Developer API Guide

SecureLINK Developer API Guide

Version 1.0

Released October 9, 2013

Table of Contents

Changes and Modifications

The table below lists changes made to the SecureLINK Developer API Guide:

Version	Changes/Modifications	Pages
1.0	Document launch.	All

Chapter 1. Introduction

1.1. Supported Integration Methods

This document assists Point-of-Sale (POS) developers directly integrating with BridgePay PathwayLINK in using SecureLINK's Point-to-Point-Encryption (P2PE) transaction services, Card Tokenization services, and Card Vaulting services. BridgePay supports the following integration methods:

- .NET Web Services
- HTTPS POST
- SOAP

1.2. Testing

A test or live merchant account with BridgePay is necessary to successfully process transactions. To acquire a test account, contact the BridgePay Developer Support Department **at developersupport@bridgepaynetwork.com** or fill out the test account request form at **http://www.bridgepaynetwork.com/testAccount.php**.

1.2.1. P2PE Transaction Services

P2PE transaction services apply to swiped (card-present) transactions that use an encrypted card reader. Because the PIN block contains a clear account number and PIN, PIN-based Debit and EBT transactions require a peripheral capable of decrypting the clear account number from an encrypted card read. Please make sure your encrypted reader is injected with a key and security format that BridgePay supports. Please contact **developersupport@bridgepaynetwork.com** with any questions.

> ⓘ For additional information about peripherals and the supported encryption strategies, please refer to **Secure Formats** on page 17.

Chapter 2. Web Service Functionality

For development and staging purposes, BridgePay provides integrators with an .asmx page that contains online documentation and a testing form for each operation.

During integration, it may be helpful to process sample transactions using the test forms to get a feel for how the various operations work or to troubleshoot integration problems.

During development and staging, use the following URL:

https://gatewaystage.itstgate.com/SmartPayments/transact3.asmx

Once certified, process all live transactions against the following URL:

https://gateway.itstgate.com/SmartPayments/transact3.asmx

> Any gateway account that performs a request that uses a SecureLINK service must have privileges to use the SecureLINK feature.

The lists below detail which Web operation requests apply to the different SecureLINK transaction services:

P2PE Transaction Service Requests

- ProcessCreditCard
- ProcessDebitCard
- ProcessEBTCard
- ProcessGiftCard
- ProcessLoyaltyCard
- GenerateCardToken
- ManageCardVault

- ProcessCreditCard
- GenerateCardToken

Card Vaulting Service Requests

- ProcessCreditCard
- ManageCardVault

> SecureLINK services integrate with the PathwayLINK Transaction Processing services. Please refer to the **PathwayLINK Transaction Processing Developer API Guide** for more information on using the transaction processing Web service operations.

2.1. P2PE Transaction Service Requests

Point-to-Point Encryption (P2PE) transaction service requests require four additional elements within the **ExtData** parameter: **SecurityInfo**, **Track1**, **Track2**, and **SecureFormat**. If the request does not contain all four elements, the transaction processes as a clear card read (non-P2PE).

The following operations support P2PE: **ProcessCreditCard**, **ProcessDebitCard**, **ProcessEBTCard**, **ProcessGiftCard**, **ProcessLoyaltyCard**, **GenerateCardToken**, and **ManageCardVault**.

The following table contains parameter descriptions and additional requirements specific to P2PE requests.

Parameter	Description
CardNum	Pass with a null value. The service populates this parameter after decrypting and reconstructing the encrypted card data.
ExpDate	Pass with a null value. The service populates this parameter after decrypting and reconstructing the encrypted card data.
MagData	Pass with a null value. The service populates this parameter after decrypting and reconstructing the encrypted card data.
NameOnCard	Pass with a null value. The service populates this parameter after decrypting and reconstructing the encrypted card data.
ExtData	Optional unless otherwise noted below. Extended data in XML format. Valid values are: • **<SecurityInfo>***SecurityInfo***</SecurityInfo>** Required. Dynamic Unique Key Per Transaction (DUKPT) Key Serial Number (KSN) delivered with each transaction. • **<Track1>***Track1***</Track1>** Required. Valid values depend on the encryption strategy. When the value of the **SecureFormat** element is: - **MagneSafeV1**: Provide encrypted **Track1** data. - **MagneSafeV2**: Provide **Track1** data. - **SecureMagV2**: Provide encrypted **Track2** data. - **SecureMag**: Provide encrypted **Track1** and **Track2** data. • **<Track2>***Track2***</Track2>** Required when SecureFormat is **MagneSafeV1** or **MagneSafeV2**. When the value of the SecureFormat element is: - **MagneSafeV1**: Provide encrypted **Track2** data. - **MagneSafeV2**: Provide **Track2** data. - **SecureMagV2**: Provide encrypted **Track2** data.

