API Specification

Click-to-Pay APS Feature

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Author** |
| Original | August 7, 2017 | Peter Rankin |

## Introduction

The purpose of this document is to outline the API allowing ERP systems to communicate with the “click-to-pay” feature, integrating invoices with the APS gateway. ERP systems can upload invoices and query payments. Customers can click the auto-generated link provided by the API to pay invoices.

## Processing Flow

There are two process flows: *Real Time* and *Batch.* Many ERP systems are real time in that any changes made to an invoice are posted immediately throughout the linked databases and there is not a separate journal and update to make those postings. Batch systems do the updates as a discrete process usually with reporting and journal updates. This API specification allows you to mix the two process flows depending on your ERP system needs.

## Daily Processing

APS ClickToPay requires that you upload the invoice information to the gateway and it will return a link code. This code uniquely identifies the invoice within the ClickToPay database so that the merchant can send a link, including the code, to the customer (see section 4.2). When the customer clicks on this link, the invoice displays in the browser with option to pay via credit card or ACH transaction.

Periodically, the ERP system must retrieve the payment transactions in order to create cash receipts. After each payment is applied to the appropriate invoice in the ERP system, that payment transaction in the APS ClickToPay gateway must be tagged as retrieved so that the gateway will now allow it to be downloaded again. This simplifies retrieval of payments to only unposted payments unless otherwise specified.

## Real Time Processing

ERP packages like Microsoft Dynamics and SAP use the real time model. As such, you will need to use the Invoice/Create (3.1) endpoint the invoice as the first line of the invoice is accepted. All subsequent lines or changes would use the Invoice/Update (3.6) endpoint. Alternately, you can use a service to poll for invoice changes and upload if change is detected. If the invoice is deleted, use the Invoice/Delete (3.2) endpoint.

Using either a batch program or a scheduled job, you would periodically scan for payments using /payments/query (3.8) and post the payments against the invoices. Immediately after posting the payment, you must tag the payment as retrieved using /payments/markRetrieved (3.7) using the code received in the /payments/query transaction so that the same payment doesn’t show available again. If you receive payment from another source (i.e. not Click To Pay), you would need to use the /invoices/logExternalPayment (3.4) endpoint to post the payment so the invoice on the gateway shows the current balance if not fully paid.

## Batch Processing

ERP packages like Sage 100 use either a batch or blended real time approach. Invoices are created or updated as they are accepted in the entry programs and deleted if the invoice is deleted from the entry program using the same endpoints as in section 1.4. Otherwise, the invoices can be created during journal pre-scans or updating.

Linking to the cash receipts entry or by a download that creates the cash receipts is the simplest method to download the payments using the /payments/query endpoint (3.8). After posting as part of the journal update, use the /payments/markRetrieved (3.7) to mark the payment as retrieved using the token received during the query. If you receive payment from another source (i.e. not Click To Pay), you would need to use the /invoices/logExternalPayment (3.4) endpoint to post the payment so the invoice on the gateway shows the current balance if not fully paid.

Note: If you have high volume, you can get hundreds of payments. The /payment/query is designed to retrieve the unposted payments in blocks of 50 to 100 payments to allow you to “throttle” the flow and keep from timing out the connection. You will need to send the block size and the starting block, which always begins at 1. The endpoint will return the last block number send which becomes the new block number for the next block.

You will repeat this process until isEndOfSet is true.

# Common Request Attributes

The following parameters and results are common to all requests and are not duplicated in specific API call documentation.

