



BridgeComm Interface API Guide

BridgeComm Interface API Guide

Version 2.3.7

October 23, 2015

Copyright © 2011-2015, BridgePay Network Solutions, Inc. All rights reserved.

The information contained herein is the confidential and proprietary property of BridgePay Network Solutions, Inc. and may not be used, distributed, modified, disclosed, or reproduced without the express written permission.

Table of Contents

Chapter 1. Introduction	4
1.1. Overview	4
1.2. Overall Process	6
1.3. Configuration.....	6
1.4. Transaction Request Types.....	6
1.5. Client Identifiers	7
1.6. Response Handling and Pass-Through Data	7
1.7. Merchant Lookup	7
1.8. Password Management.....	9
1.9. Hardware Encryption	10
1.10. Testing.....	11
1.11. Purchase Tokens.....	11
Chapter 2. Implementation	12
2.1. Web Service Operation & Data Contracts	12
2.1.1. Base Request.....	12
2.1.2. Base Response	13
2.2. Specific Request Layouts	14
2.2.1. One-Time Token (Request Type 000).....	14
2.2.1.1. Response	16
2.2.2. Multi-Use Token (Request 001)	17
2.2.2.1. Response	19
2.2.3. Multi-Use Token Security Code (Request 002)	20
2.2.3.1. Response	21
2.2.4. Generate Encryption Key (Request 003).....	22
2.2.4.1. Response	23
2.2.5. Authorization (Request 004)	24
2.2.5.1. Response	32
2.2.6. BIN Lookup (Request 005)	35
2.2.6.1. Response	36
2.2.7. Check Password Expiration (Request 006).....	37
2.2.7.1. Response	38
2.2.8. Change Password (Request 007).....	39
2.2.8.1. Response	40
2.2.9. Get Merchant Info (Request 011)	41
2.2.9.1. Response	42
2.2.10. Void/Refund (Request 012)	43
2.2.10.1. Response	45
2.2.11. Tokenize Account (Request 013)	46
2.2.11.1. Response	47
2.2.12. Capture (Request 019).....	48
2.2.12.1. Response	50
2.2.13. Initiate Settlement (Request 020).....	51
2.2.13.1. Response	52
2.2.14. Manage Gift Card (Request 022).....	53
2.2.14.1. Response	55
2.2.15. Ping (Request 099).....	56
2.2.15.1. Response	57
2.3. Generic Error Response.....	58

A. Appendix	59
A.1. ResponseCode Values	59
A.2. Base64-Encoded SOAP Message Sample	63
A.3. Including Merchant Information.....	64
A.4. Industry Specific Fields.....	65
A.5. Track Data Handling.....	74
A.6. Purchase Tokens	75
A.7. EMV Processing.....	77
A.8. Non Gateway Transactions (Check/Cash)	79
A.9. Check21 Processing.....	81
A.10. TPI Emulation	82

Changes and Modifications

The table below lists changes made to the BridgeComm Interface API Guide:

Version	Changes/Modifications	Pages
2.0	Document re-write and reorganization	All
2.1	Additional LEVEL II / LEVEL III elements documented in request type 004	pp. 18 - 24
2.1.2	SecureLink Fields and Process Added	Page 18
2.1.2	Explanation of Level III acceptance	Page 19
2.1.3	Refinement of Level II Item Fields	pp. 22 – 25
2.1.3	Added Appendix A.4 Merchant Lookup fields	Page 49
2.1.4	Added Request Types 011, 012 & 019	pp. 8, 43 – 48
2.1.5	Removed deprecated/disused request types.	
2.2.0	New Release Documentation	
2.3.0	New Release Documentation	
2.3.1	Corrected lodging formatting section	53
2.3.2	Corrected MSRKey to MSRKS, Added VoiceAuthCode element, Added Encryption Key Request data and Healthcare Fields, changes to Lodging	8, 12-27, 41, 51-55
2.3.3	Added PartialAuthorization element and Track Data Handling change information	12-13, 15-16, 22-23, 25, 27, 56
2.3.4	Added ACH Tokenization Method (013), Added authorize by WalletPaymentMethodID feature. Expanded Transaction Industry to include SEC codes for ACH. Expanded Lodging Information. Included information regarding Purchase Tokens.	4, 10, 12, 23, 24, 27-29, 38, 45, 46, 59, 60, 64, 65
2.3.5	Added EMV, Check Verification and Cash support.	24,26,28-30, 67-71
2.3.6	Added Gift Card Management Request (022) for future release.	53-55
2.3.7	Added TPI Emulation documentation, Car Rental fields, Ping Method (099). Modified Amount fields, removed Currency Code on requests, added Contract Id and corrected Lodging Field information and Response Codes.	4, 6, 12, 24-34, 59-62, 65-71, 82

Chapter 1. Introduction

1.1. Overview

The BridgePay Communication API provides an Internet-facing web services interface for requesting a variety of payment related services in real-time. The interface is a Windows Communication Foundation (WCF) web service using the request/response architecture. It provides five **access methods** that allow for different formatting options for interacting with the core communication library, and the ability to request a one-time purchase token for more secure access:

Access Method	Description
string ProcessRequest (string requestMsg) (https://url/PaymentService/RequestHandler.svc)	This access method accepts a base64 encoded XML formatted request message and returns a base64 encoded XML formatted response.
string ProcessJsonRequest (string requestMsg) (https://url/PaymentService/RequestHandler.svc)	This access method accepts a base64 encode JSON formatted request message and returns a base64 encoded JSON formatted response.
string ProcessMessage (https://url/PaymentService/Default.aspx?...)	This access method accepts an HTML formatted query string with parameters and returns an HTML formatted response, transformed by a client selected XSLT transform.
string AcquirePurchaseToken (string userName, string password, string certificationId, int transactionAmount, string purchaserInfo, string transactionInfo) (https://url/PaymentService/ActionService.svc)	This access method access specific parameters to pre-authenticate a purchase. When used, the non-sensitive details of a transaction are used to create a one-time purchase token that can be sent with the transaction request. A GUID defining the purchase token is returned upon a successful request.
Response ProcessCreditCard (string UserName, string TransType, string CardNum, string ExpDate, string MagData, string NameOnCard, string Amount, string InvNum, string PNRef, string ZIP, string Street, string CVNum, string ExtData) (https://url/PaymentService/TPIRequestHandler.asmx)	This access method provides limited functionality for migrating from the legacy TPI gateway to the new BridgePay gateway. This method must not be used for new integrations, especially for those interested in using EMV.

Merchant Identification

A core component of the transaction processing cycle is identifying the merchant account used for the authorization action. The merchant account to be used can be identified by a variety of methods. This process and the method used are referred to as the “merchant lookup strategy” and the “merchant lookup pattern”. The strategy selected defines the pattern used. A strategy can be as simple as “client code” or as complex as “client code, store ID, & terminal ID”. The system is designed to support whatever method the calling system or service needs to be able to manage the merchant accounts being utilized. The details of the merchant lookup strategy is defined and discussed during your implementation process.

Pass-Through Fields

In addition, the service request system supports the submission of extraneous, non-payment data elements that are submitted specifically as “pass through” information to be used for reporting and accounting purposes. These elements may include items like: invoice number, operator ID, service code, or any other reporting data that an interfacing system might need.

Request/Response Structure

Requests are formatted and sent to the appropriate handler/processor, and BridgePay sends a corresponding response in real-time. BridgePay returns responses to the requesting system as a web service response message. The response includes the unique transaction identifier, request type, response code, and may include a tokenized representation of payment information, BIN response, or other data elements based on the initial request type.

1.2. Overall Process

The general process cycle for a submitted service request is as follows:

- Web Service request received by BridgePay.
- Superficial evaluation of request performed to determine general “properness” and to retrieve client access credentials.
- Client access credentials used to validate client access, assign proper access rolls and to access client configuration for details on request and response formats.
- Request message evaluated based on request format details.
- Request processed.
- Results of request are formatted based on response message format details.
- Response message returned to requesting system.

1.3. Configuration

In order for any system to successfully communicate with the BridgePay gateway, the system (or user) must be setup for access. This includes assigning the system (or user) a user ID, a password and defining the type of access that the system (or user) will need. Multiple users at a site could share a common configuration but it is not required. Each connection profile will contain the supported access options and the appropriate response format.

1.4. Transaction Request Types

BridgeComm is designed to support a variety of services related to payment processing. These services are defined as request types. Depending upon the access method used, some request types may not be available. The following table lists the currently available services, the request types used to access the service and which access methods support them.

Service (and service code)	Request	XML	JSON	Query String	TPI
One Time Use Token	000	✓			
Multi-Use Token	001	✓		✓	
Multi-Use Token CSV	002	✓			
Process Payment	004	✓	✓	✓	✓
Is Debit (BIN lookup)	005	✓		✓	
Seconds Until Password Expires	006	✓			
Change Password	007	✓			
Get Merchant Data	011	✓		✓	
Void/Refund	012	✓	✓	✓	✓
Tokenize Account	013	✓			
Capture	019	✓			✓
Initiate Settlement	020	✓			
Manage Gift Card	022				
Ping	099	✓			✓

1.5. Client Identifiers

Before any processing can begin, BridgeComm must identify the client used to access the system. This is provided by a field called the **ClientIdentifier** field. This field does not indicate a user, merchant or other processing entity of that nature. Instead, it identifies an integration methodology and the rights and privileges therein.

Each Client Identifier defines what request types are available, where and how to retrieve merchant lookup information, and what specific access method is permitted.

Your Client Identifier will be provided to you by BridgePay before you begin testing on the BridgeComm test platform.

1.6. Response Handling and Pass-Through Data

In some cases, specific response formats are generated to meet client needs. Depending on the integration method used, different options are available for response handling.

ProcessMessage

For those integrating to BridgeComm using the ProcessMessage access method, a custom XSLT will be generated that will transform the data into the style and format you require. Only the standard response elements are available for those integration via this access method.

ProcessRequest/ProcessJsonRequest

For those integrating to BridgeComm using the ProcessRequest or ProcessJsonRequest access methods, clients may require that the response contain specific information that was included in the proceeding request. This type of information is referred to as “pass-through data”. BridgeComm can be instructed to return pass-through data to the requesting system in every successful response. This feature is currently available for the **ProcessRequest** and **ProcessJsonRequest** access methods only. Details on how to instruct BridgeComm to return these “pass-through” data elements can be found later in this document.

1.7. Merchant Lookup

An important feature of the BridgePay gateway is its ability to store customer merchant account information (on behalf of customers) for transaction processing and settlement. The gateway supports storing merchant account information for each customer that uses BridgePay services.

As the gateway has no knowledge of the internal structure of any of the systems that will be utilizing gateway services, a flexible merchant account location solution has been developed. As part of initial setup and configuration, each service solution that will make use of the BridgePay gateway works with the technical team to define their merchant account lookup key (or strategy). This key can be comprised of as many components as is required to support the granularity needed by the utilizing system. The key can be comprised of existing data elements or can be extensions that only exist to facilitate merchant account lookup. The only requirement is that the lookup components must be supplied in each service call.

You can also find more information about Merchant Lookup in Appendix A.3.

Examples of various lookup strategies are as follows:

- ClientIdentifier
- ClientIdentifier : CustomerCode
- ClientIdentifier : CustomerCode : ProductType
- StoreId : TerminalID
- ClientIdentifier : StoreId : ProductType

The data elements used for merchant account lookup are entirely flexible and defined by the customer during implementation.

EXAMPLES

To process an authorization via the ProcessRequest access method for a merchant whose lookup strategy uses the ClientIdentifier:Gateway:MerchantKey merchant lookup pattern, the RequestMessage element would have the following tags embedded within it:

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  .
  .
  .
  <requestMessage>
    <Gateway>XXX</Gateway>
    <MerchantKey>XXX</MerchantKey>
    <MerchantPassword>XXX</MerchantPassword>
  </requestMessage>
</requestHeader>
```

To process an authorization via the ProcessMessage access method for a merchant whose lookup strategy uses the ClientIdentifier:Gateway:MerchantKey merchant lookup pattern, parameters would be added to the query string:

```
https://url/PaymentService/Default.aspx? ... &ClientIdentifier=XXX&Gateway=XXX&MerchantKey=XXX& ...
```


To process an authorization via the `ProcessJsonMessage` access method for a merchant whose lookup strategy uses the `ClientIdentifier:Gateway:MerchantKey` merchant lookup pattern, parameters would be added to the JSON object:

```
{
  "dataElements": [
    .
    .
    .
    { "name": "clientidentifier", "value": "XXX" },
    { "name": "gateway", "value": "XXX" },
    { "name": "merchantkey", "value": "XXX" },
    .
    .
    .
  ]
}
```

For those customers you are not using specialized lookup patterns, the BridgePay merchant id and merchant account id must be provided on every message. These are provided in the MyBridgePay product at the time of boarding. All of the examples shown in this documentation will assume that the client is sending merchant lookup information using these IDs. If your implementation is customized with a different merchant lookup pattern, these parameters will not be necessary.

1.8. Password Management

In accordance with industry best practices, BridgePay users should change their passwords a minimum of every 90 days. BridgeComm provides service requests to support these user maintenance activities.

- **Check Password Expiration Request (006)** – Returns the number of seconds remaining before the current user's password will expire.
- **Update Password Request (007)** – Updates the user's password and resets the expiry timer.

If the user's password is not changed before it expires, the next time that user makes a request through BridgeComm, the request will fail and return a response indicating that the password has expired (code "10023"). Until the user's password is updated, the user will be unable to perform any other requests in the BridgePay system.

Note: Service accounts are generally set to not expire. A service account is an account that is not accessed or used by an end user to log into the system, the account is only used by a software system to facilitate transaction processing between systems.

1.9. Hardware Encryption

Some requests can accept hardware encrypted data. If the client uses devices that encrypt track data, that client must be specifically configured with an indicator that hardware encryption will be utilized as well as what type of decryption methods will be needed. In addition to accepting the data encrypted, there are other data elements that must be passed in with the request. These fields should be added as XML element tags in the RequestMessage element, JSON object or as query parameters in the request just as any other field would be added. For example, to process an authorization using the ProcessRequest access method for a transaction using a SecureMagV2 device, the RequestMessage element would have the following additional tags embedded within it:

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  .
  .
  .
  <requestMessage>
    <MSRKSN>XXX</MSRKSN>
    <SecureFormat>XXX</SecureFormat>
    <BDKSlot>XXX</BDKSlot>
  </requestMessage>
</requestHeader>
```

When using the ProcessMessage access method, they would be added in a similar fashion as follows:


```
https://url/PaymentService/Default.aspx? ... &MSRKey=XXX&SecureFormat=XXX&BDKSlot=XXX& ...
```

When using ProcessJsonRequest access method, they would be added in a similar fashion as follows:

```
{
  "dataElements": [
    .
    .
    .
    { "name": "msrkey", "value": "XXX" },
    { "name": "secureformat", "value": "XXX" },
    { "name": "bdkslot", "value": "XXX" },
    .
    .
    .
  ]
}
```

1.10. Testing

In the live environment, transactions must be initiated over an SSL-protected connection using a minimum of 128-bit encryption to protect sensitive data. Initial testing can be performed over an unencrypted connection; however, testing must be performed over an SSL-encrypted channel before a client can be certified to production. Only test card data may be transmitted *-no live payment information should ever be transmitted to the test system.*

 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://bridgepaynetwork.com/testAccount.php>.

