
	Incidencia – Comisión MX\$0 en servicios de tarjeta de débito		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

Control de versiones

Versión	Fecha	Autor	Descripción
1.0	27-09-2024	Juan Pérez	Se revisa el flujo de las incidencias y se determina posible causa.

 Incidencia – Comisión MX\$0 en servicios de tarjeta de débito			
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

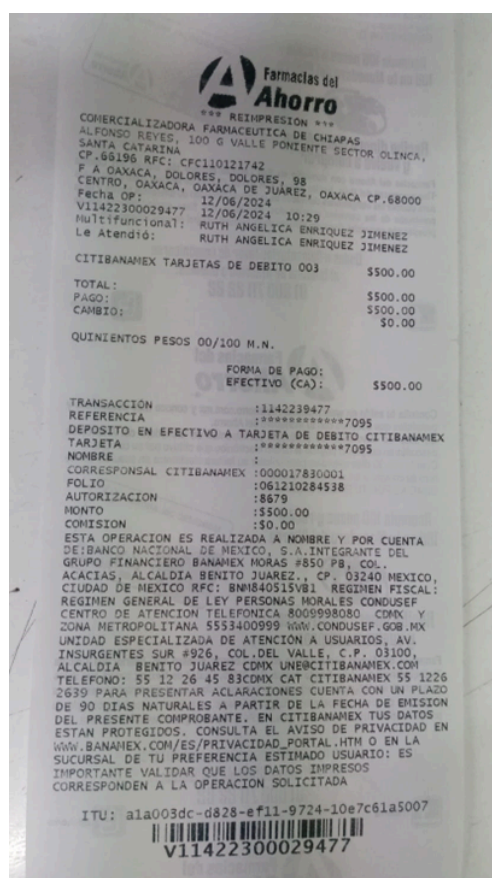
Descripción general

Para los servicios **HSBC DEPÓSITO A TARJETA DE DEBITO (Prosa)** y **CITIBANAMEX TARJETAS DE DEBITO 0035 (Guga21)** consumidos desde el punto de venta, no se está cobrando una comisión puesto que esta aparece con el monto de **MX\$0** al ingresar la referencia y continuar al siguiente paso.


Se toma como muestra la venta **V11422300029477**:

Sucursal	1142
Caja	23
Transacción	V11422300029477
Fecha Transacción	12/06/2024 10:29
Fecha Actualización Versión	27/05/2024 06:43

Se adjunta el ticket



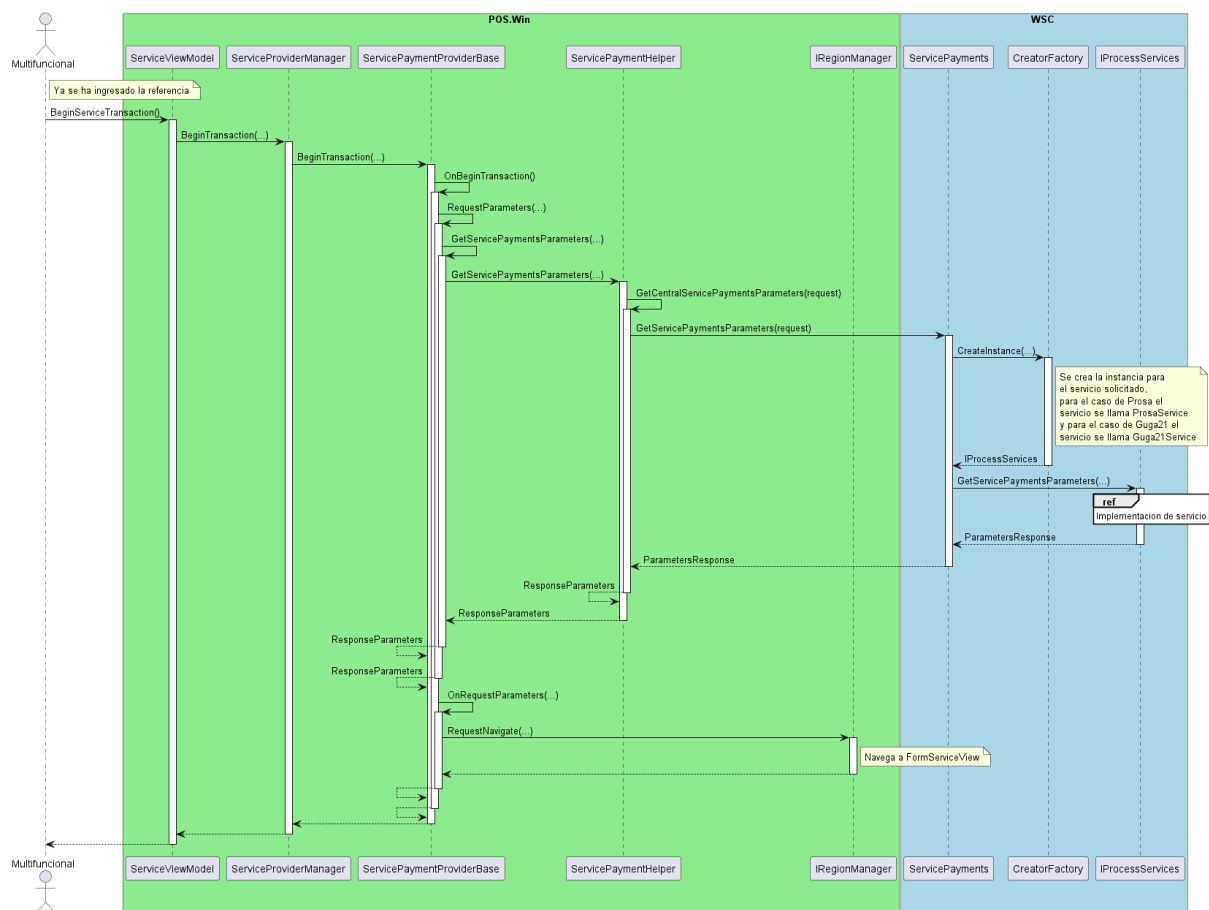
[Ver en pantalla completa](#)

<div>  <div> Incidencia – Comisión MX\$0 en servicios de tarjeta de débito </div> </div>			
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

