

# **Causa raíz NO se envía el folio de orden en setInicioTrans2 hacia Origis**

**Autor: Huilver Nolasco Aguilar**

**Fecha: Marzo del 2025**

## Contenido

<b>Descripción general.....</b>	<b>3</b>
<b>Causa.....</b>	<b>3</b>
<b>Solución.....</b>	<b>10</b>

## Descripción general

Recientemente se agregó un nuevo método en origis llamado setInicioTrans2 el cual soporta nuevos campos relacionados con la orden, en el caso particular del folio de orden el dato que debe mandarse es el campo sPedidoNumero, sin embargo se detectó que para algunas ventas que son de SAD, este dato viaja en 0 y para otras dicho dato si viaja correctamente.

## Causa

La venta que se tomó para análisis es la venta V10982000088983 que corresponde al pedido 7007373420APP.

Lo primero que se hace es revisar los datos generales del pedido, así como sus formas de pago

--consulta de cabecero de venta					
select idTransaccion,fechaTransaccion,sadOrderId from trnTransaccionesCab where folioTransaccion='V10982000088983';					
idTransaccion	fechaTransaccion	sadOrderId			
596E46EF-5EE2-EF11-95ED-10E7C6374B88	2025-02-03 12:44:48.247	FF719F1B-87CF-4FC4-892B-83CE31D0D617			
--cabecero de orden asociado a la venta					
select sadOrderId,ordenFolio,codigoFormaPago,monedero from sadOrdenes					
where sadOrderId = (select sadOrderId from trnTransaccionesCab where folioTransaccion='V10982000088983');					
sadOrderId	ordenFolio	codigoFormaPago	monedero		
FF719F1B-87CF-4FC4-892B-83CE31D0D617	7007373420APP	41	9690000002657		
--formas de pago de la orden					
select sadOrderId,fp.codigoFormaPago,fpcat.descripcion,fpcat.tipoFormaPago,monedero,referenciaTarjeta from sadCentralOrdenFormasPago fp					
join genFormasDePagoCat fpcat [l..n->1] on fp.codigoFormaPago=fpcat.codigoFormaPago					
where fp.sadOrderId=(select sadOrderId from trnTransaccionesCab where folioTransaccion='V10982000088983');					
sadOrderId	codigoFormaPago	descripcion	tipoFormaPago	monedero	referenciaTarjeta
FF719F1B-87CF-4FC4-892B-83CE31D0D617	41	PAYPAL	SAD3		

Como se puede observar en la imagen anterior esta venta corresponde a una orden que fue pagada con PAYPAL(41) y por lo tanto es correcto que el campo **monedero y referencia tarjeta no tengan dato**

Despues se procede a corroborar que lo reportado en el log de setInicioTrans2 de origis contra el log de monedero que se tiene para estar seguro que se trata del mismo caso.

setInicioTrans2	sTarjeta=9690000002657, sSucId=1098, sCajalId=20, sTerminalId=175327, sOrigen=2, <b>sPedidoNumero=0</b> , sReferenciaNumero=0, sCarritoNumero=0, sAuthCode=d98c9ff72fb6dbd0843d61e81bcd2bd
-----------------	---

Dado que en la operación de commitTrans se registra de ser el caso el número de monedero, el folio de autorización y el número de venta, con esto se puede corroborar que se trata del mismo caso

