria de masa que consiste en un disco duro 468 y una unidad 471 para leer el CD-ROM 474. Además, el ordenador 450 incluye dispositivos de entrada 477 (por ejemplo, un teclado y un ratón) y dispositivos de salida 480 (por ejemplo, un monitor y una impresora). Se usa una Tarjeta de Interfaz de Red 483 para conectar el ordenador 450 a la red. Una unidad de puente 486 interconecta el bus de sistema 453 con el bus local 465. Cada microprocesador 456 y la unidad de puente 486 pueden operar como agentes maestros que solicitan un acceso al bus de sistema 453 para transmitir información. Un árbitro 489 gestiona la concesión del acceso con exclusión mutua al bus de sistema 453. Se aplican consideraciones similares si el sistema tiene una topología diferente o se basa en otras redes. Como alternativa, los ordenadores tienen una estructura diferente, incluyen unidades equivalentes o constan de otras entidades de procesamiento de datos (tal como PDA, teléfonos móviles y similares).

Los servicios de EDIFACT que se usan en el mecanismo de sincronización de contexto de PNR y que transportan los datos de PNR se componen de la siguiente información:

- El estado de sincronización local de la plataforma que solicita o proporciona los datos de contexto de PNR. Esta información se requiere para determinar el número de transacciones de usuario que tienen que retirarse u aplicarse en el contexto local. También permite optimizaciones para reducir la cantidad de datos intercambiados entre las plataformas.
- El contexto de PNR serializado, que puede estar completo en caso de la primera sincronización o puede componerse de actualizaciones de contexto cuando la plataforma que solicita la última versión del contexto ya tiene una copia local de contexto. El contexto serializado es una PASBCQ que es en sí mismo un mensaje EDI.

El hecho de que el contexto se serializa en un mensaje EDI permite deshacerse de las particularidades de representación de datos de la plataforma que proporciona el mismo. De hecho, el contenido de datos se normaliza y es independiente de la representación de datos de plataforma.

La Sincronización de Contexto Distribuido se basa en 5 diferentes servicios que puede cualquiera implementarse en ambas plataformas TPF y OBE o ser específicos a una plataforma maestra/esclava. Cada servicio transmitirá el DCX junto con el mensaje, especialmente porque el DCX permitirá que las comprobaciones de consistencia se ejecuten en el estado de la sincronización de contexto. Cada servicio se describe en este punto con referencia a una estructura de EDIFACT (solicitud y respuesta) y un diagrama de caso de uso para mostrar datos intercambiados.

CONSEGUIR contexto: PCSGRQ/R

Este servicio, como se muestra en las Figuras 5a y 5b se usa para recuperar de una plataforma remota la última versión de su contexto. Debería implementarse en todas las plataformas en las que pueda leerse el contexto.

Datos intercambiados

En la consulta, el cliente debería proporcionar su estado de sincronización. Se compone de una lista de 3 valores <Platform, Context Key, Version>.

En la respuesta, el servidor debería proporcionar su estado de sincronización asociado con el contexto serializado con sus versiones. Ya que pueden encontrarse errores, la respuesta debería contener un grupo de errores que describen el problema.

Versiones de serialización de CONSEGUIR contexto PCSVRQ/R

Este servicio, mostrado en las Figuras 6a y 6b se usa para recuperar todas las versiones relacionadas con la serialización del contexto. Este servicio debería implementarse en OBE. En OBE, el contexto se basa en un Modelo de SBR, que se serializa en un mensaje EDIFACT llamado PASBCQ. El PASBCQ se compone de varios objetos binarios grandes que tienen una versión asociada. Ya que TPF es el maestro de las versiones de PASBCQ, el OBE tiene que recuperar las versiones actuales de serialización de TPF cuando no las conoce aún, para poder escribir un mensaje consistente. Esta situación se produce cuando sistemas externos están solicitando actualizaciones no solicitadas directamente en OBE.

Datos intercambiados

En la consulta, el cliente no debería necesitar proporcionar ningún dato.

En la respuesta, el servidor debería proporcionar sus versiones usadas para serializar el mensaje PASBCQ EDIFACT. Debería componerse de un objeto binario grande que contiene todas las versiones usadas para serializar, junto con la versión de objeto binario grande. Ya que pueden encontrarse errores, la respuesta debería contener un grupo de errores que describen el problema.

IMPULSAR contexto: PCSPUQ/R

Este servicio (véase la Figura 7a y 7b) se usa para impulsar la última versión del contexto local a una plataforma remota para actualizar el contexto en la última. Debería implementarse en plataformas que no son maestras del contexto. Este servicio es una optimización del mecanismo de sincronización para tener el maestro de contexto actualizado. En este caso, el maestro no usaría el servicio de CONSEGUIR Contexto. Ya que en TPF las llamadas de cliente son costosas, esto podría ayudar a reducir el consumo de recursos en la misma.

Datos intercambiados

En la consulta, el cliente debería proporcionar su estado sincronizado asociado con el contexto serializado con sus versiones.

En la respuesta, el servidor debería proporcionar únicamente un acuse de recibo. Ya que pueden encontrarse errores, la respuesta debería contener un grupo de errores que describen el problema.

IMPULSAR y EOT contexto: PCSEUQ/R

Este servicio (Figura 8a y 8b) se usa para impulsar la última versión del contexto local a una plataforma remota para actualizar el contexto en la última y llamar al proceso de Fin de Transacción en este contexto. Debería implementarse en todas las plataformas que requieren actualizar el contexto de PNR y llamar al proceso Fin de Transacción en estas modificaciones. Este servicio se requiere para gestionar entradas de procesamiento tales como Petición de Impresión de Billete (TTP) o Acuse de Recibo de Billete procedente de un sistema externo.

Datos intercambiados

En la consulta, el cliente debería proporcionar su estado sincronizado asociado con el contexto serializado con sus versiones.

En la respuesta, el servidor debería proporcionar únicamente un acuse de recibo. Ya que pueden encontrarse errores, la respuesta debería contener un grupo de errores que describen el problema.

IGNORAR propagación de contexto: PCSINQ/R

Este servicio se usa para ignorar todas las modificaciones que se han hecho en un SBR a través de las diferentes plataformas implicadas en los flujos. TPF recibirá la entrada "IG" y a continuación propagará la consulta a las OBE que tienen un contexto de SBR registrado en el DCX, de modo que pueden limpiar sus contextos. Se muestra en las Figuras 9a y 9b.

Datos intercambiados

En la consulta, el cliente no debería necesitar proporcionar ningún dato.