Diagnóstico


Servicio Prosa

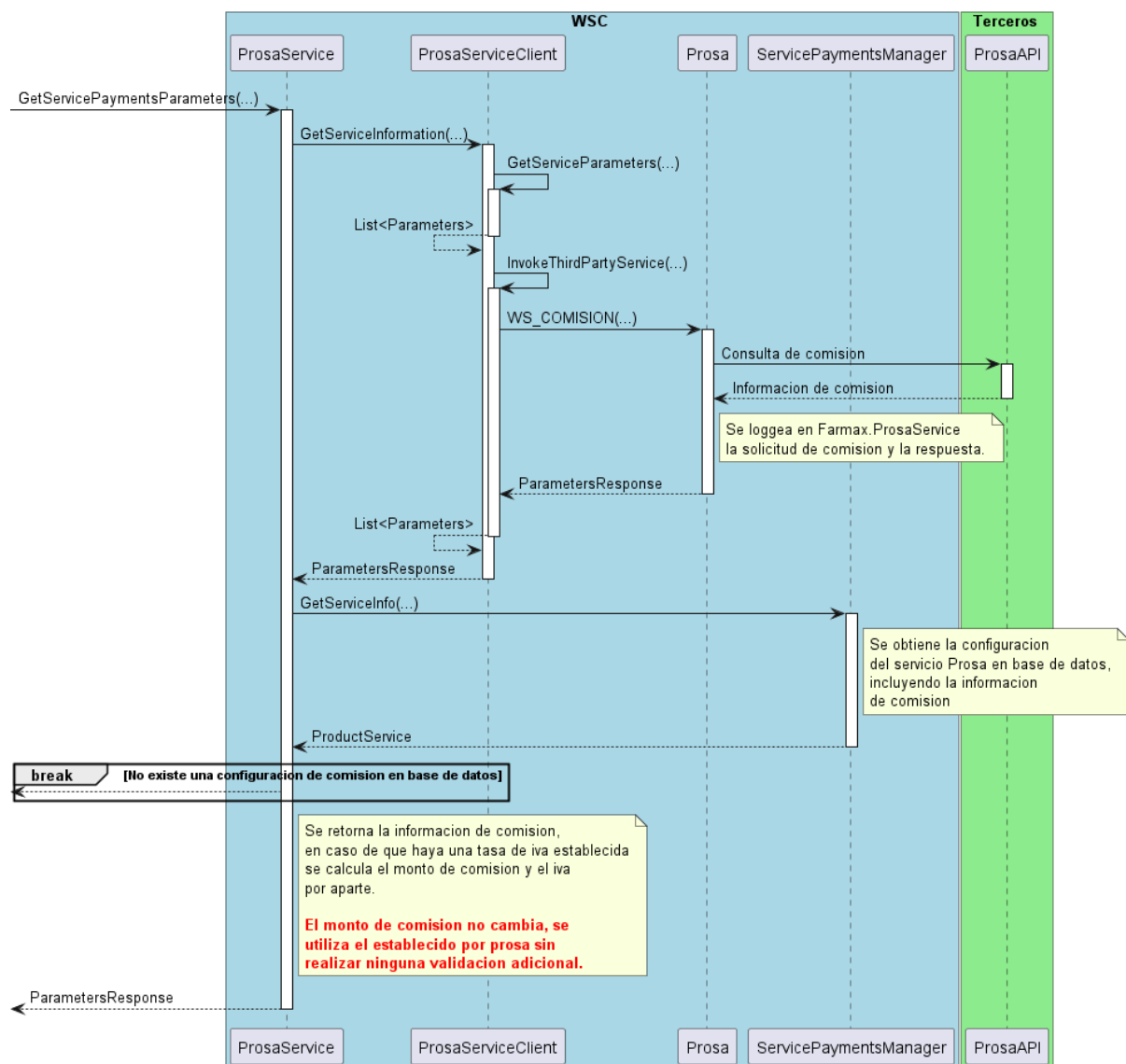
Se revisa lo que ocurre al enviar la referencia para obtener la información de comisión, se debe tener en cuenta que el proveedor del servicio **HSBC DEPÓSITO A TARJETA DE DEBITO** es **Prosa**, por lo que tiene su propia implementación al momento de realizar las operaciones relacionadas a este servicio, la primera parte del flujo aplica a todos los servicios y se expone en el siguiente diagrama de secuencia técnico:




[Ver en pantalla completa](#)

Se utiliza el método **IProcessService.GetServicePaymentsParameters(...)**, esta interfaz es implementada por cada servicio que necesite procesar sus servicios de una manera específica, la implementación para **Prosa** es **ProsaService**, la implementación de esta operación de consulta se expone en el siguiente diagrama de secuencia técnico:

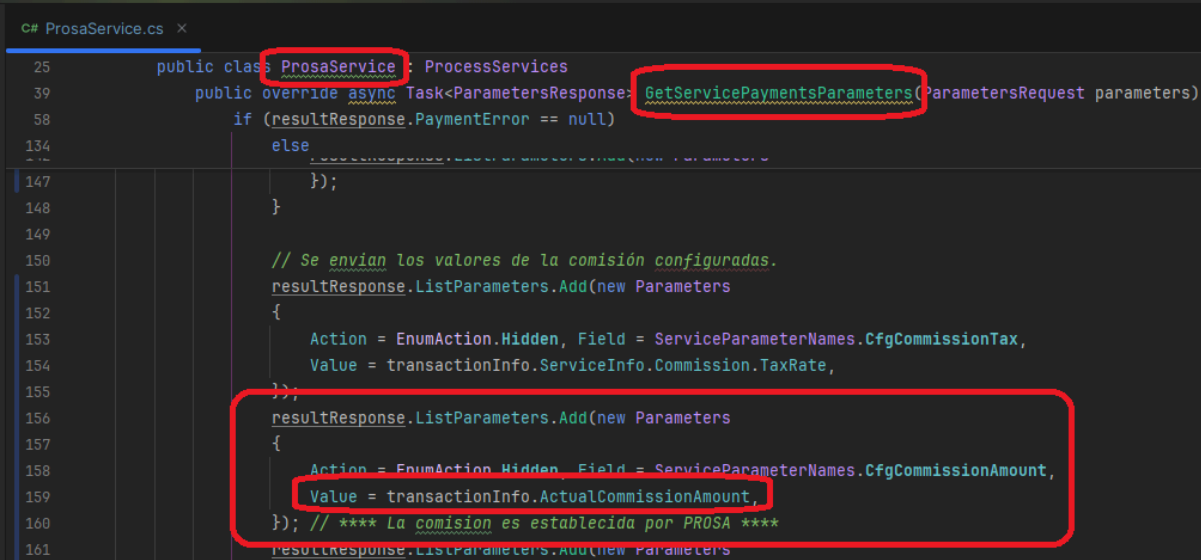
<div>  <div> Incidencia – Comisión MX\$0 en servicios de tarjeta de débito </div> </div>			
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024



[Ver en pantalla completa](#)

	<h1>Incidencia – Comisión MX\$0 en servicios de tarjeta de débito</h1>		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

Se encuentra que el monto de comisión se establece en base a lo que se recibe del servicio de **Prosa**, la configuración establecida en base de datos solamente se utiliza para calcular el **IVA** de la comisión establecida por **Prosa**, no se realiza ningún tipo de validación para asegurarse de que la comisión sea igual a la establecida en base de datos, solamente se valida que la configuración exista para poder calcular el **IVA** en base a la comisión obtenida del servicio de **Prosa**.