	• **<SecureFormat>***SecureFormat***</SecureFormat>** Required. Encryption strategy used for decryption and reconstruction of MagData. Valid Values are: **SecureMag**, **MagneSafeV1**, **MagneSafeV2**, **SecureMagV2**. > These values are **case-sensitive**.

2.1.1. Examples

The following examples show different functions available with this Web service operation. Change the credentials and payment information fields when running test transactions.

2.1.1.1. ProcessCreditCard with MagneSafeV1 Encryption

The following example processes a credit sale transaction with a MagneSafeV1-encrypted read.

Parameter	Value
UserName	Test
Password	123
TransType	Sale
Amount	13.34
ExtData	<SecurityInfo>9012160B02B90E00000C</SecurityInfo><Track1>DAE1E6C2AAE1309642E5DA7E9FC37DA73C5CA61E4E98EBC6202250385316CA82EC9213A55AB507930369B7F0AC80F9E09F68A470EA028667</Track1><Track2>88C3E2DF745FBE0414AFF920FD62E11311057AD804B9FC17E4B935E2BF271390CFF7C05FDE7C9C80</Track2><SecureFormat>MagneSafeV1</SecureFormat>

Response

```
<?xml version="1.0" encoding="utf-8" ?>
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://TPISoft.com/SmartPayment/">
   <Result>0</Result>
   <RespMSG>Approved</RespMSG>
   <Message>AP</Message>
   <AuthCode>MC1334</AuthCode>
   <PNRef>107271</PNRef>
   <HostCode>0023</HostCode>
   <GetCommercialCard>False</GetCommercialCard>
   <ExtData>CardType=MASTERCARD,BatchNum=0060
      <BatchNum>0060</BatchNum>
```

```
      <ReceiptData>
           <Trans_Id>MCC1730340714</Trans_Id>
      </ReceiptData>
   </ExtData>
</Response>
```

2.1.1.2. GenerateCardToken with MagneSafeV2 Encryption

The following example generates a card token with a MagneSafeV2-encrypted read.

Parameter	Value
UserName	Test
Password	123
ExtData	<SecurityInfo>9012160B02B90E00000C</SecurityInfo><Track1>>%E240A1B04162E52E8^TGATE/TESTCARD^500011910000E907CBB5</Track1><Track2>;4012000008000026=121010110000004 56789?</Track2><SecureFormat>MagneSafeV2</SecureFormat>

Response

```
<?xml version="1.0" encoding="utf-8" ?>
<TokenResponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://TPISoft.com/SmartPayments/">
   <Result>100</Result>
   <Message>DigestExists</Message>
   <Token>5424188727421732</Token>
</TokenResponse>
```

2.1.1.3. ManageCardVault with SecureMag Encryption

The following example creates a card vault record and generates a card token with a SecureMag-encrypted read.

Parameter	Value
UserName	Test
Password	123
Transaction	Create
ExtData	<SecurityInfo>9012160B02B90E00000C</SecurityInfo><Track1>EA02EB27DAFC3BE1AE43C318D96F39E9AB90D55B0978614D7860C895721DD0E9EA358FDDCA4A0FAADE3D1E0E69D91FCB31B70E73B8FB5E49F4E4AF6219AB9B6F75CFFF8A6385DE6645426EC7E9DDD128C7D63E662B2C6E969E3CEE75789F66C48251B69A6B9C79CDC570FD984867ACA2</Track1><SecureFormat>SecureMag

Parameter	Value
	</SecureFormat>

Response

```
<?xml version="1.0" encoding="utf-8" ?>
<CardVaultResponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://TPISoft.com/SmartPayments/">
   <Result>0</Result>
   <Message>Success</Message>
   <Token>5424188727421732</Token>
   <CustomerPaymentInfoKey>53</CustomerPaymentInfoKey>
   <ExpDate>1012</ExpDate>
   <NameOnCard>TGATE/TESTCARD</NameOnCard>
   <Street />
   <Zip />
</CardVaultResponse>
```

2.2. Card Vaulting Service Requests

The following section discusses tokenization and managing card vault information using a single request.

2.2.1. ManageCardVault

This Web service operation manages card tokenization and card vault information **independently** of a transaction. This operation can create or manage card tokens and vault records using a single request.