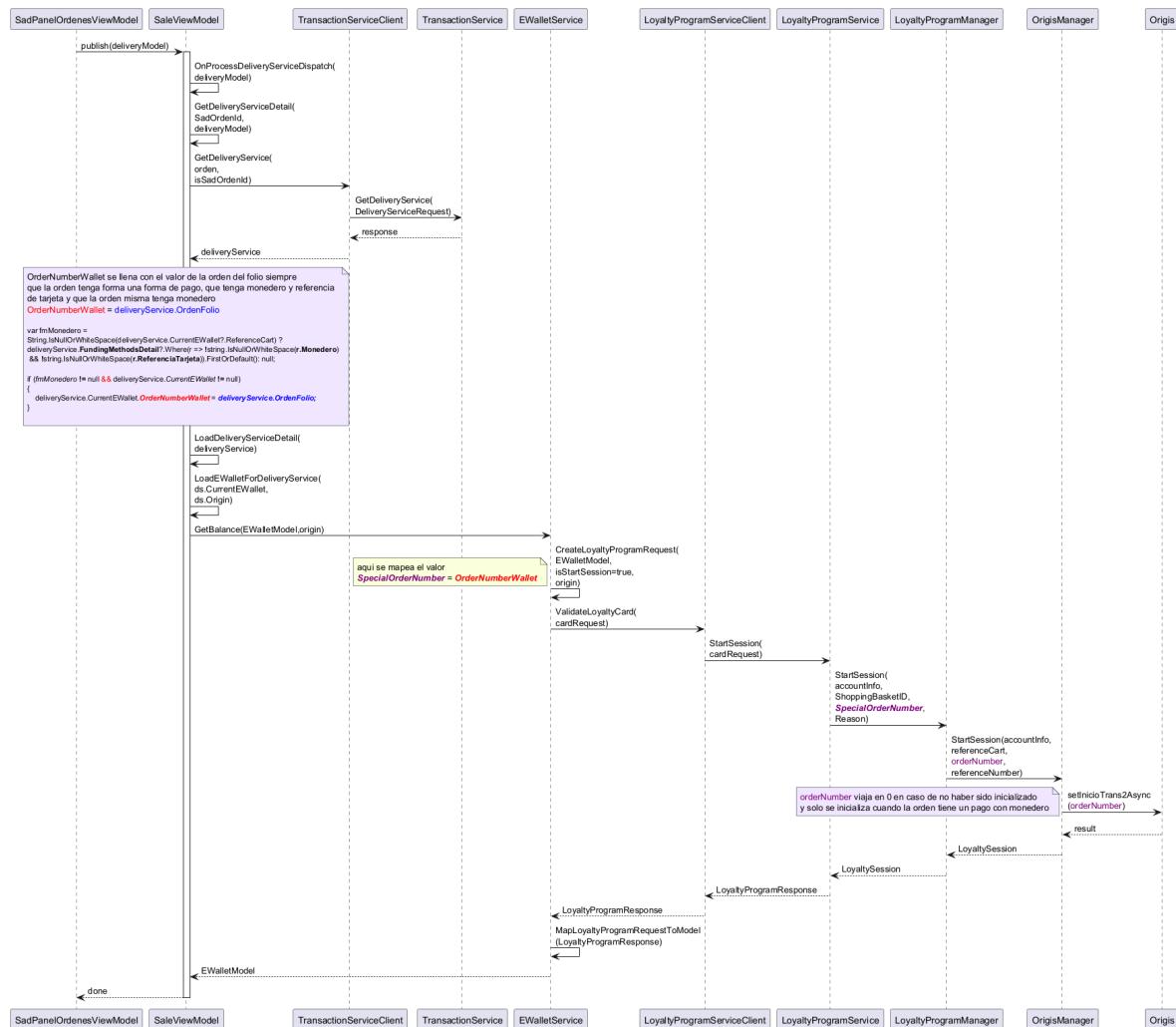
```
[2025-02-03 12:42:47:236],FARMAX_WSC,MONEDERO,INFO,,Request:<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Header>
<Action s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">https://pos.monederodelahorro.net/online2/commitTrans</Action>
</s:Header>
<s:Body xmlns:s="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<commitTrans xmlns="https://pos.monederodelahorro.net/online2/">
<sTarjeta>9690000002657</sTarjeta>
<sSucId>1098</sSucId>
<sCajaId>20</sCajaId>
<sTerminalId>175327</sTerminalId>
<sSessionId>14352336926</sSessionId>
<sArticulos>7501298289067,1,747,747|0650240057946,2,22,90002400,45,80004800|7506472801310,1,10,0000,10,0000</sArticulos>
<sNumTicket>V10982000088983</sNumTicket>
<sSaldoRedimido>0</sSaldoRedimido>
<sImporteTotal>802,0000</sImporteTotal>
<sFechaOperacion>20250203</sFechaOperacion>
<sEntregaDomicilio>2</sEntregaDomicilio>
<sFormaPago>0</sFormaPago>
<sAuthCode>d98c9ff72fb6dbd0843d61e81bcd2bd</sAuthCode>
</commitTrans>
</s:Body>
</s:Envelope>
```

Se observa que el número de tarjeta 9690000002657 y el numero de autorizacion d98c9ff72fb6dbd0843d61e81bcd2bd coinciden, por lo tanto la venta V10982000088983 es la correcta para el análisis.