You may use the following cards in testing:

Card Type	Number	Expiry Date	Security Code
MasterCard	5439750001500347		
Visa	4005550000000019		
Discover	6001111111111117		
Diners	36999999999999		
AMEX	374255312721002		

If you implement swiped, card-present transactions (Credit, Debit, EBT, Gift Cards), please send a request to developersupport@bridgepaynetwork.com, and we will ship a set of test cards to you.

1.11. Purchase Tokens

Purchase Tokens provide a more secure way of passing authentication data to BridgePay from integrated applications such as websites that require sending username and password information as part of the request.

Conceptually, the integrating software calls a method on BridgeComm to pre-authenticate a specific transaction request. BridgeComm then records the imminent transaction, generates and return a GUID identifying the purchase token, and begins a 15 minute count-down timer on its usage. When sending the *actual* transaction to BridgeComm for processing, the merchant replaces their username and password information with the previously acquired purchase token GUID. BridgeComm then compares the details of the transaction with the referenced purchase token.

If the transaction details match, the transaction is processed and the purchase token is consumed, unable to be used again. If the transaction details do not match, the transaction is declined and the purchase token is consumed, unable to be used again. If the purchase token remains unused for 15 minutes, the token is consumed, unable to be used.

For more information regarding Purchase Tokens, how to acquire one and how to consume one, see Appendix A.6 Purchase Tokens.

Chapter 2. Implementation

This chapter describes how to integrate with the BridgeComm Interface.

2.1. Web Service Operation & Data Contracts

The request and response parameters for the ProcessRequest, ProcessJsonRequest and ProcessMessage access methods are identical. The difference between the contracts is in the formatting of the request and response strings.

- **ProcessRequest** – Base64- encoded XML formatted request and response.
- **ProcessJsonRequest** – Base64- encoded JSON formatted request and response.
- **ProcessMessage** – HTML-formatted request; custom-formatted response.

NOTE: In all tables below, “Length” refers to the maximum length of a value and this may not indicate the “standard” length.

NOTE: In all tables below, an “R” in the first column indicates a required value. Failure to supply a required value as part of a service request will result in an error. A required value in a response will always be present.

2.1.1. Base Request

The base request message requires the following parameters:

	Parameter	Data Type	Length	Description
R	ClientIdentifier	string	50	Unique value to identify the application that is integrating with BridgeComm. Provided by BridgePay.
R	TransactionID	string	60	Unique value to identify the transaction. Generated by the integrating application.
R	RequestType	numeric	3	Identifies the type of request message submitted. Valid values are: <ul style="list-style-type: none">• 000 – One-Time Token Request• 001 – Multi-Use Token Request• 002 – Multi-Use Token Security Code Request• 004 – Authorization Request• 005 – BIN Lookup Request• 006 – Check Password Expiration Request• 007 – Update Password Request• 011 – Get Merchant Info• 012 – Void/Refund• 019 – Capture• 020 – Initiate Settlement• 022 – Manage Gift Card Request• 099 – Ping

	Parameter	Data Type	Length	Description
R	RequestDateTime	numeric	19	The date and time the request was sent. The default format is YYYYMMDDHHMMSS using military time. This format can be overridden using the validationFormat attribute.
R	User	string	150	Provided by BridgePay. Not required when consuming a Purchase Token.
R	Password	string	255	Initially provided by BridgePay. Not required when consuming a Purchase Token.
R	requestMessage	string	n/a	XML-formatted request message string. Required for ProcessRequest only. (See section 2.2 for details)

2.1.2. Base Response

The base response message returns the following parameters:

	Parameter	Data Type	Length	Description
R	TransactionID	string	60	BridgeComm echoes back the TransactionID value from the request message.
R	RequestType	numeric	3	BridgeComm echoes back the RequestType value from the request message. See 2.1.1 Error! Reference source not found. for valid values.
R	ResponseCode	numeric	5	The five-digit response code describing the results of the transaction. See Appendix A.1 for valid values.
R	responseMessage	string	n/a	XML-formatted response message string. Provided by ProcessRequest only. See section 2.2 for details.

2.2. Specific Request Layouts

The following sections describe the specific layouts for each request and response for each request type.

- When using **ProcessRequest**, the request data elements are embedded within a parent element called **requestMessage**. Response data elements are embedded within a parent element called **responseMessage**.
- When using **ProcessJsonRequest**, the request data elements are added into the JSON request. Response data elements are included in the JSON response object.
- When using **ProcessMessage**, the request data elements are added as query parameters to the request string. Response data elements may be provided in the response if the XSLT is coded to return them.

2.2.1. One-Time Token (Request Type 000)

Use the one-time token request to generate a single-use token for authorizing a transaction.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
1	PaymentAccountNumber	numeric string	13-19	Credit card number. Required for keyed transactions.
1	ExpirationDate	string	7	Credit card's expiration date. Supported formats are: MMY, MMY, MM/YY, and MM/YYYY. Required for keyed transactions.
1	SecurityCode	numeric string	3-4	Card Verification Number. A 3 or 4-digit security code that is printed on the front or back of a card but not encoded on the magnetic stripe data. Required for keyed transactions.
2	TrackData	string		The track 1 data from the credit card. Required for swiped transactions. Alternate element "Track" may also be used for this item.
	MSRKey	string		When present, indicates that Hardware Encryption is being used and provides the MagStripeReader's key for decryption.
	SecureFormat	string		When present, indicates the format to use for decrypting the track data.
	BDKSlot	numeric		When present, indicates the Base Derived Key slot to use as an override for Hardware Decryption.
	Track1	string		Overrides TrackData/Track fields. Validated against acceptable characters in card swipe.
	Track2	string		Overrides TrackData / Track fields. Validated against acceptable characters in card swipe.
	EncryptionID	string		The encryption ID identifying the type of encryption used on the sensitive card holder data. See 2.2.4 Generate Encryption Key Request.

The parameters marked with a “1” are co-dependent. If one of these parameters is used, all of the co-dependent parameters must be used. The parameters marked with a “2” are co-dependent. If one of these parameters is used, all of the co-dependent parameters must be used. Etc.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>000</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <PaymentAccountNumber></PaymentAccountNumber>
    <ExpirationDate></ExpirationDate>
    <SecurityCode></SecurityCode>
    <TrackData></TrackData>
    <EncryptionID></EncryptionID>
    <Track1></Track1>
    <Track2></Track2>
    <MSRKey></MSRKey>
    <SecureFormat></SecureFormat>
    <BDKSlot></BDKSlot>
  </requestMessage>
</requestHeader>
```

2.2.1.1. Response

The response message is included in the **GetToken** element and returns the following data elements:

	Data Element	Data Type	Length	Description
R	Token	Numeric	22	The token to be used for authorization.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<GetToken>
  <TransactionID></TransactionID>
  <RequestType>000</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage>
    <Token></Token>
  </responseMessage>
</GetToken>
```


2.2.2. Multi-Use Token (Request 001)

This request method is used to tokenize a single card entry without the CVV so that it can be stored in a wallet. It is similar to “One-Time Token Request (000)”.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
1	PaymentAccountNumber	string	13-16	Credit card number. Required for keyed transactions.
1	ExpirationDate	string	7	Credit card's expiration date. Supported formats are: MMY, MMY, MM/YY, and MM/YYYY. Required for keyed transactions.
2	TrackData Track	string		The track 1 data from the credit card. Required for swiped transactions. Alternate element “Track” may also be used for this item.
	MSRKey	string		When present, indicates that Hardware Encryption is being used and provides the MagStripeReader's key for decryption.
	SecureFormat	string		When present, indicates the format to use for decrypting the track data.
	BDKSlot	numeric		When present, indicates the Base Derived Key slot to use as an override for Hardware Decryption.
	Track1	string		Overrides TrackData / Track fields. Validated against acceptable characters in card swipe.
	Track2	string		Overrides TrackData / Track fields. Validated against acceptable characters in card swipe.
	EncryptionID	string		The encryption ID identifying the type of encryption used on the sensitive card holder data. See 2.2.4 Generate Encryption Key Request.

The parameters marked with a “1” are co-dependent. If one of these parameters is used, all of the co-dependent parameters must be used. The parameters marked with a “2” are co-dependent. If one of these parameters is used, all of the co-dependent parameters must be used. Etc.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>001</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <PaymentAccountNumber></PaymentAccountNumber>
    <ExpirationDate></ExpirationDate>
    <TrackData></TrackData>
    <Track1></Track1>
    <Track2></Track2>
    <MSRKey></MSRKey>
    <SecureFormat></SecureFormat>
    <BDKSlot></BDKSlot>
    <EncryptionID></EncryptionID>
  </requestMessage>
</requestHeader>
```

2.2.2.1. Response

The response message is included in the **GetToken** element and returns the following data elements:

	Data Element	Data Type	Length	Description
R	Token	numeric	22	The token to be used for authorization.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<GetToken>
  <TransactionID></TransactionID>
  <RequestType>001</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage>
    <Token></Token>
  </responseMessage>
</GetToken>
```

2.2.3. Multi-Use Token Security Code (Request 002)

Use the multi-use token security code request to attach a card security code to a payment token so that BridgePay can send the card security code value for authorization.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
R	Token	numeric	22	Token value returned previously by a "001" request.
R	SecurityCode	string	3-4	CVV / CVV2 value (3 or 4 characters long).
	EncryptionID	string		The encryption ID identifying the type of encryption used on the sensitive card holder data. See 2.2.4 Generate Encryption Key Request.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>002</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <Token></Token>
    <SecurityCode></SecurityCode>
    <EncryptionID></EncryptionID>
  </requestMessage>
</requestHeader>
```

2.2.3.1. Response


The response message is included in the **WalletSecurityCode** element and returns only the base response data elements.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<WalletSecurityCode>
  <TransactionID></TransactionID>
  <RequestType>002</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage />
</WalletSecurityCode>
```

2.2.4. Generate Encryption Key (Request 003)

If enabled in the Client Identifier provided by BridgePay, you can generate an encryption key to use for encrypting sensitive data before sending the data to BridgeComm. This is an extra layer of encryption that is not necessary for PCI requirements since all communications are handled via Secure Socket Layer. However, if you are building a web site that may need to store the data momentarily in a form, you may desire to encrypt the data for security purposes before transmitting it.

 This message requires no additional parameters from the base request.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>003</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage />
</requestHeader>
```

2.2.4.1. Response

The response message is included in the **EncryptionKey** element and returns the following responseMessage parameters:


	Data Element	Data Type	Length	Description
	Algorithm	string		The encryption algorithm used. BridgeComm supports 2048 bit RSA. Valid value is RSA.
	CreateDate	DateTime		Date the public key was created in YYYY/MM/DD HH:MM:SS.SSS format.
	ID	numeric		The encryption ID used for the request messages with encrypted sensitive cardholder data.
	KeySize	numeric		The size of the public key in bits. BridgeComm supports 2048 bit RSA. Valid value is 2048.
	PublicKey	string		The public key to use for encrypting sensitive cardholder data elements.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<EncryptionKey>
  <TransactionID></TransactionID>
  <RequestType>003</RequestType>
  <ResponseCode></ResponseCode>
  <responseMessage>
    <Algorithm></Algorithm>
    <CreateDate></CreateDate>
    <ID></ID>
    <KeySize></KeySize>
    <PublicKey></PublicKey>
  </responseMessage>
</EncryptionKey>
```

2.2.5. Authorization (Request 004)

Use the authorization request to initiate a single transaction request. Please note at the time of this writing **Level III** transactions are only Valid for TSYS and First Data Nashville.