## Parameters

The following parameters are common to all requests and are always required.

|  |  |
| --- | --- |
| Name | Description |
| apiKey | The API key used to access the API and authenticate the merchant. These can be created from the C2P portal, or you can obtain one from your system administrator. |

## Results

The following information is included in all results.

|  |  |
| --- | --- |
| Name | Description |
| errorMessage | Any detailed information about an error in the request will be included here |
| statusClassCode | (Integer) The overall status of the response. 1=Success, 2=Authorization failure, 3=Error |
| statusCode | (Integer) A 5-digit status code for the request. The first digit is the statusClassCode. You can use this code to look up more specific information on the error. |

For global codes, the second digit is always 0 (codes specific to one or more endpoints have a non-zero second digit). Here is a list of global status codes (also included in the status code appendix):

* 10000: Success
* 20000: Authorization failure (no more specific detail given, for security purposes)
* 30000: Unspecified system error
* 30001: General validation error
* 30002: Invalid operation exception; for instance, attempting to mark an invoice as paid which is already marked as such.
* 30003: Item not found exception; the specified item to query or manipulate does not exist.

## Format

The preferred request and response format is JSON, which will be used for the samples in this document. For support in using XML, contact us.

At least one sample request and response is included with each endpoint. Here is a sample for /invoices/query:

{  
 "apiKey": "1234567890",  
 "customerId": "34567",  
 "invoiceId": "INV 349210"  
}

And the response:

{  
 "statusClassCode": 1,  
 "isPaid": false,  
 "outstandingAmount": 50.25  
}

When nested parameters are specified in the spec (e.g., payments/invoices/amount), their structure is predictable using JSON arrays; i.e., a sample response for /payments/query:

{

"statusClassCode": 1,

"payments": [

{

"date": "20170131T120000Z",

"invoices": [

{

"amount": 100,

"invoiceId": "INV 9385",

"remainingBalance": 45.25

},

{

"amount": 26.53,

"invoiceId": "INV 2953",

"remainingBalance": 0

}

]

},

. . .

]

}

## Document Conventions

Parameters are listed alphabetically according to their section.

### Parameter Types

Unless otherwise specified, parameter types are strings. Default formats are as follows unless otherwise specified (often they are specified explicitly for clarity):

* Currency: ISO 4217
* Country: ISO 3166-1 alpha-3
* State/province: ISO 3166-2
* Dates/times: ISO 8601, in a format compatible with [round-trip “O” format](https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-date-and-time-format-strings#Roundtrip) in C#. UTC preferred. E.g., 2018-01-01T12:00:00Z
* Decimal: Standard JavaScript number format; e.g., “param1”: -1234567.89
* Integer: Standard JavaScript integer format; e.g., “param1”: 12345
* Boolean: Standard JavaScript format. E.g., “param1”: true

### Parameter Decorators

In the spec, parameters may be decorated with asterisks (\*) or other symbols to indicate a certain status (i.e., being required for Level III transactions). The decorators are not part of the parameter name; they just indicate something unique about that parameter. E.g., it may be listed as param1\* in the spec, but just pass in param1 when making the API call (no asterisk).

# Methods

## POST /invoices/create

Create an invoice in the click-to-pay system

### Examples

Request:

{

apiKey: "12345",

amount: 54.24,

companyId: "aw\_sc",

customerId: "jsmith1980",

currency: "USD",

customerName: "John Smith",

invoiceId: "6833",

customerEmail: "john.smith@example.com",

orderDescription: "My Order",

poNumber: "12345",

orderId: "23456",

shipFromPostal: "34567",

shippingAmount: 8.12,

summaryCommodityCode: "56789",

shipping: {

line1: "123 Some Lane",

line2: "",

city: "Greenville",

country: "USA",

firstName: "John",

lastName: "Smith",

postalCode: "12345",

state: "AK"

},

billing: {

line1: "123 Some Lane",

line2: "",

city: "Greenville",

country: "USA",

firstName: "John",

lastName: "Smith",

postalCode: "12345",

state: "AK"

},

products: [

{

commodityCode: "253",

customItemCode: "code123",

description: "Basket Ball",

discount: 1,

erpItemCode: "253",

extendedPrice: 30,

tax: 1,

unitOfMeasure: "EACH",

unitPrice: 15,

quantity: 2,

sku: "basket-ball"