En la respuesta, el servidor debería proporcionar únicamente un acuse de recibo. Ya que pueden encontrarse errores, la respuesta debería contener un grupo de errores que describen el problema. El método descrito anteriormente también se representa en el diagrama mostrado en la Figura 10. El método realiza un mecanismo de sincronización, en un método de reservas que opera en un sistema de múltiples servidores, para garantizar que se

usa el registro de PNR más actualizado durante una transacción de usuario a través de al menos dos servidores del sistema de múltiples servidores, en el que se mantiene una versión de contexto local del PNR dentro de cada servidor del sistema de múltiples servidores, estando los servidores interconectados a través de un bus de sistema. El método comienza en el círculo negro 1001 y a continuación va a la caja 1003 en la que se mantiene la versión más actualizada de PNR en un área de almacenamiento accesible por todos los servidores implicados en la transacción. El sistema de reservas de múltiples servidores recibe una petición de usuario que necesita una acción de actualización del PNR (etapa 1005) por ejemplo en un servidor "A". El bus de sistema (por ejemplo, el ESB, pero más generalmente cualquier medio de encaminamiento) determina y selecciona qué servidor necesita implicarse para procesar la petición. En las etapas 1007 y 1009 el servidor seleccionado comprueba si el PNR de contexto local es el más actualizado comparando el PNR de contexto local con la información disponible en el PNR de contexto compartido en el ESB: si el PNR de contexto local no es el más actualizado (es decir otro servidor ha modificado el PNR desde la última actualización por el servidor seleccionado (si hay alguno)) el servidor seleccionado obtiene la versión más actualizada de PNR (etapa 1011). A continuación, el servidor seleccionado realiza la actividad solicitada y modifica el PNR de contexto local (etapa 1013) que también se convierte en la versión más actualizada: tal información se pasa a continuación (etapa 1015) al PNR de contexto compartido en ESB para que se haga accesible a todos los otros servidores del sistema de múltiples servidores. Los detalles de la forma en que se transmiten la información entre los servidores, el ESB y el usuario se ha analizado en los párrafos anteriores.

Se apreciará que pueden hacerse alteraciones y modificaciones a lo anterior sin alejarse del alcance de la divulgación. Naturalmente, para satisfacer requisitos locales y específicos, un experto en la materia puede aplicar a la solución descrita anteriormente muchas modificaciones y alteraciones. Particularmente, aunque la presente divulgación se ha descrito con un cierto grado de particularidad con referencia a realización o realizaciones preferidas de la misma, debería entenderse que son posibles diversas omisiones, sustituciones y cambios en la forma y detalles así como otras realizaciones; además, se concibe expresamente que elementos específicos y/o etapas de método descritas en conexión con cualquier realización divulgada de la divulgación pueden incorporarse en cualquier otra realización como una cuestión general de elección de diseño.

Se aplican consideraciones similares si el programa (que puede usarse para implementar cada realización de la divulgación) se estructura de una forma diferente, o si se proporcionan módulos o funciones adicionales; análogamente, las estructuras de memoria pueden ser de otros tipos o pueden sustituirse con entidades equivalentes (no necesariamente consistentes de medio de almacenamiento físico). Además, la solución propuesta se presta para implementarse con un método equivalente (teniendo etapas similares o adicionales, incluso en un orden diferente). En cualquier caso, el programa puede tener cualquier forma adecuada para usarse mediante o en conexión con cualquier sistema de procesamiento de datos, tal como software externo o residente, firmware o microcódigo (ya sea en código objeto o en código fuente). Además, el programa puede proporcionarse en cualquier medio usable por ordenador; el medio puede ser cualquier elemento adecuado para contener, almacenar, comunicar, propagar o transferir el programa. Ejemplos de tal medio son discos fijos (en los que el programa puede precargarse), discos extraíbles, cintas, tarjetas, alambres, fibras, conexiones inalámbricas, redes, ondas de radiodifusión y similares; por ejemplo, el medio puede ser del tipo electrónico, magnético, óptico, electromagnético, infrarrojos o semiconductor.

En cualquier caso, la solución según la presente divulgación se presta a ser efectuado con una estructura de hardware (por ejemplo, integrada en un chip de material semiconductor) o con una combinación de software y hardware.

REIVINDICACIONES

1. Un método de sincronización, en una aplicación de reservas que opera en un sistema de múltiples servidores, intercambiando la aplicación de reservas (101) información relacionada con viajes por medio de una estructura de datos de Registro de Nombres de Pasajeros (PNR), para garantizar que se usa el PNR más actualizado durante una transacción de usuario a través de al menos dos servidores del sistema de múltiples servidores, en el que una versión de contexto local del PNR se mantiene dentro de cada servidor del sistema de múltiples servidores, estando los servidores interconectados a través de un medio de encaminamiento, incluyendo el método las etapas de:

- mantenimiento en un área de almacenamiento de contexto compartido (1003), accesible por todos los servidores del sistema de múltiples servidores, de información acerca de la ubicación de la última versión modificada de PNR;
- en respuesta a una petición de usuario que provoca que uno de los servidores seleccionado modifique la versión de contexto local de PNR realizando las siguientes acciones:

    - comprobar en el área de almacenamiento de contexto compartido qué servidor modificó por última vez el PNR;
    - si el servidor que modificó por última vez el PNR es diferente del servidor seleccionado, obtener la versión más actualizada de PNR;
    - modificar la versión de contexto local de PNR para satisfacer la petición de usuario;
    - actualizar el área de almacenamiento de contexto compartido para reflejar la última versión modificada de PNR.

2. El método de sincronización de la reivindicación 1 en el que los servidores del sistema de múltiples servidores intercambian información con el medio de encaminamiento y el usuario por medio de un sistema de Arquitectura Orientada al Cliente (SOA).

3. El método de sincronización de la reivindicación 2 en el que los mensajes incluyen un encabezamiento de mensaje que comprende información acerca de la última versión modificada de PNR.

4. El método de sincronización de cualquier reivindicación anterior en el que información acerca de la última versión modificada de PNR incluye un puntero a la última versión modificada de PNR.

5. El método de sincronización de cualquier reivindicación anterior en el que el medio de encaminamiento incluye un bus de sistema, por ejemplo, un Bus de Servicios de Empresa (ESB).

6. Un programa informático que comprende instrucciones para efectuar las etapas del método según una cualquiera reivindicación anterior, cuando dicho programa informático se ejecuta en un ordenador.

7. Un producto de programa informático que incluye medios legibles por ordenador que incorpora el programa informático de la reivindicación 6.

8. Un sistema de procesamiento de datos de multiples servidores de reservas, incluyendo un método de sincronización para garantizar que se usa el registro de PNR más actualizado durante una transacción de usuario a través de al menos dos servidores del sistema de múltiples servidores, en el que una versión de contexto local del PNR se mantiene (1003) dentro de cada servidor del sistema de múltiples servidores, estando los servidores interconectados a través de un medio de encaminamiento, en el que el sistema comprende uno o más componentes adaptados para realizar el método de cualquier reivindicación 1 a 5.

