Documentación del programa TurkeyeArchive

Contenido

[Resumen de funcionamiento 2](#_Toc434397958)

[Clases que componen la solución 3](#_Toc434397959)

[Main 3](#_Toc434397960)

[DataLayer 3](#_Toc434397961)

[Cbl\_fact\_turquia 3](#_Toc434397962)

[Cbl\_fact\_turquia\_2363 3](#_Toc434397963)

[Cbl\_fact\_turquia\_det\_2364 3](#_Toc434397964)

[SendFile 3](#_Toc434397965)

[XmlInvoice 3](#_Toc434397966)

[Funcionamiento 4](#_Toc434397967)

# Resumen de funcionamiento

El programa está escrito en Java, y será ejecutado en el servidor AIX de producción.

Se planifica que se ejecutará mediante un programa COBOL nativo que realizará una llamada a un módulo COBOL administrado, que le pasará los parámetros de conexión a la clase Java. Para obtener los datos de conexión, el módulo COBOL administrado llamará al módulo DbUsuPass escrito en COBOL nativo.

Finalmente, se ha creado BASE0190 (COBOL nativo) que llama a BASE0188 (COBOL administrado). Este último se encarga de llamar a DbUsuPass para obtener las credenciales de conexión a la BD y pasárselas luego a TurkeyeArchive.jar.

El programa hará un fetch de todos las filas de la tabla Cbl\_fact\_turquia\_2363 con ESTADO\_2363 = blanco, y sus líneas relacionadas en Cbl\_fact\_turquia\_det\_2364. Esta información la almacenará en un array en memoria, que será explotada por diversos threads que arrancarán simultáneamente, e invocarán a la función “sendEArchiveXml2” para generar la factura correspondiente. Previo a cada llamada, se cambiará el estado de la columna ESTADO\_2363 a “P” (Pendiente). Una vez finalizada la generación, se cambiará el valor de ESATADO\_2363 a “F”, se actualizarán las columnas URL\_FACTURA\_2363 e ID\_FACTURA\_2363 con los datos devueltos por la llamada a la función del WebService, y se actualizará la fila en la BD.

Si deseamos detener el programa de forma controlada, bastará crear el archivo $VOL/EXPL/TMP/TurkeyeArchive.stop. El programa detendrá todas las transacciones y eliminará el archivo.

# Clases que componen la solución

## Main

Clase principal donde se define el punto de entrada. En esta clase se recuperan los registros de la BD y se crean los threads que procesaran cada una de las filas de las facturas pendientes.

## DataLayer

Clase para ejecutar sentencias en la BD.

## Cbl\_fact\_turquia

Representa una factura. Se compone de una instancia de **Cbl\_fact\_turquia\_2363** (cabeceras) y varias instancias de **Cbl\_fact\_turquia\_det\_2364** (líneas).

Contiene una función para crear en modo DEBUG una muestra lógica de una factura: **createSample(int)**.

## Cbl\_fact\_turquia\_2363

Cabecera de una factura. Además de los datos de la cabecera, instancia una lista con sus líneas (**Cbl\_fact\_turquia\_det\_2364**).

Contiene también una función para crear una muestra específica de líneas (llamada por **Cbl\_fact\_turquia.createSample(int)** en modo DEBUG).

## Cbl\_fact\_turquia\_det\_2364

Representa una línea de factura. Instanciada por **Cbl\_fact\_turquia\_2363**.

## SendFile

Esta clase es la que ejecuta la llamada al webservice. Se instancia como un thread (en realidad, varios threads concurrentes), y trabaja con las facturas almacenadas en *SendFile.arCbl\_fact\_turquia*.

## XmlInvoice

Representa el Xml de la factura. Contiene una función para crear el Xml y otra para parsearlo.

# Funcionamiento

**Main.Runit()**

Punto de entrada.

Instancia la matriz *SendFile.arCbl\_fact\_turquia* (matriz compartida por todos los threads ejecutados en la clase **SendFile**) y llama a la función **Main.getAllCbl\_fact\_turquia()**. Esta rutina deberá recuperar todas las facturas de la BD y almacenarlas en esa matriz.