},

{

commodityCode: "254",

customItemCode: "code234",

description: "Air Pump",

discount: 0,

erpItemCode: "254",

extendedPrice: 16,

tax: 1,

unitOfMeasure: "EACH",

unitPrice: 15,

quantity: 1,

sku: "air-pump"

}

]

}

Response:

{

"clickToPayLinkCode": "GIWdGvmfNTuNE89KDHqErpnnrc40siaj",

"errorMessage": null,

"statusClassCode": 1,

"statusCode": 10000,

"exceptionId": null

}

### Parameters

|  |  |
| --- | --- |
| Name | Description |
| \*amount | (Required, currency) The amount of the invoice in the specified currency |
| billing/address1 | First line of the billing address |
| billing/address2 | Second line of the billing address |
| billing/cityβ | The name of the city for shipping |
| billing/country | (ISO 3166-1 alpha-3) The code of the country for shipping |
| billing/firstNameβ | The first name for the billing address |
| billing/lastNameβ | The last name for the billing address |
| billing/postalCodeβ | The postal code for shipping |
| billing/state | (ISO 3166-21; e.g., SC for South Carolina) The state or province for the billing address |
| \*companyId | (Required) The ID in the ERP of the company whose customer is to be invoiced |
| \*currency | (Required, ISO 4217) The currency in which the total and line item amounts are stored |
| customerEmail | The customer’s e-mail address. Required if “sendClickToPayEmail” is set to “true”. |
| \*customerName | (Required) The name of the customer |
| \*customerId | (Required) The unique ID of this customer in the ERP system. Invoices with the same customer ID and the same company ID will be automatically linked, so it is important for this value to be correct. |
| \*invoiceId | A unique ID for the invoice document as generated by the ERP (i.e., the ERP’s invoice ID). Must be unique across all invoices for the same customer. |
| isLevel3 | (Boolean) Set to “true” if you want the C2P API to do some preliminary checking to try to ensure that all level-3 data is present. If it is not, an error code is returned (50001), and the invoice is not created. |
| orderIdβ | An ID for the order, which can be distinct from the invoice ID |
| orderDescriptionβ | A description for the order |
| poNumber | Typically, the customer’s purchase order number |
| products/commodityCode | The APS-assigned commodity code for this product |
| products/custItemCode | (Default: erpItemCode, falls back to sku) An item code, which can sometimes be unique per customer, for this product. This is displayed, as the primary product identifier, to the customer when viewing the invoice on the C2P poral. |
| products/descriptionβ | The description of the product |
| products/discountβ | (Decimal) The discount applied to each unit, or 0.00 for no discount |
| products/erpItemCode | The unique identifier for this product in the ERP system |
| products/extendedPrice | (Decimal) The customer price for this detail line before tax and discount |
| products/taxβ | (Currency) -1.00 for tax exempt; 0.00 for no tax |
| products/unitOfMeasureβ | Acceptable values: EACH (TODO: Get more) |
| products/unitPriceβ | (Currency) The cost of each unit before the tax and discount are applied |
| products/quantityβ | (Decimal) The quantity of products for this line item |
| products/skuβ | A unique identifier for this product. Displayed to the customer in the portal if erpItemCode and custItemCode are not set. |
| sendClickToPayEmail | (Boolean) If true, send an e-mail to the customer with the click-to-pay link provided. Requires that “customerEmail” be set. |
| shipFromPostalβ | The postal address from which the item will be shipping |
| shipping/address1β | The first line of the shipping street address |
| shipping/address2 | The second line of the shipping street address |
| shipping/cityβ | The name of the city for shipping |
| shipping/country | (ISO 3166-1 alpha-3) The code of the country for shipping |
| shipping/firstNameβ | The first name for the shipping address |
| shipping/lastNameβ | The last name for the shipping address |
| shipping/postalCodeβ | The postal code for the shipping address |
| shipping/stateβ | (ISO 3166-2; e.g., SC for South Carolina) |
| shippingAmountβ | (Currency) The amount of the invoice’s total to go toward shipping |
| tax | (Currency, 0.00 for tax-exempt orders) The total amount of tax on the order |
| timestamp | (ISO 8601, UTC preferred; e.g., 20170131T120000Z, defaults to current date/time) The time the invoice was billed |