9. Un sistema desplegado en un sistema de procesamiento de datos para implementar el método de cualquier reivindicación 1 a 5.

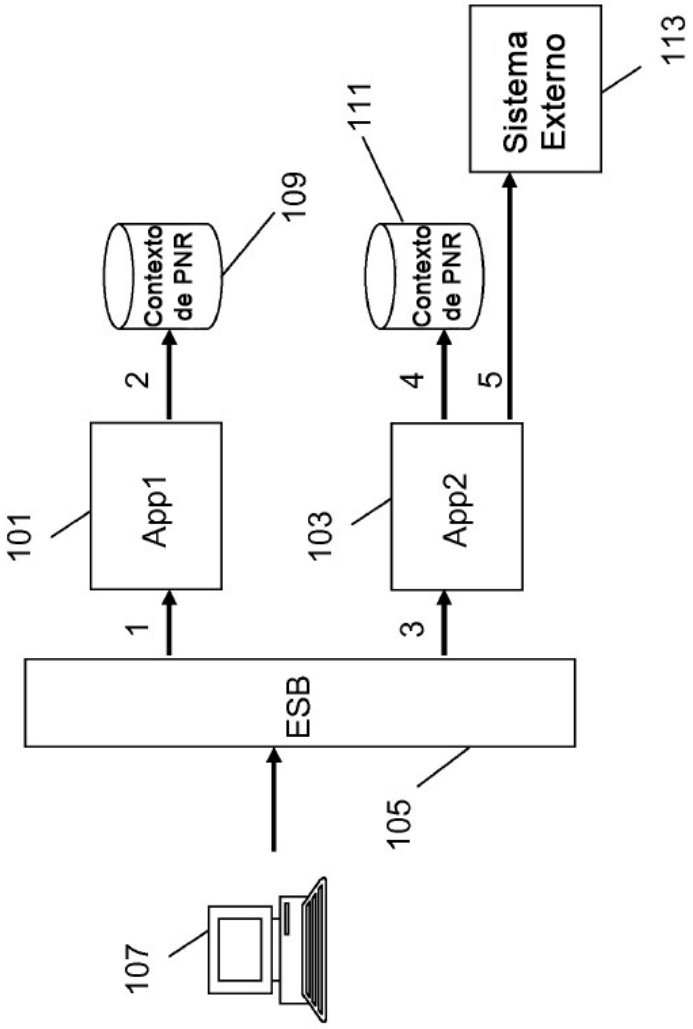
Sistema Externo
113
111
Contexto de PNR
109
Contexto de PNR
2
4
5
App1
101
App2
103
1
3
ESB
105
107

Fig. 1

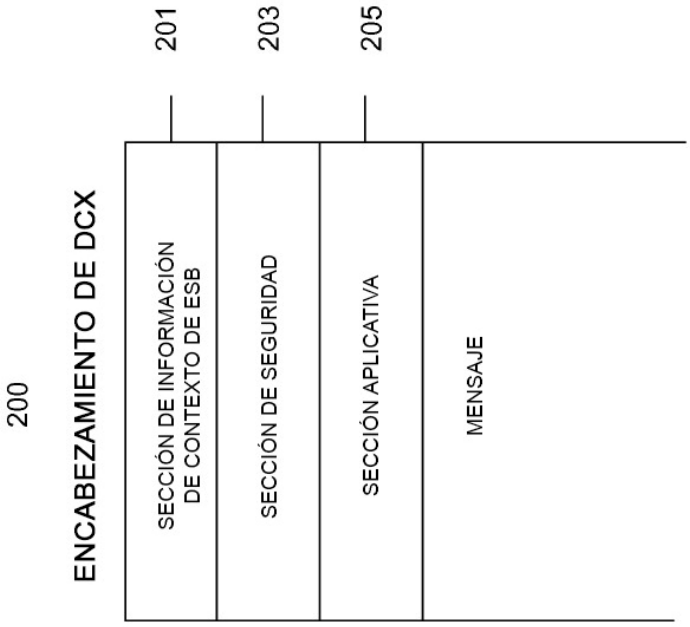
200
ENCABEZAMIENTO DE DCX
201
203
205
SECCIÓN DE INFORMACIÓN DE CONTEXTO DE ESB
SECCIÓN DE SEGURIDAD
SECCIÓN APLICATIVA
MENSAJE