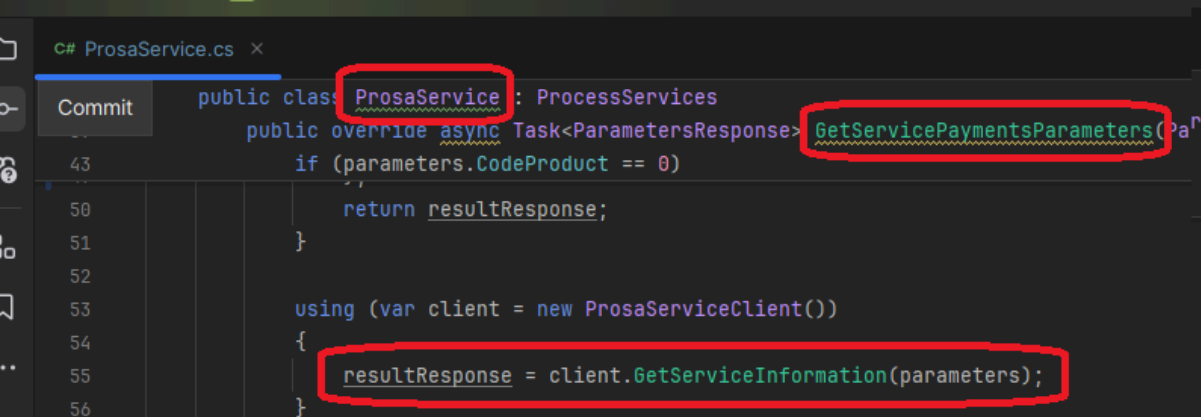
```

25 public class Prosaservice : ProcessServices
39 public override async Task<ParametersResponse> GetServicePaymentsParameters(parameters)
58 if (resultResponse.PaymentError == null)
134 else
147 }
148 }
149
150 // Se envian los valores de la comision configuradas.
151 resultResponse.ListParameters.Add(new Parameters
152 {
153     Action = EnumAction.Hidden, Field = ServiceParameterNames.CfgCommissionTax,
154     Value = transactionInfo.ServiceInfo.Commission.TaxRate,
155 });
156 resultResponse.ListParameters.Add(new Parameters
157 {
158     Action = EnumAction.Hidden, Field = ServiceParameterNames.CfgCommissionAmount,
159     Value = transactionInfo.ActualCommissionAmount,
160 }); // **** La comision es establecida por PROSA ****
161 resultResponse.ListParameters.Add(new Parameters

```

[Ver en pantalla completa](#)

Este valor es obtenido directamente de prosa, primero se realiza la solicitud de consulta y se obtiene la respuesta del servicio:




```

43 if (parameters.CodeProduct == 0)
50 return resultResponse;
51 }
52
53 using (var client = new ProsaserviceClient())
54 {
55     resultResponse = client.GetServiceInformation(parameters);
56 }

```

[Ver en pantalla completa](#)

	<h1>Incidencia – Comisión MX\$0 en servicios de tarjeta de débito</h1>		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

Luego se asigna directamente como el monto de comisión:

```

c# ProsaService.cs x
25 public class ProsaService : ProcessServices
39 public override async Task<ParametersResponse> GetServicePaymentsParameters(parametersRequest parameters)
58 if (resultResponse.PaymentError == null)
86
87 // Obtiene la comisión del integrador. El monto de la comisión que proporciona el integrador ya tiene el IVA incluido.
88 var commissionParameter =
89 resultResponse.ListParameters.FirstOrDefault(p => p.Field == ProsaParameterNames.Commission);
90
91 if (commissionParameter != null && Convert.ToDecimal(commissionParameter.Value) > 0)
92 transactionInfo.ActualCommissionAmount =
93 Convert.ToDecimal(commissionParameter.Value); // Comisión que se va a cobrar
94

```

[Ver en pantalla completa](#)

Las operaciones para este servicio solamente se loguean si existe un error en la respuesta del servicio o si en la configuración para la propiedad **Prosa.AutoLog** es **true** por lo que no es posible verificar lo que responde el servicio para esta operación:

```

c# ServicePayments.cs c# Guga21Service.cs c# ProsaService.cs c# ProsaServiceClient.cs c# Prosa.cs x
27 internal sealed class Prosa
434 public static List<Parameters> WS_COMISION(List<Parameters> parameters)
445 try
494 else
495 {
496 responseDescription = comisionResp.ResponseDescription;
497 }
498 }
499
500 if (responseCode != (int)ProsaResponseCode.SuccessfulProcess || Prosa.AutoLog)
501 {
502 Log.WriteInfo(ProsaService.SubModule, string.Format("WS_COMISION request : {0}", logRequest));
503 Log.WriteInfo(ProsaService.SubModule, string.Format("WS_COMISION response {0}: {1}", responseCode == (int)ProsaResponseCode.UnknownError ? "Un
504
505

```


[Ver en pantalla completa](#)

```

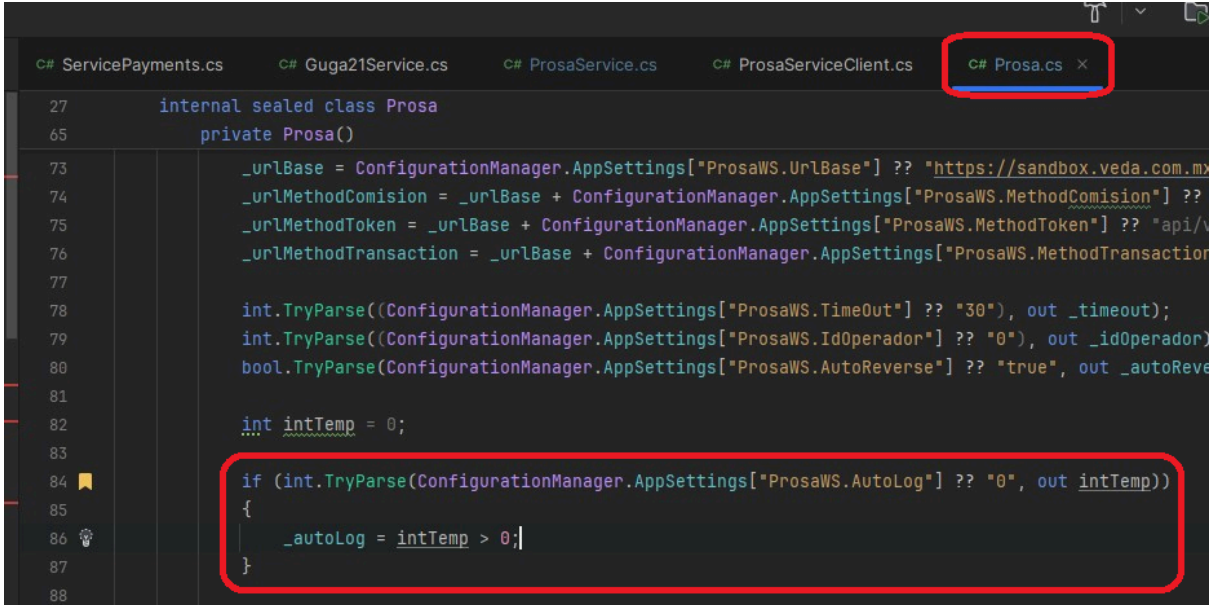
c# ServicePayments.cs c# Guga21Service.cs c# ProsaService.cs c# ProsaServiceClient.cs c# Prosa.cs x
27 internal sealed class Prosa
117 }
118
119 /// <summary>
120 /// Atributo que establece por configuracion si se maneja logeo
121 /// </summary>
122 public static bool AutoLog
123 {
124 get { return _prosa._autoLog; }
125 }
126