 This message may require additional data required for merchant lookup. See section 1.7 for more information.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
	MerchantCode	numeric		Provided by MyBridgePay at the time of boarding. The element is necessary unless the client is using a different merchant lookup solution (See Section 1.7)
	MerchantAccountCode	numeric		Provided by MyBridgePay at the time of boarding. The element is necessary unless the client is using a different merchant lookup solution (See Section 1.7)
O	PurchaseToken	GUID	36	Provided by the AcquirePurchaseToken method. Used in place of User and Password in the base response.
1	PaymentAccountNumber	numeric	13-19	When present, must follow general Credit Card conventions. Supports MC, VI, AX, and DI. Required if track data not present.
	Token	Numeric	22	When present, represents the tokenized card number received from a token request. This is used in place of the PaymentAccountNumber.
O	WalletPaymentMethodID	Guid	36	This is Wallet Payment Method ID provided by the Wallet API for the payment method. When using this field, no other payment identifiers are necessary (i.e. PaymentAccountNumber, Track, BankAccountNum, etc).
1	ExpirationDate	string	7	Expiration date of the card. If present, must follow one of these formats: MMYYY, MM/YY, MMYYYY, MM/YYYY. Required if track data not present.
1	SecurityCode	numeric	3-4	When present, must be 3 or 4 digits long CVV / CVV2 of the card. Required if track data not present.
R	Amount	integer	8	The amount requested for authorization. Processed as implied decimal. \$1.25 would be represented as 125.
2	TrackData or Track	string		When present, can contain alphanumeric characters and spaces, periods, slashes (/), percent symbols (%), caret symbols (^), question marks (?), semicolons (;), equality symbols (=) or hyphens (-). Can use hardware encryption. Can be called Track or TrackData in the request.
	EncryptionID	string		The encryption ID identifying the type of encryption used on the sensitive card holder data. See 2.2.4 Generate Encryption Key Request.
	MSRKey	string	50	When present, indicates that Hardware Encryption is being used and provides the MagStripeReader's key for decryption.
	SecureFormat	string	50	When present, indicates the format to use for decrypting the track data. Possible values: MagneSafeV1, MagneSafeV2, SecureMag and SecureMagV2.
	BDKSlot	numeric	2	When present, indicates the Base Derived Key slot to use as an override for Hardware Decryption.

	Data Element	Data Type	Length	Description
	Track1	string		Overrides TrackData / Track fields. Validated against acceptable characters in card swipe. Can use hardware encryption.
	Track2	string		Overrides TrackData / Track fields. Validated against acceptable characters in card swipe. Can use hardware encryption.
R	TransactionType	string	11	Can contain letters and hyphens. Possible values: sale, sale-auth, credit, credit-auth, increment, sale-info credit-info
R	TransIndustryType	string	2	Must either match a 2-letter industry type for credit cards or a 3-letter SEC code for ACH transactions. All caps. Credit Card Supported values are: RE (Retail), RS (Restaurant), EC (eCommerce), DM (Direct Marketing), LD (Lodging) and CR (Car Rental). ACH Supported values are: CCD (Corporate Credit or Debit), PPD (Prearranged Payment and Deposit Entry), POP (Point Purchase Entry), TEL (Telephone Initiated Entry), WEB (Internet Initiated Entry), C21 (Check 21)
R	TransCatCode	String	1	All caps. Supported values are: B (BillPayment), R (Recurring), I (Installment), H (Healthcare).
O	VoiceAuthCode	String	15	Authorization Code for Voice Authorizations only. If authorization was achieved outside of the network (by phone or other means) you may send it through this data element to utilize the existing authorization instead of acquiring a new authorization.
O	PartialAuthorization	String	5	Must contain the literal true or the literal false . Setting this field to true indicates that a partial authorization is acceptable for this transaction. The default, if this element is not present, is false.
	SwipeResult	future Use		Future use.
	PINBlock	future Use		Future use.
	PINKey	future Use		Future use.
3	BankAccountNum	numeric		Must contain at least one number. The bank account number for ACH transactions.
3	RoutingNum	numeric	10	Can contain up to 10 numbers. The routing number for ACH transactions.
R	AcctType	string	1	Single character, must match pre-defined types: R (Credit Card & Branded Debit Cards), D (Unbranded Debit Cards), S (Bank Account Savings), C (Bank Account Checking), F (EBT Food Stamp), H (EBT Cash Benefit), G (Gift Card), L (Fleet), K (Check), A (Cash)
	InvoiceNum	string	24	Alphanumeric and dashes accepted. User supplied data.
	PONum	string	24	Alphanumeric and dashes accepted. User supplied data.
	CustomerAccountCode	string	24	Alphanumeric and dashes accepted. User supplied data.
	PaymentType	string	60	Alphanumeric, dashes and spaces accepted. User supplied data.
	AccountHolderName	string	150	Any upper or lower case letter, hyphens, and spaces are allowed. Account Holder's Name.
R	HolderType	string	1	Single character, either P (Personal account) or O (Organization account).
	FeeAmount	integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Additional Fee Amount.
	TipAmount	integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Additional Tip Amount.

	Data Element	Data Type	Length	Description
	AccountStreet	string	128	Can contain alphanumerics, spaces, hyphens, periods and slashes. Street on the account
	AccountZip	string	10	Postal Code on the Payment Method Account. Accepted formats are Canadian, UK and US (5 and 9 digit variation) postal codes. Alpha characters must be all upper-case.
	AccountPhone	String	20	Billing phone number. Numerics only. No special characters allowed.
O	ContractId	numeric	12	A valid BridgePay Recurring Billing System Contract Id. Provides a way to tie a transaction to a contract defined in BridgePay's Recurring Billing System.
	TaxRate	integer	5	Processed as implied decimal. \$1.25 would be represented as 125. REQUIRED FOR LEVEL II.
	TaxAmount	integer	8	Processed as implied decimal. 5.5% would be represented as 550. Additional Tax Amount. REQUIRED FOR LEVEL II/III
	TaxIndicator	string	1	P (Provided), N (Not Provided), or E (Exempt). REQUIRED FOR LEVEL II/III.
	ShipToName	string	100	Shipping address name. REQUIRED FOR LEVEL II/III.
	ShipToStreet	string	128	Shipping address street. REQUIRED FOR LEVEL II/III.
	ShipToCity	string	50	Shipping address city. REQUIRED FOR LEVEL II/III.
	ShipToState	String	2	Shipping address state. REQUIRED FOR LEVEL II/III.
	ShipToZip	string	15	Shipping address postal code. Accepted formats are Canadian, UK and US (5 and 9 digit variation) postal codes. Alpha characters must be all upper-case. REQUIRED FOR LEVEL II/III.
	ShipToCountryCode	string	2	Shipping address country code. REQUIRED FOR LEVEL II/III. ISO 3166-1 alpha-2 codes.
	ShippingOriginZip	string	10	Postal code of the origin of the shipment. Accepted formats are Canadian, UK and US (5 and 9 digit variation) postal codes. Alpha characters must be all upper-case. REQUIRED FOR LEVEL III.
	DiscountAmount	integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Additional discount amount. REQUIRED FOR LEVEL III.
	ShippingAmount	Integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Additional shipping amount. REQUIRED FOR LEVEL III.
	DutyAmount	Integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Additional Duty Amount. REQUIRED FOR LEVEL III.
	TaxInvoiceCode	String	15	Must be at least 1 and up to 15 characters. When separate VAT invoice is produced within the context of the order, unique identifier of this invoice. REQUIRED FOR LEVEL III.
	LocalTaxAmount	Integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Additional Local Tax Amount. REQUIRED FOR LEVEL III.
	LocalTaxIndicator	string	1	P (Provided), N (Not Provided), or E (Exempt). REQUIRED FOR LEVEL III.
	NationalTaxAmount	integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Additional National Tax Amount. REQUIRED FOR LEVEL III.
	NationalTaxIndicator	string	1	P (Provided), N (Not Provided), or E (Exempt). REQUIRED FOR LEVEL III.

	Data Element	Data Type	Length	Description
	OrderCode	string	17	Unique identifier assigned to the order associated with this transaction in submitter's/merchant's front-end/ inventory system. REQUIRED FOR LEVEL III.
	OrderDate	string	8	Date in format YYYYMMDD. Date when the order associated with the transaction was placed. REQUIRED FOR LEVEL III.
	CommodityCode	string	4	Acquirer designated standardized code that classifies the group of items associated with this order/transaction. REQUIRED FOR LEVEL III.
	CustomerAccountTaxID	string	13	VAT registration number of the customer responsible for this transaction. REQUIRED FOR LEVEL III.
	CheckImageFront	string	64000	Base64 text encoded version of the TIFF image of the front of the check being processed.
	CheckImageBack	string	64000	Base64 text encoded version of the TIFF image of the back of the check being processed.
	MICR	string	255	MICR data scanned by a check reader.
	EMVTags	string	255	Combined EMV tags presented in TLV format.
	EntryModeType	string	2	Method of entering the transaction. Possible values: MX = Manual with Signature, SX = Swipe/Scan with Signature, HX = Chip with Signature, CX = Contactless with Signature, SP = Swipe with PIN, HP = Chip with PIN
	EntryMediumType	string	2	The medium type used to initiate the transaction. Possible values: NP = Not Present, MC = Magnetic Card, CC = Chip Card, CH = Check
	EntryPINModeType	string	1	Indicates whether the input source supports PIN entry. Possible values: X = unknown, S = supported, U = Unsupported, N = Inoperative, O = Offline.
	TerminalCapabilities	string	255	Describes the terminal's capabilities. Can include any combination of the following tags separated by pipe characters: unknown, unused, manual, stripe, barcode, qrcode, ocr, irc, contactless, signature, rfid, micr or mobile .
	ItemCount	numeric	12	The number of items included in the Item Collection. REQUIRED FOR LEVEL III.
	Items	collection	n/a	Multiple based on contents Individual items. REQUIRED FOR LEVEL III.

The “Items” collection is a repeatable tag that consists of the following group of elements:

	Parameter	Data Type	Length	Description
R	ItemCode	String	12	Unique identifier assigned to this item in the submitter's inventory system.
R	ItemCommodityCode	String	12	Acquirer designated standardized code that classifies this item.
R	ItemDescription	String	35	Short description of the item.
R	ItemQuantity	decimal	12	Quantity of item units purchased as part of this transaction. Up to 4 decimal places.
R	ItemUnitCostAmt	integer	12	Processed as implied decimal. \$1.25 would be represented as 125. Cost of a single unit of the item.

	Parameter	Data Type	Length	Description
R	ItemUnitMeasure	string	12	Unit of measure used to quantify the items purchased/refunded (e.g. kg, lb., inch).
R	ItemTaxRate	integer	12	Processed as implied decimal. 5.5% would be represented as 550. Rate of the tax (if any) charged on this item.
R	ItemTaxAmount	Integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Amount of tax charged for this item.
R	ItemTaxIndicator	string	1	P (Provided), N (Not provided), or E (Exempt).
R	ItemTaxCode	string	4	Acquirer designated value classifying the tax that was charged for this item.
R	ItemDiscountRate	Integer	8	Processed as implied decimal. 5.5% would be represented as 550. Rate of discount (if any) that was applied to this item.
R	ItemDiscountAmount	Integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Total amount of discount applied to this item.
R	ItemTotalAmount	Integer	8	Processed as implied decimal. \$1.25 would be represented as 125. Total amount paid for the item (including tax and discount).
R	ItemsCredit	string	1	True (Item is being returned), False (Default if not passed - Item is being purchased).

XML Request Layout

```

<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>004</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <MerchantCode></MerchantCode>
    <MerchantAccountCode></MerchantAccountCode>
    <PurchaseToken></PurchaseToken>
    <PaymentAccountNumber></PaymentAccountNumber>
    <WalletPaymentMethodID></WalletPaymentMethodID>
    <ExpirationDate></ExpirationDate>
    <SecurityCode></SecurityCode>
    <Amount></Amount>          <TrackData></TrackData>
    <EncryptionID></EncryptionID>
    <TransactionType></TransactionType>
    <TransIndustryType></TransIndustryType>
    <TransCatCode></TransCatCode>
    <VoiceAuthCode></VoiceAuthCode>
    <PartialAuthorization></PartialAuthorization>
    <MSRKey></MSRKey>
    <SwipeResult></SwipeResult>
    <PINBlock></PINBlock>
    <PINKey></PINKey>
    <BankAccountNum></BankAccountNum>
    <RoutingNum></RoutingNum>
    <AcctType></AcctType>
    <InvoiceNum></InvoiceNum>
    <PONum></PONum>
    <CustomerAccountCode></CustomerAccountCode>
    <AccountHolderName></AccountHolderName>
    <HolderType></HolderType>
    <FeeAmount></FeeAmount>
    <TipAmount></TipAmount>
  </requestMessage>
</requestHeader>