Fig. 2

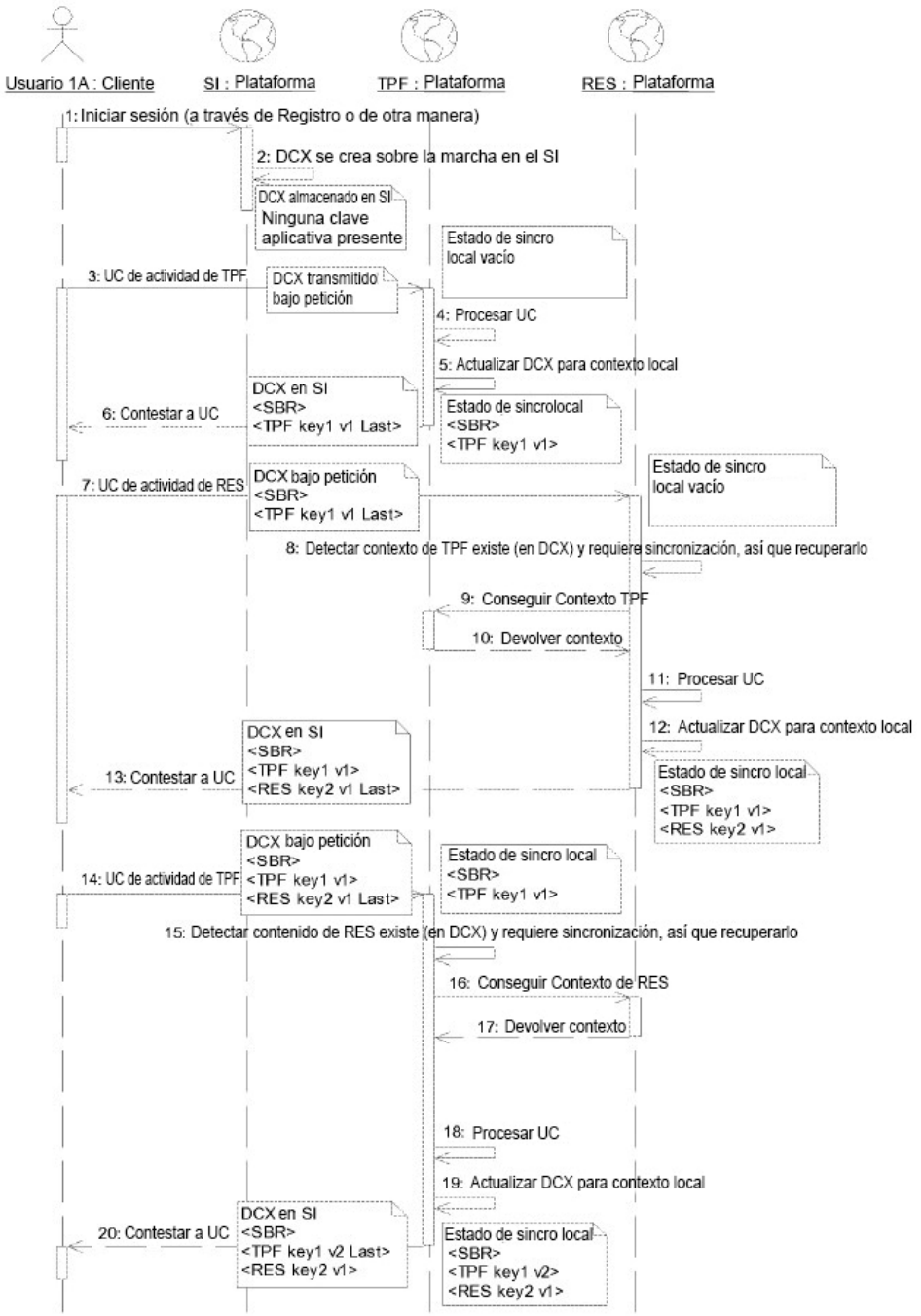
Usuario 1A : Cliente
SI : Plataforma
TPF : Plataforma
RES : Plataforma
1: Iniciar sesión (a través de Registro o de otra manera)
2: DCX se crea sobre la marcha en el SI
DCX almacenado en SI
Ninguna clave aplicativa presente
Estado de sincro local vacío
3: UC de actividad de TPF
DCX transmitido bajo petición
4: Procesar UC
5: Actualizar DCX para contexto local
DCX en SI
<SBR>
<TPF key1 v1 Last>
6: Contestar a UC
Estado de sincrolocal
<SBR>
<TPF key1 v1>
Estado de sincro local vacío
7: UC de actividad de RES
DCX bajo petición
<SBR>
<TPF key1 v1 Last>
8: Detectar contexto de TPF existe (en DCX) y requiere sincronización, así que recuperarlo
9: Conseguir Contexto TPF
10: Devolver contexto
11: Procesar UC
12: Actualizar DCX para contexto local
DCX en SI
<SBR>
<TPF key1 v1>
<RES key2 v1 Last>
13: Contestar a UC
Estado de sincro local
<SBR>
<TPF key1 v1>
<RES key2 v1>
DCX bajo petición
<SBR>
<TPF key1 v1>
<RES key2 v1 Last>
14: UC de actividad de TPF
Estado de sincro local
<SBR>
<TPF key1 v1>
15: Detectar contenido de RES existe (en DCX) y requiere sincronización, así que recuperarlo
16: Conseguir Contexto de RES
17: Devolver contexto
18: Procesar UC
19: Actualizar DCX para contexto local
DCX en SI
<SBR>
<TPF key1 v2 Last>
<RES key2 v1>
20: Contestar a UC
Estado de sincro local
<SBR>
<TPF key1 v2>
<RES key2 v1>

Fig. 3

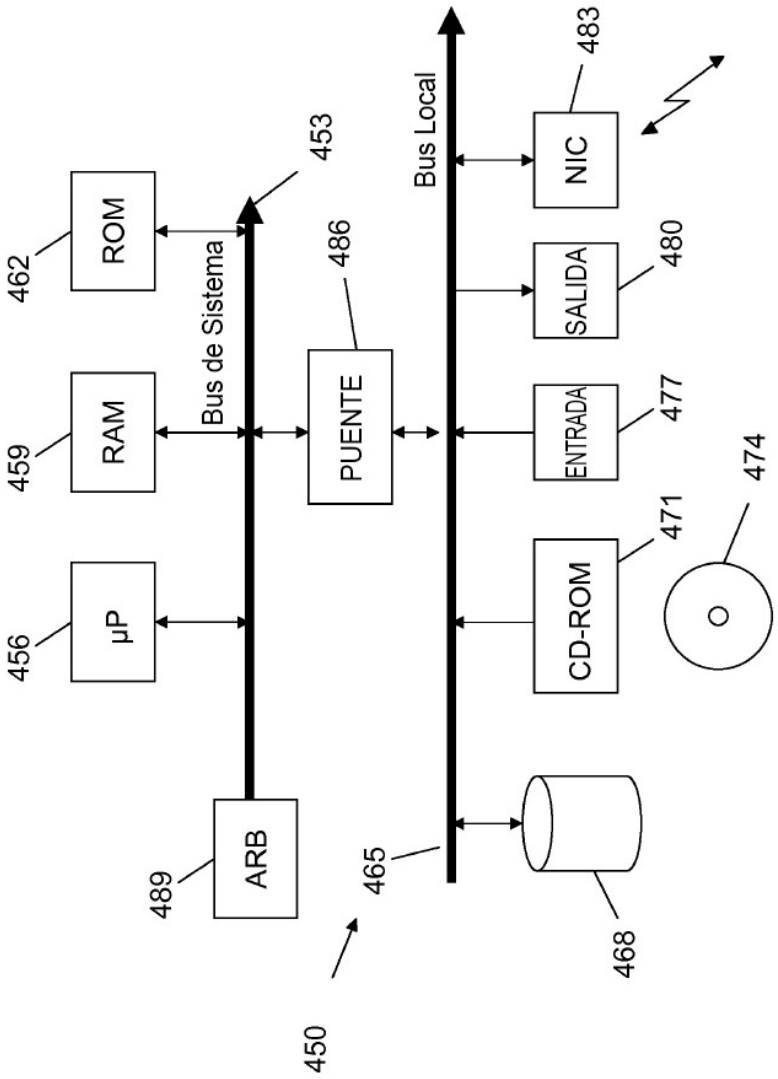
ROM
462
RAM
459
μP
456
Bus de Sistema
453
ARB
489
PUENTE
486
Bus Local
NIC
483
SALIDA
480
ENTRADA
477
CD-ROM
471
474
465
468
450