```

[Ver en pantalla completa](#)

	Incidencia – Comisión MX\$0 en servicios de tarjeta de débito		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

El valor de esta propiedad es establecida por la configuración **ProsaWS.AutoLog**, obtiene el valor de **false por defecto**, como se observa en la siguiente figura.




```

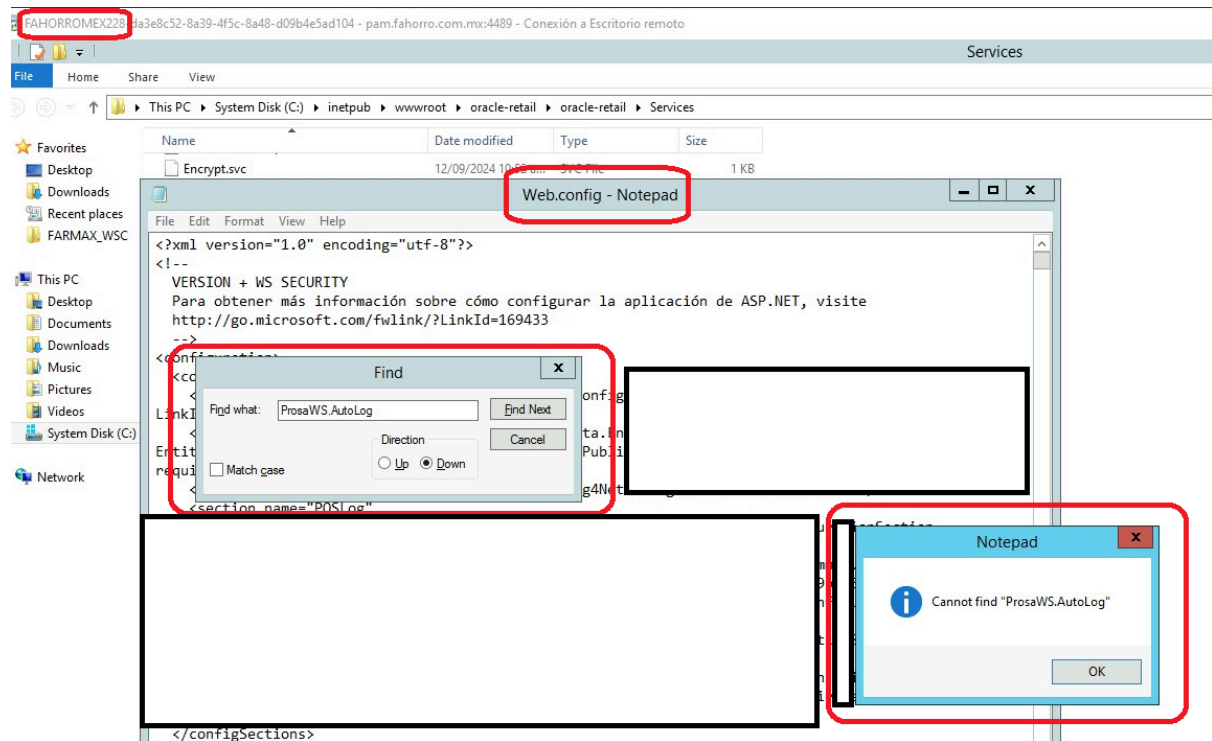
C# ServicePayments.cs  C# Guga21Service.cs  C# ProsaService.cs  C# ProsaServiceClient.cs  C# Prosa.cs x
27     internal sealed class Prosa
65     private Prosa()
73         _urlBase = ConfigurationManager.AppSettings["ProsaWS.UrlBase"] ?? "https://sandbox.veda.com.mx";
74         _urlMethodComision = _urlBase + ConfigurationManager.AppSettings["ProsaWS.MethodComision"] ??
75         _urlMethodToken = _urlBase + ConfigurationManager.AppSettings["ProsaWS.MethodToken"] ?? "api/\
76         _urlMethodTransaction = _urlBase + ConfigurationManager.AppSettings["ProsaWS.MethodTransaction
77
78         int.TryParse(ConfigurationManager.AppSettings["ProsaWS.TimeOut"] ?? "30", out _timeout);
79         int.TryParse(ConfigurationManager.AppSettings["ProsaWS.IdOperador"] ?? "0", out _idOperador);
80         bool.TryParse(ConfigurationManager.AppSettings["ProsaWS.AutoReverse"] ?? "true", out _autoReverse);
81
82         int _intTemp = 0;
83
84         if (int.TryParse(ConfigurationManager.AppSettings["ProsaWS.AutoLog"] ?? "0", out _intTemp))
85         {
86             _autoLog = _intTemp > 0;
87         }
88

```

[Ver en pantalla completa](#)


Según la configuración actual de los servicios, se toma como muestra el servidor **FAHORROMEX228**, no se establece explícitamente un valor para la configuración **ProsaWS.AutoLog** por lo que las solicitudes y respuestas de las operaciones que no tengan error no se loguean, es el caso para la incidencia.