This URL to access this Web service is:

https://gatewaystage.itstgate.com/smartpayments/transact3.asmx?op=ManageCardVault

The following table contains parameter descriptions.

Parameter	Description
Username	Required. Username assigned in the payment server.
Password	Required. Password for the username assigned in the payment server.
Transaction	Required. Type of transaction. Valid Values are **Create** and **Update**.
CustomerPaymentInfoKey	Required for **Update** TransType. This is the unique identifier for the vault record for the merchant.
Token	Required for **Update** TransType. The token is associated with customer payment info key.

Parameter	Description
CardNumber	Required for **Create** TransType. Credit card number to tokenize and vault.
ExpDate	Required. Expiration date for the credit card number.
NameOnCard	Optional. The name of the card holder.
Street	Optional. The card holder's billing address.
Zip	Optional. The card holder's ZIP code.
ExtData	Required for P2PE transactions. Extended data in XML format. Valid Values are: • **<SecurityInfo>***SecurityInfo***</SecurityInfo>** Key Serial Number of P2PE transaction. • **<Track1>***Track1***</Track1>** Encrypted Track1 of P2PE transaction. • **<Track2>***Track2***</Track2>** Encrypted Track2 of P2PE transaction. • **<SecureFormat>***SecureFormat***</SecureFormat>** Secure format of P2PE transaction.

2.2.1.1. Examples

The following examples show different functions available with this Web service operation. Change the credentials and payment information fields when running test transactions.

2.2.1.1.1. Tokenization and Vault Record with Clear Card Read

The following example creates a token and vault record with a clear card read.

Parameter	Value
UserName	Test
Password	123
Transaction	Create
CardNumber	4012000646380026
ExpDate	0914
NameOnCard	John Doe
Street	1234 Main Street
Zip	45678

Response

```
<?xml version="1.0" encoding="utf-8" ?>
<CardVaultResponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://TPISoft.com/SmartPayment/">
   <Result>0</Result>
   <Message>Success</Message>
   <Token>4012000646380026</Token>
   <CustomerPaymentInfoKey>54</CustomerPaymentInfoKey>
   <ExpDate>0914</ExpDate>
   <NameOnCard>John Doe</NameOnCard>
   <Street>1234 Main Street</Street>
   <Zip>45678</Zip>
</CardVaultResponse>
```

2.2.1.1.2. Tokenization and Vault Record with SecureMag Encryption

The following example creates a card vault record and generates a card token with a SecureMag-encrypted read.

Parameter	Value
UserName	Test
Password	123
Transaction	Create
ExtData	<SecurityInfo>9012160B02B90E00000C</SecurityInfo><Track1>EA02EB27DAFC3BE1AE43C318D96F39E9AB90D55B0978614D7860C895721DD0E9EA358FDDCA4A0FAADE3D1E0E69D91FCB31B70E73B8FB5E49F4E4AF6219AB9B6F75CFFF8A6385DE6645426EC7E9DDD128C7D63E662B2C6E969E3CEE75789F66C48251B69A6B9C79CDC570FD984867ACA2</Track1><SecureFormat>SecureMag</SecureFormat>

Response

```
<?xml version="1.0" encoding="utf-8" ?>
<CardVaultResponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://TPISoft.com/SmartPayments/">
   <Result>0</Result>
   <Message>Success</Message>
   <Token>5424188727421732</Token>
   <CustomerPaymentInfoKey>53</CustomerPaymentInfoKey>
   <ExpDate>1012</ExpDate>
   <NameOnCard>TGATE/TESTCARD</NameOnCard>
   <Street />
   <Zip />
</CardVaultResponse>
```

2.2.1.1.3. Update Token and Vault Record

The following example updates the card vault data for the record and token specified.

Parameter	Value
UserName	Test
Password	123
Transaction	Update
CustomerPaymentInfoKey	54
Token	4012000646380026
ExpDate	0914
NameOnCard	Jane Doe
Street	1234 Main Street
Zip	45678

Response

```
<?xml version="1.0" encoding="utf-8" ?>
<CardVaultResponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://TPISoft.com/SmartPayment/">
   <Result>0</Result>
   <Message>Success</Message>
   <Token>4012000646380026</Token>
   <CustomerPaymentInfoKey>54</CustomerPaymentInfoKey>
   <ExpDate>0914</ExpDate>
   <NameOnCard>Jane Doe</NameOnCard>
   <Street>1234 Main Street</Street>
   <Zip>45678</Zip>
</CardVaultResponse>
```

2.2.2. Tokenization and Card Vaulting in a Transaction

You can also manage tokenization and card vault information by passing the **CardVault** tag and its nested elements within the ExtData of a transaction request.

