GET Technologies	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

Descripción general

El problema identificado se presenta en el proceso de devolución para una venta SAD que su venta fue con pago mixto: Tarjeta (Adyen, NetPay, PayPal) y MdA, los casos detectados que generan discrepancias en montos al devolver a la tarjeta son los siguientes.

1. Para una orden SAD que ha sido entregada, se devuelve el monto total de la venta a MdA y se genera una devolución adicional aplicada a la tarjeta con la cual se realizo la venta por un monto que se obtiene actualmente con la operación aritmética: **importe de la devolución - importe pagado de la venta con MdA.**

2. Para una orden SAD que ha sido entregada, se devuelve un monto menor al total de la venta a MdA y se genera una devolución adicional aplicada a la tarjeta con la cual se realizó la venta por un monto que se obtiene actualmente con la operación aritmética: **importe de la devolución - importe pagado de la venta con MdA**.

3. Para una orden SAD que ha sido entregada, se devuelve un monto menor al total de la venta solo a la tarjeta y si este valor es menor al importe pagado de la venta con MdA, se genera actualmente en la base de datos una devolución adicional con un monto negativo, pero no se aplica el pago a la tarjeta.

4. Para una orden SAD que ha sido entregada, se devuelve un monto menor al total de la venta solo a la tarjeta y si este valor es mayor al importe pagado de la venta con MdA, se genera una devolución adicional por un monto que se obtiene actualmente con la operación aritmética: **importe de la devolución - importe pagado de la venta con MdA**.

5. Para una orden SAD que NO ha sido entregada, se devuelve el monto total de la venta a MdA y se genera una devolución adicional aplicada a la tarjeta Adyen o PayPal con la cual se realizó la venta por un monto que se obtiene actualmente con la operación aritmética: **costo real de la orden SAD - importe pagado de la venta con MdA**.

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

Diagnóstico

El siguiente diagnóstico se realizó en agosto 2025, tener en cuenta que las capturas presentadas en esta sección son de ese tiempo, incluida la reproducción del problema, por lo que se presenta primero el diagnóstico realizado entonces y se aborda después el estado del código actualmente:

Los escenarios antes listados se presentan independientemente el tipo de tarjeta con la cual se pagó la venta, (Adyen, PayPal o NetPay) al realizar el cálculo aritmético que da como resultado el monto a devolver a la tarjeta. El diagnostico se realiza en distintas clases que se encuentran en el proyecto DEV_SadCentralLibrary.

Adyen y PayPal (orden SAD entregada)

La ejecución del proceso de la sucursal para las devoluciones de ordenes SAD que fueron su venta se pagó de forma mixta con Tarjeta Adyen y MdA o Tarjeta Paypal y MdA, se inicia en la clase StoreProcess, ejecutando el método Execute el cual a su vez ejecuta el método BusinessLogicReturns.ProcessSadReturns

```
try
{
    BusinessLogicReturns.ProcessSadReturns(_storeId,_tools,_client);
}
catch(Exception ex)
{
    Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2135,"Ocurrio un excepción en el proceso de devoluciones de ordenes. Excepción:{0}",ex.Message));
}
```

Ver en pantalla completa

Dentro del método BusinessLogicReturns.ProcessSadReturns se obtiene el monto pagado con MdA en la venta.

```
IList<SADORDENFORMASPAGOV2> dbfundingMethods = tools.ExecuteReportObjects<SADORDENFORMASPAGOV2>(new SADORDENFORMASPAGOV2() { SADORDENID = saddb.SADORDENID, CODIGOFORMAPAGO = 14 });

Decimal walletAlreadyPaid = 0;
if (dbfundingMethods != null)
{
    walletAlreadyPaid = dbfundingMethods.Where(r => r.CODIGOFORMAPAGO == 14).Sum(r => r.MONTO) ?? 0;
}
```

Ver en pantalla completa

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

Adyen (orden SAD entregada)

Dentro del método BusinessLogicReturns.ProcessSadReturns, si la venta fue pagada con Tarjeta Adyen se obtiene el encabezado de la devolución (VENDEVOLUCIONESCAB) y se genera un registro en la tabla “SADDEVOLUCIONESADYEN” en la base de datos Central, el monto a devolver se calcula actualmente con la operación aritmética: **Subtotal + IVA de la devolución - el importe pagado de la venta con MdA**. El problema reside que no se considera si ya existe un movimiento para la devolución con el importe completo de la venta o parcial.

```
if(ordendb.PAGADAECOMMERCE && GENFORMASDEPAGOCAT.CheckIsAdyenTender(ordendb.CODIGOFORMAPAGO))
{
    SADDEVOLUCIONESADYEN adyenDev = tools.ExecuteReportObject(new SADDEVOLUCIONESADYEN() { SADORDENID = ordendb.SADORDENID, IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

    if (adyenDev == null)
    {
        adyenDev = new SADDEVOLUCIONESADYEN();
        adyenDev.SADORDENID = ordendb.SADORDENID;
        adyenDev.IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES;
        adyenDev.CODIGOSUCURSAL = saddb.CODIGOSUCURSAL;
        adyenDev.REGISTROACTIVO = true;
        adyenDev.TIPODEVOLUCION = SADDEVOLUCIONESADYEN.TIPO_DEVOLUCION_VALUES.DEVOLUCION_TOTAL_CANCELACION;
        adyenDev.MONTO = Math.Round(ordendb.COSTOREALTOTAL - walletAlreadyPaid, 2);
        adyenDev.SYNC_TRANSFERIDO = "N";

        s.InsertObjectWithID(adyenDev);
    }
}
```

Ver en pantalla completa

Al encontrarse información en la tabla “SADDEVOLUCIONESADYEN”, se envían las transacciones a Adyen para aplicar el pago desde el método ProcessAdyenReturn de la clase BusinessLogicAdyen, el cual se ejecuta desde el método Execute de la clase StoreProcess.

```
try
{
    BusinessLogicAdyen.ProcessAdyenReturn(_storeId, _tools, _client);
}
catch (Exception ex)
{
    Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2218,"Ocurrio una excepción en el proceso de devoluciones/cancelaciones de ordenes Adyen. Excepción:{0}",ex.Message));
}
```

Ver en pantalla completa

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

PayPal (orden SAD entregada)

BusinessLogicReturns.ProcessSadReturns, si la venta fue pagada con Tarjeta PayPal se obtiene el encabezado de la devolución (VENDEVOLUCIONESCAB) y se genera un registro en la tabla SADDEVOLUCIONESPAYPAL en la base de datos Central, el monto a devolver se calcula actualmente con la operación aritmética: **Subtotal + IVA de la devolución - el importe pagado de la venta con MdA**. El problema reside que no se considera si ya existe un movimiento para la devolución con el importe completo de la venta o parcial.

```
if (ordendb.PAGADAECOMMERCE && GENFORMASDEPAGOCAT.CheckIsPayPalTender(ordendb.CODIGOFORMAPAGO))
{
    SADDEVOLUCIONESPAYPAL payPalDev = tools.ExecuteReportObject(new SADDEVOLUCIONESPAYPAL() { SADORDENID = ordendb.SADORDENID, IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

    if (payPalDev == null)
    {
        payPalDev = new SADDEVOLUCIONESPAYPAL();
        payPalDev.SADORDENID = ordendb.SADORDENID;
        payPalDev.IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES;
        payPalDev.CODIGOSUCURSAL = saddb.CODIGOSUCURSAL;
        payPalDev.REGISTROACTIVO = true;
        payPalDev.TIPODEVOLUCION = SADDEVOLUCIONESPAYPAL.TIPO_DEVOLUCION_VALUES.DEVOLUCION_TOTAL;
        payPalDev.MONTO = Math.Round(ordendb.COSTOREALTOTAL - walletAlreadyPaid, 2);
        payPalDev.SYNC_TRANSFERIDO = "N";

        s.InsertObjectWithID(payPalDev);
    }
}
```