Finalmente, instancia la clase *oStub* y crea los threads configurados en NUM\_THREADS y lanza la función **SendFile()** (pasándole la instancia de oStub, la conexión a la BD y el número de thread) por cada uno de ellos.

**Main.getAllCbl\_fact\_turquia()**

Recuperar todas las facturas de la BD.

En modo DEBUG, lo que hace es llamar a la función **Main.createListSample()** y a **DataLayer.createSamples()**.

**Main.createListSample()**

Se encarga de crear registros ficticios llamando a **Cbl\_fact\_turquia.createSample(int)** (donde *int* representa el número de muestra que se crea). Cada una de las filas creadas se añade a la matriz *SendFile.arCbl\_fact\_turquia*.

**DataLayer.createSamples()**

Se encarga de crear en la BD los registros de pruebas con los que trabajaremos, obteniéndolos de *SendFile.arCbl\_fact\_turquia*.

**SendFile()**

Implementa la función run(), que es ejecutada automáticamente por el thread.

**SendFile.run()**

Recorre la matriz *SendFile.arCbl\_fact\_turquia* para procesar todas las facturas pendientes (ESTADO\_2364 = BLANCO). Para ello, por cada elemento de la matriz instancia *Cbl\_fact\_turquia*.

**SendFile.callWebService()**

Llama a oStub.sendEArchiveXml2

# Pendientes

Errors i’ve seen in the xml file as below;

1. ID (Invoice Number) has to be in Ministry of Finance Format like ABC2015000000001.
2. UUID has to be a uniqueidentifier(GUID) type
3. Note is missing. It has to contain the amount in words.
4. DocumentCurrencyCode has to be TRY if Turkish Lira or codes in ISO 4217.
5. There are more InvoiceLine’s as stated in LineCountNumeric as 1.
6. You need to add below section for Shipping Method (“ELEKTRONIK” for electronic, “KAGIT” for paper in DocumentType)

<AdditionalDocumentReference>

                                                       <ID>gonderimSekli</ID>

                                                       <IssueDate>2015-09-04</IssueDate>

                                                       <DocumentType>ELEKTRONIK</DocumentType>

                                       </AdditionalDocumentReference>

1. In AccountingSupplierParty, PartyIdentification schemeID="VKN" value has to be a 10 digit number
2. In AccountingSupplierParty, PostalAddress has to contain BuildingNumber, CitySubdivisionName, CityName, PostalZone, Country
3. In AccountingSupplierParty, PartyTaxScheme section TaxScheme should not contain TaxTypeCode
4. In AccountingSupplierParty, Contact and sub-elements are missing.
5. In AccountingCustomerParty, PostalAddress has to contain BuildingNumber, CitySubdivisionName, CityName, PostalZone, Country
6. In AccountingCustomerParty, PartyTaxScheme section TaxScheme should not contain TaxTypeCode
7. In AccountingCustomerParty, Contact and sub-elements are missing.
8. In InvoiceLine, AllowanceCharge has to contain MultiplierFactorNumeric, Amount, BaseAmount
9. In InvoiceLine, TaxTotal has to contain TaxSubtotal section and sub-elements.
10. In InvoiceLine, Item Description and Name should be sub-elements instead  of attributes.

You can compare the XML you’ve generated againts the sample in the documentation. These items needs to be fixed for sending a successful invoice xml.

DB-EXTRN-STRING-CONN=SGA\_OWNER/SGA2015@DBDMNGP

# Registro de logs (con fecha en nombre)

* Facturas no OK
* Facturas con error (null)
* Suma de OK
* Suma de KO
* Suma de error (null)
* Hora de inicio
* Hora de fin
* Tiempo de fetch de BD y número de filas cargadas
* Tiempo de procesamiento (llamadas al web service)
* Excepciones

Campos a incluir en la tabla de logs:

* Level (INFO, WARNING, ERROR, etc)
* Usuario
* Tty
* Ip
* Fecha
* Hora
* Texto

# Envío de emails

* Si hay KO o errores, enviar un email con la relación a [eqp.base@mango.com](mailto:eqp.base@mango.com)