A continuación se describe de manera general cómo es que con una orden SAD se interactúa con Origis enfocado al método setInicioTrans2 .

Despues de que la orden ha llegado a la sucursal para su atención, dicha orden puede ser visualizada en el panel de órdenes de las cajas, una vez que se atiende la orden esta debe generar una comanda pero no sin antes haber obtenido la información de la orden, es decir su información general, el detalle de productos así como sus formas de pago. Despues de obtener dicha información esta se coloca dentro del flujo del POS para la captura de productos y su despacho, justo en ese mapeo de los datos de la orden hacia los datos de la venta y al ejecutar la comanda, se dispara el inicio de sesión con origis y es ahí donde el número de pedido(folio orden) se notifica, sin embargo en el mapeo establecido entre la orden y la venta asociada, el número de pedido se mapea solo bajo ciertas condiciones que básicamente son, que la orden tenga un monedero asociado y que tenga una forma de pago mda, si esto no se cumple la consecuencia es que termina mandando 0 por defecto.

Se agrega un diagrama de secuencia para tener más detalle:



La imagen original se puede ver aquí  [sad\\_enviofolioorden\\_setiniciotrans.png](#)

En código los lugares claves son

1.-Generación de comanda de la orden

Code editor showing a C# file with code related to order processing. The code handles different order statuses (P, I, D) and performs actions like publishing events and closing the application.

```

354
355
356 >     /// Procesa la acción para el botón de acción 1 de las órdenes ...
357
358     public void ProcessButton1Action(SadOrderDisplayModel order)
359     {
360         switch (order.Status)
361         {
362             // orden de servicio pendiente
363             case "P":...
364
365             // orden de servicio con comanda impresa, y pendiente de despachar
366             case "I":
367                 if (order.OrderType == SADORDENES.TIPO_ORDEN_VALUES.SERVICIO_DOMICILIO_PROGRAMADA
368                     && DateTime.Now.AddMinutes(PosApplicationContext.Instance.SystemConfiguration.TiempoDespachoOrdenProgramada) < order.OrderScheduleDateTime){...}
369             else if (order.OrderType == SADORDENES.TIPO_ORDEN_VALUES.SERVICIO_DOMICILIO && order.ContainsVouchers){...}
370
371             this._eventAggregator.GetEvent<ProcessDeliveryServiceDispatchEvent>().Publish((new DeliveryServiceModel() { SadOrdenId = order.OrderID, Origin = order.OriginDisplay }));
372             this.Close();
373
374             break;
375
376             // orden de servicio en despacho
377             case "D":...
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396

```

## 2.-Inicia la atención de la orden

Code editor showing a C# file with code related to order processing. The code handles the OnProcessDeliveryServiceDispatch event, checking if there are sales details and then clearing the sale view model and loading delivery service details.

```

10385     /// </summary>
10386     /// <param name="deliveryModel">identificador del SAD</param>
10387     [Usage & Juan Murillo]
10388     private void OnProcessDeliveryServiceDispatch(DeliveryServiceModel deliveryModel)
10389     {
10390         var confirmed = this.Sale.SaleDetailList.Count > 0;
10391         if (confirmed){...}
10401         else{...}
10406
10407         if (confirmed)
10408         {
10409             this.ClearSaleVm();
10410             this.LoadDeliveryServiceDetail(this.GetDeliveryServiceDetail(deliveryModel.SadOrdenId.ToString(), isSadOrdenId: true, deliveryModel.Origin));
10411         }

```

## 3.-Se consulta la información de la orden mediante la solicitud de un servicio de till service

Code editor showing a C# file with code related to order processing. The code defines a private method GetDeliveryServiceDetail that uses a TransactionServiceClient to get the delivery service detail for a given order ID.

```

9964     private DeliveryServiceModel GetDeliveryServiceDetail(string orden, bool isSadOrdenId = false, string origin = "")
9965     {
9966         ...
9970
9971         try
9972         {
9973             using (var service = new TransactionServiceClient())
9974             {
9975                 deliveryService = service.GetDeliveryService(orden, isSadOrdenId);
9976                 deliveryService.Origin = origin;
9977             }
9978         }
9979         return deliveryService;