Ver en pantalla completa

Al encontrarse información en la tabla SADDEVOLUCIONESPAYPAL, se envían las transacciones a PayPal para aplicar el pago desde el método ProcessPayPalReturn de la clase BusinessLogicPayPal. Este método se ejecuta desde el método Execute de la clase StoreProcess.

```
    try
    {
        BusinessLogicPayPal.ProcessPayPalReturn(_storeId, _tools, _client);
    }
    catch (Exception ex)
    {
        Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2267,"Ocurrio una excepción en el proceso de devoluciones de ordenes PayPal. Excepción:{0}",ex.Message));
    }
```

Ver en pantalla completa

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

NetPay (orden SAD entregada)

La ejecución del proceso para devoluciones de ordenes SAD cuya venta se pagó de forma mixta con NetPay y MdA, se inicia en la clase SadService generando un evento tipo notificación de la clase TransactionSADNotifyStatusV3.

```
server.AddCustomTransaction(TransactionIdentifiers.TransactionNotifyEvents, typeof(TransactionSADNotifyStatusV1), _tools,
    GlobalSettingsConfiguration.Int(GlobalSettingsConfiguration.Parameters.CONFIG_SAD_NOTIFY_EVENTS_QUEUE));

server.AddCustomTransaction(TransactionIdentifiers.TransactionNotifyEventsv2, typeof(TransactionSADNotifyStatusV2), _tools,
    GlobalSettingsConfiguration.Int(GlobalSettingsConfiguration.Parameters.CONFIG_SAD_NOTIFY_EVENTS_QUEUE));

server.AddCustomTransaction(TransactionIdentifiers.TransactionNotifyEventsv3, typeof(TransactionSADNotifyStatusV3), _tools,
    GlobalSettingsConfiguration.Int(GlobalSettingsConfiguration.Parameters.CONFIG_SAD_NOTIFY_EVENTS_QUEUE));
```

Ver en pantalla completa

Desde la clase TransactionSADNotifyStatusV3 se ejecuta el método “Process” el cual a su vez ejecuta el método RegisterOrderReturnNetPay que registrara en base de datos las devoluciones SAD para NetPay.

```
public override ProcessResult Process()
{
    ProcessResult rc = new ProcessResult();
    TX_NOTIFY_STATUS_REQUEST_V2 request = this.RequestObjects.Find(p => p is TX_NOTIFY_STATUS_REQUEST_V2) as TX_NOTIFY_STATUS_REQUEST_V2;
    TX_RETURN_INFO returnInfo = this.RequestObjects.Any(p => p is TX_RETURN_INFO) ? this.RequestObjects.Find(p => p is TX_RETURN_INFO) as TX_RETURN_INFO : null;

    if (request == null)
    {
        return rc.SetAndLog(SubModule, AxNLS.ErrorData(1823638247, 2064, "No se recibio el request TX_NOTIFY_STATUS_REQUEST"));
    }

    DatabaseTools tools = this.ProcessObject as DatabaseTools;


    if ((rc = RegisterNotification(tools, request)).IsError)
    {
        return rc;
    }



    if (returnInfo != null)
    {
        SADORDENES order = tools.ExecuteReportObject(new SADORDENES() { SADORDENID = returnInfo.OrderID });

        if (order == null)
        {
            return rc.SetAndLog(SubModule, AxNLS.ErrorData(1823638247, 2149, "La orden especificada en la notificación de devolución no fue encontrada. Orden:{0}", returnInfo.OrderID));
        }
        //HNA:FIXPAGOMDA Net Pay: 07-03-2025::Se tomaba el monto de a forma de pago de monedero pero de la tabla que vive en local y no en central y dado que no se sincroniza pues la tabla viene vacia
        //de ahi el problema a que el monto de monedero se toma como 0 y de ahi el calculo incorrecto
        var pagoMonedero = tools.ExecuteReportObject(new SADCENTRALORDENFORMASPAGOV2 { SADORDENID = returnInfo.OrderID, CODIGOFORMAPAGO = TENDER_ID_MONEDERO });

        Decimal walletAlreadyPaid = pagoMonedero?.MONTO ?? 0;

        if (order.REQUIERELIQUIDACION && order.REQUIERELIQUIDACIONNETPAY)
        {
            if ((rc = RegisterOrderReturnNetPay(tools, order, returnInfo, walletAlreadyPaid)).IsError)
            {
                return rc;
            }
        }
    }

    return rc;
}
```

Ver en pantalla completa

GET Technologies	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

Dentro del método RegisterOrderReturnNetPay se obtiene el encabezado de la devolución (VENDEVOLUCIONESCAB) para obtener el monto de la devolución, actualmente se realiza con la operación aritmética: **(Subtotal + IVA + comisión bancaria) – (descuento + ajuste devolución)**. Posteriormente se genera un registro en la tabla "SADCENTRALNETPAYDEVOLUCION" en la base de datos Central, el monto a devolver a la tarjeta el cual se calcula aritméticamente: monto de la devolución - el importe pagado de la venta con MdA. El problema reside que no se considera si ya existe un movimiento para la devolución con el importe completo de la venta o parcial.

```
SADCENTRALNETPAYLIQUIDACION liqdb = tools.ExecuteReportObject(new SADCENTRALNETPAYLIQUIDACION() { SADORDENID = returnInfo.OrderID });
VENDEVOLUCIONESCAB devCab = tools.ExecuteReportObject(new VENDEVOLUCIONESCAB() { IDVENDEVOLUCIONES = returnInfo.ReturnID });

if (devCab != null && devCab.STATUS != VENDEVOLUCIONESCAB.Status.Applied)
{
    Logger.AddWarningSubModule(
        TransactionSADNotifyStatusV3.SubModule,
        AxNLS.Data(
            1823638247,
            2309,
            "Se recibió una devolución de NetPay sin completar, por lo tanto no se realizará la solicitud de devolución a NetPay. Orden:{0}, Status:{1}",
            returnInfo.OrderID,
            devCab.STATUS));
    return rc;
}

if (liqdb != null && devCab != null)
{
    SADCENTRALNETPAYDEVOLUCION dev = tools.ExecuteReportObject(new SADCENTRALNETPAYDEVOLUCION() { SADORDENID = returnInfo.OrderID, IDVENDEVOLUCIONES = returnInfo.ReturnID });

    decimal montoDevuelto = (devCab.SUBTOTAL + devCab.IVA + devCab.COMISIONBANCARIA) - (devCab.DESCUENTO + (devCab.AJUSTEDEVOLUCION.HasValue ? devCab.AJUSTEDEVOLUCION.Value : 0));

    if (dev == null)
    {
        Logger.AddInfoSubModule(SubModule, AxNLS.Data(1823638247, 2134, "Se registrara una devolución de NetPay para la orden {0}", returnInfo.OrderID));


        dev = new SADCENTRALNETPAYDEVOLUCION();
        dev.SADORDENID = returnInfo.OrderID;
        dev.IDVENDEVOLUCIONES = returnInfo.ReturnID;
        dev.CODIGO = String.Empty;
        dev.MENSAJE = String.Empty;
        dev.TIPOTRANSACCION = String.Empty;
        dev.TRANSACCIONID = String.Empty;

        dev.FECHAREGISTRO = DateTime.Now;
        dev.DEVOLUCIONAPLICADA = false;
        dev.FECHADEVOLUCION = dev.FECHAREGISTRO;
        dev.REINTENTOSDEVOLUCION = 0;
        dev.FECHAULTIMOINTENTO = dev.FECHAREGISTRO;
        dev.REGISTROACTIVO = true;
        dev.MONTODEVUELTO = montoDevuelto - walletAlreadyPaid;
        dev.TIPODEVOLUCION = montoDevuelto == Convert.ToDecimal(liqdb.MONTO) ?
            SADCENTRALNETPAYDEVOLUCION.TIPO_DEVOLUCION_VALUES.DEVOLUCION_TOTAL_CANCELACION : SADCENTRALNETPAYDEVOLUCION.TIPO_DEVOLUCION_VALUES.DEVOLUCION_PARCIAL;

        Session ns = tools.CreateSession(SubModule);

        ns.InsertObjectWithID(dev);

        try
        {
            ns.Execute();
        }
        catch (Exception ex)
        {
            return rc.SetAndLog(SubModule, AxNLS.ErrorData(1823638247, 2129, "Excepción al registrar la devolución para el sistema NetPay. Excepción:{0}", ex.Message));
        }
    }
```