The **ProcessCreditCard** Web service operation supports this functionality.

The following table contains parameter descriptions.

Parameter	Description
ExtData	**<CardVault>** Required. Introduces information and directions for card vaulting functionality. **<Transaction>***Transaction***</Transaction>** Required. The type of transaction. Valid values are: **Create**, **Read.**

Parameter	Description
	<CustomerPaymentInfoKey>*CPIK***</CustomerPaymentInfoKey>** Required for **Read**. Unique identifier for the vault record in use. **<Token>***Token***</Token>** Required for **Read**. Token associated with vault record in use. **<ExpDate>***ExpDate***</ExpDate>** Required for **Read**. Expiration date for the card associated with the vault record in use. **</CardVault>**

2.2.3. Examples

The following examples show different functions available with this Web service operation. Change the credentials and payment information fields when running test transactions.

2.2.3.1. Credit Sale with Tokenization and Card Vaulting

The following example creates a token and card vault record within a credit sale transaction using a clear card read.

Parameter	Value
UserName	Test
Password	123
TransType	Sale
CardNumber	6011001811520009
ExpDate	1012
MagData	371449635398431=05121015432112345678
Amount	5.52
ExtData	<CardVault><Transaction>Create</Transaction></CardVault>

Response

```
<?xml version="1.0" encoding="utf-8" ?>
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://TPISoft.com/SmartPayment/">
   <Result>0</Result>
   <RespMSG>Approved</RespMSG>
   <Message>AP</Message>
   <AuthCode>DI0552</AuthCode>
   <PNRef>112197</PNRef>
   <HostCode>0032</HostCode>
```

```
   <GetCommercialCard>False</GetCommercialCard>
   <ExtData>CardType=DISCOVER,BatchNum=0060
      <BatchNum>0060</BatchNum>
      <ReceiptData>
            <Trans_Id>055120935100513</Trans_Id>
            <Val_Code>0100</Val_Code>
      </ReceiptData>
      <CardVault>
            <Result>0</Result>
            <Token>6011001811520009</Token>
            <CustomerPaymentInfoKey>57</CustomerPaymentInfoKey>
            <ExpDate>1012</ExpDate>
      </CardVault>
   </ExtData>
</Response>
```

2.2.3.2. Credit Sale Using Card Vault Information

The following example processes a credit sale using the card vault record and token for customer and payment information.

Parameter	Value
UserName	Test
Password	123
TransType	Sale
Amount	6.08
ExtData	<CardVault><Transaction>Read</Transaction><CustomerPaymentInfoKey>57</CustomerPaymentInfoKey><Token>6011001811520009</Token><ExpDate>1012</ExpDate></CardVault>

Response

```
<?xml version="1.0" encoding="utf-8" ?>
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://TPISoft.com/SmartPayment/">
   <Result>0</Result>
   <RespMSG>Approved</RespMSG>
   <Message>AP</Message>
   <AuthCode>DI0608</AuthCode>
   <PNRef>112206</PNRef>
   <HostCode>0032</HostCode>
   <GetCVResult>S</GetCVResult>
   <GetCVResultTXT>Service Not Requested</GetCVResultText>
   <GetCommercialCard>False</GetCommercialCard>
   <ExtData>CardType=DISCOVER,BatchNum=0060
      <BatchNum>0060</BatchNum>
```

```
      <ReceiptData>
            <Trans_Id>55120931100516</Trans_Id>
            <Val_Code>0100</Val_Code>
      </ReceiptData>
   </ExtData>
</Response>
```

Chapter 3. Secure Formats

The SecureLINK decryption service supports three encryption strategies from two major peripheral manufacturers, IDTECH (SecureMag) and MagTek (MagneSafe 1.0, MagneSafe 2.0).

SecureLINK is agnostic to the actual type of peripheral device. SecureLINK accepts transactions from any peripheral that employs an accepted encryption strategy.

The following sections provide notes specific to each supported encryption strategy.

For more information regarding the data format and structure for your peripheral's encryption strategy, please see the **Device Drivers and Documentation** appendix for links to manufacturer resources.

3.1. IDTECH

3.1.1. SecureMag & SecureMagV2

With SecureMag, ISO/ABA cards send both clear and encrypted data. All other cards only send clear data.