```

```

<AccountStreet></AccountStreet>
<AccountZip></AccountZip>
<AccountPhone></AccountPhone>
<ContractId></ContractId>
<TaxRate></TaxRate>
<TaxAmount></TaxAmount>
<TaxIndicator></TaxIndicator>
<ShipToName></ShipToName>
<ShipToStreet></ShipToStreet>
<ShipToCity></ShipToCity>
<ShipToState></ShipToState>
<ShipToZip></ShipToZip>
<ShipToCountryCode></ShipToCountryCode>
<ShippingOriginZip></ShippingOriginZip>
<DiscountAmount></DiscountAmount>
<ShippingAmount></ShippingAmount>
<DutyAmount></DutyAmount>
<TaxInvoiceCode></TaxInvoiceCode>
<LocalTaxAmount></LocalTaxAmount>
<LocalTaxIndicator></LocalTaxIndicator>
<NationalTaxAmount></NationalTaxAmount>
<NationalTaxIndicator></NationalTaxIndicator>
<OrderCode></OrderCode>
<OrderDate></OrderDate>
<CommodityCode></CommodityCode>
<CustomerAccountTaxID></CustomerAccountTaxID>
<CheckImageFront></CheckImageFront>
<CheckImageBack></CheckImageBack>
<MICR></MICR>
<EMVTags></EMVTags>
<EntryModeType></EntryModeType>
<EntryMediumType></EntryMediumType>
<EntryPINModeType></EntryPINModeType>
<TerminalCapabilities></TerminalCapabilities>
<ItemCount>2</ItemCount>
<Item>
    <ItemCode></ItemCode>
    <ItemCommodityCode></ItemCommodityCode>
    <ItemDescription></ItemDescription>
    <ItemQuantity></ItemQuantity>
    <ItemUnitCostAmt></ItemUnitCostAmt>
    <ItemUnitMeasure></ItemUnitMeasure>
    <ItemTaxAmount></ItemTaxAmount>
    <ItemTaxRate></ItemTaxRate>
    <ItemTaxIndicator></ItemTaxIndicator>
    <ItemTaxCode></ItemTaxCode>
    <ItemDiscountRate></ItemDiscountRate>
    <ItemTotalAmount></ItemTotalAmount>
    <ItemIsCredit></ItemIsCredit>
</Item>
<Item>
    <ItemCode></ItemCode>
    <ItemCommodityCode></ItemCommodityCode>
    <ItemDescription></ItemDescription>
    <ItemQuantity></ItemQuantity>
    <ItemUnitCostAmt></ItemUnitCostAmt>
    <ItemUnitMeasure></ItemUnitMeasure>
    <ItemTaxAmount></ItemTaxAmount>
    <ItemTaxRate></ItemTaxRate>
    <ItemTaxIndicator></ItemTaxIndicator>
    <ItemTaxCode></ItemTaxCode>
    <ItemDiscountRate></ItemDiscountRate>
    <ItemTotalAmount></ItemTotalAmount>
    <ItemIsCredit></ItemIsCredit>

```

```

        </Item>
    </requestMessage>
</requestHeader>

```

JSON Request Layout

```

{ "request": {
  "dataElements": [
    { "name": "ClientIdentifier", "value": "" },
    { "name": "TransactionID", "value": "" },
    { "name": "RequestType", "value": "" },
    { "name": "RequestDateTime", "value": "" },
    { "name": "User", "value": "" },
    { "name": "Password", "value": "" },
    { "name": "MerchantCode", "value": "" },
    { "name": "MerchantAccountCode", "value": "" },
    { "name": "PurchaseToken", "value": "" },
    { "name": "PaymentAccountNumber", "value": "" },
    { "name": "WalletPaymentMethodID", "value": "" },
    { "name": "ExpirationDate", "value": "" },
    { "name": "SecurityCode", "value": "" },
    { "name": "Amount", "value": "" },
    { "name": "TrackData", "value": "" },
    { "name": "EncryptionID", "value": "" },
    { "name": "TransactionType", "value": "" },
    { "name": "TransIndustryType", "value": "" },
    { "name": "TransCatCode", "value": "" },
    { "name": "VoiceAuthCode", "value": "" },
    { "name": "MSRKey", "value": "" },
    { "name": "SwipeResult", "value": "" },
    { "name": "PINBlock", "value": "" },
    { "name": "PINKey", "value": "" },
    { "name": "BankAccountNum", "value": "" },
    { "name": "RoutingNum", "value": "" },
    { "name": "AcctType", "value": "" },
    { "name": "InvoiceNum", "value": "" },
    { "name": "PONum", "value": "" },
    { "name": "CustomerAccountCode", "value": "" },
    { "name": "AccountHolderName", "value": "" },
    { "name": "HolderType", "value": "" },
    { "name": "FeeAmount", "value": "" },
    { "name": "TipAmount", "value": "" },
    { "name": "AccountStreet", "value": "" },
    { "name": "AccountZip", "value": "" },
    { "name": "AccountPhone", "value": "" },
    { "name": "ContractId", "value": "" },
    { "name": "TaxRate", "value": "" },
    { "name": "TaxAmount", "value": "" },
    { "name": "TaxIndicator", "value": "" },
    { "name": "ShipToName", "value": "" },
    { "name": "ShipToStreet", "value": "" },
    { "name": "ShipToCity", "value": "" },
    { "name": "ShipToState", "value": "" },
    { "name": "ShipToZip", "value": "" },
    { "name": "ShipToCountryCode", "value": "" },
    { "name": "ShippingOriginZip", "value": "" },
    { "name": "DiscountAmount", "value": "" },
    { "name": "ShippingAmount", "value": "" },
    { "name": "DutyAmount", "value": "" },
    { "name": "TaxInvoiceCode", "value": "" },
    { "name": "LocalTaxAmount", "value": "" },
    { "name": "LocalTaxIndicator", "value": "" },
    { "name": "NationalTaxAmount", "value": "" },
  ]
}

```

```

{ "name": "NationalTaxIndicator", "value": "" },
{ "name": "OrderCode", "value": "" },
{ "name": "OrderDate", "value": "" },
{ "name": "CommodityCode", "value": "" },
{ "name": "CustomerAccountTaxID", "value": "" },
{ "name": "CheckImageFront", "value": "2" },
{ "name": "CheckImageBack", "value": "2" },
{ "name": "MICR", "value": "2" },
{ "name": "EMVTags", "value": "2" },
{ "name": "EntryModeType", "value": "2" },
{ "name": "EntryMediumType", "value": "2" },
{ "name": "EntryPINModeType", "value": "2" },
{ "name": "TerminalCapabilities", "value": "2" },
{ "name": "ItemCount", "value": "2" },
{ "name": "Items": [
  { "dataElements": [
    { "name": "ItemCode", "value": "" },
    { "name": "ItemCommodityCode", "value": "" },
    { "name": "ItemDescription", "value": "" },
    { "name": "ItemQuantity", "value": "" },
    { "name": "ItemUnitCostAmt", "value": "" },
    { "name": "ItemUnitMeasure", "value": "" },
    { "name": "ItemTaxAmount", "value": "" },
    { "name": "ItemTaxRate", "value": "" },
    { "name": "ItemTaxIndicator", "value": "" },
    { "name": "ItemTaxCode", "value": "" },
    { "name": "ItemDiscountRate", "value": "" },
    { "name": "ItemTotalAmount", "value": "" },
    { "name": "ItemIsCredit", "value": "" }
  ],
  { "dataElements": [
    { "name": "ItemCode", "value": "" },
    { "name": "ItemCommodityCode", "value": "" },
    { "name": "ItemDescription", "value": "" },
    { "name": "ItemQuantity", "value": "" },
    { "name": "ItemUnitCostAmt", "value": "" },
    { "name": "ItemUnitMeasure", "value": "" },
    { "name": "ItemTaxAmount", "value": "" },
    { "name": "ItemTaxRate", "value": "" },
    { "name": "ItemTaxIndicator", "value": "" },
    { "name": "ItemTaxCode", "value": "" },
    { "name": "ItemDiscountRate", "value": "" },
    { "name": "ItemTotalAmount", "value": "" },
    { "name": "ItemIsCredit", "value": "" }
  ]
} ]
} ]
} }

```

2.2.5.1. Response

The response message is included in the **Auth** element and returns the following data elements:

	Data Element	Data Type	Length	Description
	Token	string	22	Either the token submitted with the original authorization request or a tokenized version of the submitted payment account number if no token was provided in the request.
	AuthorizationCode	String	50	The transaction authorization code returned from the processor.
	ReferenceNumber	string		For future use.
	GatewayResult	string		Echo of the ResponseCode.
	AuthorizedAmount	integer	8	Implied decimal, amount of the charge that was authorized.
	OriginalAmount	integer	8	Implied decimal, amount that was requested for authorization.
	ExpirationDate	String	6	Echo back of the expiration date.
	AVSResult	String		Unipay AVS Match Result Code.
	AVSMessage	String		Unipay AVS Match Result Message.
	StreetMatchMessage	String		For future use.
	ZipMatchMessage	String		For future use.
	CVResult	String		Unipay CVV/CVV2 Match Result Code.
	CVMessage	String		Unipay CVV/CVV2 Match Result Message.
	IsCommercialCard	Boolean		For future use.
	GatewayTransID	Integer		Transaction ID from the gateway (used as Reference Number for Voids / Refunds).
	GatewayMessage	String		Message from the gateway.
	InternalMessage	String		Provides more information from the gateway and processor regarding the results of the transaction request.
	Balance	Integer		For future use.
	CashBackAmount	Integer	8	For future use.
	TransactionCode	String		Echo back of the TransactionCode.
	CardType	String		For future use.
	RemainingAmount	integer	8	For future use.
	TransactionDate	string		
	IsoCountryCode	string		
	IsoCurrencyCode	string		
	IsoTransactionDate	string		
	IsoRequestDate	string		
	NetworkReferenceNumber	string		
	NetworkTerminalId	string		
	MaskedPan	string		
	ResponseTypeDescription	string		
	MerchantCategoryCode	string		

	Data Element	Data Type	Length	Description
*	ReceiptTagData	string		Returned tag data for generating the receipt.
*	IssuerTagData	string		Returned tag data from the issuer.

* ReceiptTagData and IssuerTagData are only returned when the authorization is processed as an EMV transaction.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<Auth>
  <TransactionID></TransactionID>
  <RequestType>004</RequestType>
  <ResponseCode>0000</ResponseCode>
  <ResponseDescription></ResponseDescription>
  <responseMessage>
    <Token></Token>
    <AuthorizationCode></AuthorizationCode>
    <ReferenceNumber></ReferenceNumber>
    <GatewayResult></GatewayResult>
    <AuthorizedAmount></AuthorizedAmount>
    <OriginalAmount></OriginalAmount>
    <ExpirationDate></ExpirationDate>
    <AVSMessage></AVSMessage>
    <AVSResult></AVSResult>
    <StreetMatchMessage></StreetMatchMessage>
    <ZipMatchMessage></ZipMatchMessage>
    <CVMessage></CVMessage>
    <CVResult></CVResult>
    <IsCommercialCard></IsCommercialCard>
    <GatewayTransID></GatewayTransID>
    <GatewayMessage></GatewayMessage>
    <InternalMessage></InternalMessage>
    <Balance></Balance>
    <CashBackAmount></CashBackAmount>
    <TransactionCode></TransactionCode>
    <TransactionDate></TransactionDate>
    <IsoCountryCode></IsoCountryCode>
    <IsoCurrencyCode></IsoCurrencyCode>
    <IsoTransactionDate></IsoTransactionDate>
    <IsoRequestDate></IsoRequestDate>
    <NetworkReferenceNumber></NetworkReferenceNumber>
    <NetworkMerchantId></NetworkMerchantId>
    <NetworkTerminalId></NetworkTerminalId>
    <MaskedPan></MaskedPan>
    <MerchantCategoryCode></MerchantCategoryCode>
    <ResponseTypeDescription></ResponseTypeDescription>
    <CardType></CardType>
    <RemainingAmount></RemainingAmount>
    <ReceiptTagData></ReceiptTagData>
    <IssuerTagData></IssuerTagData>
  </responseMessage>
</Auth>
```

JSON Response Layout

```
{
  "TransactionID" : "",
  "RequestType" : "004",
  "ResponseCode" : "0000",
  "ResponseDescription" : "",
  "Token" : "",
  "AuthorizationCode" : "",
```

```
"ReferenceNumber" : "",
"GatewayResult" : "",
"AuthorizedAmount" : "",
"OriginalAmount" : "",
"ExpirationDate" : "",
"AVSMessage" : "",
"AVSResult" : "",
"StreetMatchMessage" : "",
"ZipMatchMessage" : "",
"CVMessage" : "",
"CVResult" : "",
"IsCommercialCard" : "",
"GatewayTransID" : "",
"GatewayMessage" : "",
"InternalMessage" : "",
"Balance" : "",
"CashBackAmount" : "",
"TransactionCode" : "",
"TransactionDate" : "",
"IsoCountryCode" : "",
"IsoCurrencyCode" : "",
"IsoTransactionDate" : "",
"IsoRequestDate" : "",
"NetworkReferenceNumber" : "",
"MerchantCategoryCode" : "",
"NetworkMerchantId" : "",
"NetworkTerminalId" : "",
"MaskedPan" : "",
"ResponseTypeDescription" : "",
"CardType" : "",
"RemainingAmount" : "",
"ReceiptTagData" : "",
"IssuerTagData" : "",
}
```

2.2.6. BIN Lookup (Request 005)

Use the BIN lookup request to determine the card brand of the specified BIN.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
R	BIN	numeric	11	The 6 to 11-digit BIN number to be looked up. As this is only a BIN number, it need not be encrypted and does not support being encrypted.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>005</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <BIN></BIN>
  </requestMessage>
</requestHeader>
```

2.2.6.1. Response

The response message is included in the **BINLookup** element and returns the following data elements:


	Data Element	Data Type	Length	Description
R	CardIdentifier	string		The card type for the submitted BIN.

Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<BINLookup>
  <TransactionID></TransactionID>
  <RequestType>005</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage>
    <CardIdentifier></CardIdentifier>
  </responseMessage>
</BINLookup>
```

2.2.7. Check Password Expiration (Request 006)

Use the check password expiration request to determine the amount of time remaining before the user's password expires.

 This message requires no additional parameters from the base request.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>006</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage />
</requestHeader>
```

2.2.7.1. Response

The response message is included in the **CheckPasswordExpiration** element and returns the following data elements:

	Data Element	Data Type	Length	Description
R	SecondsRemaining	numeric	7	The number of seconds remaining until the user's current password expires. If the account is a "service account", the value returned will be: 7776000. The maximum value that can be returned is 7776000.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<CheckPasswordExpiration>
  <TransactionID></TransactionID>
  <RequestType>006</RequestType>
  <ResponseCode>00000</ResponseCode>
  <ResponseDescription>Successful Request</ResponseDescription>
  <responseMessage>
    <SecondsRemaining>2548139</SecondsRemaining>
  </responseMessage>
</CheckPasswordExpiration>
```

2.2.8. Change Password (Request 007)

Use the change password request to update the user's password.

Passwords must contain a minimum of 6 characters, with at least one upper case letter, one lower case letter, and one numeric character.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
R	NewPassword	string		The new password for the user.
R	ConfirmPassword	string		The re-entered new password for the user, to confirm the new value.

Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>007</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <NewPassword></NewPassword>
    <ConfirmPassword></ConfirmPassword>
  </requestMessage>
</requestHeader>
```

2.2.8.1. Response

The response message is included in the **UpdatePassword** with no extra data elements.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<UpdatePassword>
  <TransactionID></TransactionID>
  <RequestType>007</RequestType>
  <ResponseCode>00000</ResponseCode>
</UpdatePassword>
```


2.2.9. Get Merchant Info (Request 011)

Use the merchant info request to retrieve the default merchant information for your credentials.

The Get Merchant Info request can use the PurchaseToken without consuming the purchase token. Simply provide the PurchaseToken along with the request, excluding the User and Password fields, to retrieve the merchant information.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
O	PurchaseToken	GUID	36	Provided by the AcquirePurchaseToken method. Used in place of User and Password in the base response.

Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>011</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <PurchaseToken></PurchaseToken>
  </requestMessage>
</requestHeader>
```

2.2.9.1. Response

The response message is included in the **GetMerchantInfo** element and returns the following data elements:


	Data Elements	Data Type	Length	Description
R	MerchantCode	integer	12	The default merchant code assigned to the user.
R	MerchantAccountCode	integer	12	The default merchant account code assigned to the user.
R	MerchantName	string	50	The name assigned to the merchant account.
R	GatewayResults	string	5	Results from the Gateway (5 digit code).
R	GatewayTransID	integer	12	Transaction ID for the request.
R	GatewayMessage	string	255	Message from the Gateway regarding this request.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<GetMerchantInfo>
  <TransactionID></TransactionID>
  <RequestType>011</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage>
    <MerchantCode></MerchantCode>
    <MerchantAccountCode></MerchantAccountCode>
    <MerchantName></MerchantName>
    <GatewayResults></GatewayResults>
    <GatewayTransID></GatewayTransID>
    <GatewayMessage></GatewayMessage>
  </responseMessage>
</GetMerchantInfo>
```

2.2.10. Void/Refund (Request 012)

Use the void/refund request to issue a void against an unsettled authorization or a refund against a settled transaction.

 This message may require additional data required for merchant lookup. See section 1.7 for more information.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
	MerchantCode	numeric		Provided by MyBridgePay at the time of boarding. The element is necessary unless the client is using a different merchant lookup solution (See Section 1.7)
	MerchantAccountCode	numeric		Provided by MyBridgePay at the time of boarding. The element is necessary unless the client is using a different merchant lookup solution (See Section 1.7)
R	Amount	integer	8	Processed as implied decimal. \$1.25 would be represented as 125. The amount requested to be refunded / voided.
R	ReferenceNumber	integer	12	The GatewayTransId returned as a part of the response to request 004.
R	TransactionType	string	6	Can contain letters and hyphens. Must match a one of the defined transaction types in Unipay. Should use "void" or "refund" only for this request type.
R	TransactionCode	string		Must match the TransactionID in the requestHeader.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>012</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <MerchantCode></MerchantCode>
    <MerchantAccountCode></MerchantAccountCode>
    <Amount></Amount>
    <ReferenceNumber></ReferenceNumber>
    <TransactionType></TransactionType>
    <TransactionCode></TransactionCode>
  </requestMessage>
</requestHeader>
```

JSON Request Layout

```
{
  "dataElements": [
    { "name": "ClientIdentifier", "value": "" },
    { "name": "TransactionID", "value": "" },
    { "name": "RequestType", "value": "" },
    { "name": "RequestDateTime", "value": "" },
    { "name": "User", "value": "" },
    { "name": "Password", "value": "" },
    { "name": "MerchantCode", "value": "" },
    { "name": "MerchantAccountCode", "value": "" },
    { "name": "Amount", "value": "" },
    { "name": "TransactionType", "value": "" },
    { "name": "TransactionCode", "value": "" },
    { "name": "ReferenceNumber", "value": "" }
  ]
}
```

2.2.10.1. Response

The response message is included in the **VoidRefund** element and returns the following data elements:

	Data Element	Data Type	Length	Description
R	ReferenceNumber			The reference number of the refund or of the original authorization if this was a void.
R	TransactionCode			The TransactionCode that was send with the request.
R	RemainingAmount	numeric	8	Any amount of the original authorization remaining after this void / refund. This amount is in decimal format (i.e. \$19.95 = 19.95).
R	ResponseType	string		Action that was performed: "Void" or "refund".
R	MerchantAccountCode			The merchant account code used to process the original authorization and the void or refund.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<VoidRefund>
  <TransactionID></TransactionID>
  <RequestType>012</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage>
    <ReferenceNumber></ReferenceNumber>
    <TransactionCode></TransactionCode>
    <RemainingAmount></RemainingAmount>
    <ResponseType></ResponseType>
    <MerchantAccountCode></MerchantAccountCode>
  </responseMessage>
</VoidRefund>
```

JSON Response Layout

```
{
  "TransactionID" : "",
  "RequestType" : "",
  "ResponseCode" : "",
  "ReferenceNumber" : "",
  "TransactionCode" : "",
  "RemainingAmount" : "",
  "ResponseType" : "",
  "MerchantAccountCode" : ""
}
```

2.2.11. Tokenize Account (Request 013)

Use the Tokenize Account request to tokenize a bank account number for ACH transactions.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
R	BankAccountNum	numeric		Must contain at least one number. The bank account number for ACH transactions.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>013</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <BankAccountNum></ BankAccountNum >
  </requestMessage>
</requestHeader>
```

2.2.11.1. Response

The response message is included in the **GetToken** element and returns the following data elements:


	Data Element	Data Type	Length	Description
R	Token	Numeric	22	The 22-digit token to be used for authorization.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<GetToken>
  <TransactionID></TransactionID>
  <RequestType>012</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage>
    <Token></Token>
  </responseMessage>
</GetToken>
```

2.2.12. Capture (Request 019)

This request method is used to confirm a previously authorized sale. This is used for merchants that are not setup to AutoConfirm their transactions. Transactions that are not confirmed will be voided at settlement time.

 This message may require additional data required for merchant lookup. See section 1.7 for more information. This message may also require additional data for lodging check-out transactions.

The data elements for this request type are as follows:

	Parameter	Data Type	Length	Description
	MerchantCode	numeric		Provided by MyBridgePay at the time of boarding. The element is necessary unless the client is using a different merchant lookup solution (See Section 1.7)
	MerchantAccountCode	numeric		Provided by MyBridgePay at the time of boarding. The element is necessary unless the client is using a different merchant lookup solution (See Section 1.7)
R	Amount	integer	8	Processed as implied decimal. \$1.25 would be represented as 125. The amount to be confirmed. For new authorizations, can be 100% or less of the original auth amount. For previously confirmed transactions, can be 100% of the previously confirmed amount. For lodging transactions, can be greater or less than the total authorized for the lodging key.
C	ReferenceNumber	numeric		The GatewayTransId returned as part of the auth response (004). Conditionally Required – Not required for Lodging Transactions
R	TransactionType	string		Can contain letters and hyphens. Must match a one of the defined transaction types in Unipay. Should use “capture” only for this request type.
R	TransactionCode	string		Must match the TransactionID in the requestHeader
C	FolioNumber	String	15	Hotel folio number. Required when capturing a lodging transaction.
C	CheckInDate	Date		The date of actual or anticipated check-in. Required when capturing a lodging transaction.
C	Token	Numeric	22	When present, represents the tokenized card number received from a token request or authorization. Required when capturing a lodging transaction.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>019</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <MerchantCode><MerchantCode>
    <MerchantAccountCode></MerchantAccountCode>
    <Amount></Amount>
    <ReferenceNumber></ReferenceNumber>
```



```
<TransactionType></TransactionType>  
<TransactionCode></TransactionCode>  
<FolioNumber></FolioNumber>  
<CheckInDate></CheckInDate>  
<Token></Token>  
  </requestMessage>  
</requestHeader>
```

2.2.12.1. Response

The response message is included in the **Capture** element and returns the following data elements:


	Parameter	Data Type	Length	Description
R	ReferenceNumber	string		Provider supplied reference number.
R	GatewayResult	string		Echo of ResponseCode.
R	GatewayTransID	string		Echo of TransactionID.
R	GatewayMessage	string		Echo of ResponseDescription.
R	TransactionCode	string		The TransactionCode that was send with the request.
R	ResponseType	string		Action that was performed: "capture".
R	MerchantAccountCode	string		The merchant account code used to process the original authorization and the void or refund.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<Capture>
  <TransactionID></TransactionID>
  <RequestType>019</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage>
    <ReferenceNumber></ReferenceNumber>
    <GatewayResult></GatewayResult>
    <GatewayTransID></GatewayTransID>
    <GatewayMessage></GatewayMessage>
    <TransactionCode></TransactionCode>
    <ResponseType></ResponseType>
    <MerchantAccountCode></MerchantAccountCode>
  </responseMessage>
</Capture>
```

2.2.13. Initiate Settlement (Request 020)

This request method is used to initiate a settlement request immediately. All unsettled transactions will be submitted to the processors for settlement.

 This message requires no specific additional parameters apart from the base request, other than elements required for merchant lookup. See section 1.7 for more information.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>020</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <MerchantCode></MerchantCode>
    <MerchantAccountCode></MerchantAccountCode>
  </requestMessage>
</requestHeader>
```

2.2.13.1. Response

The response message is included in the **CloseCycle** element and returns the following data elements:


	Parameter	Data Type	Length	Description
R	CycleCode	string		The Batch ID for the submitted transactions.
R	GatewayResult	string		Echo of ResponseCode.
R	GatewayTransID	string		Echo of TransactionID.
R	GatewayMessage	string		Echo of ResponseDescription.
R	TransactionCode	string		The TransactionCode that was send with the request.
R	ResponseType	string		Action that was performed: "closeCycle".
R	MerchantAccountCode	string		The merchant account code used to process the original transactions.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<CloseCycle>
  <TransactionID></TransactionID>
  <RequestType>020</RequestType>
  <ResponseCode>00000</ResponseCode>
  <responseMessage>
    <CycleCode></CycleCode>
    <GatewayResult></GatewayResult>
    <GatewayTransID></GatewayTransID>
    <GatewayMessage></GatewayMessage>
    <TransactionCode></TransactionCode>
    <ResponseType></ResponseType>
    <MerchantAccountCode></MerchantAccountCode>
  </responseMessage>
</CloseCycle>
```

2.2.14. Manage Gift Card (Request 022)

This request method is used to activate, deactivate or reactivate a gift card.

 This request type is not currently available in BridgeComm and is only provided for future reference.

The data elements for this request type are as follows:

	Data Element	Data Type	Length	Description
	MerchantCode	numeric		Provided by MyBridgePay at the time of boarding. The element is necessary unless the client is using a different merchant lookup solution (See Section 1.7)
	MerchantAccountCode	numeric		Provided by MyBridgePay at the time of boarding. The element is necessary unless the client is using a different merchant lookup solution (See Section 1.7)
1	PaymentAccountNumber	numeric	13-19	When present, must follow general Credit Card conventions. Required if track data not present.
R	Amount	integer	8	Amount of money, available to a customer after a card is activated or returned upon deactivation of a card. Processed as implied decimal. \$1.25 would be represented as 125.
	MSRKey	string	50	When present, indicates that Hardware Encryption is being used and provides the MagStripeReader's key for decryption.
	SecureFormat	string	50	When present, indicates the format to use for decrypting the track data. Possible values: MagneSafeV1, MagneSafeV2, SecureMag and SecureMagV2.
	BDKSlot	numeric	2	When present, indicates the Base Derived Key slot to use as an override for Hardware Decryption.
2	Track	string		When present, can contain alphanumeric characters and spaces, periods, slashes (/), percent symbols (%), caret symbols (^), question marks (?), semicolons (;), equality symbols (=) or hyphens (-). Can use hardware encryption.
	Track1	string		Overrides Track field. Validated against acceptable characters in card swipe. Can use hardware encryption.
	Track2	string		Overrides Track field. Validated against acceptable characters in card swipe. Can use hardware encryption.
R	TransactionType	string	11	Possible values: activate, deactivate, reactivate
R	TransIndustryType	string	2	Must match a 2-letter industry type for credit cards. All caps. Supported values are: RE (Retail), RS (Restaurant), EC (eCommerce), DM (Direct Marketing) and LD (Lodging).
	EMVTags	string	255	Combined EMV tags presented in TLV format.
	LaneCode	String	10	Lane number associated with the transaction (when applicable).

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>020</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    <MerchantCode></MerchantCode>
    <MerchantAccountCode></MerchantAccountCode>
    <PaymentAccountNumber></PaymentAccountNumber>
    <Amount></Amount>
    <MSRKey></MSRKey>
    <SecureFormat></SecureFormat>
    <BDKSlot></BDKSlot>
    <Track></Track>
    <Track1></Track1>
    <Track2></Track2>
    <TransactionType></TransactionType>
    <TransIndustryType></TransIndustryType>
    <EMVTags></EMVTags>
    <LaneCode></LaneCode>
  </requestMessage>
</requestHeader>
```

2.2.14.1. Response

The response message is included in the **Gift** element and returns the following data elements:


	Parameter	Data Type	Length	Description
R	ResponseType	string	10	Possible Values: activate , deactivate , reactivate
R	MerchantAccountCode	integer		Merchant Account Code used for card management request
R	TransactionCode	string	60	Transaction Code used for card management request
R	GatewayTransId	integer		Identifier in gateway
R	GatewayMessage	string	255	Generated gateway response message
R	CashbackAmount	integer	8	Amount of money, left on a card and returned to customer upon deactivation. Only present on deactivate responses.
R	GatewayResult	string	3	Gateway generated response code

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<Gift>
  <TransactionID></TransactionID>
  <RequestType>022</RequestType>
  <ResponseCode>00000</ResponseCode>
  <ResponseDescription>Successful Request</ResponseDescription>
  <responseMessage>
    <ResponseType></ResponseType>
    <MerchantAccountCode></MerchantAccountCode>
    <TransactionCode></TransactionCode>
    <GatewayTransID></GatewayTransID>
    <GatewayMessage></GatewayMessage>
    <CashbackAmount></CashbackAmount>
    <GatewayResult></GatewayResult>
  </responseMessage>
</Gift>
```

2.2.15. Ping (Request 099)

Use the ping request to determine the status of the gateway.

 This message requires no additional parameters from the base request.

XML Request Layout

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>099</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage />
</requestHeader>
```


2.2.15.1. Response

The response message is included in the **Ping** element with no extra data elements.