Fig. 4

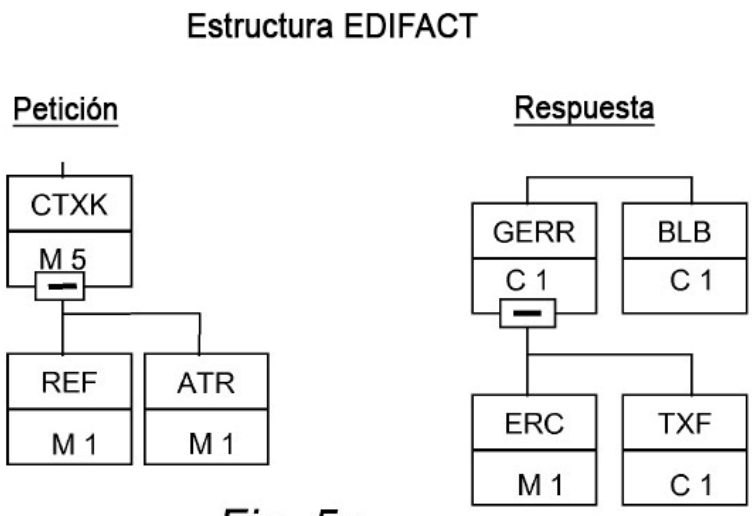
Estructura EDIFACT
Petición
CTXK
M 5
REF
M 1
ATR
M 1
Respuesta
GERR
C 1
BLB
C 1
ERC
M 1
TXF
C 1

Fig. 5a

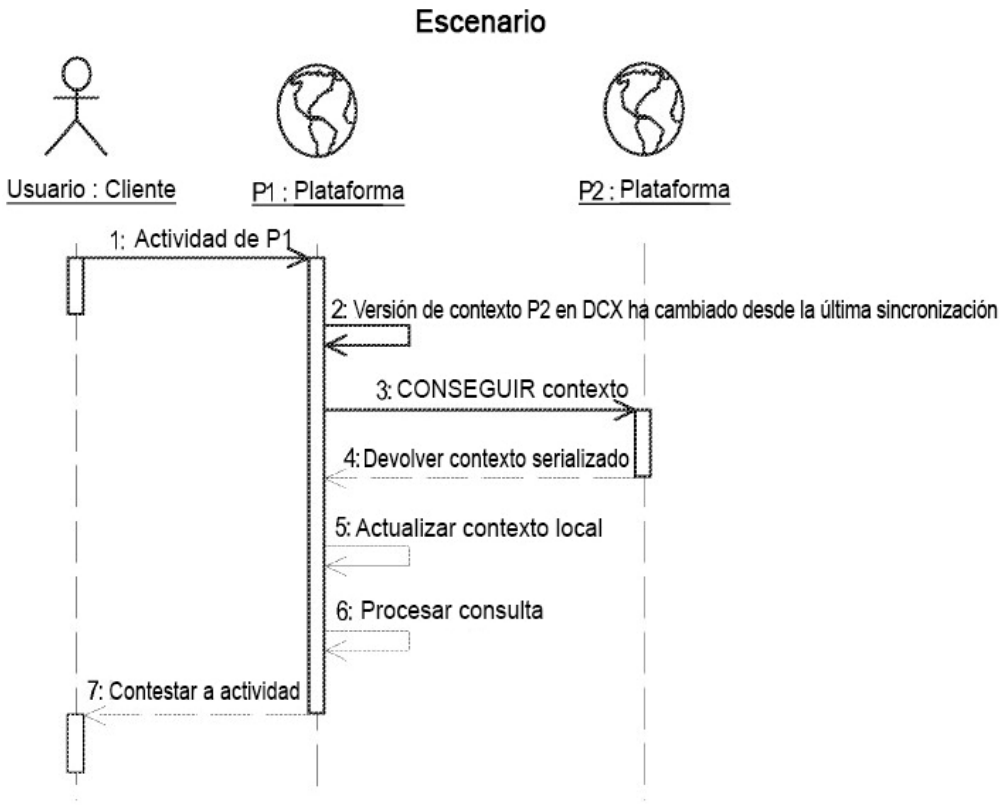
Escenario
Usuario : Cliente
P1 : Plataforma
P2 : Plataforma
1: Actividad de P1
2: Versión de contexto P2 en DCX ha cambiado desde la última sincronización
3: CONSEGUIR contexto
4: Devolver contexto serializado
5: Actualizar contexto local
6: Procesar consulta
7: Contestar a actividad

Fig. 5b

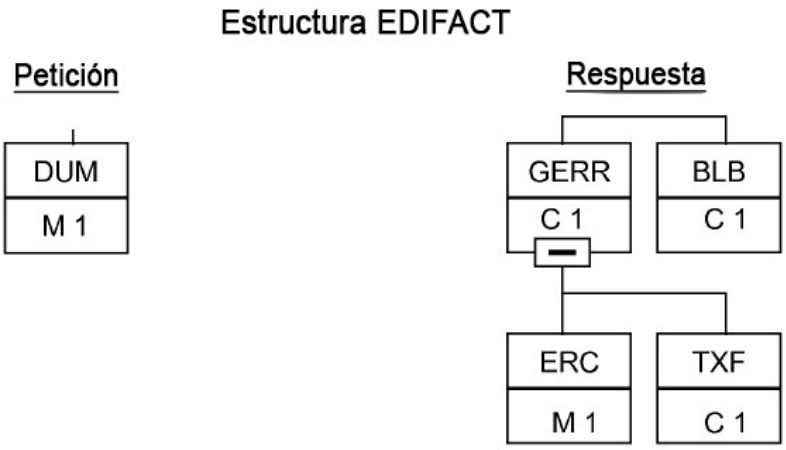
Estructura EDIFACT
Petición
Respuesta
DUM
M 1
GERR
C 1
BLB
C 1
ERC
M 1
TXF
C 1