Ver en pantalla completa

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

Para aplicar la devolución a NetPay se ejecuta el método Execute de la clase CentralProcess, el cual a su vez ejecuta el método CheckNetPayReturns de la clase BusinessLogicNetPay.

```
// Aplica las devoluciones de NetPay en caso de ordenes liquidadas devueltas

try
{
    BusinessLogicNetPay.CheckNetPayReturns(_tools, _webConfig);
}
catch (Exception ex)
{
    Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2122,"Ocurrio un excepción en el proceso de devoluciones de NetPay. Excepción:{0}",ex.Message));
}
```

Ver en pantalla completa

En el método CheckNetPayReturns de la clase CheckNetPayReturns se obtienen los registros almacenados de la tabla SADCENTRALNETPAYLIQUIDACION para registro por registro aplicar el pago en NetPay.

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

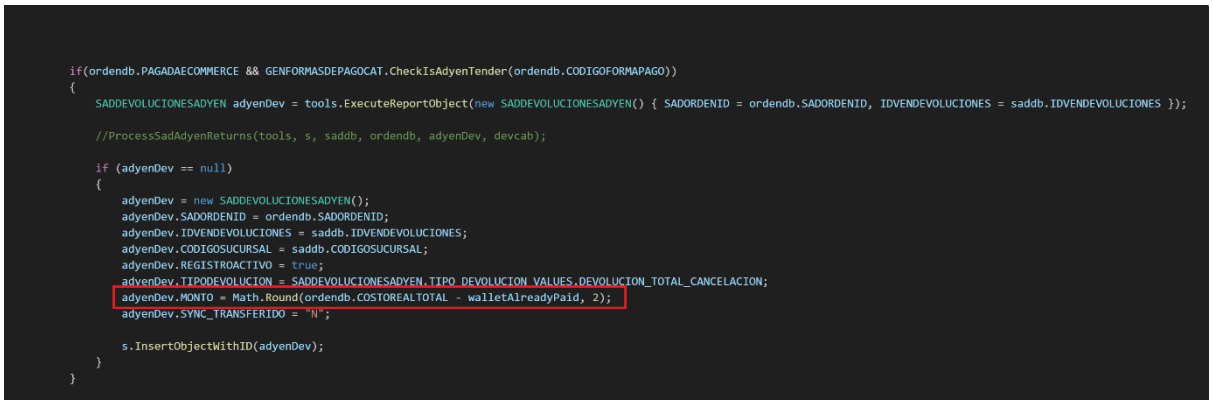
```
/// <summary>
/// Se revisa si existen devoluciones de NETPAY pendientes
/// </summary>

public static void CheckNetPayReturns(DatabaseTools tools, WebServiceConfig webConfig)
{

    if (DateTime.Now.Subtract(_netPayReturnLastChecking).TotalMinutes < NetPayReturnsCheckingMinutes)
    {
        return;
    }

    _netPayReturnLastChecking = DateTime.Now;


    IList<SADCENTRALNETPAYDEVOLUCION> pendingReturns = tools.ExecuteReportObjects<SADCENTRALNETPAYDEVOLUCION>(new SADCENTRALNETPAYDEVOLUCION()
    {
        REGISTROACTIVO = true
    });

    if (pendingReturns.Count == 0)
    {
        return;
    }

    Logger.AddInfoSubModule(SubModule, AxNLS.Data(1823638247, 2116, "Se encontraron {0} orden(es) pendiente(s) de devolución en NETPAY", pendingReturns.Count));


    List<NetPayReturnContainer> returnsToProcess = new List<NetPayReturnContainer>();

    foreach (SADCENTRALNETPAYDEVOLUCION returndb in pendingReturns)
    {
        SADCENTRALNETPAYLIQUIDACION netPaySettlement = tools.ExecuteReportObject(new SADCENTRALNETPAYLIQUIDACION() { SADORDENID = returndb.SADORDENID });

        if (netPaySettlement == null)
        {
            Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2148,"La liquidación de la orden {0} con NetPay no fue encontrada",returndb.SADORDENID));
            continue;
        }

        SADORDENES order = tools.ExecuteReportObject(new SADORDENES() { SADORDENID = returndb.SADORDENID });

        if (order == null)
        {
            Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2223,"La orden {0} con NetPay no fue encontrada",returndb.SADORDENID));
            continue;
        }

        VENDEVOLUCIONESCAB dev = tools.ExecuteReportObject(new VENDEVOLUCIONESCAB() { IDVENDEVOLUCIONES = returndb.IDVENDEVOLUCIONES});

        if (dev == null)
        {
            Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2303,"La devolución {0} no fue encontrada",returndb.IDVENDEVOLUCIONES));
            continue;
        }


        NetPayReturnContainer c = new NetPayReturnContainer();
        c.ReturnTracking = returndb;
        c.Settlement = netPaySettlement;
        c.Order = order;

        returnsToProcess.Add(c);
    }


    if (returnsToProcess.Count == 0)
    {
        return;
    }


    ExecuteReturns(tools, webConfig, returnsToProcess);
}
```

Ver en pantalla completa

Adyen (orden SAD no entregada)

