Getting Started with Payments Essentials

Contents

Audience and Purpose

This guide is written for application developers who want to use the Payments Processor API to integrate Payments services into their order management system. It describes the basic steps you must complete in order to get started with the ProcessorAPI.

Implementing the Payments services requires software development skills. You mustuse the Processor API request and reply fields to integrate the services into your existing order management system.

Conventions

Note and Important Statements

📝	A *Note* contains helpful suggestions or references to material not contained in the document.

⚠	An *Important* statement contains information essential to successfully completing a task or learning a concept.

Text and Command Conventions

Convention	Usage
Bold	■ Field and service names in text; for example: Include the **ics_applications** field. ■ Items that you are instructed to act upon; for example: Click **Save**.
`Screen text`	■ XML elements. ■ Code examples and samples. ■ Text that you enter in an API environment; for example: Set the **ccAuthService_run** field to `true`.

CHAPTER 1

Payment Processing with Payments

Payments Essentials is a payment solution that enables you to manage your payment transactions. Payments supports multiple payment methods and integration methods.

Understanding the Payment Industry

In the e-commerce industry, multiple organizations work together to make online transactions possible. The following table describes some of the most important types of organizations in this industry.

Table 1 Types of Organizations in the Payment Industry

Type of Organization	Description
Merchant	A person or company that sells goods or services.
Merchant (acquiring) bank	A bank that provides businesses with accounts to accept credit card or check payments.
Card association	Organizations, such as Visa, Mastercard, and Discover, that have business relationships with the banks that issue your customers' cards.
Payment processor	An organization that processes payment requests, such as credit card authorizations and settlements, and routes them to the appropriate card associations according to their guidelines. Your merchant bank's processor relationship determines which payment processor you use.
Payment gateway	An organization, such as Payments, that enables merchants to securely send order information to and receive it from payment processors in the appropriate format.

CHAPTER 2

Steps for Getting Started

Registering and Logging In

To start using Payments services, register for a Payments account. After registering, you receive a confirmation email with your Payments username. Your password to the Payments Test Business Center is the one you associate with yourusername during registration.

A Payments partner can also provide you with a Payments account. If you register through one of these partners, you receive two confirmation emails: one containing your unique Payments username and the other containing your temporary password to the Payments Business Center. The first time you log in to the Business Center, the system prompts you to create a permanent password.

If you don’t receive these emails, contact Customer Support for further assistance. If you have a Payments merchant account, the Customer Support phone number is includedin your approval email.

Becoming Familiar with the Business Center

The Payments Business Center is a powerful and secure web portal designed to helpyou manage your customers’ orders. It includes the Virtual Terminal should you choose to manually enter customers’ orders. So, becoming familiar with the Business Center is crucial to managing your business payments efficiently.

For business Center training videos, go to **Support > Training**.

The Business Center offers several options for processing order information and reducing fraud.

Initially, your Business Center account is in test mode, which enables you to become familiar with the Business Center by running test transactions. After you have completed testing, you can request to “Go Live” to begin processing real transactions. See "Going Live," page 11.

Installing and Testing the Processor API

Step 1 Generate security keys.

To ensure that you transmit information to Payments securely and to authenticate your transactions as belonging to your account, you need to generate security keys before you process transactions.

⚠ You must generate two transaction security keys—one for the Payments production environment and one for the test environment. For information about generating and using security keys, see *Creating and Using Security Keys* (PDF | HTML).

Step 2 Install a Processor API client.

a Choose a Processor API client based on your platform and level of programming experience.

b Learn how to use the API and client for running credit card orders by reading *Credit Card Services Using the Processor API*. For information about processing electronic checks, see *Electronic Check Services Using the Processor API*.

c Download the client and its related documentation, which tells you how to install, test, and use the client.

d Install the client by following the instructions in the documentation you downloaded with the client.

e Run the samples included with the client to ensure that the connection with Payments is established. After you are comfortable with the samples, add code to integrate the client with your own application.