	<h1>Incidencia – Comisión MX\$0 en servicios de tarjeta de débito</h1>		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024



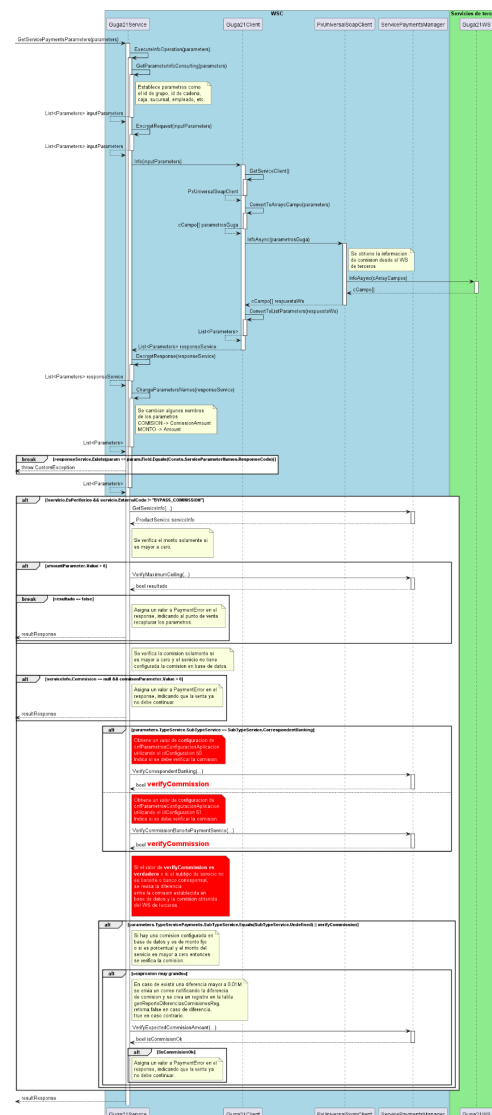
[Ver en pantalla completa](#)

Esto quiere decir que no es posible saber lo que se recibió de parte del servicio de prosa al momento de realizar las transacciones.

	Incidencia – Comisión MX\$0 en servicios de tarjeta de débito		
	Autor	Versión	Fecha de elaboración
Juan Pérez	1.0	27-09-2024	27-09-2024


Servicio Guga21

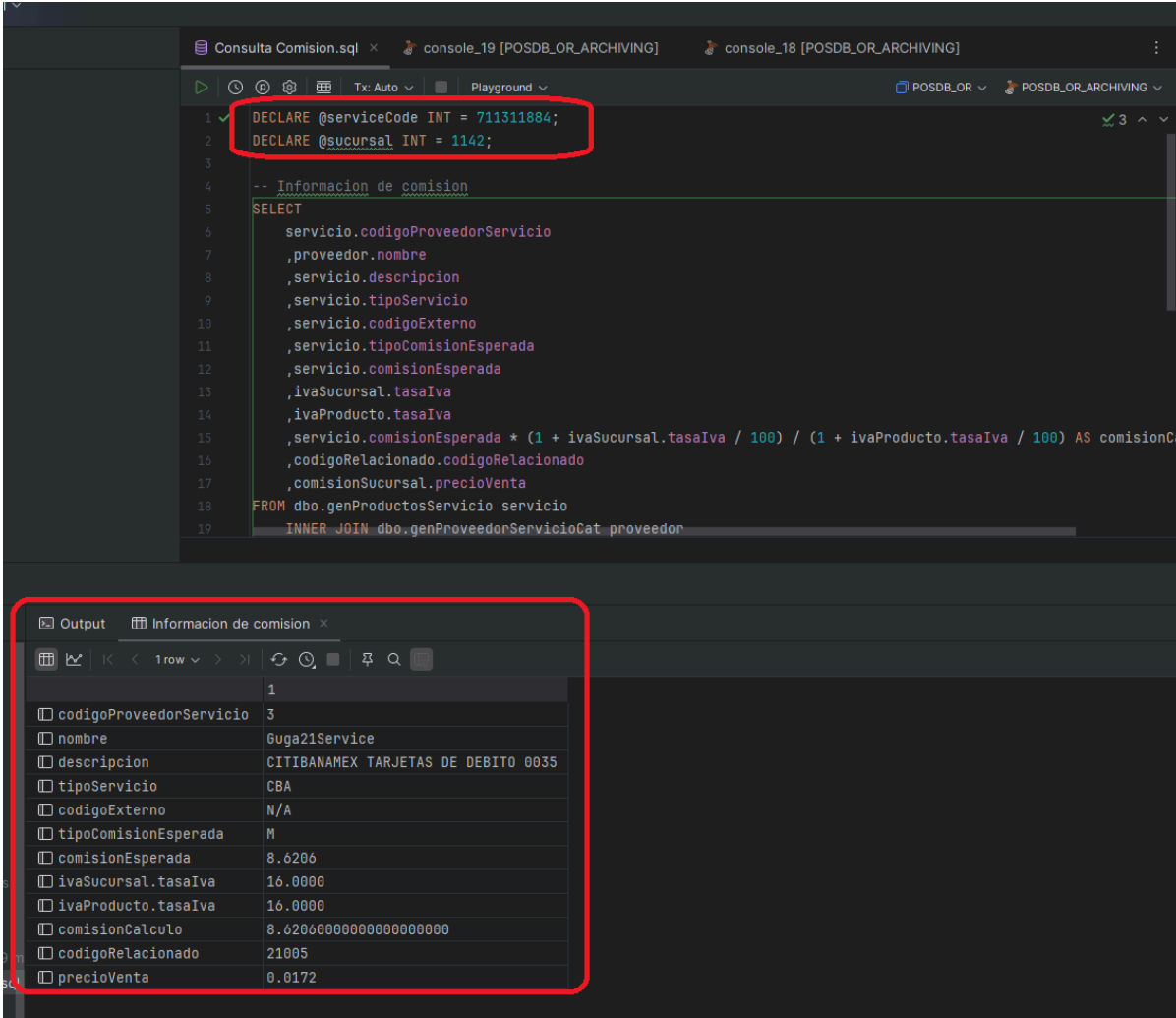
Para el caso del servicio **CITIBANAMEX TARJETAS DE DEBITO 0035**, el proveedor es **Guga21**, el flujo para consultar la información de comisión cambia, la implementación de este servicio se encuentra en **Guga21Service**, el flujo de consulta de comisión se expresa en el siguiente diagrama de secuencia técnico, se recomienda [ver en pantalla completa](#):



[Ver en pantalla completa](#)

Se observa que la validación depende de si existe una comisión configurada en base de datos, esto se puede verificar utilizando el script [Consulta Comision.sql](#), se utiliza la sucursal **1142** para realizar la consulta, es la sucursal que se toma como muestra, se observa que el servicio se encuentra correctamente configurado:

<div>  <div> Incidencia – Comisión MX\$0 en servicios de tarjeta de débito </div> </div>			
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024



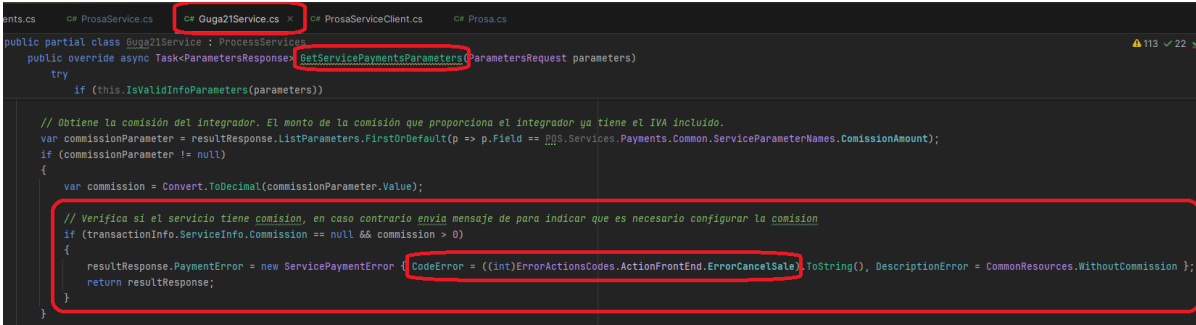
The screenshot shows a SQL query in a database tool. The query declares two variables: `@serviceCode INT = 711311884;` and `@sucursal INT = 1142;`. It then performs a SELECT query on the `dbo.genProductosServicio` table, joined with `dbo.genProveedorServicioCat`. The output shows a single row of data for the service 'Guga21Service'.