β Required for Level III processing

### Result

|  |  |
| --- | --- |
| Name | Description |
| clickToPayLink | The secure token which can be used to construct a link to the C2P portal, where the user can pay the invoice. For details on constructing the link, please see 4.2: Click-to-Pay Link Creation. |

### Errors

#### [50001] Level III validation error

The invoice was marked as Level III yet did not pass preliminary Level III inspection.

#### [50002] Company does not exist

If a company with the specified company ID does not exist, an error is thrown. The company must first be created on the web site before an invoice can be uploaded to it. (TODO: Will this be too much hassle? Should we auto-create the company if it does not exist?)

#### [50003] Invoice already exists

If another active, completed invoice with the same ID for the same customer already exists, an error is thrown.

## POST /invoices/delete

This deletes an invoice completely from the C2P system. Only invoices with no payments yet associated with them can be deleted.

### Examples

Request:

{

apiKey: "",

companyId: "aw\_sc",

customerId: "jsmith1980",

invoiceId: "test\_temp\_10"

}

Response:

{

"errorMessage": null,

"statusClassCode": 1,

"statusCode": 10000,

"exceptionId": null

}

### Parameters

|  |  |
| --- | --- |
| Name | Description |
| \*invoiceId | (Required) The ERP-assigned ID of the invoice to be deleted |
| \*companyId | (Required) The ERP-assigned ID of the company containing the invoice to be deleted |
| \*customerId | (Required) The ERP-assigned ID of the customer whose invoice is to be deleted |

### Errors

#### [30005] Invoice cannot be deleted

This error occurs when the invoice cannot be safely deleted from the C2P system. For instance, one or more payments may have already been made against this invoice.

## POST /invoices/getCode

Get the code or token used to generate a web link that can be given to a customer allowing them to pay the invoice.

### Examples

Request:

{

apiKey: "",

companyId: "aw\_sc",

customerId: "jsmith1980",

invoiceId: "test\_11"

}

Response:

{

"clickToPayLinkCode": "sp4pen0CrH7lr-O9P8SvA-4r1-ew3Dqh",

"errorMessage": null,

"statusClassCode": 1,

"statusCode": 10000,

"exceptionId": null

}

### Parameters

|  |  |
| --- | --- |
| Name | Description |
| \*invoiceId | (Required) The ERP-assigned ID of the invoice to be deleted |
| \*companyId | (Required) The ERP-assigned ID of the company containing the invoice to be deleted |
| \*customerId | (Required) The ERP-assigned ID of the customer whose invoice is to be deleted |

### Results

|  |  |
| --- | --- |
| Name | Description |
| clickToPayLinkCode | The secure token which can be used to build a web link allowing the user to pay the invoice |

## POST /invoices/logExternalPayment

Log an invoice payment by a source external to the click-to-pay system (e.g., paid through the ERP directly). That way, the invoice doesn’t show up as unpaid when that customer logs into the click-to-pay system.

### Examples

Request:

{

apiKey: "",

amount: 15,

companyId: "aw\_sc",

customerId: "jsmith1980",

date: "2017-09-13",

invoiceId: "4818924"

}

Response:

{

"isPaid": false,

"outstandingAmount": 35,

"id": "112",

"errorMessage": null,

"statusClassCode": 1,

"statusCode": 10000,

"exceptionId": null

}

### Parameters

|  |  |
| --- | --- |
| Name | Description |
| \*customerId | (Required) The ERP-assigned customer ID associated with the invoice |
| \*invoiceId | (Required) The ERP-assigned invoice ID |
| \*amount | (Required, currency) The amount of the payment |

### Results

|  |  |
| --- | --- |
| Name | Description |
| id | The unique ID of this external payment in the API system as assigned by our C2P gateway |

### Errors

#### Invalid Amount

This error code is returned when an invalid amount is passed in (i.e., more than the outstanding balance, or negative).