Step 3 Test your account.

To ensure that Payments successfully receives your order information, Payments strongly recommends that you run test transactions after implementing your chosen connection method. The test environment gives you the opportunity to troubleshoot and correct any connection issues. To verify that your implementation is correct, request test transactions, including authorizations, captures, and credits. You can test using your owncredit card or, if you prefer to use test credit card numbers, use those provided in the following table with any future expiration date.

Table 2 Test Credit Card Numbers

Credit Card Type	Test Account Number[1]
Visa	4111 1111 1111 1111
Mastercard	5555 5555 5555 4444

Table 2 Test Credit Card Numbers (Continued)

Credit Card Type	Test Account Number[1]
American Express	3782 8224 6310 005
Discover	6011 1111 1111 1117
JCB	3566 1111 1111 1113
Diners Club	3800 000000 0006

1 remove spaces when sending to Payments.

Testing Information is available.

Going Live

After you have successfully tested your account and are ready to process real transactions, you can take your account live. Because go-live requests must be in writing, you must submit a go-live request in an eTicket. For more information on submitting an eTicket, see "Submitting an eTicket," page 14.

Request that Customer Support enable your merchant ID in production. Indicate which Payments services you would like to have enabled (for example, credit card processing, Decision Manager, or Payer Authentication). Be sure to include identifiers foryour payment processor. If you are not sure who your payment processor is, contact your merchant acquirer.

Managing Your Orders

After you begin processing orders, you might need to review your transactions for various reasons. For example, you might need to view all the orders processed on a specific day, check order details, or verify whether orders were approved or declined. With the Transaction Search feature of the Business Center, you can:

- Capture authorizations
- Review, credit, or void sales
- Create subscriptions from your customers' orders

Tracking and Reconciling Your Orders

The following table describes the values that you can use to track and reconcile your orders.

Table 3 Values for Tracking and Reconciling Orders

Value	Description
Request ID	The request ID is a unique identifier that Payments assigns to each request and returns in each reply message. You can use the request ID to do the following: ▪ Identify a transaction in a Payments report ▪ Search for a transaction in the Business Center ▪ Discuss a specific request with Customer Support ▪ Link a follow-on request to a primary request For specific field names, see the user guide for the payment method you are using.
Merchant Reference Code	The merchant reference code is an order tracking number that you generate and send in your requests so that you can track orders as they move through the phases of processing with Payments. Payments recommends that you use a unique merchant reference code for each order and use the same merchant reference code for all the requests associated with the order. This enables you to efficiently track the order in the Payments reportsand, on the Transaction Search pages on the Business Center. For specific field names, see the user guide for the payment method you are using.
Reconciliation ID	For most Payments services, the reply message includes a unique reconciliation ID that is assigned by Payments. For most payment processors, you can use this value to reconcile the transactions in your Payments reports with the transactions in your processor reports. For details, such as specific field names and information about processors that do not support reconciliation, see the user guide for the payment method you are using.
Additional Tracking Values	A few services provide additional tracking values. See the order tracking information in the user guide for the payment method you are using.

Accessing Reports

In addition to the Transaction Search feature in the Business Center, Payments generates predefined and on-demand reports to help you manage your orders. If at any time you feel that these reports are not necessary, you can disable them.

You can access your report settings by logging in to the Business Center with your administrative login credentials. The following table describes the information to enter for your administrative login credentials.

Table 4 Information for Administrative Login Credentials

Value	Description
Merchant ID	Your regular merchant ID followed by _acct.
User Name	Your regular merchant ID followed by _admin.
Password	The password for this account should be the same as the original password that you chose for your merchant ID.

Payments recommends that you periodically download and save your reports for reference. Your reports remain available for approximately one year in the Business Center.

The reports that are the most useful for reconciliation are:

- Transaction Request
- Payment Batch Detail Report
- Payment Events Report
- Payment Batch Summary Report

For detailed information about reports, see the *Business Center Reporting User Guide* (PDF | HTML).