Column	Value
codigoProveedorServicio	3
nombre	Guga21Service
descripcion	CITIBANAMEX TARJETAS DE DEBITO 0035
tipoServicio	CBA
codigoExterno	N/A
tipoComisionEsperada	M
comisionEsperada	8.6206
ivaSucursal.tasaIva	16.0000
ivaProducto.tasaIva	16.0000
comisionCalculo	8.620600000000000000000000
codigoRelacionado	21005
precioVenta	0.0172

[Ver en pantalla completa](#)

Se encuentra que el servicio está correctamente configurado.

Si no hubiera configuración y la comisión fuera **mayor a 0**, se producirá un error, así como se expresa en el [diagrama](#) y en el código:




The screenshot shows a C# code snippet. A red box highlights a comment and a code block that checks if the commission is greater than 0. If it is, it returns a `ServicePaymentError` with the message 'Without Commission'.

```

// Verifica si el servicio tiene comision, en caso contrario envia mensaje de para indicar que es necesario configurar la comision
if (transactionInfo.ServiceInfo.Commission == null && commission > 0)
{
    resultResponse.PaymentError = new ServicePaymentError { CodeError = ((int)ErrorActionsCodes.ActionFrontEnd.ErrorCancelSale).ToString(), DescriptionError = CommonResources.WithoutCommission };
    return resultResponse;
}

```

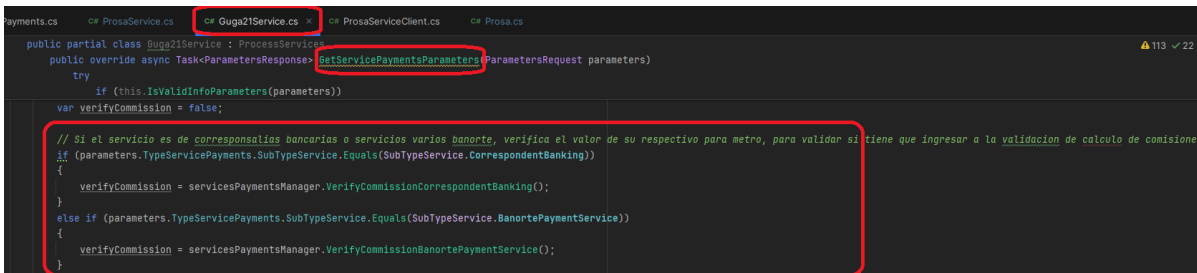
[Ver en pantalla completa](#)

<div>  <div> Incidencia – Comisión MX\$0 en servicios de tarjeta de débito </div> </div>			
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

Luego se consultan los parámetros de configuración de aplicación para determinar si se debe verificar la diferencia entre la comisión en base de datos y la comisión obtenida de **Guga21**, estos parámetros son:

- **VerifyCommissionCorrespondentBanking** con id **50**
- **VerifyCommissionBanortePaymentService** con id **51**

Como se puede apreciar en la siguiente figura:



```

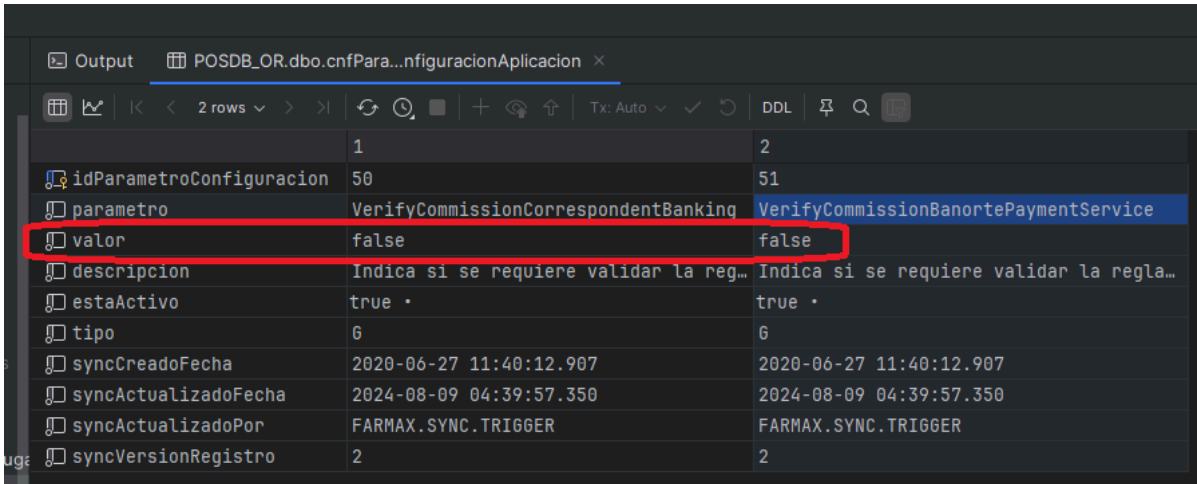
public partial class Guga21Service : ProcessService
{
    public override async Task<ParametersResponse> GetServicePaymentsParameters(ParametersRequest parameters)
    {
        try
        {
            if (this.IsValidInfoParameters(parameters))
            {
                var verifyCommission = false;

                // Si el servicio es de correspondencias bancarias o servicios varios banorte, verifica el valor de su respectivo parametro, para validar si tiene que ingresar a la validacion de calculo de comisiones
                if (parameters.TypeServicePayments.SubTypeService.Equals(SubTypeService.CorrespondentBanking))
                {
                    verifyCommission = servicesPaymentsManager.VerifyCommissionCorrespondentBanking();
                }
                else if (parameters.TypeServicePayments.SubTypeService.Equals(SubTypeService.BanortePaymentService))
                {
                    verifyCommission = servicesPaymentsManager.VerifyCommissionBanortePaymentService();
                }
            }
        }
        catch { }
    }
}

```

[Ver en pantalla completa](#)

Se pueden consultar con el script [Consulta parámetros guga21.sql](#):




idParametroConfiguracion	1	2
parametro	VerifyCommissionCorrespondentBanking	VerifyCommissionBanortePaymentService
valor	false	false
descripcion	Indica si se requiere validar la regla...	Indica si se requiere validar la regla...
estaActivo	true	true
tipo	6	6
syncCreadoFecha	2020-06-27 11:40:12.907	2020-06-27 11:40:12.907
syncActualizadoFecha	2024-08-09 04:39:57.350	2024-08-09 04:39:57.350
syncActualizadoPor	FARMAX.SYNC.TRIGGER	FARMAX.SYNC.TRIGGER
syncVersionRegistro	2	2

[Ver en pantalla completa](#)

Se encuentra que la verificación está desactivada tanto para banorte como para banco correspondencial.