## POST /invoices/query

Query the status of an invoice

### Examples

Request:

{

apiKey: "test\_XweF4yRa5jxrwW17zDHNT8C8YgAt1l",

companyId: "aw\_sc",

customerId: "jsmith1980",

invoiceId: "test\_4818923"

}

Response:

{

"isPaid": true,

"outstandingAmount": 0,

"errorMessage": null,

"statusClassCode": 1,

"statusCode": 10000,

"exceptionId": null

}

### Parameters

|  |  |
| --- | --- |
| Name | Description |
| \*companyId | (Required) The ERP-assigned ID of the company containing the customer whose invoice to query |
| \*customerId | (Required) The ID of the customer whose invoice to query |
| \*invoiceId | (Required) The invoice ID to query |

### Results

|  |  |
| --- | --- |
| Name | Description |
| isPaid | (Boolean) Whether the invoice has been paid |
| outstandingAmount | (Currency) The amount on the invoice still outstanding, or 0.00 if the invoice has been paid in full |

## POST /invoices/update

Update an invoice after it has been initially submitted. Any information provided in the optional parameters will overwrite previously provided information. Line items are overwritten, not appended. A previously submitted invoice with the same invoice, customer, and company IDs must already exist.

### Examples

Request:

{

apiKey: "",

amount: 50,

companyId: "aw\_sc",

customerId: "jsmith1980",

currency: "USD",

customerName: "John Smith",

invoiceId: "12345",

customerEmail: "john.smith@example.com",

orderDescription: "My Order",

poNumber: "12345",

orderId: "12345",

shipping: {

line1: "123 Some Lane",

line2: "",

city: "Greenville",

country: "USA",

firstName: "John",

lastName: "Smith",

postalCode: "12345",

state: "AK"

}

}

Response:

{

"clickToPayLinkCode": "W5HcS-qTbpF8DQgB4rgxnSf6r3lVUoEo",

"errorMessage": null,

"statusClassCode": 1,

"statusCode": 10000,

"exceptionId": null

}

### Parameters

Required parameters include:

* amount
* companyId
* customerId
* invoiceId

Documentation for each of these parameters is stored under 3.1.2 (parameters for POST /invoices/create). All other parameters required by POST /invoices/create which are not required here are optional parameters in this action.

### Result

This is the same as for POST invoices/create.

### Errors

#### [40001]: Amount updated

If the amount of the invoice was changed because of this update (courtesy notice status)

#### [50004]: Invoice does not exist

If an invoice with the same number and customer number and company number does not already exist in the current instance, then this error is thrown.

## POST /payments/markRetrieved

Mark a payment as retrieved by the ERP system. This can be helpful in allowing the ERP to query only payments not marked as retrieved.

### Examples

Request:

{

apiKey: "",

id: 104

}

Response:

{

"errorMessage": null,

"statusClassCode": 1,

"statusCode": 10000,

"exceptionId": null

}

### Parameters

|  |  |
| --- | --- |
| Name | Description |
| \*id | (Required) The C2P-assigned ID of the payment to mark as retrieved |

## POST /payments/query

This returns a list of payments ordered from most to least recent.

### Examples

Request:

{

apiKey: "",

companyId: "aw\_sc",

customerId: "jsmith1980",

blockNumber: 1,

numItemsPerBlock: 2,

minDate: "2018-01-01T12:00:00Z",

maxDate: "2018-03-01T12:00:00Z",

notRetrievedOnly: false

}

Response:

{

"blockNumber": 1,

"isEndOfSet": false,

"numItemsPerBlock": 2,

"payments": [

{

"amount": 1,

"currency": "USD",

"customerId": "JSMITH1980",

"date": "2/6/2018 2:00:51 PM",

"isMarkedRetrieved": false,

"id": "505936",

"invoices": [

{

"amount": 1,

"invoiceId": "test\_11",

"remainingBalance": 4

}

]