Support Center

For more information about Payments services, log in to the Business Center and click **Support > Support Center**. At the Support Center, you can search the knowledge base, submit an eTicket, and browse for documentation.

Submitting an eTicket

Step 1 Log in to the Business Center:

- Live: https://Payments.com/sbc/

Step 2 Click **Support Center** near the top-left side of the page.

The Support Center appears.

Step 3 Click **Create eTicket**.

Step 4 Enter the following information on the Create eTicket screen:

- **First Name**: the first name of the person submitting the eTicket.
- **Last Name**: the last name of the person submitting the eTicket.
- **Phone Number**: the daytime phone number of the person submitting the eTicket.
- **Email Address**: the email address of the person submitting the eTicket.
- **Sensitive**: if you check this box, only the user submitting this eTicket and the Administrator on this account will be able to view this information.

⚠ Do not include personally identifiable information in your eTicket such as credit card numbers, card verification codes (CVV/CVC/CID), Social Security Numbers, or passwords.

- **Issue Type:** choose the most appropriate issue.
- **Summary**: enter a summary of the problem or request.
- **Description**: enter a detailed description of the problem or request.

Step 5 Click **Next**.

The eTicket is created.

CHAPTER 3

Processor API Basics

The Payments Processor API enables you to access Payments services using name-value pairs, XML, or SOAP toolkit. This document describes the name-value pairs and XML interfaces. For information about the SOAP interface, see the *SOAP Toolkits forWeb Services Developer Guide*.

Choosing a Client

The Processor API clients include the following components:

- Client libraries for communicating with Payments services
- Security libraries that digitally sign the messages
- Sample code that shows how to digitally sign the messages and use the client libraries
- SOAP proxy classes

The Processor API clients are available in various combinations of programming languages (ASP/COM, C/C++, Java, .NET, PHP, Perl), platforms (Windows, Solaris, Linux), and interfaces (name-value pairs, XML, SOAP). Choose a client SDK from the Processor API Clients page. After downloading an SDK, Payments recommendsthat you read the accompanying documentation, install the client, and test the client.

Choosing an API Version

Payments updates the Processor API regularly with new API fields and new functionality. With each update, the API version number increases. For the latest versionof the API, see the XML schema:

https://sbc.com/commerce/transactionProcessor/xml

Payments recommends that you use the latest version of the Processor API to take advantage of the full functionality of Payments services. See the *Processor API Release Notes* (PDF | HTML) for information about updates to the API.

When you configure your Processor API client, you must indicate which version of theAPI to use. The documentation for your client explains how to do it.

Name-Value Pairs or XML

The name-value pair interface is based on the XML schema. Most of the Payments documentation uses name-value pairs when discussing the Processor API. If you are using XML, you can easily translate the name-value pairs to the corresponding XML elements as described in "Correlating XML Elements and Name-Value Pair Fields," page 16.

Name-Value Pairs

When you use name-value pairs, a request includes the required name-value pairs for the services that you are requesting. Your Processor API client digitally signs and sends your request. Then you parse the reply message, which consists of name-value pairs.

XML

When you use XML, a request includes the required XML elements and attributes for the services that you are requesting. Your Processor API client digitally signs and sends your request. Then you parse the XML reply message.

Correlating XML Elements and Name-Value Pair Field Names

XML element names and name-value pair field names are related in the following ways:

- Each name-value pair field name matches the corresponding XML element name.
- The XML schema shows hierarchy with an underscore (_) separating the name of the parent element from the name of the child element.

In addition to separating the names of the parent elements from the names of the child elements, underscores can be included in the names of parent elements and child elements.

For example, the XML schema has a `<billTo>` element with several child elements. The following table shows some of the `<billTo>` child element names in the XML schema and the corresponding name-value pair field names.