```

#### 4.-Se consulta el servicio

```
C# ITransactionPayment    C# BankTender    C# GenericTender    C# ILoyaltyProgramService    C# SadPanelOrdenesViewModel    C# TransactionServiceClient.partial

81 >     /// Metodo que obtiene una orden de servicio ...
82  IUsage & Juan Murillo
83     public DeliveryServiceModel GetDeliveryService(string orden, bool isSadOrdenId = false)
84     {
85         ...
86         DeliveryServiceResponse response = this.Useservice((ITransactionService transactionService) => transactionService.GetDeliveryService(request));
87
88         return this.MapDeliveryServiceResponse(response);
89     }
90
91     private void MapDeliveryServiceResponse(DeliveryServiceResponse response)
92     {
93         ...
94     }
95 }
```

5.-Se hace el mapeo de los datos de la orden hacia un response, haciendo notar el mapeo relevante de los datos que más adelante se usan, por ejemplo está el monedero de la forma de pago de la orden, así como su referencia de tarjeta y los datos del monedero de la orden en el currentEwallet



The screenshot shows a .NET IDE interface with a code editor containing C# code. The code is part of a class named `DeliveryServiceModel` and is responsible for mapping a response from the `DeliveryService` to a `DeliveryServiceModel`. The code uses `foreach` loops to iterate through lists of `DeliveryServiceFundingMethod` objects and map them to `DeliveryServiceFundingMethodModel` objects. It also handles the `Monedero` and `ReferenciaTarjeta` fields. The code editor includes features like syntax highlighting, code completion, and navigation bars at the top.

```
private DeliveryServiceModel MapDeliveryServiceResponse(DeliveryServiceResponse response)
{
    var orden = response.DeliveryServices[0];

    string orderNumberWallet = orden.OrdenFolio;
    foreach (DeliveryServiceFundingMethod fm in orden.DeliveryServiceFundingMethods ?? new List<DeliveryServiceFundingMethod>())
    {
        var item = new DeliveryServiceFundingMethodModel
        {
            ...
            Monedero = fm.Monedero,
            ...
            ReferenciaTarjeta = fm.ReferenciaTarjeta,
        };

        dsm.FundingMethodsDetail.Add(item);
    }

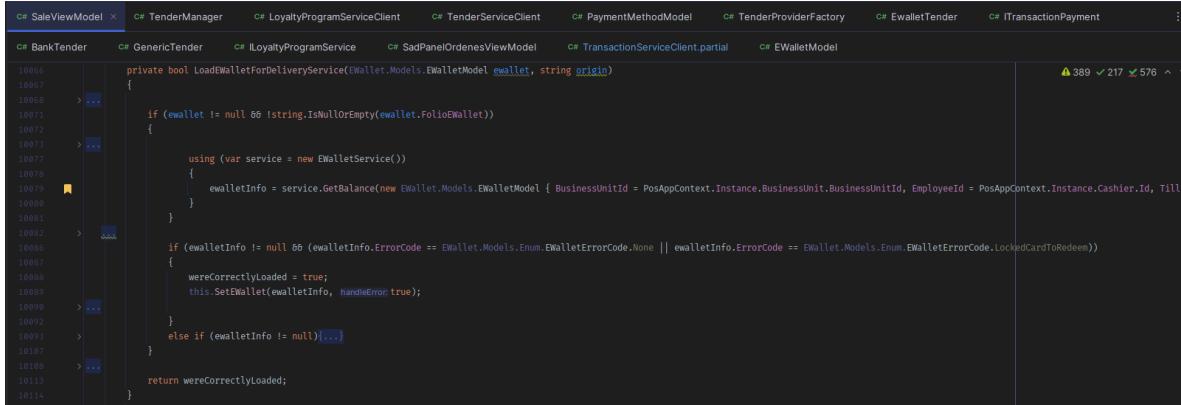
    if (!string.IsNullOrEmpty(orden.Monedero))
    {
        dsm.CurrentEWallet = new EWallet.Models.EWalletModel() { FolioEWallet = orden.Monedero, ReferenceCart = referenceCart, OrderNumberWallet = orderNumberWallet, ReferenceNumberWallet = referenceNumberWallet };
    }
}

return dsm;
}
```