},

{

"amount": 1,

"currency": "USD",

"customerId": "JSMITH1980",

"date": "2/6/2018 1:40:32 PM",

"isMarkedRetrieved": false,

"id": "505935",

"invoices": [

{

"amount": 1,

"invoiceId": "test\_10",

"remainingBalance": 4

}

]

}

],

"errorMessage": null,

"statusClassCode": 1,

"statusCode": 10000,

"exceptionId": null

}

### Parameters

|  |  |
| --- | --- |
| Name | Description |
| blockNumber | (Integer, default 1) The 1-based ordinal of the block or “page” to retrieve |
| \*companyId | (Required) The ID of the company whose payments to query |
| \*customerId | The customer whose payments to query |
| maxDate | (Date/time) The latest payment date/time to be included in the query results. Defaults to no limit. This is an exclusive boundary, so for example, if midnight on February 1 is passed, anything in and before January will be included, but nothing from February. |
| minDate | (Date/time, default today UTC minus 7 days) The earlest payment date/time to be included in the report. This is an inclusive boundary, so any payments falling exactly on this time will be included. |
| notRetrievedOnly | (Boolean, default false) If set to “true”, only those payments not yet marked as retrieved by the ERP will be fetched. This allows the ERP to not have to sort through lots of repeat entries if polling for new entries. POST /payments/markRetrieved must be called to mark a payment as retrieved. |
| numItemsPerBlock | (Integer, default 50, max 100) The number of items to request per block in the response. If you set this value, be sure to keep it consistent throughout all pages requested in the same operation. |

### Results

|  |  |
| --- | --- |
| Name | Description |
| blockNumber | (Integer, default 1) The 1-based ordinal of the block that was retrieved |
| payments/amount | (Currency) The total amount of the payment (sum of all invoice payment line items) |
| payments/currency | (ISO 4217) The currency in which the payment was made. Returns what was passed into POST /invoices/create, and so ISO 4217 compliance depends on the value passed at invoice creation. |
| payments/date | (Date/time) The date and time the payment was made (UTC) |
| payments/id | The unique ID of the payment in the online system. ERP programmers can use this to keep track of payments. |
| payments/isMarkedRetrieved | (Boolean) Whether this payment has been marked as retrieved. To mark it as retrieved, call POST /payments/markRetrieved. |
| payments/invoices/amount | (Currency) The amount of the payment which went toward this invoice |
| payments/invoices/invoiceId | The ERP-assigned invoice ID |
| payments/invoices/ remainingBalance | The remaining balance on the invoice (helpful so the ERP doesn’t have to keep track of this manually or perform a separate invoice status query) |

# Appendixes

## Appendix #1: Error Codes

The list below is sorted ascending by error code number. Each has a brief description of the exception, and when applicable, a link to a more detailed description.

* 10000: Success
* 20000: Authorization failure (no more specific detail given, for security purposes)
* 30000: Unspecified system error
* 30001: General validation error
* 30002: Invalid operation exception; for instance, attempting to mark an invoice as paid which is already marked as such.
* 30003: Item not found exception; the specified item to query or manipulate does not exist.
* 30004: Malformed exception; the request could not be parsed or understood. Examples include invalid JSON or an improperly formatted date/time string
* 30005: Invoice cannot be deleted from the system as requested because, e.g., payments have already been made against it (3.2.2.1)
* 40001: Amount updated. See 3.4.4.1
* 50001: Level III validation error (3.1.4.1)
* 50002: Company does not exist (3.1.4.2)
* 50003: Invoice already exists (3.1.4.3)
* 50004: Invoice does not exist (3.4.4.2)

## Click-to-Pay Link Creation

To create a click-to-pay link from the secure token returned from the API, follow this format:

https://portal.apsclicktopay.com/Invoices/ClickToPay?code=TOKEN