Example XML Schema Names and Name-Value Pair Field Names

XML Schema Names	Corresponding Name-Value Pair Field Names
`<billTo>`	
`<city>`	**billTo_city**
`<country>`	**billTo_country**
`<postalCode>`	**billTo_postalCode**
`</billTo>`	

If you are using SOAP, the complex types in the XML schema translate to classes of the same name. For example, the `<billTo>` complex type in the schema translates to a `BillTo` class in the SOAP client.

Numbered Elements

The Payments XML schema includes several numbered elements. You can include these complex elements more than once in a request. For example, when a customer order includes more than one item, you must include multiple `<item>` elements in your request. Each item is numbered, starting with `0`. The XML schema uses an `id` attribute in the item's opening tag to indicate the number. For example:

```
<item id="0">
```

As a name-value pair field name, this tag is called `item_0`. In this portion of the field name, the underscore before the number does not indicate hierarchy in the XML schema. The item fields are generically referred to as `item_#_<element name>` in the documentation.

Below is an example of the numbered `<item>` element and the corresponding name-value pair field names. If you are using the Simple Object Access Protocol (SOAP), the client contains a corresponding `Item` class.

Example 1 Numbered XML Schema Element Names and Name-Value Pair Field Names

XML Schema Element Names	Corresponding Name-Value Pair Field Names

`<item id="0">`
`  <unitPrice>`
`  <quantity>`
`</item>`
	
	`item_0_unitPrice`
	`item_0_quantity`

`<item id="1">`
`  <unitPrice>`
`  <quantity>`
`</item>`
	
	`item_1_unitPrice`
	`item_1_quantity`

When a request in XML format includes an `<item>` element, the element must include an `id` attribute. For example: `<item id="0">`.

Constructing and Sending Requests

A request for a Payments service includes general information and information specificto the service that you are requesting. General information includes information about:

- You, the merchant
- The customer and the form of payment
- The items that the customer is buying

To indicate which service you are requesting, set the `run` attribute for the service to `true`. For example, to request the credit card authorization service, set the `run` attribute for the service as shown in the following table.

Example Setting the run Attribute for the Credit Card Authorization Service

Name-Value Pair:	`ccAuthService_run=true`

XML:	`<ccAuthService run="true">`
	`.`
	`.`
	`.`
	`</ccAuthService>`

MultiByte Characters and Special Characters

Payments supports multibyte characters for all services except Delivery Address Verification, Payment Tokenization, and Recurring Billing. Before implementing any of these services, contact Customer Support to discuss multibyte character support. All of the Processor API clients support UTF-8.

Unless otherwise noted, all field names are case sensitive, and all fields accept special characters such as `@`, `#`, and `%`.

The values of the **item_#_** fields must not contain carets (^) or colons (:) because these characters are reserved for use by Payments services.

The values of all request fields must not contain new lines or carriage returns. However, they can contain embedded spaces and any other printable characters. All leading and trailing spaces are removed.

Data Types

For more information about these data types, see the World Wide Web Consortium (W3C) XML Schema Part 2: Datatypes Second Edition.

Table 5 Data Type Definitions

Data Type	Description
Date and time	Format is YYYY-MM-DDThh:mm:ssZ, where: ▪ T separates the date and the time ▪ Z indicates Coordinated Universal Time (UTC), also known as Greenwich Mean Time (GMT) **Example** 2020-01-11T22:47:57Z equals January 11, 2020, at 22:47:57 (10:47:57 p.m.).
Integer	Whole number {..., -3, -2, -1, 0, 1, 2, 3, ...}
String	Sequence of letters, numbers, spaces, and special characters

Using Items or a Grand Total in a Request

For some services, you must specify the amount of the transaction. You can specify the amount either in a grand total for the entire transaction or in a separate amount for each product that the customer is purchasing.

Items

Items are the products that your customers purchase from you. When you send a request for a service that requires an amount, you can send item-specific information, such as the quantity of each item ordered and the unit price for each item. The items are referred to as **item_0**, **item_1**, **item_2**, and so on. Payments uses the information you provide for each item to calculate the grand total for the order.