From the masked track data, the following elements are available for local checking:

- BIN recognition and lookup (4 digits)
- Name field
- Last 4 digits for receipt printing

For more information regarding the structure and format of the data coming off of your peripheral, please refer to **Device Drivers and Documentation** appendix for links to manufacturer resources.

3.2. MagTek

3.2.1. MagneSafe 1.0

With MagneSafe 1.0, ISO/ABA cards send both clear and encrypted data. All other cards only send clear data.

From the masked track data, the following elements are available for local checking:

- BIN recognition and lookup (4 digits)
- Name field
- Last 4 digits for receipt printing

For more information regarding the structure and format of the data coming off of your peripheral, please refer to **Device Drivers and Documentation** appendix for links to manufacturer resources.

3.2.2. MagneSafe 2.0

MagneSafe 2.0 follows a format preservation encryption method to solve the problem of encrypting data while maintaining data formats compatible with legacy software and network protocols.

With this format preservation, the data available for local checking includes:

- BIN recognition and lookup (6 digits)
- Name field
- Last 4 digits for receipt printing
- PAN check digit routine
- Expire data check
- Service Code check
- PIN offset, language codes, and discretionary data
- Normalcy checking (e.g., track lengths, content, SS/EE characters)

For ISO/ABA swiped cards, the format code in Track1 is **E**, signifying this is a MagneSafe 2.0 message and that both Track1 and Track2 read without error. Track2 data is masked.

In order to pass SecureLINK encryption, you process the following elements from the swipe:

- Track1
- Track2
- SecurityInfo (DUKPT KSN)

> Though encrypted Track1 and encrypted Track2 are also available, these elements represent MagneSafe 1.0 encryption strategy.

For more information regarding the structure and format of the data coming off of your peripheral, please refer to **Device Drivers and Documentation** appendix for links to manufacturer resources.

3.2.3. Magnesa.NET Pass-Though Service

SecureLINK supports a pass-through to the Magensa.NET decryption service for MagneSafe 1.0 and 2.0-encrypted transaction requests that do not validate against the KSID lookup for BridgePay's decryption service. In most cases, the device is injected with a factory Magensa.NET key, and BridgePay will forward the decryption to Magnesa.

If your devices have a Magensa.NET key injected in them, follow the same format as a regular Magnesa decryption request. SecureLINK has the intelligence to identify and forward transactions that have the Magnesa.NET key. After decryption, SecureLINK puts the clear card information into the transaction and forwards it to PathwayLINK for transaction processing.

A. Appendix

A.1. Device Drivers and Documentation

Please note, for some manufacturers, you may need a non-disclosure agreement (NDA) directly with them to access their documentation and downloads.

A.1.1. IDTECH

To access IDTECH documentation and drivers, please visit:

http://www.idtechproducts.com/download/swipe-readers/cat_view/94-swipe-readers/110-securemag-magstripe-reader--idre-series-.html

A.1.2. MagTek

To access MagTek documentation and drivers, please visit:

http://www.magtek.com/support/documentation/sec_card_read_auth.asp

http://www.magtek.com/support/software/programming_tools/

A.2. SecureLINK-Specific Error Codes

The response returned when processing a Transact.asmx Web service operation (e.g., ProcessCreditCard), contains a **Result** response field. The payment processor returns Result values for approvals (0) and declines (12, 13). All other values for Result are error codes from the payment gateway.

The following table contains the descriptions for error codes that are specific to SecureLINK processing. When programmatically validating a transaction's result, you should use this value instead of any other response message fields (e.g., **RespMSG**, **Message**) describing the result.

Value	Result Code	Description
501	Decryption Failure	Unable to decrypt secure transaction due to invalid data or invalid decryption keys.
502	Unregistered KSID	Secure device's ID is not registered as a SecureLINK device.
503	InvalidSecureFormat	Unrecognized or missing SecureFormat in ExtData.
504	InvalidSecureTrack1	Invalid or missing Track1 in ExtData.
505	InvalidSecureTrack2	Invalid or missing Track2 in ExtData.
506	InvalidSecurityInfo	Invalid or missing SecurityInfo in ExtData.

Value	Result Code	Description
507	NeedSecureLinkAccount	User is not authorized for SecureLINK transactions.
508	NeedSecureLinkEULA	User has not yet signed the SecureLINK license agreement.
509	InvalidDecryptedTrack1	Track1 decrypted OK but is malformed. Causes: Invalid decryption keys or SecureFormat.
510	InvalidDecryptedTrack2	Track2 decrypted OK but is malformed.