6.-Se cargan los datos de la orden para que sea procesada por el POS, en donde se puede observar que el llenado de OrdenNumberWallet depende de que exista una forma de pago de la orden que tenga monedero y que tenga referenciaTarjeta y que la orden tenga los datos del monedero en currentEwallet. Sin embargo la Orden en cuestión fue pagada con PayPal y **su forma de pago no tiene monedero ni referencia de tarjeta** así que el dato OrdenNumberWallet **no se llena**

```
9991     private void LoadDeliveryServiceDetail(DeliveryServiceModel deliveryService)
9992     {
9993         if (deliveryService == null){...}
9994
9995         // Si no hay numero de carrito buscamos si en la forma de pago de monedero se cuenta con alguno
9996         var fmMonedero :DeliveryServiceFundingMethodModel = String.IsNullOrEmptyOrWhiteSpace(deliveryService.CurrentWallet?.ReferenceCart) ?
9997             deliveryService.FundingMethodsDetail?.Where(r => r.DeliveryServiceFundingMethodModel != null && !string.IsNullOrEmptyOrWhiteSpace(r.Monedero) && !string.IsNullOrEmptyOrWhiteSpace(r.ReferenciaTarjeta));
9998
9999         if (fmMonedero != null && deliveryService.CurrentWallet != null)
10000         {
10001             deliveryService.CurrentEWallet.OrderNumberWallet = deliveryService.OrdenFolio;
10002         }
10003
10004         if (Ithis.LoadEWalletForDeliveryService(deliveryService.CurrentEWallet, deliveryService.Origin))
10005         {
10006             Log.Write(string.Format("Servicio a domicilio FOLIO: {0}, no se logro procesar monedero", deliveryService.OrdenFolio));
10007         }
10008
10009         this.Sale.CurrentDeliveryService = deliveryService;
10010     }
10011
10012 }
```

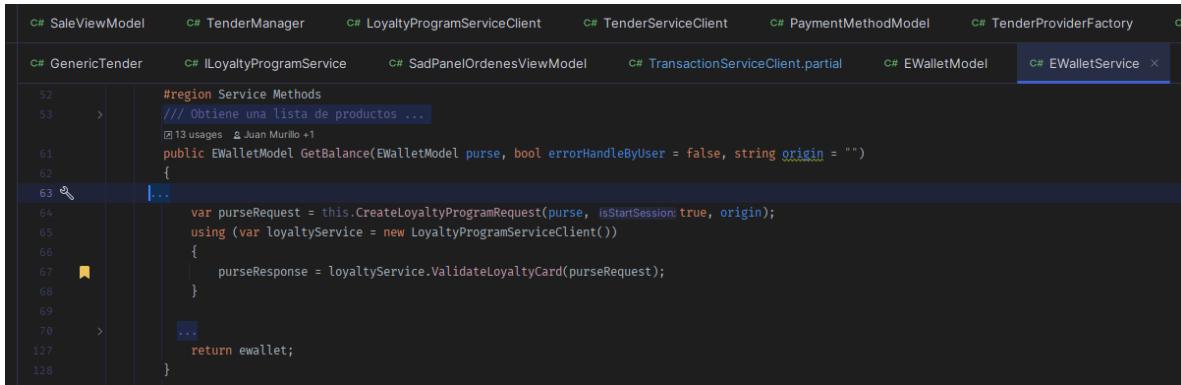
7.-Se consultan los datos del monedero hacia Origis en caso de que se cuente con el monedero, para despues colocar dicha informacióen en el POS



```

10866 private bool LoadEWalletForDeliveryService(EWallet.Models.EWalletModel ewallet, string origin)
10867 {
10868     > ...
10869     if (ewallet != null && !string.IsNullOrEmpty(ewallet.FolioEWallet))
10870     {
10871         > ...
10872         using (var service = new EWalletService())
10873         {
10874             ewalletInfo = service.GetBalance(new EWallet.Models.EWalletModel { BusinessUnitId = PosApplicationContext.Instance.BusinessUnit.BusinessUnitId, EmployeeId = PosApplicationContext.Instance.Cashier.Id, TillId = PosApplicationContext.Instance.Till.TillId });
10875         }
10876         if (ewalletInfo != null && (ewalletInfo.ErrorCode == EWallet.Models.Enum.EWalletErrorCode.None || ewalletInfo.ErrorCode == EWallet.Models.Enum.EWalletErrorCode.LockedCardToRedeem))
10877         {
10878             wereCorrectlyLoaded = true;
10879             this.SetEWallet(ewalletInfo, handleError: true);
10880         }
10881         > ...
10882         else if (ewalletInfo != null) {...}
10883     }
10884     > ...
10885     return wereCorrectlyLoaded;
10886 }
10887
10888
10889
10890
10891
10892
10893
10894
10895
10896
10897
10898
10899
10900
10901
10902
10903
10904
10905
10906
10907
10908
10909
10910
10911
10912
10913
10914