The values for the **item_#_** fields must not contain carets (`^`) or colons (`:`) because these characters are reserved for use by Payments services.

Required Item-Level Fields

The value that you use for the **item_#_productCode** field determines which fields are required. If you omit the **item_#_productCode** field, its value defaults to `default`. If the value of the **item_#_productCode** field is one of the values in the bulleted list below, then the only required field is **item_#_unitPrice**:

- `default`
- a value related to shipping and handling

If you do not set the **item_#_quantity** field, it defaults to `1`.

If you do not set the **item_#_productCode** field to one of the values in the previous list, the following fields are required:

- item_#_unitPrice
- item_#_quantity
- item_#_productName
- item_#_productSKU

The **item_#_taxAmount** field is always optional.

Specifying Tax

To include tax for an item, use the **item_#_taxAmount** field. This value is the total tax for the entire quantity of that item. In other words, its value is not multiplied by the value of **item_#_quantity**.

Example Specifying Tax

```
item_0_unitPrice=10.00
item_0_quantity=5
item_0_taxAmount=4.00
```

The grand total for this transaction is (10.00 * 5) + 4.00 = 54.00.

Specifying Freight Charges

To include a shipping and handling charge for the order, you must include an additional item with the **item_#_productCode** field set to one of the following values:

- `shipping_only`
- `handling_only`
- `shipping_and_handling`

Example Specifying Freight Charges

```
item_0_unitPrice=10.00
item_0_quantity=5
item_0_taxAmount=4.00
item_1_unitPrice=4.95
item_1_quantity=1
item_1_productCode=shipping_only
```

The grand total for this transaction is (10.00 * 5) + 4.00 + (4.95 * 1) = 58.95.

Grand Total

Instead of using an itemized total, you can send a grand total for the order in the **purchaseTotals_grandTotalAmount** field. If you provide item-level information in addition to the grand total, Payments uses the grand total amount for the order total; Payments does not use the item-level information to calculate the order total. The item-level information is displayed on the Transaction Details page on the Business Center.

Coupons

You can offer your customers virtual coupons at your web store. Payments defines a coupon as a non-taxable, fixed amount deducted from an order total. Some examples of coupons you might offer are:

- Register now and get $100 off your purchase!
- Spring clearance! Get $10 off any order!
- Thank you for ordering again within 30 days! We’re taking $5 off your order!

How Coupons are Processed

This sequence summarizes how Payments processes a request that includes a coupon:

1. All items are totaled and then the coupon amounts are deducted, resulting in an order subtotal.

2. The order subtotal and order tax total are added to get an order grand total.

3. The order grand total is used by the services in your request.

   For example, if you requested an electronic check debit in the request with the coupon, the electronic check debit service uses the order grand total as the amount to charge.

Coupon Constraints

You cannot use coupons to do the following:

- Apply a discount to a specific item in a multi-item order
- Apply a discount to a specific item before tax is calculated
- Apply a percentage discount

The total coupon amount cannot be greater than the order grand total. Calculate your order totals before you send your requests to Payments so that you do not send orderswith negative subtotals. Payments returns an error for orders with negative subtotals.

Including a Coupon in the Request

To request a coupon with an order, include in the request an item with the product code set to `coupon`. For example, if your request contains two items, **item_0** and **item_1**, request a coupon by adding **item_2**. The following example shows how to specify a 10 USD coupon. The **quantity**, **productName**, and **productSKU** fields are required.

```
item_2_unitPrice=10.00
item_2_quantity=1
item_2_productCode=coupon
item_2_productName=Spring Clearance
item_2_productSKU=349209
```

Requesting a Follow-On Service

Request IDs and Request Tokens

⚠ Request tokens are required in follow-on request messages only for PayPal Express Checkout, China Processing, the Ingenico ePayments processor and the Atos processor. If you are using other payment methods and processors, you are not required to include request tokens in your request messages.