Dentro del método BusinessLogicReturns.ProcessSadReturns, si la orden SAD no fue entregada y la venta fue pagada con Tarjeta Adyen se obtiene el encabezado de la orden SAD (SADORDENES) y se genera un registro en la tabla “SADDEVOLUCIONESADYEN” en la base de datos Central, el monto a devolver se calcula actualmente con la operación

aritmética: **costo real de la orden SAD - el importe pagado de la venta con MdA**. El problema reside que no se considera si ya existe un movimiento para la devolución con el importe completo de la venta o parcial.



```
if(ordendb.PAGADAECOMMERCE && GENFORMASDEPAGOCAT.CheckIsAdyenTender(ordendb.CODIGOFORMAPAGO))
{
    SADDEVOLUCIONESADYEN adyenDev = tools.ExecuteReportObject(new SADDEVOLUCIONESADYEN() { SADORDENID = ordendb.SADORDENID, IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

    //ProcessSadAdyenReturns(tools, s, saddb, ordendb, adyenDev, devcab);

    if (adyenDev == null)
    {
        adyenDev = new SADDEVOLUCIONESADYEN();
        adyenDev.SADORDENID = ordendb.SADORDENID;
        adyenDev.IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES;
        adyenDev.CODIGOSUCURSAL = saddb.CODIGOSUCURSAL;
        adyenDev.REGISTROACTIVO = true;
        adyenDev.TIPODEVOLUCION = SADDEVOLUCIONESADYEN.TIPO_DEVOLUCION_VALUES.DEVOLUCION_TOTAL_CANCELACION;
        adyenDev.MONTO = Math.Round(ordendb.COSTOREALTOTAL - walletAlreadyPaid, 2);
        adyenDev.SYNC_TRANSFERIDO = "N";

        s.InsertObjectWithID(adyenDev);
    }
}
```

Ver en pantalla completa

Al encontrarse información en la tabla “SADDEVOLUCIONESADYEN”, se envían las transacciones a Adyen para aplicar el pago desde el método ProcessAdyenReturn de la clase BusinessLogicAdyen, el cual se ejecuta desde el método Execute de la clase StoreProcess.

```
try
{
    BusinessLogicAdyen.ProcessAdyenReturn(_storeId, _tools, _client);
}
catch (Exception ex)
{
    Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2218,"Ocurrio una excepción en el proceso de devoluciones/cancelaciones de ordenes Adyen. Excepción:{0}",ex.Message));
}
```

Ver en pantalla completa

PayPal (orden SAD no entregada)

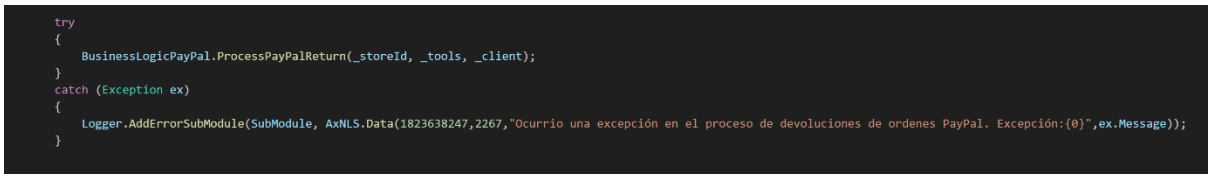
Dentro del método BusinessLogicReturns.ProcessSadReturns, si la orden SAD no fue entregada y la venta fue pagada con PayPal se obtiene el encabezado de la orden SAD (SADORDENES) y se genera un registro en la tabla “SADDEVOLUCIONESPAYPAL” en la

base de datos Central, el monto a devolver se calcula actualmente con la operación aritmética: **costo real de la orden SAD - el importe pagado de la venta con MdA**. El problema reside que no se considera si ya existe un movimiento para la devolución con el importe completo de la venta o parcial.

```
if (ordendb.PAGADAECOMMERCE && GENFORMASDEPAGOCAT.CheckIsPayPalTender(ordendb.CODIGOFORMAPAGO))
{
    SADDEVOLUCIONESPAYPAL payPalDev = tools.ExecuteReportObject(new SADDEVOLUCIONESPAYPAL() { SADORDENID = ordendb.SADORDENID, IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

    if (payPalDev == null)
    {
        payPalDev = new SADDEVOLUCIONESPAYPAL();
        payPalDev.SADORDENID = ordendb.SADORDENID;
        payPalDev.IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES;
        payPalDev.CODIGOSUCURSAL = saddb.CODIGOSUCURSAL;
        payPalDev.REGISTROACTIVO = true;
        payPalDev.TIPODEVOLUCION = SADDEVOLUCIONESPAYPAL.TIPO_DEVOLUCION_VALUES.DEVOLUCION_TOTAL;
        payPalDev.MONTO = Math.Round(ordendb.COSTOREALTOTAL - walletAlreadyPaid, 2);
        payPalDev.SYNC_TRANSFERIDO = "N";

        s.InsertObjectWithID(payPalDev);
    }
}
```

Ver en pantalla completa

Al encontrarse información en la tabla SADDEVOLUCIONESPAYPAL, se envían las transacciones a PayPal para aplicar el pago desde el método ProcessPayPalReturn de la clase BusinessLogicPayPal. Este método se ejecuta desde el método Execute de la clase StoreProcess.



```
try
{
    BusinessLogicPayPal.ProcessPayPalReturn(_storeId, _tools, _client);
}
catch (Exception ex)
{
    Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2267,"Ocurrio una excepción en el proceso de devoluciones de ordenes PayPal. Excepción:{0}",ex.Message));
}
```

Ver en pantalla completa

Posibles soluciones

La posible actualización de código se realiza en métodos diferentes y esto es de acuerdo al tipo de tarjeta, el ajuste implica obtener la suma de los importes pagados e IVA de los movimientos de la tabla genMovimientosCajasDet que fueron generados para la devolución, después se es obtener aritméticamente la diferencia con el importe de la devolución de la tabla venDevolucionesCab, en caso de existir un valor mayor a cero, este será devuelto a la

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

Tarjeta (Adyen, NetPay o PayPal), almacenado el registro en tabla correspondiente de la base datos. para posteriormente aplicar el pago.

Relación tipo de tarjeta y tabla de base datos que se considera para aplicar el pago.

Tarjeta	Tabla
Adyen	SADDEVOLUCIONESADYEN
NetPay	SADCENTRALNETPAYDEVOLUCION
PayPal	SADDEVOLUCIONESPAYPAL

Adyen

Actualización del método ProcessSadReturn de la clase BusinessLogicReturns cuando la tarjeta es Adyen.