Se observa que de este valor de esta configuración depende si se valida la comisión o no, en caso de que el integrador envíe un monto de MX\$0:

	<h1>Incidencia – Comisión MX\$0 en servicios de tarjeta de débito</h1>		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024


```

31  public partial class @Guga21Service : ProcessService
185  public override async Task<ParametersResponse> GetServicePaymentParameters(ParametersRequest parameters)
189  try
191  if (this.IsValidInfoParameters(parameters))
284  verifyCommission = servicesPaymentsManager.VerifyCommissionSanortePaymentService();
285  }
286
287  // Valida si se requiere pasar por calculo de comisiones
288  if (parameters.TypeService.Equals(SubTypeService.Undefined) || verifyCommission)
289  {
290  // Solo se valida si el servicio no manda error.
291  if (resultResponse.PaymentError == null)
292  {
293  if (commissionParameter != null)
294  {
295  transactionInfo.ActualCommissionAmount = Convert.ToDecimal(commissionParameter.Value); // Comisión que se va a cobrar
296  }
297
298  if (transactionInfo.ServiceInfo.Commission != null && (transactionInfo.ServiceInfo.Commission.CommissionType == BusinessLogic.ServicePayments.DTO.Enum.CommissionCalculationType.FixedAmount
299  || (transactionInfo.ServiceInfo.Commission.CommissionType == BusinessLogic.ServicePayments.DTO.Enum.CommissionCalculationType.Percentage && transactionInfo.ServiceInfo.ServiceAmount > 0)))
300  {
301  // Verifica si la comisión es el monto de la comisión es el que se espera de acuerdo al catálogo.
302  if (!servicesPaymentsManager.VerifyExpectedCommissionAmount(transactionInfo, transactionInfo.ActualCommissionAmount))
303  {
304  resultResponse.PaymentError = new ServicePaymentError
305  {
306  CodeError = ((int)ErrorActionsCodes.ActionFrontEnd.ErrorCancelSale).ToString(),
307  DescriptionError = CommonResources.InvalidCommission
308  };
309  }
310  }
311  }
312  }

```

[Ver en pantalla completa](#)

De otro modo se enviará un error al punto de venta y se cancelaría el proceso de la venta, también se envía un correo electrónico notificando que la comisión no está bien configurada y se guarda un registro en **genReporteDiferenciasComisionesReg**.

	Incidencia – Comisión MX\$0 en servicios de tarjeta de débito		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

Posibles causas


Según lo encontrado en código, lo más probable es que el servicio esté contestando con una comisión de **MX\$0**, esto se maneja de distintas maneras según el servicio en cuestión. **No es posible revisar las respuestas recibidas** por parte de estos servicios en la operación de consulta de información de comisión porque el logueo de operaciones sin error se encuentra desactivado para **Prosa** y en el caso de **Guga21** no existe un flujo que loguee la operación de consulta de información.

Servicio Prosa

Para el servicio de prosa, al recibir una comisión de **MX\$0**, simplemente se toma y se continúa procesando la transacción, no existe una validación que compare el valor de la comisión obtenida directamente de **Prosa** con el valor establecido en base de datos, solamente se utiliza la tasa de **IVA** que se encuentra en base de datos para calcular el **IVA** de la comisión que ya viene incluida en el monto obtenido directamente de **Prosa**.

Servicio Guga21

En el caso de guga21, se revisa si existe una diferencia con la comisión establecida en base de datos solamente si los parametros de configuracion de **50 y 51** están establecidos en **true** para validar la comisión de servicios de banco corresponsal y banorte, respectivamente, en caso de que el servicio no esté en ninguna de estas categorías, la comisión siempre se valida, el caso de la incidencia es banorte por lo que la validación no se realiza y se continúa procesando la transacción con un monto de **MX\$0**.

	Incidencia – Comisión MX\$0 en servicios de tarjeta de débito		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

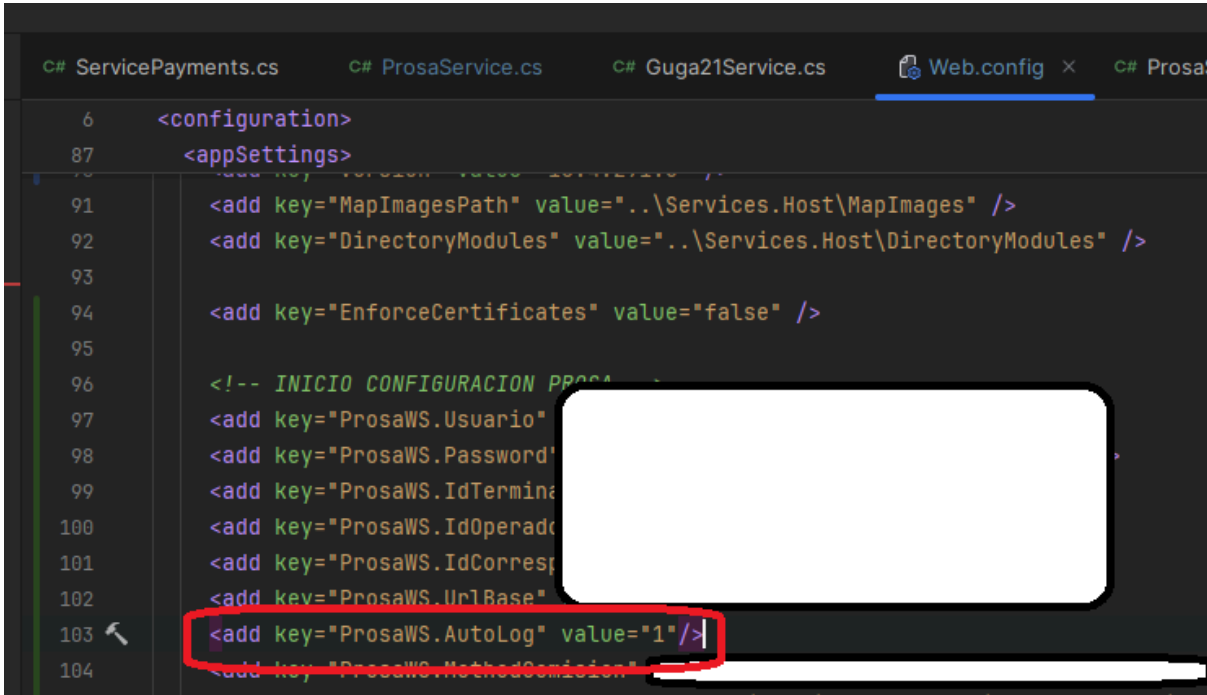
Posible solución

Partiendo de la suposición de que los servicios están contestando con el monto de **MX\$0** como comisión, se propone:

Activar el logueo de operaciones de Prosa

Se propone activar el logueo de todas las operaciones de **Prosa** y realizar nuevamente la prueba en producción para saber qué responde el servicio de **Prosa**.