Request IDs and request tokens are identifiers that Payments returns in the reply messages for all services. You need to store these values because you will need them when you send a request for a follow-on service. For the request ID, the field name depends on the names of the primary service and follow-on service.

⚠ Although the name of the request token field that you receive is the same in each reply, the value for each field is different. Therefore, you must save each request token that you receive.

Alternative Payment Methods and Processors

For PayPal Express Checkout, China Processing, the Ingenico ePayments processor, and the Atos processor, a request for a follow-on service must include a request ID and a request token. Save the request ID from the reply for the primary service and send this value in the follow-on request.

Working with Request Tokens

⚠ Request tokens are required in follow-on request messages only for PayPal Express Checkout, China Processing, the Ingenico ePayments processor, and the Atos processor. If you are using other payment methods and processors, you are not required to include request tokens in your request messages.

The request token is a unique identifier that Payments assigns to each request and returns to you in each reply. This field is an encoded string that does not contain any confidential information, such as account numbers or card verification numbers. For China Processing and Atos, you need to store this value, which is a string that can contain up to 256 characters. Depending on how you process follow-on requests, you might need to retrieve the request token value and include it in your follow-on service requests:

- If you process primary requests with an API, but process follow-on requests through the Business Center, you do not need to provide the request token data.
- If you process primary and follow-on requests with an API, you need to retrieve the request token from the primary reply message and include it in the follow-on request message:

  Primary reply: `requestTokenrequest_token=AA4JUrWguaMUGwxSWVdPS5IM`

  Follow-on request: `orderRequestTokenorder_request_ token=AA4JUrWguaMUGwxSWVdPS5IM`

Merchants Who Have Not Implemented Request Tokens

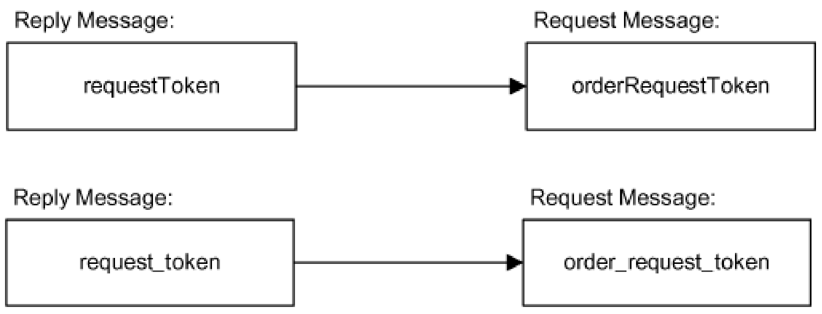
If you were a Payments merchant before July 2008 and never implemented request tokens, you have a choice:

- You can implement the request token as described in this document.
- You can let Payments store the request token data for you and retrieve it for you as necessary when you send follow-on requests.

Merchants Who Have Implemented Request Tokens

If you were a Payments merchant before July 2008 and implemented request tokens, your implementation uses a different field name for the request token that is sent to each follow-on service. With the new implementation, you use the **orderRequestTokenord** field for all follow-on services. This approach makes it easier to manage the request token values. It also reduces the likelihood of submitting a duplicate request by accidentally sending in the wrong request token value. You have a choice:

- You can continue to use your current implementation at this time. It is still supported, but Payments recommends that you update your implementation to use the **orderRequestToken** field.
- You can update your implementation to store the latest request token for a transaction in one database field to return with all follow-on services for the transaction:



- You can update your implementation without changing your database. Store the latest request token for a transaction in all of your existing request token database fields so that any follow-on request for the transaction returns the latest request token value:

The following figure uses the request field names for the credit card services. If you are using a different set of services, substitute the request field names for the services you are using.

Sending Requests

For live transactions, send requests to the location of the XML schema:

https://sbc.com/commerce/transactionProcessor

For test transactions, send requests to:

https://sbc.com/commerce /transactionProcessor

Handling Replies

After receiving a request from you, Payments responds with a reply message containing the results of the request that must be integrated into your system and anyother system that uses it. Payments recommends storing the data.