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

```
if (ordendb.PAGADAECOMMERCE && GENFORMASDEPAGOCAT.CheckIsAdyenTender(ordendb.CODIGOFORMAPAGO))
{
    SADDEVOLUCIONESADYEN adyenDev = tools.ExecuteReportObject(new SADDEVOLUCIONESADYEN() { SADORDENID = ordendb.SADORDENID, IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

    if (adyenDev == null)
    {
        VENDEVOLUCIONESCAB devcab = tools.ExecuteReportObject(new VENDEVOLUCIONESCAB() { IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

        if(devcab == null)
        {
            Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247,2217,"La transacción de devolución no fue encontrada. IDVENDEVOLUCIONES:{0}",saddb.IDVENDEVOLUCIONES));
            continue;
        }

        //Se obtienen los movimientos generados para la devolución que se encuentren aplicados
        IList<GENMOVIMIENTOSCAJASDET> movDet = tools.ExecuteReportObjects<GENMOVIMIENTOSCAJASDET>(new GENMOVIMIENTOSCAJASDET()
        {
            CODIGOMOVIMIENTOCAJA = (Guid)devcab.CODIGOMOVIMIENTOCAJA, ESTATUSFORMAPAGO = "A"
        });

        if (movDet != null && movDet.Count > 0)
        {

            decimal montoDevolucionMdA = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_MONEDERO)
                                               .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

            Logger.AddInfoSubModule(SubModule, "[Devolucion-Adyen] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a MdA (14) " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucionMdA);


            decimal montoDevolucionTarjeta = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_ADYEN_CREDIT
                                                    || a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_ADYEN_DEBIT
                                                    || a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_ADYEN_AMEX)
                                                .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

            Logger.AddInfoSubModule(SubModule, "[Devolucion-Adyen] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a tarjeta Adyen (71,72,73) " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucionTarjeta);

            decimal montoDevolucion = Math.Round(devcab.SUBTOTAL + devcab.IVA, 2);
            Logger.AddInfoSubModule(SubModule, "[Devolucion-Adyen] El monto de la devolucion para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucion);

            decimal montoParaDevolver = 0;
            if (montoDevolucionTarjeta == montoDevolucion)//si el monto del movimiento con tarjeta es el total del monto de la devolucion
            {
                montoParaDevolver = montoDevolucionTarjeta;
            }
            else if (montoDevolucionMdA < montoDevolucion)//si el monto de movimiento con MdA es menor al monto de la devolucion, se calcula el monto a devolver a tarjeta
            {
                montoParaDevolver = Math.Round(montoDevolucion - montoDevolucionMdA, 2);
            }

            Logger.AddInfoSubModule(SubModule, "[Devolucion-Adyen] El monto a devolver a la tarjeta para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES + " es " + montoParaDevolver);

            if (montoParaDevolver > 0)
            {
                adyenDev = new SADDEVOLUCIONESADYEN();
                adyenDev.SADORDENID = ordendb.SADORDENID;
                adyenDev.IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES;
                adyenDev.CODIGOSUCURSAL = saddb.CODIGOSUCURSAL;
                adyenDev.REGISTROACTIVO = true;
                adyenDev.TIPODEVOLUCION = SADDEVOLUCIONESADYEN.TIPO_DEVOLUCION_VALUES.DEVOLUCION_PARCIAL;
                adyenDev.MONTO = montoParaDevolver;
                adyenDev.SYNC_TRANSFERIDO = "N";

                s.InsertObjectWithID(adyenDev);
            }
            else
            {
                Logger.AddInfoSubModule(SubModule, "[Devolucion-Adyen] El monto total ya fue devuelto para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES);
            }
        }
        else
        {
            Logger.AddInfoSubModule(SubModule, "[Devolucion-Adyen] No se encontraron movimientos aplicados para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES);
        }
    }
}
```

Ver en pantalla completa

NetPay

Actualización del método RegisterOrderReturnNetPay de la clase TransactionSADNotifyStatusV3.

GET Technologies	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

```
SADCENTRALNETPAYLIQUIDACION liqdb = tools.ExecuteReportObject(new SADCENTRALNETPAYLIQUIDACION() { SADORDENID = returnInfo.OrderID });
VENDEVOLUCIONESCAB devCab = tools.ExecuteReportObject(new VENDEVOLUCIONESCAB() { IDVENDEVOLUCIONES = returnInfo.ReturnID });

if (devCab != null && devCab.STATUS != VENDEVOLUCIONESCAB.Status.Applied)
{
    Logger.AddWarningSubModule(
        TransactionSADNotifyStatusV3.SubModule,
        AxNLS.Data(
            1823638247,
            2309,
            "Se recibió una devolución de NetPay sin completar, por lo tanto no se realizará la solicitud de devolución a NetPay. Orden:{0}, Status:{1}",
            returnInfo.OrderID,
            devCab.STATUS));
    return rc;
}

if (liqdb != null && devCab != null)
{
    SADCENTRALNETPAYDEVOLUCION dev = tools.ExecuteReportObject(new SADCENTRALNETPAYDEVOLUCION() { SADORDENID = returnInfo.OrderID, IDVENDEVOLUCIONES = returnInfo.ReturnID });

    //Se obtienen los movimientos generados para la devolución que se encuentren aplicados
    Ilist<GENMOVIMIENTOSCAJASDET> movDet = tools.ExecuteReportObjects<GENMOVIMIENTOSCAJASDET>(new GENMOVIMIENTOSCAJASDET()
    {
        CODIGOMOVIMIENTOCAJA = (Guid)devCab.CODIGOMOVIMIENTOCAJA,
        ESTATUSFORMAPAGO = "A"
    });

    if (movDet != null && movDet.Count > 0 && dev == null)
    {
        decimal montoDevolucionMdA = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_MONEDERO)
                                           .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

        Logger.AddInfoSubModule(SubModule, "[Devolucion-NetPay] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a MdA (14) " + devCab.IDVENDEVOLUCIONES + " es " + montoDevolucionMdA);

        decimal montoDevolucionTarjeta = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_NETPAY_GENERIC
                                                        || a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_NETPAY_CREDIT
                                                        || a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_NETPAY_DEBIT
                                                        || a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_NETPAY_AMEX)
                                               .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

        Logger.AddInfoSubModule(SubModule, "[Devolucion-NetPay] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a tarjeta Adyen (63,64,65, 66) " + devCab.IDVENDEVOLUCIONES + " es " + montoDevolucionTarjeta);

        decimal montoDevolucion = (devCab.SUBTOTAL + devCab.IVA + devCab.COMISIONBANCARIA) - (devCab.DESCUENTO + (devCab.AJUSTEDEVOLUCION.HasValue ? devCab.AJUSTEDEVOLUCION.Value : 0));

        Logger.AddInfoSubModule(SubModule, "[Devolucion-NetPay] El monto de la devolucion para IDVENDEVOLUCIONES " + devCab.IDVENDEVOLUCIONES + " es " + montoDevolucion);

        decimal montoParaDevolver = 0;
        if (montoDevolucionTarjeta == montoDevolucion)//si el monto del movimiento con tarjeta es el total del monto de la devolucion
        {
            montoParaDevolver = montoDevolucionTarjeta;
        }
        else if (montoDevolucionMdA < montoDevolucion)//si el monto de movimiento con MdA es menor al monto de la devolucion, se calcula el monto a devolver a tarjeta
        {
            montoParaDevolver = Math.Round(montoDevolucion - montoDevolucionMdA, 2);
        }

        Logger.AddInfoSubModule(SubModule, "[Devolucion-NetPay] El monto a devolver a la tarjeta para IDVENDEVOLUCIONES " + devCab.IDVENDEVOLUCIONES + " es " + montoParaDevolver);

        if (montoParaDevolver > 0)
        {
            Logger.AddInfoSubModule(SubModule, AxNLS.Data(1823638247, 2134, "Se registrara una devolución de NetPay para la orden {0}", returnInfo.OrderID));

            dev = new SADCENTRALNETPAYDEVOLUCION();
            dev.SADORDENID = returnInfo.OrderID;
            dev.IDVENDEVOLUCIONES = returnInfo.ReturnID;
            dev.CODIGO = String.Empty;
            dev.MENSAJE = String.Empty;
            dev.TIPOTRANSACCION = String.Empty;
            dev.TRANSACCIONID = String.Empty;

            dev.FECHAREGISTRO = DateTime.Now;
            dev.DEVOLUCIONAPLICADA = false;
            dev.FECHADEVOLUCION = dev.FECHAREGISTRO;
            dev.REINTENTOSDEVOLUCION = 0;
            dev.FECHAULTIMOINTENTO = dev.FECHAREGISTRO;
            dev.REGISTROACTIVO = true;
            dev.MONTODEVUELTO = montoParaDevolver;
            dev.TIPODEVOLUCION = montoParaDevolver == Convert.ToDecimal(liqdb.MONTO) ?
                SADCENTRALNETPAYDEVOLUCION.TIPO_DEVOLUCION_VALUES.DEVOLUCION_TOTAL_CANCELACION : SADCENTRALNETPAYDEVOLUCION.TIPO_DEVOLUCION_VALUES.DEVOLUCION_PARCIAL;

            Session ns = tools.CreateSession(SubModule);

            ns.InsertObjectWithID(dev);

            try
            {
                ns.Execute();
            }
            catch (Exception ex)
            {
                return rc.SetAndLog(SubModule, AxNLS.ErrorData(1823638247, 2129, "Excepción al registrar la devolución para el sistema NetPay. Excepción:{0}", ex.Message));
            }
        }
        else
        {
            Logger.AddInfoSubModule(SubModule, "[Devolucion-NetPay] El monto total ya fue devuelto para IDVENDEVOLUCIONES " + returnInfo.ReturnID);
        }
    }
    else
    {
        Logger.AddInfoSubModule(SubModule, "[Devolucion-NetPay] No se encontraron movimientos aplicados para IDVENDEVOLUCIONES " + returnInfo.ReturnID);
    }
}
```