```

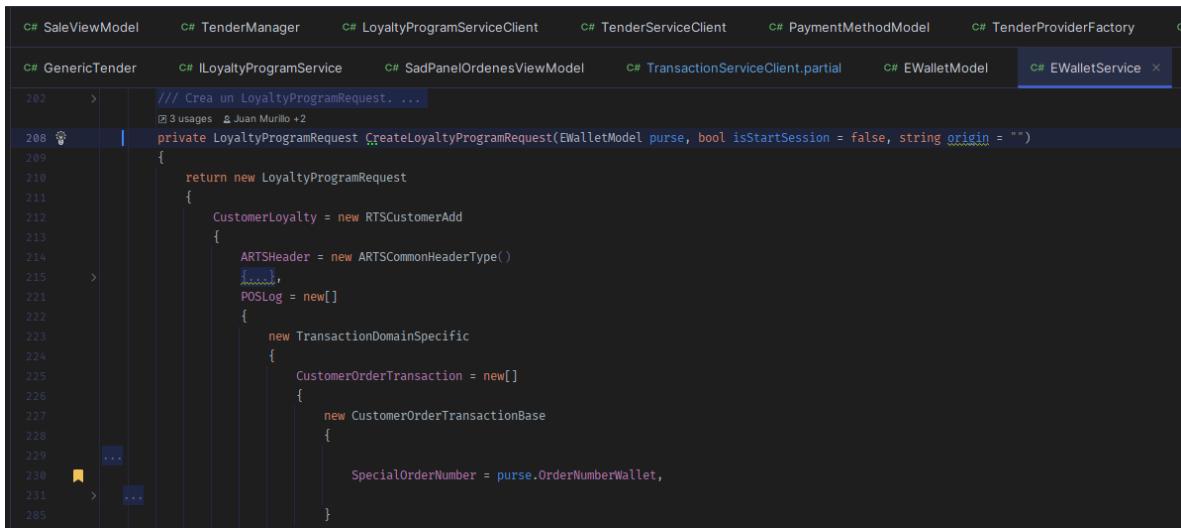


```

52     #region Service Methods
53     /// Obtiene una lista de productos ...
54     [3 usages & Juan Murillo +1]
55     public EWalletModel GetBalance(EWalletModel purse, bool errorHandleByUser = false, string origin = "")
56     {
57         ...
58         var purseRequest = this.CreateLoyaltyProgramRequest(purse, isStartSession: true, origin);
59         using (var loyaltyService = new LoyaltyProgramServiceClient())
60         {
61             purseResponse = loyaltyService.ValidateLoyaltyCard(purseRequest);
62         }
63         ...
64         return ewallet;
65     }
66
67     > ...
68
69
70     > ...
71
72     127
73     128

```

Pero antes de se llamar al servicio se genera el request mediante un mapeo en el que ahora el OrdenNumberWallet es depositado en SpecialOrdenNumber



```

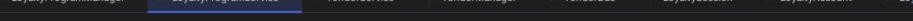
202     /// Crea un LoyaltyProgramRequest. ...
203     [3 usages & Juan Murillo +2]
204     private LoyaltyProgramRequest CreateLoyaltyProgramRequest(EWalletModel purse, bool isStartSession = false, string origin = "")
205     {
206         return new LoyaltyProgramRequest
207         {
208             CustomerLoyalty = new RTSCustomerAdd
209             {
210                 ARTSHeader = new ARTSCurrentHeaderType()
211                 {
212                     ...
213                 },
214                 POSLog = new[]
215                 {
216                     new TransactionDomainSpecific
217                     {
218                         CustomerOrderTransaction = new[]
219                         {
220                             new CustomerOrderTransactionBase
221                             {
222                                 ...
223                                 SpecialOrderNumber = purse.OrderNumberWallet,
224                             }
225                         }
226                     }
227                 }
228             }
229         };
230     }
231
232     > ...
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285