If you are using XML, you need to use an XML parser that supports namespaces because XML replies from Payments contain the namespace prefix **c:**.

Write an error handler to interpret the information in the reply message. Do not show the reply information directly to your customers. Instead, present an appropriate response that provides customers with the results of their transactions.

⚠ Because Payments can add reply fields and reason codes at any time, dothe following:

- Parse the reply data according to the names of the fields instead of their order in the reply. For more information about parsing reply fields, see the documentation for your client.
- Program your error handler to use the **decision** field to determine the result if it receives a reason code that it does not recognize.

Decisions

Every reply message includes the **decision** field, which summarizes the overall result of your request. Look at this field first to decide your course of action. The following table describes the possible values for the **decision** field. You are charged for all accepted and rejected requests. You are not charged for requests that result in errors.

Table 6 Possible Values for the Decision Field

Value	Description
`ACCEPT`	The request succeeded.
`ERROR`	A system error occurred. Errors that are caused by system problems are usually unrelated to the content of the request itself. You must design your transaction management system to correctly handle Payments system errors. Depending on which payment processor is handling the transaction, the error could indicate a valid Payments system error or it could indicate a processor rejection because of invalid data. Payments recommends that you do not design your system to endlessly resend transactions when a system occurs. Use the value of the **reasonCode** field to determine the reason for the `ERROR` decision. For more information about handling system errors and retries, see the documentation for your client.
`REJECT`	One or more of the service requests was declined. Requests can be rejected by Payments, the payment processor, or the issuing bank.For example: ■ Payments rejects a request if required data is missing or invalid. ■ The issuing bank rejects a request if the card limit is reached and funds are not available. This value can also be returned when an authorization transaction for a debit card or prepaid card is partially approved. See the information about debit cards and prepaid cards in the *Credit Card Services User Guide*. Use the value of the **reasonCode** field value to determine the reason for the `REJECT` decision.

Reason Codes

After looking at the value of the **decision** field, use the value of the **reasonCode** field to determine the reason for the decision and decide if you want to take further action:

- If the decision was `ERROR`, the reason code indicates the type of error that occurred.
- If the decision was `REJECT`, the reason code indicates the reason for the rejection and whether you can take action that might result in a successful order.

The **<service>_reasonCode** fields indicate the result of each service that you requested. For example, if you request a credit card authorization, the reply message includes the **ccAuthReply_reasonCode** field.

Payments reserves the right to add new reason codes at any time. If your error handler receives a reason code that it does not recognize, it should use the **decision** field to determine the result.

Missing or Invalid Fields

You are responsible for ensuring that the data you send to Payments is complete (no missing fields) and correct (no invalid data). Verify the data entered on your web sites and point-of-sale applications before sending the information to Payments.

If you send a request with missing or invalid information, the reply message includes the appropriate reason codes and one or more reply fields, **invalidField_0...N** or **missingField_0…N**, which list the fields you need to correct. The nature of the missing or invalid information determines the number and the content of the reply fields. For example, if three required fields are missing from your request, the reply includes at least one and up to three fields named **missingField_0, missingField_1,** and **missingField_2.** You need to correct these fields and resubmit the request.

Because the API behavior pertaining to these reply fields is always subject to change, do not use these fields to communicate with customers.

For XML, the `<missingField>` and `<invalidField>` elements are not numbered. Instead, the reply includes multiple `<missingField>` or `<invalidField>` elements.

If you are using SOAP, the reply includes an array of the missing fields and an array of the invalid fields.

Request IDs

Payments returns a request ID in the reply for every service. You must store this value because you will need it when you send a request for a follow-on service as described in "Requesting a Follow-On Service," page 23.

Although the name of the request ID field is the same in each reply, the value for the field is different in each reply. Therefore, you must save each request ID that you receive.