Ver en pantalla completa

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

PayPal

Actualización del método ProcessSadReturn de la clase BusinessLogicReturns cuando la tarjeta es PayPal.

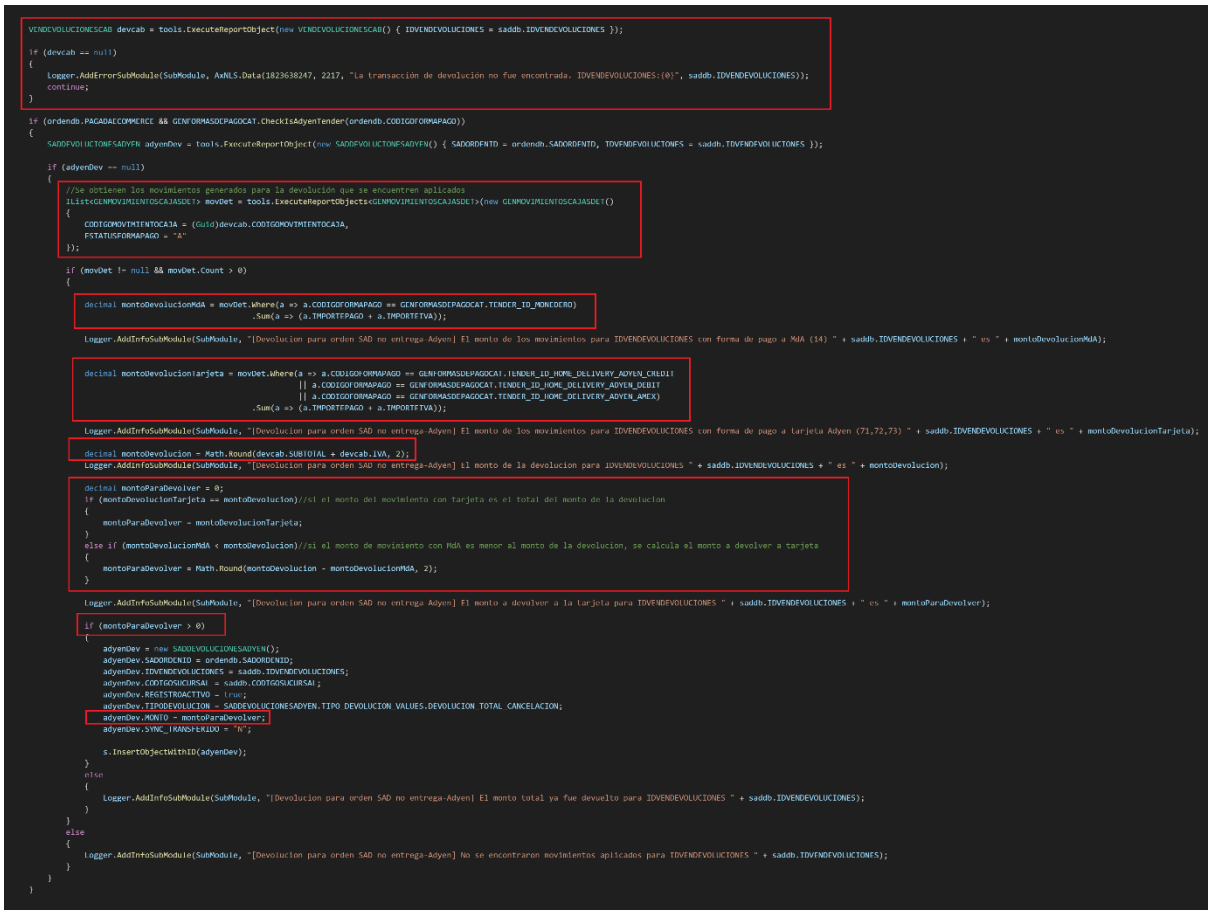
```
if (ordendb.PAGADAECOMMERCE && GENFORMASDEPAGOCAT.ChecksIsPayPalTender(ordendb.CODIGOFORMAPAGO))
{
    SADDEVOLUCIONESPAYPAL payPalDev = tools.ExecuteReportObject(new SADDEVOLUCIONESPAYPAL() { SADORDENID = ordendb.SADORDENID, IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

    if (payPalDev == null)
    {
        VENDEVOLUCIONESCAB devcab = tools.ExecuteReportObject(new VENDEVOLUCIONESCAB() { IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

        if (devcab == null)
        {
            Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247, 2217, "La transacción de devolución no fue encontrada. IDVENDEVOLUCIONES:{0}", saddb.IDVENDEVOLUCIONES));
            continue;
        }

        //Se obtienen los movimientos generados para la devolución que se encuentren aplicados
        IList<GENMOVIMIENTOSCAJASDET> movDet = tools.ExecuteReportObjects<GENMOVIMIENTOSCAJASDET>(new GENMOVIMIENTOSCAJASDET()
        {
            CODIGOMOVIMIENTOCAJA = (Guid)devcab.CODIGOMOVIMIENTOCAJA,
            ESTATUSFORMAPAGO = "A"
        });

        if (movDet != null && movDet.Count > 0)
        {
            decimal montoDevolucionMdA = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_MONEDERO)
                                               .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

            Logger.AddInfoSubModule(SubModule, "[Devolucion-PayPal] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a MdA (14) " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucionMdA);

            decimal montoDevolucionTarjeta = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_PAYPAL)
                                                   .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

            Logger.AddInfoSubModule(SubModule, "[Devolucion-PayPal] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a tarjeta Adyen (41) " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucionTarjeta);

            decimal montoDevolucion = Math.Round(devcab.SUBTOTAL + devcab.IVA, 2);
            Logger.AddInfoSubModule(SubModule, "[Devolucion-PayPal] El monto de la devolucion para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucion);

            decimal montoParaDevolver = 0;
            if (montoDevolucionTarjeta == montoDevolucion)//si el monto del movimiento con tarjeta es el total del monto de la devolucion
            {
                montoParaDevolver = montoDevolucionTarjeta;
            }
            else if (montoDevolucionMdA < montoDevolucion)//si el monto de movimiento con MdA es menor al monto de la devolucion, se calcula el monto a devolver a tarjeta
            {
                montoParaDevolver = Math.Round(montoDevolucion - montoDevolucionMdA, 2);
            }

            Logger.AddInfoSubModule(SubModule, "[Devolucion-PayPal] El monto a devolver a la tarjeta para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES + " es " + montoParaDevolver);

            if (montoParaDevolver > 0)
            {
                //Para evitar generar discrepancias se dejará devolucion_parcial dado que no afecta al enviar a PayPal,
                //lo correcto es realizar la verificacion: TIPODEVOLUCION = payPalDev.MONTO < ordendb.COSTOREALTOTAL ? DEVOLUCION_PARCIAL : DEVOLUCION_TOTAL;
                payPalDev = new SADDEVOLUCIONESPAYPAL();
                payPalDev.SADORDENID = ordendb.SADORDENID;
                payPalDev.IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES;
                payPalDev.CODIGOSUCURSAL = saddb.CODIGOSUCURSAL;
                payPalDev.REGISTROACTIVO = true;
                payPalDev.MONTO = montoParaDevolver;
                payPalDev.TIPODEVOLUCION = SADDEVOLUCIONESPAYPAL.TIPO_DEVOLUCION_VALUES.DEVOLUCION_PARCIAL;
                payPalDev.SYNC_TRANSFERIDO = "N";

                s.InsertObjectWithID(payPalDev);
            }
            else
            {
                Logger.AddInfoSubModule(SubModule, "[Devolucion-PayPal] El monto total ya fue devuelto para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES);
            }
        }
        else
        {
            Logger.AddInfoSubModule(SubModule, "[Devolucion-PayPal] No se encontraron movimientos aplicados para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES);
        }
    }
}
```