Se debe establecer el parámetro de configuración **ProsaWS.AutoLog** en el **Web.config** de **POS.Services (WSC)**, con el valor de al menos 1:



```


6      <configuration>
87     <appSettings>
91         <add key="MapImagesPath" value="..\Services.Host\MapImages" />
92         <add key="DirectoryModules" value="..\Services.Host\DirectoryModules" />
93
94         <add key="EnforceCertificates" value="false" />
95
96         <!-- INICIO CONFIGURACION PROSA -->
97         <add key="ProsaWS.Usuario" value="..." />
98         <add key="ProsaWS.Password" value="..." />
99         <add key="ProsaWS.IdTerminal" value="..." />
100        <add key="ProsaWS.IdOperador" value="..." />
101        <add key="ProsaWS.IdCorrespondiente" value="..." />
102        <add key="ProsaWS.UrlBase" value="..." />
103        <add key="ProsaWS.AutoLog" value="1" />
104        <add key="ProsaWS.MethodComision" value="..." />
105        <add key="ProsaWS.MethodTarifa" value="..." />

```

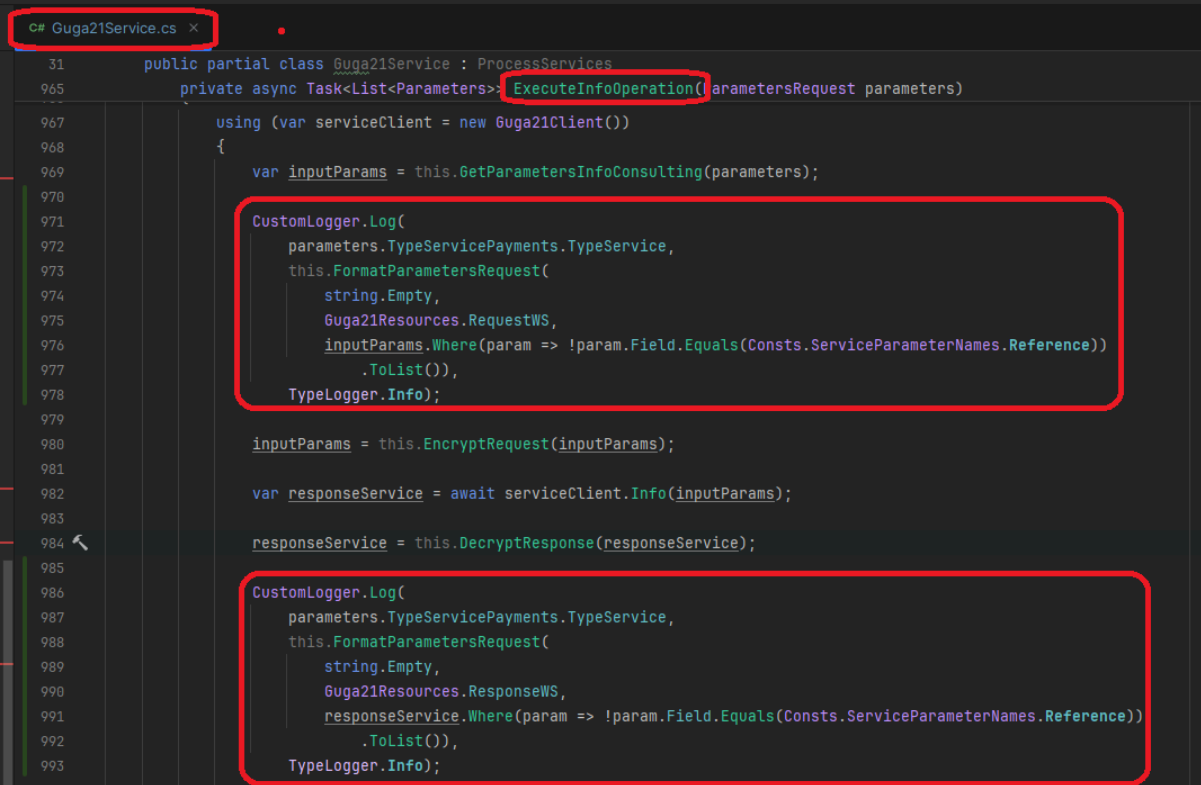
[Ver en pantalla completa](#)

Loguear operaciones de Guga21

Se propone loguear las operaciones de consulta de información de comisión de **Guga21** para conocer lo que responde el servicio y realizar nuevamente la prueba en producción.

	Incidencia – Comisión MX\$0 en servicios de tarjeta de débito		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Juan Pérez	1.0	27-09-2024	27-09-2024

Se puede realizar el siguiente cambio en el código con el propósito de realizar la prueba.



```

31 public partial class Guga21Service : ProcessServices
965 private async Task<List<Parameters>> ExecuteInfoOperation(parametersRequest parameters)
967 {
968     using (var serviceClient = new Guga21Client())
969     {
970         var inputParams = this.GetParametersInfoConsulting(parameters);
971
972         CustomLogger.Log(
973             parameters.TypeServicePayments.TypeService,
974             this.FormatParametersRequest(
975                 string.Empty,
976                 Guga21Resources.RequestWS,
977                 inputParams.Where(param => !param.Field.Equals(Consts.ServiceParameterNames.Reference))
978                     .ToList()),
979             TypeLogger.Info);
980
981         inputParams = this.EncryptRequest(inputParams);
982
983         var responseService = await serviceClient.Info(inputParams);
984
985         responseService = this.DecryptResponse(responseService);
986
987         CustomLogger.Log(
988             parameters.TypeServicePayments.TypeService,
989             this.FormatParametersRequest(
990                 string.Empty,
991                 Guga21Resources.ResponseWS,
992                 responseService.Where(param => !param.Field.Equals(Consts.ServiceParameterNames.Reference))
993                     .ToList()),
994             TypeLogger.Info);
995     }
996 }

```

[Ver en pantalla completa](#)