XML Response Layout

```
<?xml version="1.0" encoding="utf-16"?>
<Ping>
  <TransactionID></TransactionID>
  <RequestType>099</RequestType>
  <ResponseCode>00000</ResponseCode>
  <ResponseDescription>SuccessfulRequest</ResponseDescription>
</Ping>
```

2.3. Generic Error Response

All of the service responses described in the preceding sections reflect a successful result. If the results of a specific request are not successful, a “generic error” response will be returned instead of a successful response. The generic error response returns the following parameters:

	Parameter	Data Type	Length	Description
R	TransactionID			The TransactionID value submitted with the original request.
R	RequestType			The RequestType value from the original request that has failed.
R	ResponseCode	numeric	5	A five-digit response code describing the results of the transaction. See Appendix A.1 for a list of valid response values.
R	ResponseDescription	string		Description of the error response code (ResponseCode).

Response Layout

The following is an example of an error response due to an invalid client identifier (the TransactionID value would be the actual value from the initial service call that failed):

```
<?xml version="1.0" encoding="utf-16"?>
<ErrorResponse>
  <TransactionID>0312-7958-4465-0664</TransactionID>
  <RequestType>004</RequestType>
  <ResponseCode>10020</ResponseCode>
  <ResponseDescription>Invalid Client Identifier</ResponseDescription>
  <responseMessage />
</ErrorResponse>
```

The following is an example of an error response due to invalid access credentials (the TransactionID value would be the actual value from the initial service call that failed):

```
<?xml version="1.0" encoding="utf-16"?>
<ErrorResponse>
  <TransactionID>0312-7958-4465-0664</TransactionID>
  <RequestType>006</RequestType>
  <ResponseCode>10024</ResponseCode>
  <ResponseDescription>Invalid Credentials</ResponseDescription>
  <responseMessage />
</ErrorResponse>
```

A. Appendix

A.1. ResponseCode Values

The following table displays the possible numeric values and descriptions returned by BridgePay in the ResponseCode field. It also describes the TPI Result code that is returned for each message. Note that not all response codes between TPI and BridgePay have a one-to-one relationship.

BridgePay Code	TPI Code	Description
00000	0	Successful request
00001	0	Partial Authorization
10001	19	Missing Reference Number
10002	23	Invalid Card Number - Blank/Null
10003	1010	Invalid Card Type - Doesn't match accepted card types
10004	24	Invalid Expiration Date - Blank/Null
10005	7	Invalid Security Code - Blank/Null
10007	23	Invalid Card Number - Not Numeric
10008	23	Invalid Length for card type
10009	24	Invalid Expiration Date - Card Expired
10010	7	Invalid Security Code - Not Numeric
10011	7	Invalid Transaction ID
10012	23	Invalid Card Number - Failed Mod10
10013	24	Invalid Expiration Date Value
10014	7	Invalid Security Code Length
10017	24	Invalid Expiration Date - Invalid Month
10018	24	Invalid Expiration Date - Invalid Year
10019	24	Invalid Expiration Date
10020	1	Invalid Client Identifier
10021	1007	Invalid Request Element – Missing Element
10022	1007	Invalid Request Type
10023	1007	Password Expired
10024	1001	Invalid Credentials
10025	7	Invalid Zip – Not Numeric
10026	7	Invalid Zip – Wrong Length

BridgePay Code	TPI Code	Description
10027	4	Invalid Amount – Blank/Null
10028	4	Invalid Amount – Not Numeric
10029	7	Invalid Request Date/Time
10030	7	Invalid Token
10031	7	Invalid Track
10032	7	Invalid Track Identifier
10033	108	Invalid Void Request
10034	7	Invalid Encryption ID
10035	23	Invalid Account Number – Blank/Null
10036	23	Invalid Account Number – Not Numeric
10037	1010	Invalid Payment Type – Blank/Null
10038	1010	Invalid Payment Type – Unrecognized Payment Type
10039	23	Invalid Account Number – Account Number Does Not Exist
10040	7	Missing Required Pass-Thru Data Element
10041	7	Missing / Invalid BIN
20001	99	Tokenization Service Connection Error
20002	99	BridgePay Internal Server Error
20003	99	Client Service Unavailable
20004	99	Payment Service Sensitive Data Timeout
30004	1007	Invalid Request Message
30005	1007	Invalid Response Message
30006	1001	New Password Doesn't Match Confirmation Password
30007	1001	New Password Too Weak*
30008	23	Missing Payment Card Data
30009	99	Internal Payment Card Data Error
30010	99	Invalid Record Format
30011	5	Invalid Merchant Number (from Gateway)
30012	23	Bad Card Number (from Gateway)
30013	5	Invalid Store Number
30020	1007	Invalid Transaction Industry Type
30021	1007	Missing Transaction Industry Type
30022	99	Processing Network Unavailable

BridgePay Code	TPI Code	Description
30023	23	Invalid Account Number
30024	2	No Account
30025	12	Invalid Security Code
30026	4	Invalid Amount
30028	99	Settlement Failed
30029	99	Transaction Error
30030	99	Transaction data integrity validation error
30032	12	Denied by customer's bank
30033	50	Insufficient funds
30034	12	Hold - Pick up card
30035	99	Incorrect PIN
30036	110	Duplicate Transaction
30037	12	Card reported lost
30038	12	Card reported stolen
30039	99	Service not allowed
30040	99	Stop Recurring
30041	99	Unattempted Batch Decline
30042	99	Maximum transaction limit is exceeded at the moment. Try processing your transaction tomorrow.
30043	99	Re-enter Transaction
30044	12	Unmapped decline
30045	99	Billing profile configuration error
30046	99	Pin Try Exceeded
30047	99	Refund was not processed/received
30048	99	Chargeback received
30049	99	Processing Canceled by User
30050	1007	Invalid Transaction Category
30051	1007	Invalid Verification Status
30052	1007	Invalid Terminal Type
30053	1007	Invalid Petroleum Product Type
30060	1007	Invalid IApp User Id
30061	99	Account insert failed
30062	99	Merchant insert failed

BridgePay Code	TPI Code	Description
30070	1001	New Password Previously Used
30071	1001	Missing Clerk Id
30072	13	Call for Authorization
30073	12	Card is Restricted
30074	12	Declined due to fraud rules
30075	12	Bank Account Blacklisted
30100	1007	Invalid Account Data – Blank/Null
30101	1007	Invalid EMV Tag Data – Blank/Null
40001	99	Lodging Reauth Failed
40002	99	Missing Lodging Folio Number
80000	0	Gateway Services Available
80001	99	Gateway Services Unavailable
80002	1001	Invalid Purchase Token
90000	99	Token Store failed to encrypt a token
90001	99	Token Store failed to decrypt a token
99999		Unknown Error

A.2. Base64-Encoded SOAP Message Sample

The following message samples show the proper way to base64 encode a request message and the resulting base64-encoded response message:

Request Message

[illegible]

Response Message

```
<?xml version='1.0' encoding='utf-8'>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <ProcessRequestResponse xmlns="http://bridgepaynetsecuretx.com/requesthandler">
      <ProcessRequestResult>
        PD94bwWgdMvYc2lvbj0iMS4wIiB1bmNvZGluzZ0idXRmLETEiJ8+DQo8RXJyb3JJSZXNwb25zZSB4bWxuczp4c2k9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hLW1uc3RhbmNlIiB4bWxuczp4c2Q9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hIj4NCiAgPFYyY5ZWN0aW9uSUQ+MTAwMTwvVHJhbnNhY3Rpb25JRd4NCiAgPFJlcXVlc3RUeXB1PjAwNDwvUmVxdWVzZFR5cGU+DQogIDxSZXNwb25zZUNvZGU+MTAwMjA8L1Jlc3BvbnNlQ29kZT4NCjwvRXJyb3JJSZXNwb25zZT4=
      </ProcessRequestResult>
    </ProcessRequestResponse>
  </s:Body>
</s:Envelope>
```

A.3. Including Merchant Information

Depending upon the several factors, you may be required to send more information to BridgeComm for it to correctly identify the merchant information it should use for processing. These factors include what technology you are using to connect to BridgeComm, whether you are using BridgePay's internal merchant lookup service or an outside service, and, in some cases, the processor intended to receive the request. In your welcome package, you should receive this information from BridgePay once your account has been established.

To include this information in a request to BridgeComm, you will simply include another set of fields in the request message itself. Although there are many configurations and more are being added each day, the most common is the combination of Merchant Code and Merchant Account Code. If you are using the BridgePay internal merchant lookup service, you will be using these two fields.

To add them to the request, simply place them anywhere in the body of the requestMessage element, prior to Base64 encoding. See the example below with the Merchant Code and Merchant Account Codes highlighted.

XML Request Message

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <ClientIdentifier></ClientIdentifier>
  <TransactionID></TransactionID>
  <RequestType>004</RequestType>
  <RequestDateTime></RequestDateTime>
  <User></User>
  <Password></Password>
  <requestMessage>
    .
    .
    .
    <MerchantCode></MerchantCode>
    <MerchantAccountCode></MerchantAccountCode>
    .
    .
    .
  </requestMessage>
</requestHeader>
```

Note that not every request type requires this information and not every request type requires these exact fields. Be sure to work with your BridgePay representative when determining which message types require the merchant lookup fields and their content.

A.4. Industry Specific Fields

Depending upon the industry, you can receive better transaction rates by specifying your industry and including industry specific fields in transaction requests. This appendix provides information for each specialized industry and what fields you can use to support these transactions.

Lodging

	Parameter	Data Type	Length	Description
O	RoomNumber	String	25	Hotel room number
C	FolioNumber	String	15	Hotel folio number Required when LodgingChargeType = 'H'
C	RoomRateAmount	Numeric	8	Daily room rate. Implied decimal. Required when LodgingChargeType = 'H'
O	RoomTaxAmount	Numeric	8	Tax charged on daily rate. Implied decimal.
R	LodgingChargeType	Enum		Type of service for which this transaction is processed. <ul style="list-style-type: none"> • H = Hotel • R = Restaurant • G = Gift Shop • S = Health Spa • B = Beauty Shop • F = Convention Fee • T = Tennis • O = Golf
C	CheckInDate	Date		The date of actual or anticipated check-in. Required when LodgingChargeType = 'H'
C	CheckOutDate	Date		The date of actual or anticipated check-out. Required when LodgingChargeType = 'H'
O	StayDuration	Numeric		Indicates the length of anticipated, actual or incremental stay.
C	SpecialProgramType	Enum		Important when the cardholder did not stay at the hotel. <ul style="list-style-type: none"> • AD = Advance deposit • AR = Assured reservation • DC = Delayed charge • ES = Express service • NC = Normal charge • NS = No show charge Required when LodgingChargeType = 'H'
O	DepartureAdjAmount	Numeric	8	Additional amount charged after cardholder left the hotel.
C	LodgingItemCount	Numeric		Conditional, required if lodging items present in request. Indicates number of lodging industry items purchased as part of this transaction. See Lodging Item Records for more detail. Required when LodgingChargeType = 'H'
C	LodgingItems	Collection		Collection of lodging item records.

Lodging Item Records

Clients processing lodging industry transactions and desiring to itemize the charges for additional service can include lodging items sub-records as a part of the request message. When lodging item information is supplied, the client is required to indicate the number of sub-records in the **LodgingItemCount** field.

	Parameter	Data Type	Length	Description
R	LodgingItemType	Enum		Lodging extra charge type <ul style="list-style-type: none">• G = Gift Shop• L = Laundry• B = Mini bar• R = Restaurant• T = Telephone
R	LodgingItemAmount	Numeric	8	Amount for lodging extra charge item. Implied decimal.

EXAMPLES

To process an authorization or capture via the ProcessRequest access method for a merchant using lodging industry fields, the RequestMessage element would have the following tags embedded within it:

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  .
  .
  .
  <requestMessage>
    .
    .
    .
    <RoomNumber></RoomNumber>
    <FolioNumber></FolioNumber>
    <RoomRateAmount></RoomRateAmount>
    <RoomTaxAmount></RoomTaxAmount>
    <LodgingChargeType></LodgingChargeType>
    <CheckInDate></CheckInDate>
    <CheckOutDate></CheckOutDate>
    <StayDuration></StayDuration>
    <SpecialProgramType></SpecialProgramType>
    <DepartureAdjAmount></DepartureAdjAmount>
    <LodgingItemCount></LodgingItemCount>
    <LodgingItem>
      <LodgingItemType></LodgingItemType>
      <LodgingItemAmount></LodgingItemAmount>
    </LodgingItem>
    .
    .
    .
  </requestMessage>
</requestHeader>
```