Ver en pantalla completa

Adyen (orden SAD no entrega)

Actualización del método ProcessSadReturn de la clase BusinessLogicReturns cuando la tarjeta es Adyen y la orden SAD es no entrega.

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025



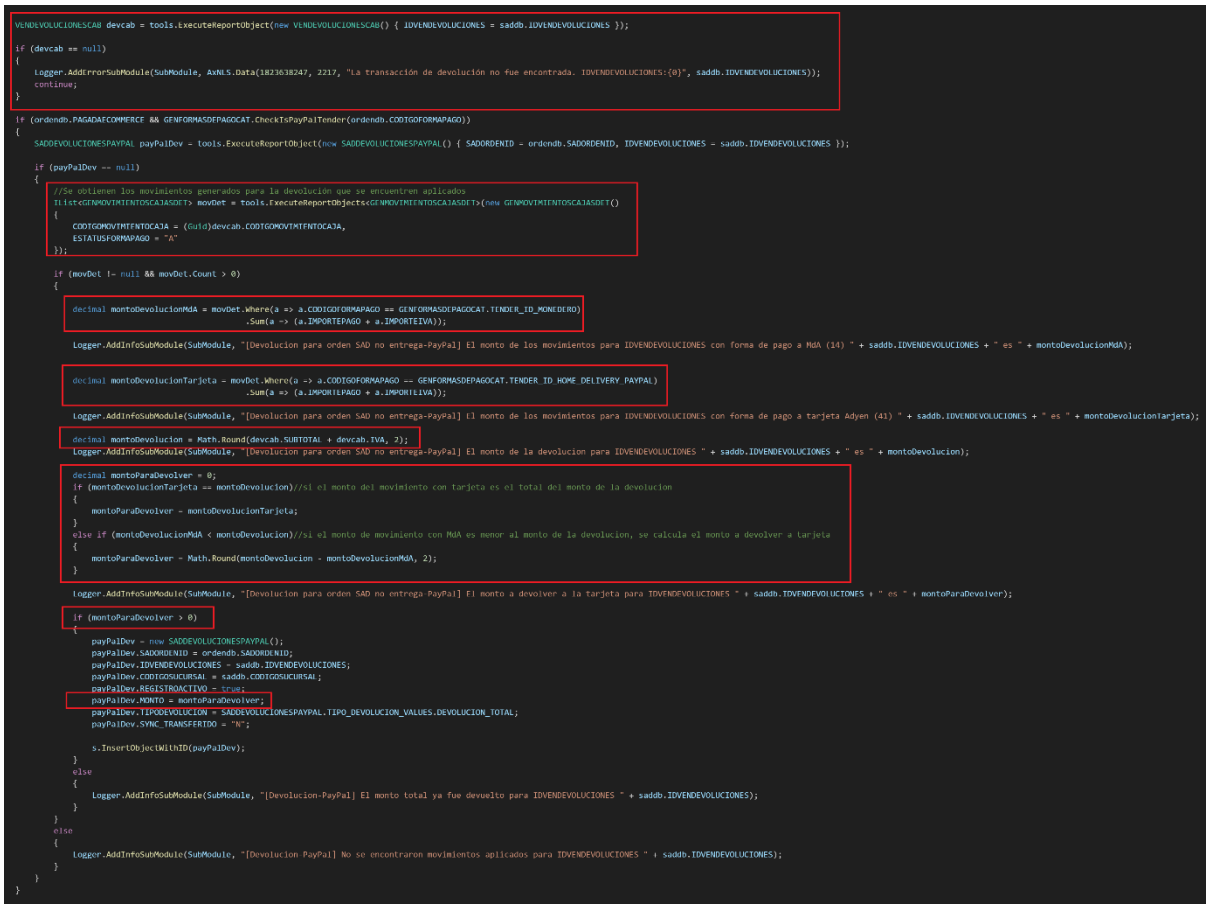
```
VENDEVOLUCIONESCAB devcab = tools.ExecuteReportObject(new VENDEVOLUCIONESCAB() { IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

if (devcab == null)
{
    Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247, 2217, "La transacción de devolución no fue encontrada. IDVENDEVOLUCIONES:{0}", saddb.IDVENDEVOLUCIONES));
    continue;
}

if (ordendb.PAGADAECOMMERCE && GENFORMASDEPAGOCAT.CheckIsAdyenTender(ordendb.CODIGOFORMAPAGO))
{
    SADDEVOLUCIONESADYEN adyenDev = tools.ExecuteReportObject(new SADDEVOLUCIONESADYEN() { SADORDENID = ordendb.SADORDENID, IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

    if (adyenDev == null)
    {
        //Se obtienen los movimientos generados para la devolución que se encuentren aplicados
        IList<GENMOVIMIENTOSCAJASDET> movDet = tools.ExecuteReportObjects<GENMOVIMIENTOSCAJASDET>(new GENMOVIMIENTOSCAJASDET()
        {
            CODIGOMOVIMIENTOCAJA = (Guid)devcab.CODIGOMOVIMIENTOCAJA,
            ESTATUSFORMAPAGO = "A"
        });

        if (movDet != null && movDet.Count > 0)
        {
            decimal montoDevolucionMdA = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_MONEDERO)
                                               .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-Adyen] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a MdA (14) " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucionMdA);

            decimal montoDevolucionTarjeta = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_ADYEN_CREDIT
                                                    || a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_ADYEN_DEBIT
                                                    || a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_ADYEN_AMEX)
                                                .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-Adyen] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a tarjeta Adyen (71,72,73) " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucionTarjeta);

            decimal montoDevolucion = Math.Round(devcab.SUBTOTAL + devcab.IVA, 2);
            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-Adyen] El monto de la devolucion para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucion);

            decimal montoParaDevolver = 0;
            if (montoDevolucionTarjeta == montoDevolucion)//si el monto del movimiento con tarjeta es el total del monto de la devolucion
            {
                montoParaDevolver = montoDevolucionTarjeta;
            }
            else if (montoDevolucionMdA < montoDevolucion)//si el monto de movimiento con MdA es menor al monto de la devolucion, se calcula el monto a devolver a tarjeta
            {
                montoParaDevolver = Math.Round(montoDevolucion - montoDevolucionMdA, 2);
            }

            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-Adyen] El monto a devolver a la tarjeta para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES + " es " + montoParaDevolver);

            if (montoParaDevolver > 0)
            {
                adyenDev = new SADDEVOLUCIONESADYEN();
                adyenDev.SADORDENID = ordendb.SADORDENID;
                adyenDev.IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES;
                adyenDev.CODTGOSUCURSAL = saddb.CODTGOSUCURSAL;
                adyenDev.REGISTROACTIVO = true;
                adyenDev.TIPODEVOLUCION = SADDEVOLUCIONESADYEN.TIPO_DEVOLUCION_VALUES.DEVOLUCION_TOTAL_CANCELACION;
                adyenDev.MONTO = montoParaDevolver;
                adyenDev.SYNC_TRANSFERIDO = "N";

                s.InsertObjectWithID(adyenDev);
            }
            else
            {
                Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-Adyen] El monto total ya fue devuelto para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES);
            }
        }
        else
        {
            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-Adyen] No se encontraron movimientos aplicados para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES);
        }
    }
}
```