Fig. 6a

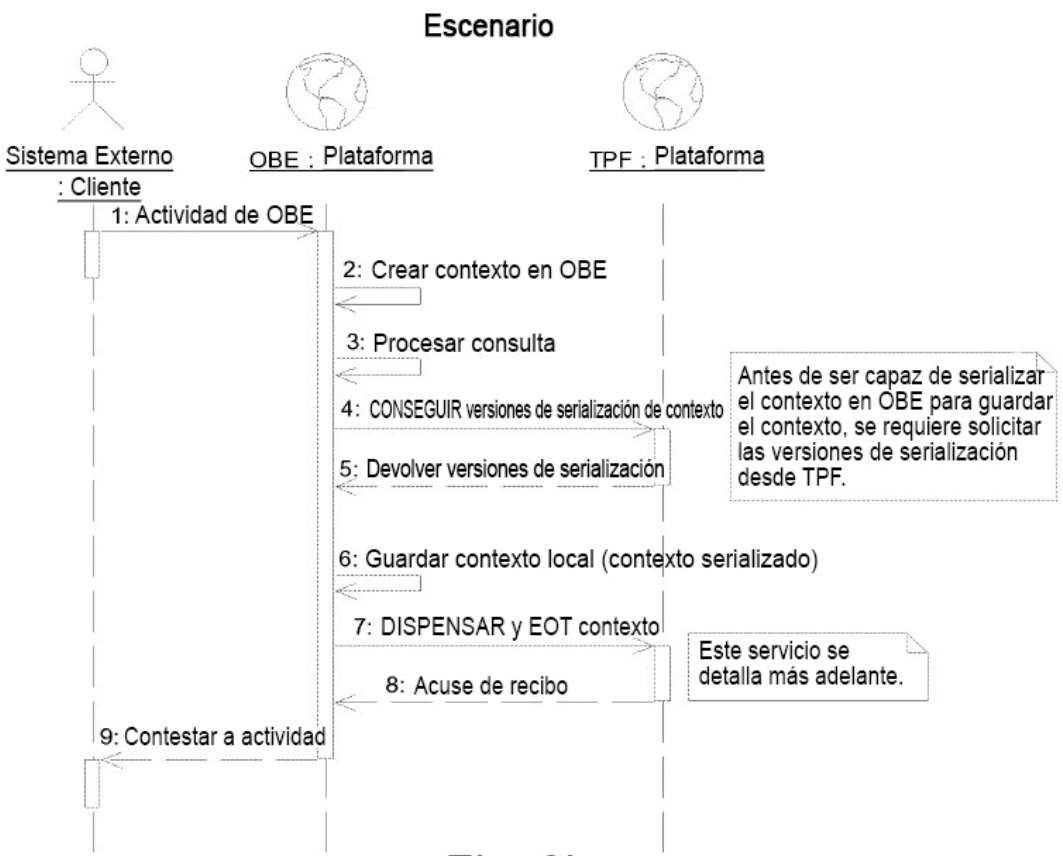
Escenario
Sistema Externo
: Cliente
OBE : Plataforma
TPF : Plataforma
1: Actividad de OBE
2: Crear contexto en OBE
3: Procesar consulta
4: CONSEGUIR versiones de serialización de contexto
5: Devolver versiones de serialización
Antes de ser capaz de serializar el contexto en OBE para guardar el contexto, se requiere solicitar las versiones de serialización desde TPF.
6: Guardar contexto local (contexto serializado)
7: DISPENSAR y EOT contexto
8: Acuse de recibo
Este servicio se detalla más adelante.
9: Contestar a actividad

Fig. 6b

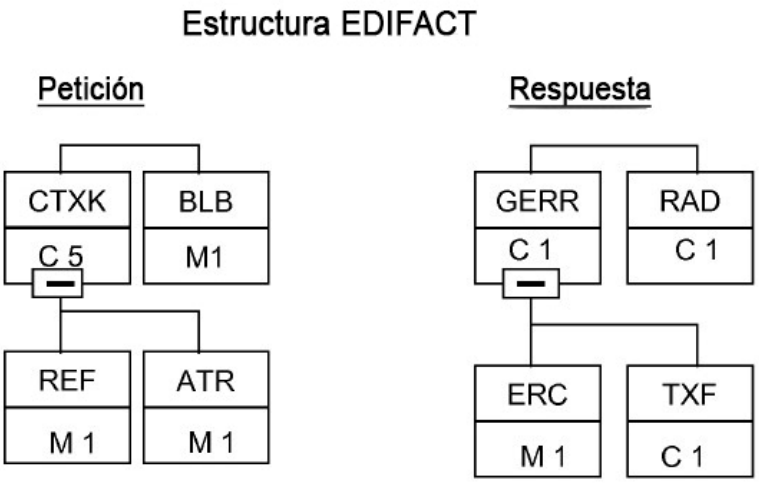
Estructura EDIFACT
Petición
Respuesta
CTXK
C 5
BLB
M1
REF
M 1
ATR
M 1
GERR
C 1
RAD
C 1
ERC
M 1
TXF
C 1

Fig. 7a

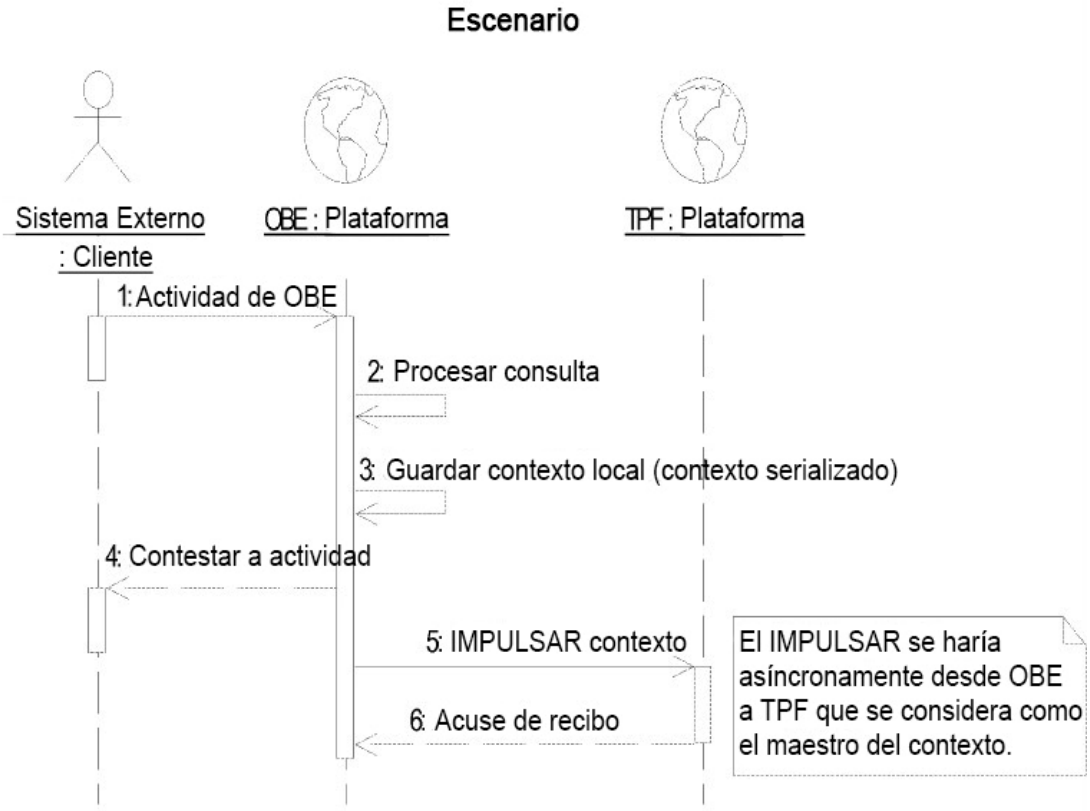
Escenario
Sistema Externo
: Cliente
OBE : Plataforma
TPF : Plataforma
1: Actividad de OBE
2: Procesar consulta
3: Guardar contexto local (contexto serializado)
4: Contestar a actividad
5: IMPULSAR contexto
6: Acuse de recibo
El IMPULSAR se haría asíncronamente desde OBE a TPF que se considera como el maestro del contexto.