To process an authorization or capture via the ProcessJsonRequest access method for a merchant using lodging industry fields, the elements would be added into the JSON request.

```
{ "request": {  
  .  
  .  
  .  
  "dataElements": [  
    { "name": "roomNumber", "value": "" },  
    { "name": "folioNumber", "value": "" },  
    { "name": "roomRateAmount", "value": "" },  
    { "name": "roomTaxAmount", "value": "" },  
    { "name": "lodgingChargeType", "value": "" },  
    { "name": "checkInDate", "value": "" },  
    { "name": "checkOutDate", "value": "" },  
    { "name": "stayDuration", "value": "" },  
    { "name": "specialProgramType", "value": "" },  
    { "name": "departureAdjAmount", "value": "" },  
    { "name": "lodgingItemCount", "value": "" },  
    { "name": "lodgingItems": [  
      { "dataElements": [  
        { "name": "lodgingItemType", "value": "" },  
        { "name": "lodgingItemAmount", "value": "" }  
      ],  
      { "dataElements": [  
        { "name": "lodgingItemType", "value": "" },  
        { "name": "lodgingItemAmount", "value": "" }  
      ] }  
    ]  
  } ]  
} }
```

Car Rental

	Parameter	Data Type	Length	Description
R	RentalAgreementNumber	string	25	Car Rental Agreement Number. Required as part of the Rental Key.
O	RentalDailyRateAmount	integer	8	Total rent amount per day. Implied decimal.
R	RentalDuration	integer	4	Car Rental Period.
O	RentalExtraChargesAmount	integer	8	Can rental special services amount. Implied decimal.
O	RentalInsuranceAmount	integer	8	Car insurance amount. Implied decimal.
O	MaxFreeMiles	integer	8	Total mileage for rent period.
O	MileRateAmount	integer	8	Total rent amount per mile. Implied decimal.
R	RentalName	string	20	Name of the person who rented the car.
O	RentalCity	string	35	City where the car rental starts.
O	RentalCountryCode	string	2	Country where the car rental starts.
R	RentalDate	date	10	Date when the car rental starts. (MM/DD/YYYY format)
O	RentalState	string	3	State where the car rental starts.
O	RentalTime	integer	4	Car rental start time in 24-Hour Format. (HHMM)
O	ReturnLocationCode	string	10	Identification code of the place where car was returned.
O	ReturnCity	string	35	City where the car is returned
O	ReturnCountryCode	string	2	Country where the car is returned.
R	ReturnDate	date	10	Date when the car should be returned.
O	ReturnState	string	3	State where the car is returned.
O	ReturnTime	integer	4	Time when the car should be returned in 24-Hour Format. (HHMM)
R	RentalSpecialProgramType	string	2	String based enum representing the special program type for the rental. Accepted values are AD (Advance Deposit), AR (Assured Reservation), DC (Delayed Charge), ES (Express Service), NC (Normal Charge), NS (No Show Charge) and FR (Frequent Renter)
O	TotalMiles	integer	8	Rented car total mileage (miles on car)
O	RentalExtraChargeItemCount	integer	4	Count of extra charges included in the transaction.
O	RentalExtraChargeItems	Collection	n/a	List of extra charges. See below.

Car Rental Item Records

Clients processing car rental industry transactions and desiring to itemize the charges for additional service can include car rental items sub-records as a part of the request message. When car rental item information is supplied, the client is required to indicate the number of sub-records in the **RentalExtraChargesItemCount** field.

	Parameter	Data Type	Length	Description
R	RentalExtraChargeItemType	Enum		Car rental extra charge type <ul style="list-style-type: none">• M = Extra Mileage• S = Gas• N = Late Return• W = One-Way Service• P = Parking Violation
R	RentalExtraChargeTypeAmount	Numeric	8	Amount for car rental extra charge item. Implied decimal.

EXAMPLES

To process an authorization or capture via the ProcessRequest access method for a merchant using car rental industry fields, the RequestMessage element would have the following tags embedded within it:

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  <requestMessage>
    <RentalName></RentalName>
    <RentalCity></RentalCity>
    <RentalState></RentalState>
    <RentalCountryCode></RentalCountryCode>
    <RentalDate></RentalDate>
    <RentalTime></RentalTime>
    <ReturnCity></ReturnCity>
    <ReturnState></ReturnState>
    <ReturnCountryCode></ReturnCountryCode>
    <RentalExtraChargesAmount></RentalExtraChargesAmount>
    <ReturnDate></ReturnDate>
    <ReturnTime></ReturnTime>
    <RentalAgreementNumber></RentalAgreementNumber>
    <RentalDailyRateAmount></RentalDailyRateAmount>
    <RentalDuration></RentalDuration>
    <RentalInsuranceAmount></RentalInsuranceAmount>
    <MaxFreeMiles></MaxFreeMiles>
    <MileRateAmount></MileRateAmount>
    <ReturnLocationCode></ReturnLocationCode>
    <RentalSpecialProgramType></RentalSpecialProgramType>
    <TotalMiles></TotalMiles>
    <RentalExtraChargesItemCount></RentalExtraChargesItemCount>
    <RentalExtraChargeItem>
      <RentalExtraChargeItemType></RentalExtraChargeItemType>
      <RentalExtraChargeItemAmount></RentalExtraChargeItemAmount>
    </RentalExtraChargeItem>
  </requestMessage>
</requestHeader>
```

To process an authorization or capture via the ProcessJsonRequest access method for a merchant using lodging industry fields, the elements would be added into the JSON request.

```
{ "request": {
  "dataElements": [
    { "name": "rentalName", "value": "" },
    { "name": "rentalCity", "value": "" },
    { "name": "rentalState", "value": "" },
    { "name": "rentalCountryCode", "value": "" },
    { "name": "rentalDate", "value": "" },
    { "name": "rentalTime", "value": "" },
    { "name": "returnCity", "value": "" },
    { "name": "returnState", "value": "" },
    { "name": "returnCountryCode", "value": "" },
    { "name": "returnTime", "value": "" },
    { "name": "rentalExtraChargesAmount", "value": "" },
    { "name": "returnDate", "value": "" },
    { "name": "rentalAgreementNumber", "value": "" },
    { "name": "rentalDailyRateAmount", "value": "" },
    { "name": "rentalDuration", "value": "" },
    { "name": "rentalInsuranceAmount", "value": "" },
    { "name": "maxFreeMiles", "value": "" },
    { "name": "mileRateAmount", "value": "" },
    { "name": "returnLocationCode", "value": "" },
    { "name": "rentalSpecialProgramType", "value": "" },
    { "name": "totalMiles", "value": "" },
    { "name": "rentalExtraChargesItemCount", "value": "" },
    { "name": "rentalExtraChargeItems": [
      { "dataElements": [
        { "name": "rentalExtraChargeItemType", "value": "" },
        { "name": "rentalExtraChargeItemAmount", "value": "" }
      ]
    },
    { "dataElements": [
      { "name": "rentalExtraChargeItemType", "value": "" },
      { "name": "rentalExtraChargeItemAmount", "value": "" }
    ]
  ]
} ]
}
```

Performing Grouped Authorizations & Captures

Initial Authorizations

The formatting and content of a transaction request to BridgeComm will identify whether the request is an initial authorization or an incremental authorization. A series of transactions for lodging or car rental are identified by a **transaction group key**. The group key is comprised of four data elements: the card number/token, the check-in date, the folio number or rental agreement number and the merchant lookup information.

When sending a grouped transaction to BridgeComm, the key is constructed based on the data elements mentioned above. If the transaction is the first transaction using that key, an initial authorization is created and the settlement delay is set to either the duration of the stay or the difference between the check-in date (rental start) and the check-out date (rental end).

Incremental Authorizations

Incremental authorizations can be generated in one of two ways. The originating system can either send an incremental request by providing the amount to increment the total, or by sending the new total. For example, if the initial authorization is \$250.00, to increase the total authorization amount to

\$300.00, the originating system can either send a \$50.00 incremental message, or a \$300.00 total message. Both operations end up in acquiring only \$50.00 more for the key.

To send an incremental authorization with only the delta amount, the message to BridgeComm will set the **TransactionType** field to **increment**, and send the incremental amount in the **Amount** field. The originating system must also include all other elements required to process the transaction.

To send an incremental authorization by specifying the new total amount, the message to BridgeComm will set the **TransactionType** field to **sale-auth**, and send the new total amount in the **Amount** field. The originating system must also include all other elements required to process the transaction.

Capture/Check-Out/Return

The capture process for a group of authorizations for a key is handled in much the same way. It is not necessary to capture each individual authorization. Instead, the merchant sends a capture request (See 2.2.11 – Capture Request) including the Card Number or Token, Check-In Date, Folio Number (or Rental Agreement Number) and Merchant Identification information. All authorizations for that key are then totaled and captured as one transaction in preparation for settlement. If the final amount of the transaction is different than the amount authorized to date, the merchant can also send the final amount with the capture message. This will trigger either a partial reversal (if the final amount is less than the total authorized to date amount) or a final incremental authorization (if the final amount is greater than the total authorized to date amount). This capture method also disables the settlement delay, allowing the transaction to be included in the next settlement batch.

Healthcare

When processing a healthcare industry transaction (using an FSA or HSA account) additional amount fields need to be specified. The only amount that is required is **HealthCareAmt**, while other amounts are optional. If medical transportation was involved, then **TransitAmt** should be added.

Depending on the type of service rendered, additional amounts can be specified to HealthCareAmt to indicate how much was spent on prescription, vision/optical, dental and clinic/hospitalization services. The total of prescription, vision, dental and clinic amounts can be less than or equal to HealthCareAmt.

	Parameter	Data Type	Length	Description
R	HealthCareAmt	Numeric	8	Indicates the portion of the amount field that was spent on healthcare-related services. Implied decimal.
O	TransitAmt	Numeric	8	Indicates the portion of the amount field that was spent on healthcare-related transportation. Implied decimal.
O	PrescriptionAmt	Numeric	8	Indicates the portion of the HealthCareAmt field that was spent on prescription drugs. Implied decimal.
O	VisionAmt	Numeric	8	Indicates the portion of the HealthCareAmt field that was spent on vision-related (optical) medical services. Implied decimal.
O	DentalAmt	Numeric	8	Indicates the portion of the HealthCareAmt field that was spent on dental medical services. Implied decimal.
O	ClinicAmt	Numeric	8	Indicates the portion of the HealthCareAmt field that was spent on hospitalization. Implied decimal.
O	IsQualifiedIIAS	Boolean		Indicates whether items purchased were verified against IIAS as qualified for healthcare purchases. Value must be True or False .

EXAMPLES

To process an authorization via the ProcessRequest access method for a merchant using healthcare industry fields, the RequestMessage element would have the following tags embedded within it:

```
<?xml version="1.0" encoding="utf-16"?>
<requestHeader>
  .
  .
  .
  <requestMessage>
    .
    .
    .
    <TransCatCode>H</TransCatCode>
    <HealthCareAmt></HealthCareAmt>
    <TransitAmt></TransitAmt>
    <PrescriptionAmt></PrescriptionAmt>
    <VisionAmt></VisionAmt>
    <DentalAmt></DentalAmt>
    <ClinicAmt></ClinicAmt>
    <IsQualifiedIIAS></IsQualifiedIIAS>
    .
    .
    .
  </requestMessage>
</requestHeader>
```


To process an authorization via the ProcessJsonRequest access method for a merchant using healthcare industry fields, the elements would be added into the JSON request.

```
{ "request": {  
  .  
  .  
  .  
  "dataElements": [  
    { "name": "transCatCode", "value": "H" },  
    { "name": "healthCareAmt", "value": "" },  
    { "name": "transitAmt", "value": "" },  
    { "name": "prescriptionAmt", "value": "" },  
    { "name": "visionAmt", "value": "" },  
    { "name": "dentalAmt", "value": "" },  
    { "name": "clinicAmt", "value": "" },  
    { "name": "isQualifiedIIAS", "value": "" },  
    .  
    .  
    .  
  ]  
} }
```

A.5. Track Data Handling

If you are sending track data to BridgeComm, there are two ways of sending the data. You may choose the integration specifics that work best for you. The options are Combined Track Information (CTI) or Split Track Information (STI).

Combined Track Information (CTI)

If your integration requires sending track information all in one field, you can use either the **TrackData** element or the **Track** element to send the data. It should contain both Track 1 and Track 2 concatenated and sent in one string.

To send the data without extra encryption, simply concatenate the two fields with no extra spaces. You must include all start and end sentinels and the field separators as well.

To send the data with hardware level encryption, your reader must support encrypting combined track data. This encrypted combined track data would then be placed in the **TrackData** or **Track** element.

To send the data with RSA level encryption, you must use the encryption details received from the **Generate Encryption Key** method (003) to encrypt the data. Then send the encrypted string in the **TrackData** or **Track** element.

Split Track Information (STI)

If your integration requires sending track information in separate fields, you can use the **Track1** and **Track2** data elements to send the data.

To send the data without extra encryption, simply send the appropriate data elements in the appropriate fields. You must include all start and end sentinels and the field separators as well.

To send the data with hardware level encryption, your reader must provide the encrypted data in separate fields. This encrypted combined track data would then be placed in the **Track1** and **Track2** elements respectively.

RSA level encryption is not available for Split Track Information. To use RSA level encryption, you will need to send the data using the CTI method described above.

A.6. Purchase Tokens

If you wish to use Purchase Tokens to further secure your payment transactions, you must first acquire a purchase token from BridgeComm. A Purchase Token is valid for exactly one transaction and only for 15 minutes after creation. This appendix describes how to acquire and consume a purchase token.

Acquiring a Purchase Token

To acquire a purchase token, your application must consume the `AcquirePurchaseToken` method on the `ActionService` web service.