[Ver en pantalla completa]

PayPal (orden SAD no entrega)

Actualización del método ProcessSadReturn de la clase BusinessLogicReturns cuando la tarjeta es PayPal y la orden SAD es no entrega.

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025



```
VENDEVOLUCIONESCAB devcab = tools.ExecuteReportObject(new VENDEVOLUCIONESCAB() { IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

if (devcab == null)
{
    Logger.AddErrorSubModule(SubModule, AxNLS.Data(1823638247, 2217, "La transacción de devolución no fue encontrada. IDVENDEVOLUCIONES:{0}", saddb.IDVENDEVOLUCIONES));
    continue;
}

if (ordendb.PAGADAFCOMMERCE && GENFORMASDEPAGOCAT.CheckIsPayPalTender(ordendb.CODTGOFORMAPAGO))
{
    SADDEVOLUCIONESPAYPAL payPalDev = tools.ExecuteReportObject(new SADDEVOLUCIONESPAYPAL() { SADORDENID = ordendb.SADORDENID, IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES });

    if (payPalDev == null)
    {
        //Se obtienen los movimientos generados para la devolución que se encuentren aplicados
        IList<GENMOVIMIENTOSCAJASDET> movDet = tools.ExecuteReportObject<GENMOVIMIENTOSCAJASDET>(new GENMOVIMIENTOSCAJASDET()
        {
            CODTGOMOVTMTFNTOCAJA = (Guid)devcab.CODTGOMOVTMTFNTOCAJA,
            ESTATUSFORMAPAGO = "A"
        });

        if (movDet != null && movDet.Count > 0)
        {
            decimal montoDevolucionMdA = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_MONEDERO)
                                               .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-PayPal] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a MdA (14) " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucionMdA);

            decimal montoDevolucionTarjeta = movDet.Where(a => a.CODIGOFORMAPAGO == GENFORMASDEPAGOCAT.TENDER_ID_HOME_DELIVERY_PAYPAL)
                                               .Sum(a => (a.IMPORTEPAGO + a.IMPORTEIVA));

            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-PayPal] El monto de los movimientos para IDVENDEVOLUCIONES con forma de pago a tarjeta Adyen (41) " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucionTarjeta);

            decimal montoDevolucion = Math.Round(devcab.SUBTOTAL + devcab.IVA, 2);
            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-PayPal] El monto de la devolucion para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES + " es " + montoDevolucion);

            decimal montoParaDevolver = 0;
            if (montoDevolucionTarjeta == montoDevolucion)//si el monto del movimiento con tarjeta es el total del monto de la devolucion
            {
                montoParaDevolver = montoDevolucionTarjeta;
            }
            else if (montoDevolucionMdA < montoDevolucion)//si el monto de movimiento con MdA es menor al monto de la devolucion, se calcula el monto a devolver a tarjeta
            {
                montoParaDevolver = Math.Round(montoDevolucion - montoDevolucionMdA, 2);
            }

            Logger.AddInfoSubModule(SubModule, "[Devolucion para orden SAD no entrega-PayPal] El monto a devolver a la tarjeta para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES + " es " + montoParaDevolver);

            if (montoParaDevolver > 0)
            {
                payPalDev = new SADDEVOLUCIONESPAYPAL();
                payPalDev.SADORDENID = ordendb.SADORDENID;
                payPalDev.IDVENDEVOLUCIONES = saddb.IDVENDEVOLUCIONES;
                payPalDev.CODTGOSUCURSAL = saddb.CODTGOSUCURSAL;
                payPalDev.REGISTROACTIVO = true;
                payPalDev.MONTO = montoParaDevolver;
                payPalDev.TIPODEVOLUCION = SADDEVOLUCIONESPAYPAL.TIPO_DEVOLUCION_VALUES.DEVOLUCION_TOTAL;
                payPalDev.SYNC_TRANSFERIDO = "N";

                s.InsertObjectWithID(payPalDev);
            }
            else
            {
                Logger.AddInfoSubModule(SubModule, "[Devolucion-PayPal] El monto total ya fue devuelto para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES);
            }
        }
        else
        {
            Logger.AddInfoSubModule(SubModule, "[Devolucion-PayPal] No se encontraron movimientos aplicados para IDVENDEVOLUCIONES " + saddb.IDVENDEVOLUCIONES);
        }
    }
}
```

Ver en pantalla completa

	FIX – Devoluciones SAD con pago duplicado a Tarjeta	
Autor	Versión	Fecha de elaboración
Alan Cedeño	1.0	27-06-2025

Posibles causas

La posible causa que genera una devolución duplicada con forma de pago Adyen, NetPay o PayPal es derivada a que no se esta considerando el monto ya devuelto, al considerarlos se evitaría la duplicidad ya que se obtendría la diferencia de lo ya pagado contra lo que resta por devolver.