Fig. 7b

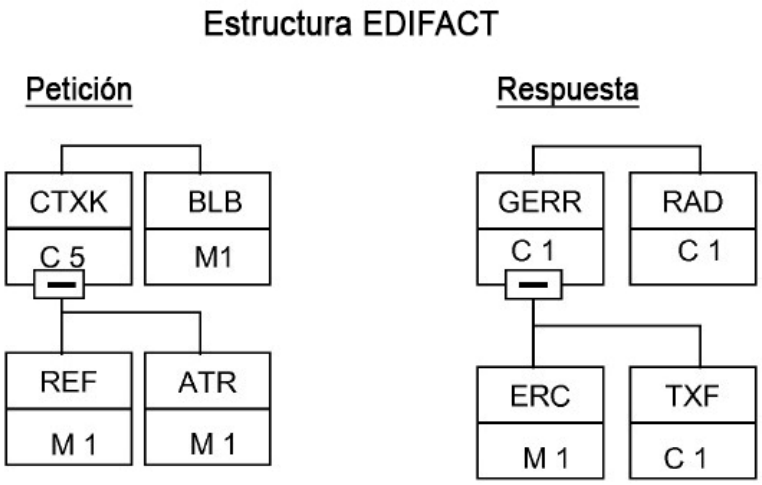
Estructura EDIFACT
Petición
Respuesta
CTXK
C 5
BLB
M1
REF
M 1
ATR
M 1
GERR
C 1
RAD
C 1
ERC
M 1
TXF
C 1

Fig. 8a

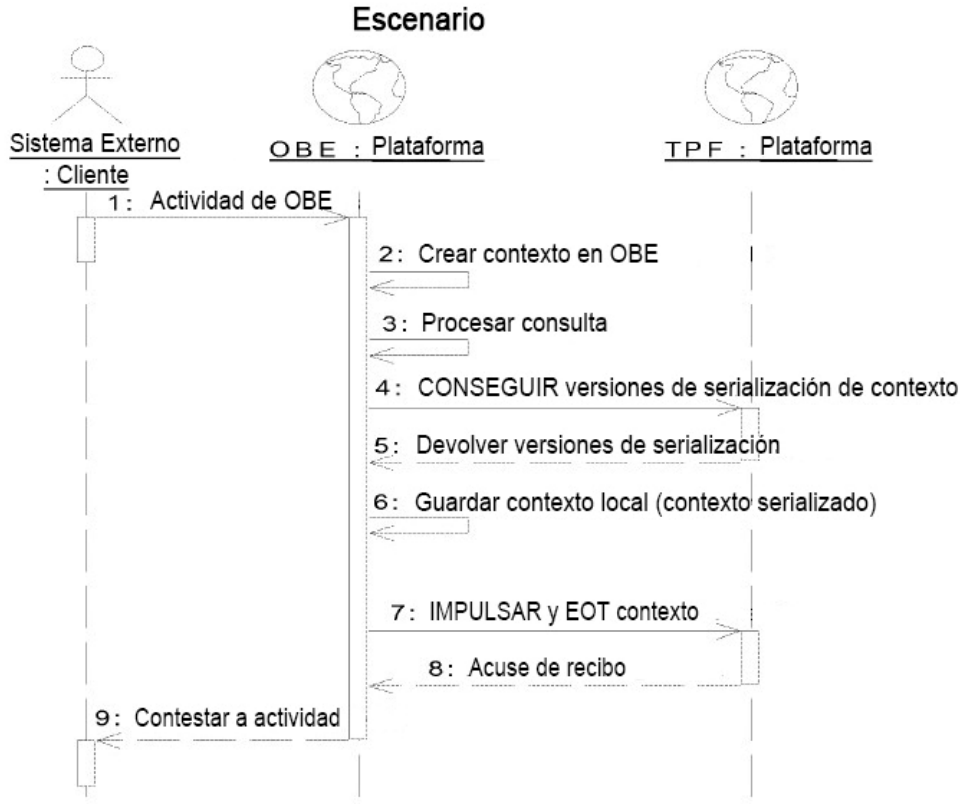
Escenario
Sistema Externo
: Cliente
OBE : Plataforma
TPF : Plataforma
1: Actividad de OBE
2: Crear contexto en OBE
3: Procesar consulta
4: CONSEGUIR versiones de serialización de contexto
5: Devolver versiones de serialización
6: Guardar contexto local (contexto serializado)
7: IMPULSAR y EOT contexto
8: Acuse de recibo
9: Contestar a actividad

Fig. 8b

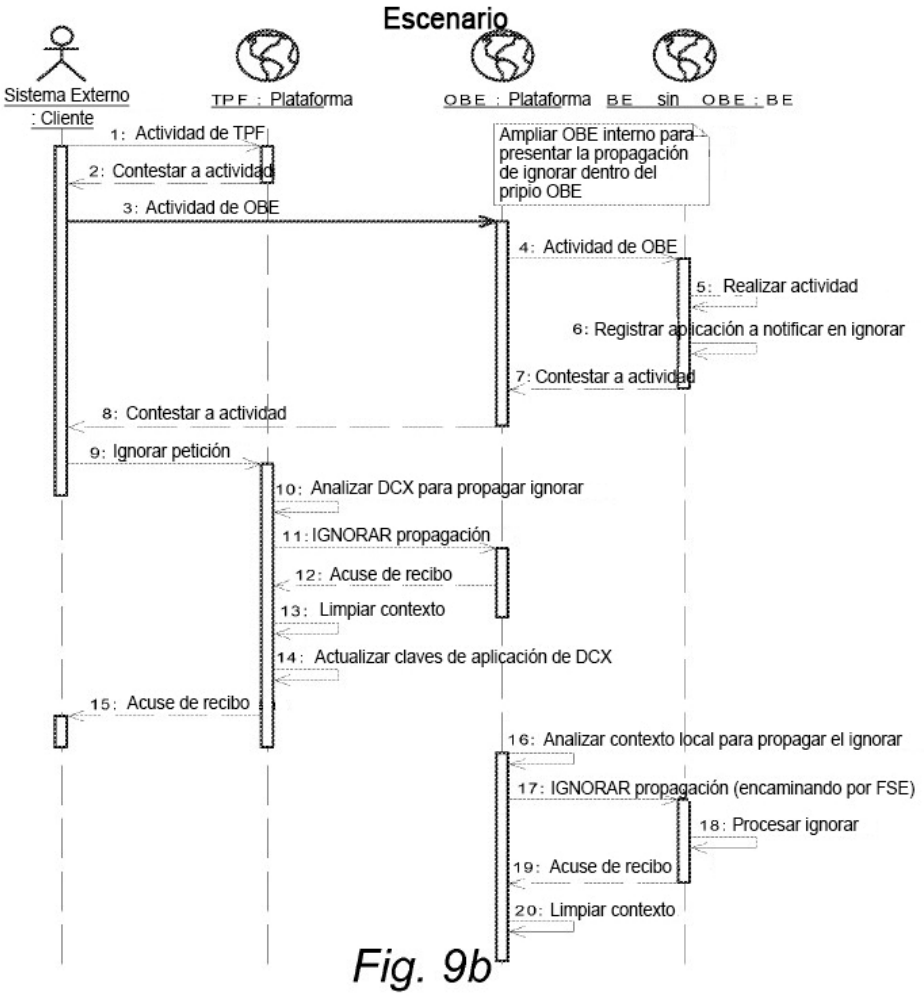
Escenario
Sistema Externo
: Cliente
TPF : Plataforma
OBE : Plataforma
BE sin OBE : BE
1: Actividad de TPF
2: Contestar a actividad
3: Actividad de OBE
Ampliar OBE interno para presentar la propagación de ignorar dentro del pripio OBE
4: Actividad de OBE
5: Realizar actividad
6: Registrar aplicación a notificar en ignorar
7: Contestar a actividad
8: Contestar a actividad
9: Ignorar petición
10: Analizar DCX para propagar ignorar
11: IGNORAR propagación
12: Acuse de recibo
13: Limpiar contexto
14: Actualizar claves de aplicación de DCX
15: Acuse de recibo
16: Analizar contexto local para propagar el ignorar
17: IGNORAR propagación (encaminando por FSE)
18: Procesar ignorar
19: Acuse de recibo
20: Limpiar contexto