```
string AcquirePurchaseToken(string userName, string password, string certificationId, int transactionAmount, string purchaserInfo, string transactionInfo)
```

	Parameter	Data Type	Length	Description
R	userName	String		The username assigned for accessing BridgeComm
R	password	String		The password assigned to the user for accessing BridgeComm
O	certificationId	String		Not required. Default to and empty string. Reserved for future use.
R	transactionAmount	Int	8	The implied decimal amount of the transaction being preauthorized. (e.g. 125 for a \$1.25 authorization)
O	purchaserInfo	String		Not required. A merchant provided string identifying the customer account the transaction is for. Compared with CustomerAccountCode element in the transaction request.
O	transactionInfo	String		Not required. A merchant provided string identifying the invoice the transaction is for. Compared with the InvoiceNum element in the transaction request.

When BridgeComm receives a request for a new purchase token, BridgeComm will first look to see if a purchase token for a similar request using the same data elements has already been generated and not yet consumed. If so, no new purchase token will be generated and the service will return an empty string.

If no existing purchase token exists, one will be generated and returned to the merchant.

Consuming a Purchase Token

Once the merchant has received a purchase token, the merchant has 15 minutes to use the token. To consume the token, the merchant must send the purchase token in the authorization request and all information used to create the purchase token must be included in the request.

If the merchant included the **purchaserInfo** in the request to acquire a purchase token, that data must be included in the CustomerAccountCode element of the transaction request.

If the merchant included the **transactionInfo** in the request to acquire a purchase token, that data must be included in the InvoiceNum element of the transaction request.

Example

1. Merchant calls **AcquirePurchaseToken**
 - **AcquirePurchaseToken**("user", "password", "", 500, "12345", "98765")
2. Merchant receives PurchaseToken
 - returnValue = "0E0D6C55-AF5E-49AE-8DE7-001B08A550CF"
3. Merchant calls **ProcessRequest**
 - **ProcessRequest**("<requestHeader>
<ClientIdentifier>SOAP</ClientIdentifier>
<TransactionId>1234567890</TransactionId>
<RequestType>004</RequestType>
<RequestDateTime>2015-01-01</RequestDateTime>
<requestMessage>
...
<PurchaseToken>0E0D6C55-AF5E-49AE-8DE7-001B08A550CF </PurchaseToken>
<Amount>500</Amount>
<CustomerAccountCode>12345</CustomerAccountCode>
<InvoiceNum>98765</InvoiceNum>
...
</requestMessage>
</requestHeader>")

A.7. EMV Processing

Processing EMV transactions through BridgeComm requires that the merchant provide a few more details in the transaction request. If the EMV reader is an encrypted reader, the merchant will also need to send the encrypted swipe data to BridgeComm as well to achieve tokenization.

Field Requirements

- **EMVTags** – This element contains all the EMV tags returned from the EMV reader, combined into a single string using TLV encoding.
- **EntryModeType** – This field identifies how the transaction was processed by the terminal.
- **EntryMediumType** – This field identifies what type of media was used to collect the transaction.
- **EntryPINModeType** – This field identifies whether the media used provides PIN features.
- **TerminalCapabilities** – This field identifies the capabilities of the terminal used to collect the data.

EMV Fields Example (Unencrypted)

```
<requestHeader>
  <ClientIdentifier>ClientId</ClientIdentifier>
  <TransactionID>XXXX-XXXX-XXXX-XXXX</TransactionID>
  <RequestType>004</RequestType>
  <RequestDateTime>01/01/2015</RequestDateTime>
  <User>mbpUser</User>
  <Password>mbpPassword</Password>
  <requestMessage>
    <transindustrytype>RE</transindustrytype>
    <TransactionType>sale-auth</TransactionType>
    <AcctType>R</AcctType>
    <Amount>1000</Amount>
    <MerchantCode>1000</MerchantCode>
    <MerchantAccountCode>1001</MerchantAccountCode>
    <HolderType>P</HolderType>

    <EMVTags>4F07A0000000031010500A5669736120446562697457104450649200973734D19092260003680F82023C00
8407A000000003101095050800000009A031504309B02E8009C01005F2A0208405F3401019F0206000000005009F0
607A00000000310109F0902008C9F100706010A03A020009F120A566973612044656269749F1A0208409F1E08383032
34373636319F21032109149F2608FA940EE08082F8A59F2701809F3303E0B8C89F34031E03009F3501229F360200109
F37043F2BE80F9F3901059F4005F000F0A0019F4104000003819F530152</EMVTags>
    <EntryMedium>CC</EntryMedium>
    <EntryMode>HP</EntryMode>
    <EntryPINMode>X</EntryPINMode>

    <TerminalCapabilities>unknown|unused|manual|stripe|barcode|qrcode|ocr|icc|contactless|signature
|rfid|micr</TerminalCapabilities>  </requestMessage>
</requestHeader>
```

EMV Fields Example (Encrypted)

```
<requestHeader>
  <ClientIdentifier><ClientId/>ClientIdentifier>
  <TransactionID>XXXX-XXXX-XXXX-XXXX/>TransactionID>
  <RequestType>004/>RequestType>
  <RequestDateTime>01/01/2015/>RequestDateTime>
  <User>mbpUser/>User>
  <Password>mbpPassword/>Password>
  <requestMessage>
    <transindustrytype>RE/>transindustrytype>
    <TransactionType>sale-auth/>TransactionType>
    <AcctType>R/>AcctType>
    <Amount>1000/>Amount>
    <MerchantCode>1000/>MerchantCode>
    <MerchantAccountCode>1001/>MerchantAccountCode>
    <HolderType>P/>HolderType>

  <Track>9990A46CC78D0CDACA4CB5A3F9A73A4CEF1127B642BD11A4CF89E290EB96CFF1C745CC6AB9D64E53/>Track>
    <MSRKS>910701000000002000CC/>MSRKS>
    <SecureFormat>SecureMag/>SecureFormat>

  <EMVTags>4F07A000000004101050104465626974204D61737465724361726457105128570000000856D17086220000
0000820239008407A0000000041010950504002000009A031505089B02E8009C01005F2A0208405F3401009F0206000
0000012009F0607A00000000410109F090200029F10120110A000002A00000000000000000000FF9F121044656269
74204D6173746572436172649F1A0208409F1E0838303138393539369F21031116219F26087FE93875A4F4F3899F270
1809F3303E0B8C89F34031E03009F3501229F360202229F37044BB2D57D9F3901059F4005F000F0A0019F4104000005
659F530152/>EMVTags>
    <EntryMedium>CC/>EntryMedium>
    <EntryMode>HP/>EntryMode>
    <EntryPINMode>X/>EntryPINMode>

  <TerminalCapabilities>unknown|unused|manual|stripe|barcode|qrcode|ocr|icc|contactless|signature
|rfid|micr/>TerminalCapabilities>
  </requestMessage>
</requestHeader>
```

A.8. Non Gateway Transactions (Check/Cash)

BridgeComm is able to record transactions that did not occur within the gateway such as Check and Cash transactions. To record a transaction of this nature, the merchant must send specific values to identify it as a Cash or Check transaction.

Field Requirements

- **TransactionType** – When sending a Check transaction or a Cash transaction, this field must be set to **sale-info** for transactions that represent income to the merchant or **credit-info** for transactions that represent refunds or other credits the merchant is making for a customer.
- **AccountType** – If the request is recording a cash transaction, the AccountType field should be set to **A**. If the request is recording a physical check transaction, the AccountType field should be set to **K**.
- **CheckImageFront** – If the request is recording a check transaction, the merchant can provide an image of the front of the check using this field.
- **CheckImageBack** – If the request is recording a check transaction, the merchant can provide an image of the back of the check using this field.
- **MICR** – If the request is recording a check transaction, the merchant can provide the MICR data associated with the check in this field.

Cash Fields Example

```
<requestHeader>
  <ClientIdentifier>clientId</ClientIdentifier>
  <TransactionID>XXXX-XXXX-XXXX-XXXX</TransactionID>
  <RequestType>004</RequestType>
  <RequestDateTime>01/01/2015</RequestDateTime>
  <User>mbpUser</User>
  <Password>mbpPassword</Password>
  <requestMessage>
    <transindustrytype>RE</transindustrytype>
    <TransactionType>sale-info</TransactionType>
    <AcctType>A</AcctType>
    <Amount>1000</Amount>
    <MerchantCode>1000</MerchantCode>
    <MerchantAccountCode>1001</MerchantAccountCode>
  </requestMessage>
</requestHeader>
```

Check Fields Example

```
<requestHeader>
  <ClientIdentifier>clientId</ClientIdentifier>
  <TransactionID>XXXX-XXXX-XXXX-XXXX</TransactionID>
  <RequestType>004</RequestType>
  <RequestDateTime>01/01/2015</RequestDateTime>
  <User>mbpUser</User>
  <Password>mbpPassword</Password>
  <requestMessage>
    <transindustrytype>RE</transindustrytype>
    <TransactionType>sale-info</TransactionType>
    <AcctType>K</AcctType>
    <CheckImageFront>base64EncodedTIFFImage</CheckImageFront>
    <CheckImageBack>base64EncodedTIFFImage</CheckImageBack>
    <MICR>:021000021:4099999992&lt;1001</MICR>
    <Amount>1000</Amount>
    <MerchantCode>1000</MerchantCode>
    <MerchantAccountCode>1001</MerchantAccountCode>
  </requestMessage>
</requestHeader>
```


A.9. Check21 Processing

Processing with Check21 services is similar to ACH processing. In order to process via Check21, the merchant must identify the transaction as a Check21 transaction and provide images of both the front and back of the physical check, along with the MICR data retrieved from the check reader.

Field Requirements

- **TransIndustryType** – When sending a Check21 transaction this field must be set to **C21**.
- **AccountType** – When sending a Check21 transaction, this field should be provided in the same manner as an ACH transaction with either **C** for checking account or **S** for saving account.
- **BankAccountNum** – As with any ACH transaction, this field must contain the account number of the transaction.
- **RoutingNum** – As with any ACH transaction, this field must contain the routing number associated with the account number.
- **CheckImageFront** – If the request is recording a check transaction, the merchant can provide an image of the front of the check using this field.
- **CheckImageBack** – If the request is recording a check transaction, the merchant can provide an image of the back of the check using this field.
- **MICR** – If the request is recording a check transaction, the merchant can provide the MICR data associated with the check in this field.

Check 21 Fields Example

```
<requestHeader>
  <ClientIdentifier>clientId</ClientIdentifier>
  <TransactionID>XXXX-XXXX-XXXX-XXXX</TransactionID>
  <RequestType>004</RequestType>
  <RequestDateTime>01/01/2015</RequestDateTime>
  <User>mbpUser</User>
  <Password>mbpPassword</Password>
  <requestMessage>
    <transindustrytype>C21</transindustrytype>
    <TransactionType>sale</TransactionType>
    <AcctType>C</AcctType>
    <BankAccountNum>021000021</BankAccountNum>
    <RoutingNum>4099999992</RoutingNum>
    <CheckImageFront>base64EncodedTIFFImage</CheckImageFront>
    <CheckImageBack>base64EncodedTIFFImage</CheckImageBack>
    <MICR>:021000021:4099999992&lt;1001</MICR>
    <Amount>1000</Amount>
    <MerchantCode>1000</MerchantCode>
    <MerchantAccountCode>1001</MerchantAccountCode>
  </requestMessage>
</requestHeader>
```

A.10. TPI Emulation

For those merchants migrating from TPI to BridgePay, BridgeComm offers limited TPI Request Emulation so that merchants can make minimal changes of their existing system to begin processing through BridgeComm.

The scope of this document does not include providing detailed information for sending TPI transactions. See the PathwayLink implementation guide for more information regarding TPI transaction processing.

Emulation Endpoint

In order to process transactions using TPI formatting through BridgeComm, you must change the endpoint to access BridgeComm's TPI Emulation endpoint.

Certification

<https://www.bridgepaynetsecuretest.com/PaymentService/TPIRequestHandler.asmx>

Production

<https://www.bridgepaynetsecuretx.com/PaymentService/TPIRequestHandler.asmx>

Emulated Methods

The following methods are available via BridgeComm's TPI Emulation Engine.

Method	Trans Type	Description
ProcessCreditCard	Auth	Processes Request Type "004" with Transaction Type "sale-auth"
	Sale	Processes Request Type "004" with Transaction Type "sale"
	Return	If a previous gateway transaction Id is provided in the PNRef field: Processes Request Type "012" with Transaction Type "refund" If no previous gateway transaction Id is provided: Processes Request Type "004" with Transaction Type "credit"
	Void	Processes Request Type "012" with Transaction Type "void"
	Reversal	Processes Request Type "012" with Transaction Type "void"
	Adjustment	Processes Request Type "019" with Transaction Type "capture"
	Force	If an auth code is provided in the AuthCode field: Processes Request Type "004" with Transaction Type "sale" If no auth code is provided: Processes Request Type "019" with Transaction Type "capture"
GetInfo	StatusCheck	Processes Request Type "099"