```

El servicio que se termina invocando es StartSession de la clase LoyaltyProgramServiceClient

```
 27     }
 28
 29 >     /// Valida la existencia de un monedero. (startSession) ...
 30     2 usages  ▲ Juan Murillo
 31     public LoyaltyProgramResponse ValidateLoyaltyCard(LoyaltyProgramRequest cardRequest)
 32     {
 33         return this.UserService((ILoyaltyProgramService loyaltyProgramService) => loyaltyProgramService.StartSession(cardRequest).Result);
 34     }
 35
 36
 37
 38
 39
```

8.-Se inicia la sesión con origis mandando el dato SpecialOrderNumber que debería contener el folio de la orden

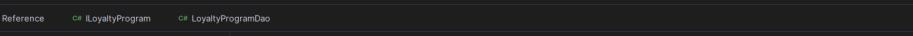


```
    /// <returns>
    [ExceptionHandlingService]
    [Sql100 usages = 1 Juan Muriel = 3]
    public async Task<LoyaltyProgramResponse> StartSession(LoyaltyProgramRequest cardRequest)
    {
        if (isSessionStart)
        {
            valid = await manager.StartSession(accountInfo, referenceCart, customerOrderTransaction.ShoppingBasketID, customerOrderTransaction.SpecialOrderNumber, referenceNumber, customerOrderTransaction.ReasonCode);
        }
    }
}
```

```
C# OrigisManager          C# LoyaltyProgramManager x C# LoyaltyProgramService      C# TenderService      C# TenderManager      C# TenderDao      C# LoyaltySession      C# LoyaltyAccount      C# LoyaltyBenefits
C# FactoryInstance        C# NovartisManager      C# Reference      C# ILoyaltyProgram      C# LoyaltyProgramDao

45     /// <param name="referenceNumber" >Número de referencia relacionado</param>
46     /// <returns>Regresa la información de la tarjeta validada en el objeto <see cref="LoyaltyAccount"/></returns>
47     public async Task<ResultStruct<LoyaltySession>> StartSession(LoyaltyAccount accountInfo, string referenceCart = null, string orderNumber = null, string referenceNumber = null)
48     {
49         var loyalty = FactoryInstance.CreateInstance(service.accountInfo.LoyaltyProgramID + this.suffix);
50         return await loyalty.StartSession(accountInfo, referenceCart, orderNumber, referenceNumber);
    
```

9.-Se manda llamar el metodo setInicioTrans2Async de origis, se puede observar que en caso de venir nulo o vacío se manda por defecto 0, que como hemos visto anteriormente orderNumber=SpecialOrderNumber y para esta orden en particular viene lleno debido al punto 6



```
public override async Task<ResultStruct<LoyaltySession>> StartSession(LoyaltyAccount accountInfo, string referenceCart = null, string orderNumber = null, string referenceNumber = null)
```

```
try
{
    bool isSendOriginActivated = DeterminaEnvioOrigenVenta(accountInfo);

    using (var serviceClient = new Monedero_del_AhorroSoapClient(MON_ENDPOINT_NAME, _checkConnection: false, enableOriginLog: true))
    {
        result = await serviceClient.setInicioTrans2Async(accountInfo.Card, accountInfo.Branch, accountInfo.lillId, accountInfo.PayrollId, isSendOriginActivated ? accountInfo.Origin : OriginSale,
            accountInfo.IsNullOrWhiteSpace(referenceCart) ? "" : referenceCart, accountInfo.IsNullOrWhiteSpace(referenceNumber) ? "" : referenceNumber, accountInfo.IsNullOrWhiteSpace(referenceNumber) ? "" : referenceNumber);
    }
}
```

## Solución

Se propone cambiar en el método

FAhorro.POS.Win.Sale.ViewModels.SaleViewModel.LoadDeliveryServiceDetail la siguiente sección de código:

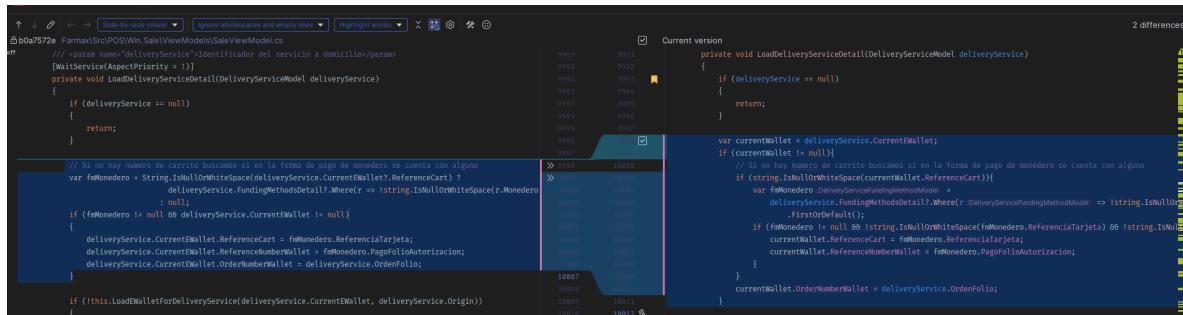
```
// Si no hay numero de carrito buscamos si en la forma de pago de monedero se cuenta con alguno
var fmMonedero = String.IsNullOrWhiteSpace(deliveryService.CurrentEWallet?.ReferenceCart) ?
    deliveryService.FundingMethodsDetail?.Where(r => !string.IsNullOrWhiteSpace(r.Monedero) &&
!string.IsNullOrWhiteSpace(r.ReferenceTarjeta)).FirstOrDefault()
    : null;
if (fmMonedero != null && deliveryService.CurrentEWallet != null)
{
    deliveryService.CurrentEWallet.ReferenceCart = fmMonedero.ReferenciaTarjeta;
    deliveryService.CurrentEWallet.ReferenceNumberWallet = fmMonedero.PagoFolioAutorizacion;
    deliveryService.CurrentEWallet.OrderNumberWallet = deliveryService.OrdenFolio;
}
```

Por esta otra:

```
var currentWallet = deliveryService.CurrentEWallet;
if (currentWallet != null){
    // Si no hay numero de carrito buscamos si en la forma de pago de monedero se cuenta con alguno
    if (string.IsNullOrWhiteSpace(currentWallet.ReferenceCart)){
        var fmMonedero =
            deliveryService.FundingMethodsDetail?.Where(r => !string.IsNullOrWhiteSpace(r.Monedero) &&
!string.IsNullOrWhiteSpace(r.ReferenceTarjeta))
            .FirstOrDefault();
        if (fmMonedero != null && !string.IsNullOrWhiteSpace(fmMonedero.PagoFolioAutorizacion) &&
!string.IsNullOrWhiteSpace(fmMonedero.PagoFolioAutorizacion)){
            currentWallet.ReferenceCart = fmMonedero.ReferenciaTarjeta;
            currentWallet.ReferenceNumberWallet = fmMonedero.PagoFolioAutorizacion;
        }
    }
    currentWallet.OrderNumberWallet = deliveryService.OrdenFolio;
}
```

que básicamente significa que coloques *OrderNumberWallet* solo condicionado a que *deliveryService.CurrentEWallet* no sea nulo y los datos *ReferenceCart* y *ReferenceNumberWallet* de *deliveryService.CurrentEWallet* se traten de llenar con los datos de la forma de pago si encuentra alguna que los tenga, pero solo si *ReferenceCart* de *deliveryService.CurrentEWallet* no trae dato

Se anexa comparativa del cambio propuesta vs la versión actual



The screenshot shows a code comparison tool with two panes. The left pane is labeled 'Current version' and the right pane is labeled 'Proposed solution'. The code is in C# and involves loading delivery service details. The proposed solution is more concise and uses null-coalescing operators to handle the null case for the current wallet's reference cart. The current version is more verbose, using a separate variable 'fmMonedero' and a 'FirstOrDefault' call. The differences are highlighted with yellow boxes and arrows, showing the removal of the 'fmMonedero' variable and the simplification of the assignment logic.