Fig. 9b

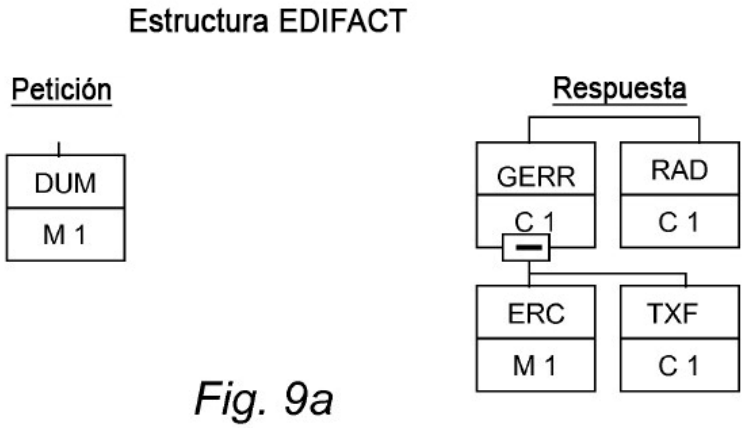
Estructura EDIFACT
Petición
DUM
M 1
Respuesta
GERR
C 1
RAD
C 1
ERC
M 1
TXF
C 1

Fig. 9a

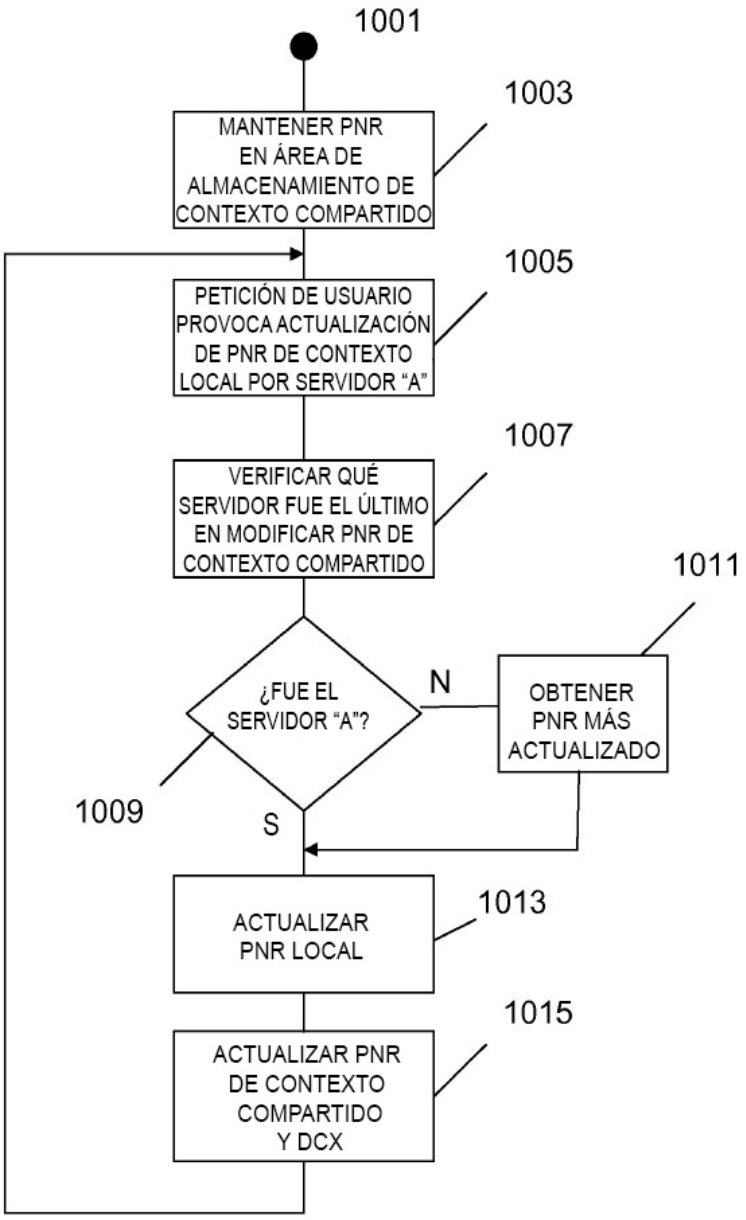
1001
1003
MANTENER PNR EN ÁREA DE ALMACENAMIENTO DE CONTEXTO COMPARTIDO
1005
PETICIÓN DE USUARIO PROVOCA ACTUALIZACIÓN DE PNR DE CONTEXTO LOCAL POR SERVIDOR "A"
1007
VERIFICAR QUÉ SERVIDOR FUE EL ÚLTIMO EN MODIFICAR PNR DE CONTEXTO COMPARTIDO
1011
¿FUE EL SERVIDOR "A"?
N
OBTENER PNR MÁS ACTUALIZADO
1009
S
ACTUALIZAR PNR LOCAL
1013
1015
ACTUALIZAR PNR DE CONTEXTO COMPARTIDO Y DCX

Fig. 10